Нова українська школа

Інна Тріщук, Олександр Лазарець

ІНФОРМАТИКА

6 клас



ДІЇ У ВИПАДКУ ФІШИНГОВОЇ АТАКИ

1 Не взаємодійте з підозрілим повідомленням

Якщо ви отримали підозрілий електронний лист або повідомлення, не натискайте на жодні покликання і не відкривайте вкладення.

2 Перевірте свої облікові записи

- Якщо ви підозрюєте, що ваші облікові дані могли бути скомпрометовані, негайно змініть паролі. Використовуйте складні паролі і, якщо можливо, увімкніть двофакторну аутентифікацію (2FA).
- Перевірте свої облікові записи на наявність підозрілої активності.

3 Виконайте сканування системи

- Переконайтеся, що адреса вебсайту починається з **https://** і містить правильний домен.
- Якщо вебсайт має незвичний вигляд або містить помилки, можливо, це підроблений сайт.

4 Використовуйте додаткові заходи безпеки

- Двофакторна аутентифікація (2FA).
- Оновлення програмного забезпечення.
- Резервне копіювання даних.

Захист власного цифрового середовища від зловмисних дій

1. Використання антивірусного програмного забезпечення
2. Оновлення програмного забезпечення
3. Використання складних паролів
4. Двофакторна аутентифікація (2FA)
5. Захист мережі

ОСНОВНІ ФУНКЦІЇ

Основні текстові функції

CONCATENATE	об'єднує кілька текстових рядків в один
UPPER	перетворює текст у верхній регістр
LOWER	перетворює текст у нижній регістр
LEN	рахує кількість символів у тексті

Математичні функції

SUM	підсумовує числа в діапазоні клітинок
AVERAGE	обчислює середнє арифметичне чисел у діапазоні
MIN і MAX	знаходить мінімальне і максимальне значення в діапазоні клітинок
ROUND	округлює число до заданої кількості десяткових знаків
POWER	підносить число до степеня
SQRT	обчислює квадратний корінь числа

Логічні функції

IF	перевіряє умову і повертає одне значення, якщо умова істинна, й інше, якщо хибна
AND	повертає істину, якщо всі умови істинні, і хибність, якщо хоча б одна умова хибна
OR	повертає істину, якщо хоча б одна умова істинна
NOT	повертає істину, якщо умова хибна, і хибність, якщо умова істинна
IFERROR	повертає задане значення у випадку помилки

Статичні функції

COUNT	рахує кількість клітинок із числовими значеннями
MODE	визначає моду (найчастіше буває значення у вибірці)
MEDIAN	обчислює медіану (середнє значення вибірки, яке ділить дані на дві рівні частини)
STDEV	обчислює стандартне відхилення, що показує, наскільки значення у вибірці відхиляються від середнього значення

Фінансові функції

PMT	обчислює щомісячний платіж за позикою на основі постійних платежів і постійної процентної ставки
CUMIPMT	обчислює загальну суму відсотків, що виплачуються за конкретний період для позики
PV	обчислює теперішню вартість інвестиції або позики на основі майбутніх виплат і процентної ставки
FV	обчислює майбутню вартість інвестиції на основі регулярних внесків, процентної ставки та кількості періодів

Інна Тріщук
Олександр Лазарець

ІНФОРМАТИКА

ПІДРУЧНИК ДЛЯ 8 КЛАСУ
закладів загальної середньої освіти

Рекомендовано Міністерством освіти і науки України



ТЕРНОПІЛЬ
НАВЧАЛЬНА КНИГА — БОГДАН
2025

УДК 004(075.2)
Т 67

Рекомендовано Міністерством освіти і науки України
(наказ Міністерства освіти і науки України
від 21.02.25 № 347)

Підручник створено за Модельною навчальною програмою
«Інформатика. 7–9 класи» для закладів загальної середньої освіти
(авт. Пасічник О.В., Козак Л.З., Ворожбит А.В.)

Тріщук І. В.
Т 67 Інформатика : підручник для 8 кл. закладів загальн. середн. освіти / І. В. Тріщук, О. Ю. Лазарець. — Тернопіль : Навчальна книга — Богдан, 2025. — 376.

ISBN 978-966-10-8987-6

Зміст підручника відповідає Державному стандарту загальної середньої освіти та Модельній навчальній програмі «Інформатика. 7–9 класи» (авт.: Пасічник О.В., Козак Л.З., Ворожбит А.В.).

Для учнів та учениць 8 класу.

УДК 004(075.2)

Завантажуйте безкоштовний інтерактивний додаток,
використовуючи детальну інструкцію, за покликанням:
https://edodatok.com/8987-6/

Тех. підтримка: admin@edodatok.com

Охороняється законом про авторське право.
Жодна частина цього видання не може бути відтворена
в будь-якому вигляді без дозволу видавництва.

ISBN 978-966-10-8987-6

© І. В. Тріщук, О. Ю. Лазарець, 2024
© Навчальна книга — Богдан, виключна
ліцензія на видання, оригінал-макет, 2025

Шановні восьмикласниці та восьмикласники!

Ви тримаєте в руках підручник з інформатики, який допоможе вам розширити й поглибити здобуті в 7-му класі знання. У цьому підручнику буде багато новинок про використання штучного інтелекту, генерування зображень, способи захистити свою інформацію та багато тем із програмування! Ми будемо навіть створювати свій YouTube-канал та публікувати власні відео!

Підручник складається із шести розділів:

- Продуктивність та безпека цифрового середовища
- Збір і опрацювання даних
- Сайт та його верстка
- Проєктування графічного інтерфейсу
- Алгоритми та їхні коди
- Стиль цифрового образу

Розділи поділено на теми, кожна з яких містить теоретичний матеріал, практичну роботу за персональним комп'ютером, домашнє завдання та перевірку знань. Після виконання практичної роботи ви зможете перевірити себе, відсканувавши QR-код та відповівши на кілька запитань. Також, скориставшись посиланням за QR-кодами, ви матимете можливість переглянути інтерактивні презентації, що допоможуть вам закріпити вивчене.

Тож старанно вивчайте теорію, вдосконалюйте практичні навички користування комп'ютером, створюйте цікаві інтерактивні проєкти, а головне — робіть усе із задоволенням!

Автори

УМОВНІ ПОЗНАЧЕННЯ



Для вчителя/вчительки: додайте тест у свою бібліотеку ClassTime, щоб бачити статистику учнів



Для учня/учениці: проскануйте QR-код та пройдіть перевірку



Додаткове завдання



Домашнє завдання



Для учня/учениці: проскануйте QR-код та виконайте завдання для кмітливих

Розділ 1

ПРОДУКТИВНІСТЬ ТА БЕЗПЕКА ЦИФРОВОГО СЕРЕДОВИЩА

Тема 1

Цифрове робоче місце. Види та можливості сучасних комп'ютерів. Архітектура та елементна база цифрових пристроїв. Процесор. Пам'ять. Кодування даних на цифрових носіях

Ключові поняття

Цифрове робоче місце
Персональні комп'ютери
Кодування даних
Одиниці вимірювання інформації

Цифрове робоче місце

Цифрове робоче місце — це поєднання всіх інструментів, технологій та ресурсів, необхідних для виконання робочих завдань.



- Обладнання
- Програмне забезпечення
- Мережеві ресурси
- Безпека

Обладнання

- **Комп'ютери та ноутбуки:** потужні та надійні пристрої для виконання різних завдань.
- **Планшети та смартфони:** для мобільного доступу до робочих ресурсів.
- **Монітори:** великі та високоякісні дисплеї для зручної роботи.
- **Периферійні пристрої**: клавіатура, миша, принтери, сканери та інше обладнання.
- **Джерела безперебійного живлення (UPS):** для захисту від перебоїв електропостачання.

Програмне забезпечення

- **Операційні системи: Windows**, **macOS**, **Linux** для ПК; **iOS**, **Android** для мобільних пристроїв.
- **Офісні додатки: Microsoft Office 365**, **Google Workspace** для створення та обробки документів.
- **Комунікаційні інструменти: Slack**, **Microsoft Teams**, **Zoom** для спілкування та відеоконференцій.
- **Хмарні сервіси: Google Drive**, **Dropbox**, **OneDrive** для зберігання та обміну файлами.
- **Системи управління проєктами: Trello**, **Asana**, **Jira** для координації роботи команд.

Мережеві ресурси

- **Мережеве обладнання:** Wi-Fi-маршрутизатори, комутатори, VPN для створення безпечного та надійного доступу до мережі.
- **Сервери та дата-центри:** для зберігання даних та роботи додатків.
- **Резервне копіювання та відновлення:** інструменти для збереження даних та їх відновлення у разі втрати.

Безпека

- **Антивірусне програмне забезпечення:** захист від вірусів та шкідливого ПЗ.
- **Фаєрволи:** захист мережі від несанкціонованого доступу.
- **Системи управління доступом:** контроль доступу до ресурсів та даних.
- **Шифрування:** захист даних під час зберігання та передачі.

Види та можливості сучасних комп'ютерів

Сучасні комп'ютери класифікуються за різними критеріями, зокрема за типом, розміром, призначенням та обчислювальною потужністю.

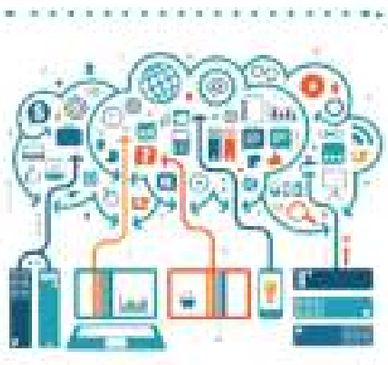
Персональні комп'ютери (ПК)



Сервери



Вбудовані комп'ютери



Поративні (мобільні) комп'ютери



Персональні комп'ютери (ПК) – це електронні пристрої, призначені для індивідуального використання.

Сервери — це потужні комп'ютери або комп'ютерні програми, які надають різноманітні послуги іншим комп'ютерам у мережі (клієнтам). Вони відіграють ключову роль в обробці та зберіганні даних, забезпеченні безперебійної роботи інтернет-сервісів, вебсайтів, корпоративних додатків та багато іншого.

Вбудовані комп'ютери є спеціалізованими обчислювальними системами, які інтегровані в інші пристрої для виконання конкретних функцій. Вони відрізняються від звичайних комп'ютерів своєю спеціалізацією та часто мають обмежені ресурси.

Портативні (мобільні) комп'ютери — це комп'ютери в яких є батарея, що дозволяє певний час працювати пристрою без підключення до електромережі.

Портативні (мобільні) комп'ютери

Мобільні комп'ютери представляють собою портативні обчислювальні пристрої, які забезпечують користувачам можливість працювати, спілкуватися та розважатися в будь-якому місці. Основна перевага мобільних комп'ютерів — їхня портативність та можливість автономної роботи від вбудованого акумулятора.



Ноутбуки (Laptops) — це портативні комп'ютери, які об'єднують дисплей, клавіатуру, сенсорну панель (трекпад) та інші компоненти в одному корпусі.

- Виконання широкого спектру завдань: від офісних додатків до мультимедійних розваг та ігор.
- Потужні моделі можуть використовуватись для графічного дизайну, програмування, відеомонтажу.

Нетбуки (Netbooks) — це малі, легкі та енергоефективні ноутбуки, призначені для базових завдань, таких як вебсерфінг та робота з документами.

- Легкість та компактність.
- Тривалий час роботи від акумулятора.
- Обмежена продуктивність у порівнянні зі звичайними ноутбуками.

Ультрабуки (Ultrabooks) — це тонкі та легкі ноутбуки з високою продуктивністю, розроблені для забезпечення тривалого часу автономної роботи та зручності у транспортуванні.

- Висока продуктивність з низьким енергоспоживанням.
- Тривалий час роботи від акумулятора.
- Тонкий та легкий корпус.

Планшети (Tablets) — це портативні комп'ютери з сенсорним екраном, які можуть використовуватися без фізичної клавіатури та миші.

- Сенсорний екран для зручного управління.
- Велика кількість додатків для різних завдань.
- Можливість підключення зовнішніх пристроїв (клавіатура, стилус).
- Легкість та портативність.

Смартфони (Smartphones) — це багатофункціональні мобільні телефони, які поєднують функції комп'ютера та комунікаційного пристрою.

- Виконання дзвінків, надсилання текстових повідомлень.
- Доступ до інтернету та електронної пошти.
- Використання додатків для різних завдань (соціальні мережі, ігри, навігація).
- Високоякісні камери для фото та відео.

Смартгодинники (Smartwatches) — це наручні комп'ютери у формі годинника, які надають доступ до різних функцій, зокрема сповіщень, відстеження фізичної активності та роботи з додатками.

- Відстеження фізичної активності (кроки, пульс, сон тощо).
- Отримання сповіщень з підключеного смартфона.
- Використання додатків для управління здоров'ям, музикою та іншими завданнями.
- Безконтактні платежі.

Персональні комп'ютери

Персональні комп'ютери (ПК) є універсальними обчислювальними пристроями, призначеними для індивідуального використання. Їх широко застосовують для виконання різноманітних завдань — від роботи з документами до ігор та мультимедіа.



Переваги настільних ПК

- *Висока продуктивність.* Можливість встановлення потужних процесорів, графічних карт та великої кількості оперативної пам'яті.
- *Модернізація.* Легкість заміни та оновлення компонентів, що дозволяє збільшувати продуктивність та розширювати функціональність.
- *Великий вибір периферії.* Можливість підключення різноманітних пристроїв, таких як принтери, сканери, додаткові монітори.
- *Надійність.* Настільні ПК зазвичай мають кращу систему охолодження, що збільшує їхню стабільність і тривалість роботи.

Недоліки настільних ПК

- *Мобільність.* Настільні ПК зазвичай громіздкі та важкі, що робить їх незручними для транспортування.
- *Простір.* Вимагають більше простору на робочому місці для розміщення системного блоку, монітора та периферійних пристроїв.
- *Енергоспоживання.* Вищий рівень енергоспоживання порівняно з портативними пристроями.

1 **Настільні комп'ютери (Desktops)**

Традиційні ПК, що складаються з окремих компонентів: системного блоку, монітора, клавіатури, миші та інших периферійних пристроїв.

- **Можливість модернізації.** Легкість заміни та оновлення компонентів.
- **Висока продуктивність.** Можливість встановлення потужних процесорів, графічних карт та великої кількості оперативної пам'яті.
- **Великий вибір периферії.** Можливість підключення різноманітних пристроїв (принтери, сканери, додаткові монітори).

2 **Робочі станції (Workstations)**

Високопродуктивні комп'ютери, призначені для професійних завдань, таких як графічний дизайн, відеомонтаж, наукові обчислення.

- **Надвисока продуктивність.** Потужні процесори, професійні графічні карти, велика кількість оперативної пам'яті.
- **Надійність.** Підвищена надійність та стабільність роботи.
- **Підтримка спеціалізованого ПЗ.** Підтримка програмного забезпечення для професіоналів.

3 **Ігрові комп'ютери (Gaming PCs)**

Комп'ютери, спеціально розроблені для ігор, з потужними графічними картами та процесорами.

- **Висока продуктивність.** Потужні графічні карти, процесори та велика кількість оперативної пам'яті.
- **Розширені можливості охолодження.** Системи охолодження для запобігання перегріву.
- **Стильний дизайн.** Часто мають яскравий дизайн з підсвічуванням.

4 **Домашні сервери (Home Servers)**

Настільні комп'ютери, які використовуються як сервери для зберігання та обробки даних в домашніх умовах.

- **Зберігання даних.** Великі обсяги жорстких дисків для зберігання файлів.
- **Обробка даних.** Потужність для обробки медіафайлів, резервного копіювання даних.
- **Віддалений доступ.** Можливість доступу до даних з будь-якого пристрою в мережі.

Настільні персональні комп'ютери (ПК) є традиційними обчислювальними пристроями, призначеними для розміщення на робочому столі. Настільні ПК забезпечують високу продуктивність і широкі можливості для модернізації.

Системний блок | Периферійні пристрої | Програмна складова | Домашні сервери (Home Servers)



Системний блок є основною частиною настільного ПК, в якому розташовані всі основні компоненти комп'ютера.

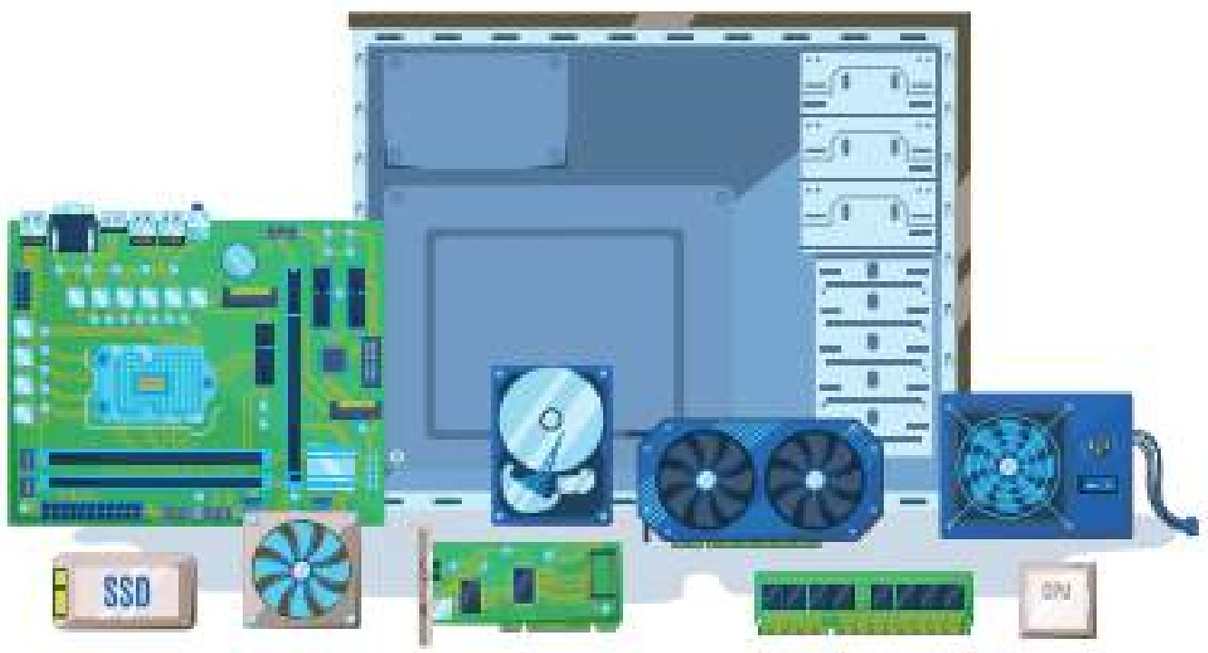
Периферійні пристрої персонального комп'ютера (ПК) є зовнішніми компонентами, які розширюють функціональні можливості комп'ютера та забезпечують взаємодію користувача з системою. Вони поділяються на пристрої введення, виведення та зберігання даних.

Програмна складова персонального комп'ютера (ПК) складається з програмного забезпечення (ПЗ), яке гарантує функціональність апаратних компонентів та взаємодію користувача з системою. Програмне забезпечення включає операційну систему, системне програмне забезпечення, прикладні програми та утиліти.

Основні завдання домашніх серверів:

- Зберігання даних (File Storage)
- Резервне копіювання (Backup)
- Медіасервери (Media Servers)
- Хостинг вебсайтів та додатків (Web and Application Hosting)
- Сервери для ігор (Game Servers)
- Управління розумним будинком (Smart Home Management)

Системний блок та його складові



Системний блок (Case)

Опис. Корпус, який містить усі внутрішні компоненти ПК.
Особливості:

- **Форм-фактор.** Визначає розмір і сумісність з материнськими платами (ATX, MicroATX, Mini-ITX).
- **Вентиляція та охолодження.** Вентилятори, радіатори та рідинні системи охолодження для підтримки оптимальної температури компонентів.
- **Дизайн та ергономіка.** Розташування портів, кнопок та можливості розширення.

Материнська плата (Motherboard)

Опис. Основна плата, яка з'єднує всі компоненти ПК і забезпечує їх взаємодію.

Основні компоненти:

- **Процесорний роз'єм (CPU Socket).** Місце встановлення центрального процесора.
- **Слоти для оперативної пам'яті (RAM Slots).** Слоти для модулів оперативної пам'яті.
- **Роз'єми для відеокарт (PCIe Slots).** Слоти для встановлення графічних карт та інших розширювальних карт.
- **Інтерфейси зберігання даних:** SATA, M.2 для підключення жорстких дисків та твердотільних накопичувачів.
- **Чипсет (Chipset).** Контролер, що забезпечує роботу і зв'язок між процесором, оперативною пам'яттю та периферійними пристроями.
- **Інтерфейси вводу/виводу (I/O Ports)**. USB, HDMI, Ethernet, аудіороз'єми тощо.

Центральний процесор (CPU)

Опис. Основний обчислювальний елемент ПК, який виконує інструкції програмного забезпечення.

Основні характеристики:

- **Тактова частота (Clock Speed).** Вимірюється в герцах (Гц) і визначає швидкість роботи процесора.
- **Кількість ядер (Cores).** Більше ядер дозволяє виконувати більше задач одночасно.
- **Кеш-пам'ять (Cache).** Вбудована високошвидкісна пам'ять для тимчасового зберігання даних.

Оперативна пам'ять (RAM)

Опис. Тимчасова пам'ять, що використовується для зберігання даних і програм під час їх виконання.

Основні характеристики:

- **Ємність (Capacity):** вимірюється в гігабайтах (ГБ) і визначає обсяг даних, які можна зберігати одночасно.
- **Швидкість (Speed):** вимірюється в мегагерцах (МГц) і визначає швидкість передачі даних.
- **Тип пам'яті:** DDR4, DDR5 тощо.

Накопичувачі (Storage)

Опис. Пристрої для постійного зберігання даних, таких як операційна система, програми та файли.

Типи накопичувачів:

- **Жорсткі диски (HDD):** механічні диски з великою ємністю і відносно низькою швидкістю.
- **Твердотільні накопичувачі (SSD):** швидкі накопичувачі без рухомих частин, з високою швидкістю читання/запису.

Графічна карта (GPU)

Опис. Пристрій для обробки графіки і виведення зображення на монітор.

Типи графічних карт:

- **Інтегровані графічні карти (Integrated Graphics):** вбудовані в процесор і підходять для базових завдань.
- **Дискретні графічні карти (Discrete Graphics):** окремі картки, що забезпечують високу продуктивність для ігор та професійних завдань.

Блок живлення (PSU)

Опис. Пристрій, що забезпечує електроживленням всі компоненти комп'ютера.

Основні характеристики:

- **Потужність (Wattage):** вимірюється у ватах (Вт) і визначає максимальну потужність, яку може забезпечити блок живлення.
- **Ефективність (Efficiency):** ККД блока живлення, що впливає на споживання електроенергії та тепловиділення.

Системи охолодження (Cooling System)

Опис. Пристрої для підтримання оптимальної температури компонентів ПК.

Типи систем охолодження:

- **Повітряне охолодження:** вентилятори та радіатори для відведення тепла від компонентів.
- **Рідинне охолодження:** водяні системи охолодження для ефективнішого відведення тепла.

Периферійні пристрої персонального ПК

- Пристрої введення
- Пристрої виведення
- Пристрої зберігання даних

Пристрої введення

Клавіатура

Опис. Пристрій для введення текстової та командної інформації.
Типи. Механічні, мембранні, бездротові, ергономічні.

Комп'ютерна миша

Опис. Пристрій для керування курсором на екрані та введення команд.
Типи. Оптичні, лазерні, трекболи, бездротові.





Графічний планшет

Опис. Пристрій для малювання та графічного дизайну за допомогою стилуса.



Вебкамера

Опис. Пристрій для зйомки відео і фотографій, а також для відеоконференцій.
Типи. Вбудовані, зовнішні.



Сканер

Опис. Пристрій для оцифрування фізичних документів і зображень.
Типи. Пласкі, портативні, барабанні, сканери для документів.



Мікрофон

Опис. Пристрій для запису звуку та участі в голосових викликах.
Типи. Настільні, петличні, студійні.



Ігрові контролери

Опис. Пристрої для управління іграми.
Типи. Джойстики, геймпади, кермові колеса.

Пристрої виведення

Монітор

Опис. Пристрій для виведення візуальної інформації.



Принтер

Опис. Пристрій для друку тексту та зображень на папері.
Типи. Лазерні, струменеві, матричні, 3D-принтери.



Навушники

Опис. Пристрої для індивідуального прослуховування звуку.



Проєктор

Опис. Пристрій для проєкції зображень на екран або стіну.
Типи. LCD, DLP, LED.



Периферійні пристрої персонального ПК

Зовнішні жорсткі диски (HDD)

Опис. Пристрої для зберігання великої кількості даних.
Типи. Портативні, настільні.



Твердотільні накопичувачі (SSD)

Опис. Пристрої для швидкого зберігання та передачі даних.
Типи. Портативні, внутрішні.



USB-флеш-накопичувачі

Опис. Компактні пристрої для зберігання та передачі даних.
Типи. З різними обсягами пам'яті, з шифруванням даних.



Оптичні накопичувачі

Опис. Пристрої для читання та запису CD, DVD, Blu-ray дисків.
Типи. Внутрішні, зовнішні.



Інші периферійні пристрої

Мережеві адаптери та роутери

Опис. Пристрої для підключення до інтернету або локальної мережі.
Типи. Wi-Fi адаптери, Ethernet адаптери.



Джерела безперебійного живлення

Опис. Пристрої для забезпечення резервного живлення ПК.

Картрідери

Опис. Пристрої для читання карт пам'яті.
Типи. Вбудовані, зовнішні.





Процесор та пам'ять комп'ютера

Процесор та оперативна пам'ять тісно взаємодіють для забезпечення високої продуктивності комп'ютера. Процесор виконує команди програм, зберігаючи й обробляючи дані в оперативній пам'яті.

Основні види пам'яті комп'ютера

- Оперативна пам'ять (RAM)
- Постійна пам'ять (ROM)
- Зовнішня пам'ять
- Кеш-пам'ять
- Жорсткі диски (HDD)
- Твердотільні накопичувачі (SSD)
- Флеш-накопичувачі (USB)
- Оптичні носії (CD, DVD, Blu-ray)

Оперативна пам'ять (RAM — Random Access Memory) є тимчасовою пам'яттю комп'ютера, яка використовується для зберігання даних і програм під час їх виконання. Вона забезпечує швидкий доступ до даних, необхідних процесору для виконання завдань.

Основні характеристики оперативної пам'яті:

- *Ємність (Capacity):* вимірюється в гігабайтах (ГБ) і визначає обсяг даних, які можуть бути одночасно збережені. Сучасні комп'ютери зазвичай оснащені оперативною пам'яттю від 4 до 32 ГБ і більше.
- *Швидкість (Speed):* вимірюється в мегагерцах (МГц) і визначає швидкість передачі даних. Вищі частоти означають швидший доступ до даних.
- *Тип пам'яті:* DDR (Double Data Rate): DDR4, DDR5 є найпоширенішими типами оперативної пам'яті в сучасних комп'ютерах.

Постійна пам'ять (ROM — Read-Only Memory) є типом пам'яті, яка зберігає дані постійно, навіть після вимкнення комп'ютера. Вона використовується для зберігання програмного забезпечення, яке потрібне для початкового завантаження системи.

Постійна пам'ять зберігає свої дані навіть після вимкнення живлення комп'ютера.

Зовнішня пам'ять забезпечує довготривале зберігання даних, які можуть бути записані, прочитані або видалені. Вона включає жорсткі диски (HDD), твердотільні накопичувачі (SSD), флеш-накопичувачі (USB) та оптичні носії (CD, DVD, Blu-ray).

Кеш-пам'ять є високошвидкісною пам'яттю, яку застосовують для тимчасового зберігання часто використовуваних даних і команд, щоб зменшити час доступу до оперативної пам'яті.

Жорсткі диски (HDD) — механічні накопичувачі, які використовують магнітні пластини для зберігання даних.

Ємність: вимірюється в гігабайтах (ГБ) або терабайтах (ТБ).

Швидкість обертання шпинделя: вимірюється в обертах за хвилину (RPM), що впливає на швидкість доступу до даних (5400 RPM, 7200 RPM, 10000 RPM).

Твердотільні накопичувачі (SSD) — накопичувачі без рухомих частин, які використовують флеш-пам'ять для зберігання даних.

Ємність: вимірюється в гігабайтах (ГБ) або терабайтах (ТБ).

Швидкість читання/запису: вища, ніж у HDD, що забезпечує швидший доступ до даних.

Флеш-накопичувачі (USB) — портативні пристрої для зберігання і перенесення даних.

Ємність: вимірюється в гігабайтах (ГБ).

Інтерфейси: USB 2.0, USB 3.0, USB-C.

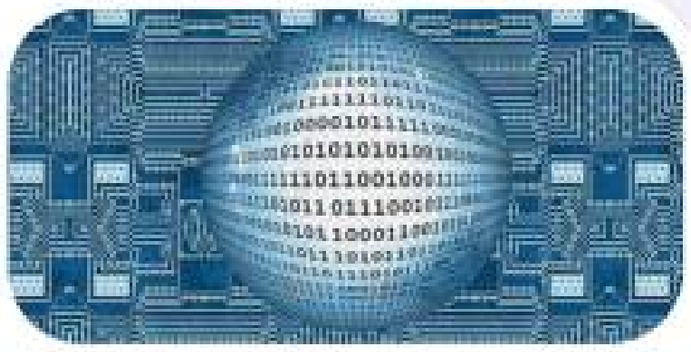
Оптичні носії (CD, DVD, Blu-ray) — накопичувачі, які використовують лазерне світло для зчитування і запису даних.

Ємність: CD – до 700 МБ, DVD – до 4.7 ГБ, Blu-ray – до 25 ГБ.

Типи дисків: одноразового запису (R), багаторазового запису (RW).

Кодування даних

Кодування даних на цифрових носіях — це процес перетворення інформації у формат, придатний для зберігання і передачі на різних цифрових пристроях.



Цей процес дозволяє зменшити обсяг файлів, не суттєво втрачаючи якість.

Типи кодування

Бінарне кодування (Binary Encoding)

Усі дані в комп'ютерах зберігаються й обробляються у вигляді бінарного коду, який складається з нулів (0) та одиниць (1).

Кожен символ, число або інструкція перетворюється на послідовність бітів.

ASCII і Unicode кодування

ASCII (American Standard Code for Information Interchange). Використовується для кодування англійських букв, цифр та деяких символів. Кожен символ кодується 7-бітним числом.

Методи стиснення даних

Без втрат (Lossless Compression)	**Із втратами** (Lossy Compression)
Стиснення, яке дозволяє повністю відновити оригінальні дані після розпакування.	Стиснення, яке видаляє деяку частину даних для досягнення більш високого ступеня стиснення, що призводить до втрати якості.
Приклади: ZIP, GZIP, PNG (для зображень), FLAC (для аудіо).	*Приклади:* JPEG (для зображень), MP3 (для аудіо), MPEG (для відео).

Одиниці вимірювання інформації

Одиниці вимірювання інформації використовуються для визначення обсягу даних, які можуть бути збережені або передані.

Одиниці вимірювання інформації

Біт (bit)	Біт є найменшою одиницею інформації і може мати одне з двох значень: 0 або 1. Один біт може використовуватися для зберігання одного значення істина/неістина, увімкнено/вимкнено.
Байт (byte)	Байт складається з 8 бітів і є основною одиницею вимірювання інформації в комп'ютерах. Один байт може зберігати один символ тексту, такий як літера «А» або цифра «7».
Кілобайт (KB)	Один кілобайт дорівнює 1024 байтам. Короткий текстовий файл може займати кілька кілобайтів.
Мегабайт (MB)	Один мегабайт дорівнює 1024 кілобайтам. Невелика цифрова фотографія або музичний файл у форматі MP3 може займати кілька мегабайтів.
Гігабайт (GB)	Один гігабайт дорівнює 1024 мегабайтам. Стандартний фільм у форматі високої якості може займати кілька гігабайтів.
Терабайт (TB)	Один терабайт дорівнює 1024 гігабайтам. Весь жорсткий диск комп'ютера може мати ємність в декілька терабайтів.
Петабайт (PB)	Один петабайт дорівнює 1024 терабайтам. Великі дата-центри або суперкомп'ютери можуть використовувати петабайти для зберігання величезних обсягів даних.
Ексабайт (EB)	Один ексабайт дорівнює 1024 петабайтам. Глобальний інтернет-трафік за кілька місяців може вимірюватися в ексабайтах.

Практична робота №1.

Індивідуальна робота

1. Оберіть три моделі комп'ютерів з різних категорій та порівняйте їх за основними параметрами (процесор, пам'ять, накопичувач, графічна карта, ціна).

   Програмне середовище: **Google Таблиці**

   Критерії аналізу:

   - Процесор (CPU)
   - Оперативна пам'ять (RAM)
   - Накопичувач (Storage)
   - Графічна карта (GPU)
   - Операційна система (OS)
   - Дисплей
   - Автономність (для ноутбуків і ультрабуків)
   - Можливості підключення
   - Мобільність
   - Додаткові функції
   - Ціна

2. В **Google Таблиці** додайте покликання та зображення.
3. Додайте короткий опис параметрів та їхнє значення.
4. Надішліть педагогу з можливістю коментування.

Колективна робота. Підбір конфігурації ПК

Мета: навчити студентів працювати в команді для підбору оптимальної конфігурації персонального комп'ютера, враховуючи основні компоненти архітектури комп'ютера та їхні характеристики.

Вимоги:

1. Кількість учасників: 3–4 учні.
2. Інструменти: **Google Таблиці** для спільної роботи.
3. Час виконання: 1 урок або домашнє завдання.
4. Результат: складання таблиці з оптимальною конфігурацією ПК для конкретних завдань (наприклад, для навчання, ігор або професійної роботи).

Етапи виконання:

Розподіл ролей (самостійно).

1. Створення **Google Таблиці**.
   - Один з учасників створює нову **Google Таблицю** та надає доступ іншим учасникам.
   - Назва таблиці: «Конфігурація ПК для [Завдання]».

2. Створення структури таблиці.
- Додати колонки: Компонент, Модель, Характеристики, Ціна, Примітки.
- Додати рядки: Процесор, Оперативна пам'ять, Графічна карта, Накопичувач, Материнська плата, Корпус, Блок живлення, Системи охолодження, Додаткові компоненти.

3. Спільне обговорення обраних компонентів.

4. Підрахунок загальної вартості ПК.

5. Перевірка на відповідність вимогам і бюджету.

6. Представлення власної конфігурації ПК.

7. Надання доступу педагогу з можливістю коментування.







Тема 2 Пристрої введення та виведення даних. Драйвери

Ключові поняття

Периферійні пристрої персонального ПК
Інтерфейс підключення
Ігрові контролери
Тип підключення

Периферійні пристрої персонального ПК

Клавіатура

Мембранна клавіатура

Мембранні клавіатури використовують гумові або силіконові мембрани під клавішами, які замикають електричні контакти при натисканні.

- Тихі та м'які натискання.
- Відносно недорогі.
- Менший термін служби в порівнянні з механічними клавіатурами.
- Підходять для офісного та повсякденного використання.

Механічна клавіатура

Механічні клавіатури використовують індивідуальні механічні перемикачі під кожною клавішею.

- Чітке тактильне і звукове зворотне відчуття.
- Довговічні (до 50–100 мільйонів натискань).
- Різноманітні типи перемикачів (лінійні, тактильні, клікні).
- Ідеальні для геймерів та професіоналів, які багато друкують.

Проєкційні клавіатури

Являє собою не набір клавіш, а невеликий проєктор, який формує світлове зображення у вигляді кнопок на гладкій поверхні. Коли ви торкаєтеся проєкції клавіші, датчики реагують на відбитий від пальця лазер і зіставляють його з відповідною кнопкою, якби ви натиснули її на звичайній клавіатурі. Проєктори дають зрозуміти, що майбутнє вже настало, дозволяють спроєктувати будь-яку розкладку і займають мало місця. Але користуватися такою клавіатурою не дуже зручно — багато випадкових спрацювань, складно використовувати десятипальцевий метод набору, а місце установки пристрою має бути ідеально рівним.

Бездротова клавіатура

Бездротові клавіатури використовують радіочастотні (RF) або Bluetooth технології для підключення до комп'ютера без використання кабелів.

- Легкість у переміщенні.
- Працюють від батарейок або акумуляторів.
- Можлива незначна затримка при введенні даних.
- Ідеальні для домашнього та офісного використання.

Ігрова клавіатура

Ігрові клавіатури спеціально розроблені для геймерів.

- Зазвичай механічні перемикачі для швидкого і чіткого натискання.
- RGB підсвітка з налаштуваннями кольорів та ефектів.
- Додаткові макроклавіші для програмованих команд.
- Висока продуктивність та довговічність.

Сенсорна клавіатура

Сенсорні клавіатури використовують сенсорні панелі замість фізичних клавіш.

- Гладка поверхня.
- Налаштовувані клавіші та функції.
- Використовуються на планшетах, смартфонах та деяких ноутбуках.

Клавіатури розрізняються за різними параметрами, такими як тип клавіш, метод підключення, ергономічність та спеціалізовані функції.



Інтерфейс підключення

- *Дротове підключення:* надійне з'єднання без затримок.
- *Бездротове підключення:* використання радіочастот (RF) або Bluetooth для підключення без кабелів.
- *Комбіноване підключення:* можливість як дротового, так і бездротового підключення.

Додаткові клавіші

- *Мультимедійні клавіші:* для управління медіа (гучність, відтворення/пауза).
- *Програмовані макроклавіші:* для налаштування користувацьких команд і функцій.
- *Клавіші для швидкого доступу:* до веббраузера, пошти, калькулятора тощо.

Розкладка клавіатури

- *Стандартна розкладка:* класичне розташування клавіш.
- *Компактна розкладка:* зменшена кількість клавіш, відсутність цифрового блоку.
- *Ергономічна розкладка:* викривлене або розділене розташування клавіш для зменшення навантаження на руки.

Ергономічний дизайн

- *Підставка для зап'ясть:* для підтримки рук і зменшення втоми.
- *Розділена конструкція:* для природнішого розташування рук.

Живлення

- *Батарейки:* використання звичайних або акумуляторних батарейок.
- *Вбудований акумулятор:* перезаряджається через USB-кабель.

Комп'ютерна миша

Оптична миша

Оптичні миші використовують світлодіод і фотодіод для відстеження руху.

- *Точність:* висока точність на більшості поверхонь.
- *Підключення:* дротове або бездротове.
- *Застосування:* підходить для офісного та повсякденного використання.

Лазерна миша

Лазерні миші використовують лазерний діод для відстеження руху, що забезпечує вищу точність і чутливість.

- *Точність:* дуже висока точність, працює на більшості поверхонь, включаючи скло.
- *Підключення:* дротове або бездротове.
- *Застосування:* ідеальна для професійної роботи та геймерів.

Механічна (кулькова) миша

Механічні миші використовують кульку для відстеження руху, яка передає дані на механічні датчики всередині миші.

- *Точність:* менша точність, ніж в оптичних та лазерних мишей.
- *Чищення:* вимагають регулярного чищення для збереження працездатності.
- *Застосування:* рідко використовуються в сучасних комп'ютерних системах.



Ергономічна миша

Ергономічні миші розроблені для зменшення навантаження на руку і зап'ясток під час використання.

- *Дизайн:* викривлені форми або вертикальні миші для природнішого положення руки.
- *Застосування:* ідеальні для користувачів, які багато працюють за комп'ютером.

Трекбол

Трекболи використовують кульку, яка розташована у верхній частині пристрою та керується пальцями або долонею.

- *Точність:* висока точність і зручність у використанні.
- *Комфорт:* не потребують переміщення миші, що знижує навантаження на руку.
- *Застосування:* підходять для графічного дизайну і роботи з кресленнями.

Сенсорна клавіатура

Сенсорні миші використовують сенсорні панелі для управління курсором замість традиційних кнопок і колеса прокрутки.

- *Інтерфейс:* гладка сенсорна поверхня.
- *Функціональність:* можливість жестового управління.
- *Застосування:* використовуються з сучасними ноутбуками і деякими настільними комп'ютерами.

При виборі комп'ютерної миші важливо звернути увагу на кілька ключових характеристик, які визначають її зручність, продуктивність та функціональність.

Тип сенсора

- *Оптичний сенсор:* використовує світлодіод для відстеження руху. Працює на більшості поверхонь.
- *Лазерний сенсор:* використовує лазерний діод для відстеження руху. Більш точний і чутливий, працює на більшій кількості поверхонь, включаючи скло.
- *Механічний сенсор (кулька):* використовує механічну кульку для відстеження руху. Вимираючий тип, потребує частого очищення.

DPI (dots per inch)

- *Високе DPI:* забезпечує високу чутливість миші, що дозволяє переміщувати курсор на великі відстані з невеликими рухами миші.
- *Регульоване DPI:* можливість змінювати DPI для різних завдань (ігри, робота з графікою тощо).

Тип підключення

- *Дротове підключення (USB):* стабільне з'єднання, відсутність затримок.
- *Бездротове підключення (Bluetooth):* зручність без кабелів, можливі незначні затримки.
- *Комбіноване підключення:* можливість використання як дротового, так і бездротового підключення.

Форма та ергономіка

- *Симетрична форма:* підходить для обох рук, зручна для більшості користувачів.
- *Ергономічна форма:* спеціально розроблена для зменшення навантаження на руку і зап'ясток.
- *Вертикальна миша:* забезпечує природніше розташування руки, зменшує навантаження на зап'ясток.

Живлення

- *Батарейки:* звичайні або акумуляторні батарейки.
- *Вбудований акумулятор:* перезаряджається через USB-кабель.
- *Час роботи:* тривалість роботи від одного заряду.

Вебкамера

Вебкамера — це цифровий пристрій, який використовується для захоплення відео та зображень і передавання їх у реальному часі через інтернет або інші комп'ютерні мережі.



Вбудовані вебкамери



Інтегровані в корпус ноутбуків або моніторів.

- *Якість відео:* зазвичай нижча якість у порівнянні із зовнішніми вебкамерами.
- *Зручність:* вбудовані та завжди доступні.
- *Обмеженість налаштувань:* менше можливостей для налаштування та поліпшення якості зображення.

Зовнішні вебкамери

Підключаються до комп'ютера або ноутбука через USB.

- *Якість відео:* широкий діапазон якості відео від стандартної до високої.
- *Мобільність:* легко переміщувати і налаштовувати.
- *Додаткові функції:* часто мають кращі налаштування, мікрофони та інші функції.

Вебкамери з функцією повороту і нахилу

Вебкамери, які можна повертати і нахиляти для кращого кадрування.

- *Керування:* ручне або дистанційне керування.
- *Гнучкість:* можливість налаштування кута огляду під час використання.

Вебкамери для стрімінгу

Вебкамери, спеціально розроблені для стрімінгу на платформах як **Twitch**, **YouTube** тощо.

- *Якість відео:* висока якість відео з високою частотою кадрів.
- *Додаткові функції:* можуть включати функції для поліпшення якості стріму, такі як вбудовані фільтри та налаштування освітлення.

Роздільна здатність

Визначає кількість пікселів, що формують зображення.

- *HD (720p):* підходить для більшості відеодзвінків.
- *Full HD (1080p):* висока якість для відеоконференцій і стрімінгу.
- *4K (2160p):* дуже висока якість для професійного використання.

Частота кадрів

Визначає, скільки кадрів за секунду може записувати вебкамера.

- *30 кадрів за секунду (fps):* Стандартна частота для більшості вебкамер.
- *60 кадрів в секунду (fps):* Плавніший відеопотік, підходить для стрімінгу і запису відео.

Автофокус

Автоматично налаштовує фокус, щоб забезпечити чітке зображення.

- *Автофокус:* постійне налаштування фокусу під час руху.
- *Фіксований фокус:* постійне налаштування, яке не змінюється.

Підключення

Визначає, як вебкамера підключається до комп'ютера.

- *USB 2.0/3.0:* стандартне підключення.
- *Wi-Fi:* бездротове підключення, зручне для мобільності.

Живлення

- *Батарейки:* звичайні або акумуляторні.
- *Вбудований акумулятор:* перезаряджається через USB-кабель.
- *Час роботи:* тривалість роботи від одного заряду.

Сканер

Сканер — це електронний пристрій, призначений для перетворення фізичних документів, зображень або об'єктів у цифровий формат. Він зчитує текст, графіку або тривимірні об'єкти і створює їх цифрову копію, яку можна зберігати, редагувати або передавати електронним шляхом.

Планшетні сканери

Найпоширеніший тип сканера, який має плоску скляну поверхню для розміщення документів або зображень.

- *Роздільна здатність:* від 600 до 4800 dpi та більше.
- *Швидкість сканування:* залежить від моделі, може варіюватися від декількох секунд до хвилини на сторінку.
- *Формат:* зазвичай до А4, але є моделі для форматів А3 і більше.
- *Застосування:* підходить для домашнього та офісного використання, сканування документів, фотографій.

Протяжні сканери

Протяжні сканери використовують автоматичну подачу документів через пристрій, що дозволяє швидко сканувати великі обсяги паперу.

- *Швидкість сканування:* висока, може досягати десятків сторінок за хвилину.
- *Автоподача документів (ADF):* дозволяє автоматично сканувати кілька сторінок.
- *Формат:* зазвичай до А4.
- *Застосування:* ідеальні для офісів з великим обсягом документів, банків, юридичних фірм.

Портативні сканери

Компактні та легкі сканери, які можна переносити і використовувати в будь-якому місці.

- *Розміри:* легкі та компактні, часто живляться від USB.
- *Функціональність:* можуть сканувати документи, візитки, чеки.
- *Застосування:* підходять для мандрівників, торгових представників, мобільних офісів.

3D-сканери

3D-сканери використовуються для створення тривимірних моделей об'єктів.

- *Точність:* висока точність сканування, визначена в міліметрах або мікронах.
- *Функціональність:* можливість сканування об'єктів різного розміру і складності.
- *Застосування:* використовуються в промисловому дизайні, архітектурі, медицині, ігровій індустрії.

Барабанні сканери

Барабанні сканери використовуються для високоякісного сканування зображень, фотографій та негативів.

- *Роздільна здатність:* висока, до 12 000 dpi та більше.
- *Якість сканування:* професійна, з високою точністю передачі кольору і деталей.
- *Застосування:* використовуються в професійній поліграфії, фотоіндустрії, архівах.



Формат сканування

Визначає максимальний розмір документів, які може сканувати пристрій.

- *A4:* стандартний розмір для більшості сканерів.
- *A3 та більше:* для великих документів і креслень.

Інтерфейс підключення

Визначає спосіб підключення сканера до комп'ютера.

- *USB:* найпоширеніший інтерфейс.
- *Wi-Fi:* безпровідне підключення для зручності використання.

Роздільна здатність

Визначає кількість крапок на дюйм (dpi), що може бути відскановано.

- *600 dpi:* підходить для текстових документів.
- *2400 dpi і більше:* підходить для фотографій і зображень високої якості.

Глибина кольору

Визначає кількість кольорів, які може розрізняти сканер.

- *24 біти:* стандарт для більшості сканерів.
- *48 біти:* висока точність кольоропередачі, підходить для професійного сканування.

Ігрові контролери

Ігрові контролери – це пристрої введення, призначені для керування діями персонажів та об'єктів у відеоіграх. Вони забезпечують інтерактивну взаємодію між гравцем і грою, надаючи можливість використовувати різні типи управління, такі як натискання кнопок, рухи аналогових стіків, натискання тригерів та сенсорні жести.









Джойстики

Джойстики є вертикальними ігровими контролерами з кнопками, які можна нахиляти в різні боки для керування іграми.

- *Точність:* висока точність керування.
- *Функціональність:* переважно використовуються в симуляторах авіапольотів.
- *Кнопки:* додаткові кнопки для виконання різних команд.

Геймпади

Геймпади є найпоширенішим типом ігрових контролерів, які зазвичай мають два аналогові стіки, кілька кнопок, хрестовину (D-pad) та тригери.

- *Зручність:* ергономічний дизайн для комфортного тримання.
- *Сумісність:* використовуються з консолями, комп'ютерами та мобільними пристроями.
- *Функціональність:* підтримка вібраційного зворотного зв'язку.

Кермо і педалі

Ці контролери імітують реальні автомобільні керма та педалі для використання в іграх з перегонами.

- *Реалістичність:* висока точність і реалістичне відчуття.
- *Функціональність:* містять педалі акселератора, гальм та зчеплення

Аркадні джойстики

Ці контролери імітують класичні аркадні автомати з великими кнопками та міцним джойстиком.

- *Довговічність:* міцна конструкція для інтенсивного використання.
- *Функціональність:* підходять для файтингів та інших аркадних ігор.

Ергономіка та дизайн

Визначає комфорт і зручність користування контролером протягом тривалого часу.

- *Форма:* ефективність дизайну для різних типів рук.
- *Матеріали:* Якість матеріалів, які використовуються для створення контролера.
- *Розташування кнопок і стіків:* зручність доступу до всіх елементів управління.

Тип підключення

Спосіб підключення контролера до пристрою.

- *Дротове підключення:* USB, для стабільного з'єднання без затримок.
- *Бездротове підключення:* Bluetooth, RF (радіочастотне), для зручності без кабелів.
- *Комбіноване підключення:* можливість використовувати як дротове, так і бездротове з'єднання.

Функціональність

Додаткові можливості, які надає контролер.

- *Підсвітка:* наявність і налаштування RGB-підсвітки.
- *Інтегровані динаміки:* додатковий звуковий зворотний зв'язок.
- *Сенсорні панелі:* наявність сенсорної панелі для додаткового управління.

Живлення

Спосіб живлення контролера.

- *Батарейки:* звичайні або акумуляторні.
- *Вбудований акумулятор:* можливість перезаряджання через USB.
- *Тривалість роботи:* час роботи від одного заряду.

Принтер

Принтер — це периферійний пристрій, який зчитує електронні дані з комп'ютера чи іншого цифрового пристрою і відтворює їх на папері або іншому матеріалі у вигляді тексту чи графіки.

Лазерні принтери

Використовують лазерний промінь для створення зображення на фотобарабані, який потім переносить тонер на папір.

- *Швидкість друку:* висока.
- *Якість друку:* висока, чудові результати на звичайному папері.
- *Вартість:* вищі початкові витрати, але низька вартість друку на одну сторінку.
- *Призначення:* офіси, де потрібний швидкий і якісний друк великих обсягів документів.

Струменеві принтери

Використовують дрібні краплі чорнила, які розпилюються на папір через мікроскопічні сопла.

- *Швидкість друку*: середня.
- *Якість друку:* висока якість зображень і фотографій.
- *Вартість:* нижчі початкові витрати, але вища вартість друку на одну сторінку.
- *Призначення:* домашнє використання, де потрібний кольоровий друк, фотографії.

Світлодіодні принтери

Працюють за схожим принципом з лазерними принтерами, але використовують світлодіодні масиви замість лазерного променя.

- *Швидкість друку:* висока.
- *Якість друку:* чудова якість тексту і графіки.
- *Вартість:* схожі з лазерними принтерами, низька вартість друку на одну сторінку.
- *Призначення:* офіси, де потрібний швидкий і якісний друк.





Матричні (голчасті) принтери

Використовують ряд голок, які вдаряють по стрічці, що переносить чорнило на папір.

- *Швидкість друку:* низька.
- *Якість друку:* низька, підходить для друку тексту.
- *Вартість:* низька вартість обслуговування і витратних матеріалів.
- *Призначення:* використовуються для друку рахунків, чеків, документів на декілька копій.

3D-принтери

Використовують технологію адитивного виробництва для створення тривимірних об'єктів.

- *Матеріали:* використовують пластик, метал, смоли та інші матеріали.
- *Призначення:* прототипування, виробництво деталей, створення моделей.

Роздільна здатність

Визначає кількість крапок на дюйм (dpi), яку може відтворити принтер.

- *600 x 600 dpi:* прийнятна якість для тексту;
- *1200 x 1200 dpi:* вища якість для графіки і тексту;
- *2400 x 1200 dpi та більше:* висока якість для фотографій і зображень.

Швидкість друку

Вимірюється у кількості сторінок за хвилину (ppm).

- *20-30 ppm:* для домашнього використання;
- *30-50 ppm:* для невеликих офісів;
- *50+ ppm:* для великих офісів і комерційного використання.

Формат паперу

Максимальний розмір паперу, який може обробляти принтер.

- *A4:* стандартний розмір для більшості принтерів;
- *A3:* для друку великих документів, плакатів;
- *інші формати:* B5, C5, DL тощо для спеціалізованих завдань.

Тип підключення

Визначає способи підключення принтера до комп'ютера чи мережі.

- *USB:* стандартне підключення для одного комп'ютера;
- *Ethernet:* мережеве підключення для спільного використання;
- *Wi-Fi:* бездротове підключення для зручності;
- *Bluetooth:* підключення до мобільних пристроїв.

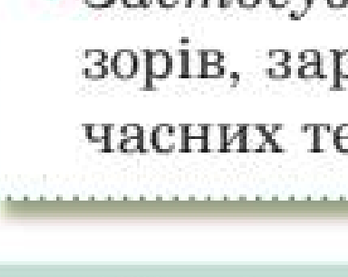
Підтримка кольору

Визначає, чи може принтер друкувати кольорові зображення.

- *чорно-білий:* для друку тексту і простих графіків;
- *кольоровий:* для друку фотографій, кольорових документів.

Монітор

Монітор — це пристрій відображення, який підключається до комп'ютера або іншого електронного пристрою для виведення візуальної інформації.



Монітори на основі електронно-променевої трубки (CRT)

Монітори, що використовують електронно-променеву трубку для відтворення зображення. Цей тип моніторів був основним до появи рідкокристалічних дисплеїв.

- *Якість зображення:* висока, з природними кольорами.
- *Розмір і вага:* великі та важкі.
- *Енергоспоживання:* високе.
- *Застосування:* використовувалися раніше для комп'ютерів і телевізорів, зараз практично не використовуються через появу більш сучасних технологій.

Рідкокристалічні монітори (LCD)

Використовують рідкі кристали для відтворення зображення. Це найбільш поширений тип моніторів на сьогоднішній день.

- *Якість зображення:* висока, з чіткими і яскравими кольорами.
- *Розмір і вага:* тонкі та легкі.
- *Енергоспоживання:* низьке.
- *Застосування:* використовуються для комп'ютерів, телевізорів, мобільних пристроїв.

Плазмові монітори

Використовують плазмові дисплеї, де зображення створюється шляхом освітлення фосфору ультрафіолетовими променями, які виробляються плазмою.

- *Якість зображення:* висока, з насиченими кольорами і глибокими чорними.
- *Розмір і вага:* великі, важчі за LCD.
- *Енергоспоживання:* високе.
- *Застосування:* використовуються переважно для великих телевізорів і дисплеїв.

Проєкційні монітори

Використовують проєкцію зображення на екран за допомогою проєктора.

- *Розмір зображення:* дуже великий, залежить від проєктора та екрана.
- *Якість зображення:* висока, але залежить від умов освітлення.
- *Енергоспоживання:* високе.
- *Застосування:* використовуються для презентацій, домашніх кінотеатрів, конференцій.

OLED-монітори

Використовують технологію адитивного виробництва для створення тривимірних об'єктів.

- *Матеріали:* використовують пластик, метал, смоли та інші матеріали.
- *Призначення:* прототипування, виробництво деталей, створення моделей.

Розмір екрана

Вимірюється по діагоналі екрана в дюймах.

- *19-24 дюйми:* зазвичай для офісного та домашнього використання.
- *27-32 дюйми:* підходить для роботи з графікою, ігор, мультимедіа.
- *34 дюйми і більше:* використовується для професійної роботи, ігор, мультимедійних систем.

Роздільна здатність

Визначає кількість пікселів на екрані, що впливає на чіткість зображення.

- *Full HD (1920x1080):* підходить для більшості завдань;
- *Quad HD (2560x1440):* краща деталізація для роботи з графікою та ігор;
- *4K Ultra HD (3840x2160):* висока чіткість для професійної роботи та розваг.

Тип матриці

Визначає технологію виготовлення екрана та впливає на якість зображення.

- *TN (Twisted Nematic):* швидкий час відгуку, нижча якість кольорів;
- *IPS (In-Plane Switching):* краща кольоропередача і кути огляду;
- *VA (Vertical Alignment):* гарна контрастність і кольоропередача;
- *OLED (Organic Light Emitting Diode):* висока якість зображення, глибокі чорні кольори.

Частота оновлення

Вимірюється в герцах (Гц) і визначає, скільки разів за секунду екран оновлює зображення.

- *60 Гц:* стандартна частота для більшості завдань;
- *120 Гц і більше:* підходить для ігор і швидкісного відео.

Тип підключення

Визначає способи підключення монітора до комп'ютера.

- *HDMI:* підтримка високої роздільної здатності та звуку;
- *DisplayPort:* висока пропускна здатність для відео та аудіо;
- *USB-C:* підключення з підтримкою передачі даних, відео і живлення.

Практична робота №2.

Індивідуальна робота

Частина 1

Порівняльна таблиця пристроїв введення та виведення даних

1. Заповніть таблицю інформацією про кожен пристрій:
   - Клавіатура
   - Миша
   - Монітор
   - Принтер
   - І ще 3 пристрої на ваш вибір.
2. Додайте фото та покликання на покупку цього пристрою.
   - Створіть ще один стовпчик у вашій таблиці з назвою «Загальна вартість».
   - Підрахуйте загальну вартість усіх доданих пристроїв.
3. Надайте доступ до таблиці з можливістю коментування вашому педагогу.

Частина 2

1. Створіть чек-лист у **Canva**.
   Алгоритм пошуку драйверів онлайн з офіційних сайтів.
2. Готовий чек-лист надішліть педагогу на електронну адресу.

Колективна робота

1. Підготувати колективну презентацію в **Google Презентаціях**.
   Склад команди: 3–4 учні.
   Тему обираємо самостійно.

   Практичні питання:

   1. Які нові моделі клавіатур і мишей ви можете знайти на сайтах великих інтернет-магазинів (Amazon, Rozetka, Eldorado тощо)? Які їхні головні характеристики?
   2. Які сучасні принтери рекомендують для домашнього використання? Як вони відрізняються від попередніх моделей?
   3. Які нові моделі моніторів з'явилися у продажу за останній рік? Які їхні головні переваги?
   4. Які нові моделі проєкторів рекомендуються для використання в освітніх установах?
   5. Які новітні технології впроваджено в сучасних аудіосистемах для комп'ютера?

Сучасні тенденції:

- Які нові технології у сфері пристроїв введення та виведення, на думку експертів, є найперспективнішими?
- Як інтеграція штучного інтелекту впливає на розвиток пристроїв введення та виведення даних?
- Які нові технології для пристроїв введення та виведення прогнозують на найближчі роки?

2. Надати доступ до презентації з можливістю коментування вашому педагогу.







Апгрейд та проблеми утилізації електронного сміття. Поняття про файлову систему та її види. Підготовка дисків для роботи та їхнє обслуговування. Поняття про логічні та фізичні ушкодження дисків

Ключові поняття

Апгрейд (модернізація) комп'ютерних пристроїв
Файлова система
Кешування (Caching)
Шифрування (Encryption)
Дефрагментація
Резервне копіювання даних

Апгрейд (модернізація) комп'ютерних пристроїв

Апгрейд — модернізація комп'ютерних пристроїв та інших електронних приладів, що дозволяє продовжити термін служби техніки, підвищити її продуктивність і відповідати сучасним вимогам.

Причини апгрейду

1. **Підвищення продуктивності**
Заміна старих компонентів на нові, більш потужні, для прискорення роботи системи.



2. **Розширення функціональних можливостей**
Додавання нових компонентів або периферійних пристроїв для виконання нових завдань.

3. **Забезпечення сумісності з новим програмним забезпеченням**
Оновлення компонентів для сумісності з сучасними операційними системами та додатками.

4. **Енергоефективність**
Використання нових компонентів, які споживають менше енергії.

Проблеми утилізації електронного сміття



Екологічні проблеми

Токсичні матеріали

Електронні пристрої містять шкідливі хімічні речовини, такі як свинець, ртуть, кадмій і бромовані антипірени, які можуть забруднювати ґрунт і воду.

Забруднення повітря

Неправильне спалювання електронного сміття призводить до викиду токсичних газів у атмосферу.

Зменшення ресурсів

Багато електронних пристроїв містять цінні матеріали, такі як золото, срібло, мідь і платина. Неправильна утилізація призводить до втрати цих ресурсів.



Соціальні проблеми

Небезпечні умови праці

У країнах, що розвиваються, де часто утилізують електронне сміття, працівники стикаються з небезпечними умовами праці.

Економічні проблеми

Витрати на утилізацію

Ефективна утилізація електронного сміття потребує значних фінансових ресурсів.

Втрата цінних матеріалів

Неправильна утилізація призводить до втрати цінних матеріалів, які можуть бути повторно використані в нових продуктах.

Файлова система

Файлова система – це організована структура, яка визначає спосіб зберігання та управління файлами і даними на пристроях зберігання, таких як жорсткі диски, SSD, флеш-накопичувачі та інші.



Файлова система надає методи для розміщення, пошуку, видалення та організації файлів у відповідні каталоги або папки.



Файл	Це найменша одиниця зберігання даних, яка містить інформацію і має унікальне ім'я.
Каталог (папка)	Це об'єкт файлової системи, що використовується для зберігання та впорядкування інших файлів або підкаталогів.
Диск (том)	Це логічна одиниця зберігання, яка може містити одну або кілька файлових систем.
Файлова структура	Це спосіб організації файлів у вигляді деревоподібної структури, де кореневий каталог містить підкаталоги і файли.

Основні функції файлової системи

1. **Зберігання даних**
   Файлова система забезпечує зберігання даних у файлах.

2. **Управління доступом**
   Вона регулює доступ до файлів, гарантуючи безпеку даних.

3. **Організація файлів**
   Файлова система допомагає організувати файли в ієрархічну структуру з каталогами і підкаталогами.

4. **Пошук файлів**
   Вона забезпечує швидкий та ефективний пошук файлів.

Види файлових систем

Різні файлові системи мають свої переваги і недоліки, і вибір залежить від конкретних потреб користувача та операційної системи.



FAT (File Allocation Table)

Одна з найстаріших файлових систем, розроблена для **MS-DOS** і **Windows**.

Переваги:

- Простота і сумісність з багатьма операційними системами.
- Підходить для флеш-накопичувачів та карт пам'яті.

Недоліки:

- Обмежений розмір файлів (до 4 ГБ у FAT32).
- Відсутність сучасних функцій безпеки і продуктивності.

NTFS (New Technology File System)

Основна файлова система для операційних систем **Windows NT** і пізніших версій.

Переваги:

- Підтримка великих файлів і томів.
- Функції безпеки, такі як права доступу і шифрування.
- Журнальована файлова система, що забезпечує надійність даних.

Недоліки: менша сумісність з іншими операційними системами (обмежена підтримка в **Linux** і **macOS**).

exFAT (Extended File Allocation Table)

Файлова система, розроблена для флеш-накопичувачів і карт пам'яті.

Переваги:

- Підтримка великих файлів і томів.
- Сумісність з більшістю сучасних операційних систем.

Недоліки: відсутність функцій безпеки і журналювання, як у NTFS.

HFS+ (Hierarchical File System Plus)

Основна файлова система для комп'ютерів **Apple** до **macOS High Sierra**.

Особливості:

- Підтримує великі файли і томи.
- Має вбудовані функції безпеки.

Кешування (Caching) – процес тимчасового зберігання часто використовуваних даних у швидкодоступній пам'яті для прискорення доступу до них.



Шифрування (Encryption) – процес перетворення даних у формат, який можна прочитати лише за допомогою ключа шифрування, для захисту конфіденційності.

Диски (жорсткі диски, SSD, флеш-накопичувачі) — основні пристрої для зберігання даних на комп'ютерах. Для забезпечення ефективної роботи необхідно правильно підготувати диски до використання і регулярно їх обслуговувати.

Форматування — це процес створення файлової системи на диску. Він очищає диск і готує його для зберігання даних.

Типи форматування:

- *Швидке форматування.* Створює нову файлову систему, не перевіряючи диск на наявність пошкоджень.
- *Повне форматування.* Очищає диск і перевіряє його на наявність пошкоджених секторів.



Розбивка на розділи (партиціювання) — це процес поділу диска на кілька логічних частин (розділів). Кожен розділ може мати свою файлову систему і використовуватися окремо.

Типи розділів:

- *Основні розділи (Primary partitions).* Можна створити до чотирьох основних розділів на диску.
- *Розширені розділи (Extended partitions).* Використовуються для створення додаткових логічних розділів всередині них.

Встановлення файлової системи. Після форматування і розбивки на розділи, на кожному розділі створюється файлова система (наприклад, NTFS, FAT32, ext4), яка визначає спосіб зберігання та організації даних на диску.

Обслуговування дисків

Дефрагментація

Дефрагментація — це процес об'єднання фрагментованих частин файлів на жорсткому диску, щоб поліпшити продуктивність доступу до них.

Інструменти для дефрагментації: вбудовані утиліти Windows (Дефрагментація диска).

Перевірка на наявність помилок

Регулярна перевірка дисків на наявність помилок допомагає виявити і виправити логічні та фізичні пошкодження.

Інструменти для перевірки: CHKDSK (Windows) — вбудована утиліта для перевірки дисків на наявність помилок.

Резервне копіювання даних

Резервне копіювання даних — це процес створення копій важливих файлів для захисту від втрати даних.

Методи резервного копіювання:

- *Локальне резервне копіювання.* Збереження копій на зовнішніх дисках або інших носіях.
- *Хмарне резервне копіювання.* Збереження копій у хмарних сервісах (Google Drive, Dropbox).

Логічні та фізичні ушкодження дисків

Логічні ушкодження виникають через програмні помилки або неправильне використання диска. Вони можуть бути виправлені за допомогою спеціальних програм.

Приклади логічних ушкоджень:

- Пошкоджені файлові системи.
- Втрачені або пошкоджені файли.
- Некоректні таблиці розділів.

Причини: некоректне вимкнення комп'ютера, програмні збої, віруси та шкідливе ПЗ.

Фізичні ушкодження виникають через механічні пошкодження або зношування апаратних компонентів диска. Вони часто незворотні і можуть вимагати заміни диска.

Ознаки фізичних ушкоджень:

- Незвичайні шуми від жорсткого диска.
- Часті збої і зависання системи.
- Неможливість доступу до даних.

Причини: механічні удари або падіння, перегрів, зношування компонентів.

Практична робота №3.

Індивідуальна робота

Частина 1

У вашій попередній практичній роботі №1 є **Google Таблиця** «Підбір конфігурації ПК».

1. Створіть новий аркуш в **Google Таблиці** та підготуйте три варіанти апгрейду вашого ПК.
2. Надайте доступ до таблиці вашому педагогу.

Частина 2

1. Знайдіть інформацію про місцеві пункти прийому електронного сміття.
2. Складіть план дій для утилізації старого комп'ютера або телефона.
3. Презентуйте свій план перед класом.

Пошукова колективна робота

1. Створіть коротку доповідь в документі в **Canva**.
2. Оберіть 5 питань, на які підготуєте коротку доповідь перед класом. Склад команди — 3–4 учні. Тривалість — 30 хв.

Питання

- Які новітні файлові системи використовуються у сучасних операційних системах?
- Які переваги APFS (Apple File System) перед HFS+?
- Які технології шифрування використовуються у файлових системах?
- Які інструменти використовуються для швидкого форматування SSD?
- Які переваги повного форматування порівняно зі швидким форматуванням?
- Чи потрібно дефрагментувати SSD? Чому?
- Які сучасні утиліти рекомендуються для дефрагментації HDD?
- Які інструменти використовуються для перевірки стану дисків у 2024 році?
- Які ознаки вказують на фізичні ушкодження жорсткого диска?
- Як часто потрібно виконувати резервне копіювання для запобігання втрати даних через фізичні ушкодження?
- Які новітні технології використовуються для переробки електронних відходів?
- Які компанії є лідерами в галузі переробки електронного сміття?
- Які сучасні методи утилізації електронного сміття є найбільш ефективними?
- Які країни лідирують у сфері екологічно чистої утилізації електронного сміття?

Тема 4 **Операційна система та її складові. Спеціальні функції операційної системи для користувачів з особливими потребами**

Ключові поняття

Операційна система (ОС)
Екранний диктор
Екранна клавіатура (On-Screen Keyboard)
Екранна лупа (Screen Magnifier)

Операційна система (ОС)

Операційна система (ОС) — це програмне забезпечення, яке керує роботою комп'ютера або іншого пристрою.

Існує багато різних ОС, розроблених для різних пристроїв і завдань.

Кожна операційна система призначена для певних видів пристроїв і має свої властивості та переваги. Вибір операційної системи залежить від типу пристрою та потреб користувача.

ОС є своєрідним мозком комп'ютера, який відповідає за виконання команд, організацію роботи і забезпечення взаємодії користувача з комп'ютером.

Функції операційної системи

- **Управління ресурсами**

ОС відповідає за керування апаратними ресурсами комп'ютера, включаючи процесор, оперативну пам'ять, диски, пристрої введення та виведення. Вона розподіляє ресурси між різними програмами і завданнями.

- **Запуск та керування програмами**

ОС запускає і припиняє роботу програм, а також надає механізми для керування ними, включаючи можливість перемикання між відкритими програмами.

- **Керування файлами і папками**

ОС дозволяє користувачам створювати, зберігати, переміщувати та видаляти файли та папки. Вона надає інтерфейс для роботи з файлами.

- **Забезпечення безпеки**

Операційна система включає механізми для захисту комп'ютера від загроз, таких як віруси, шкідливе програмне забезпечення та несанкціонований доступ. Вона контролює доступ користувачів та забезпечує конфіденційність та цілісність даних.

Операційні системи для ПК (персональних комп'ютерів)



Windows

- Підходить для широкого кола завдань.
- Широка сумісність з апаратним та програмним забезпеченням.
- Регулярні оновлення та захист.

Linux

Linux

- Безкоштовна та відкрита ОС.
- Висока налаштовуваність, підходить для програмістів та системних адміністраторів.
- Велика кількість дистрибутивів для різних потреб (Ubuntu, Fedora, Debian).

macOS

- Ідеально підходить для творчих завдань (графіка, відеомонтаж).
- Інтеграція з іншими пристроями Apple.
- Висока стабільність та безпека.

Chrome OS

- Орієнтована на роботу з вебдодатками.
- Підходить для простих завдань та навчання.
- Висока швидкість та безпека.

Налаштування операційної системи під власні потреби

1 Налаштування робочого середовища

Організація робочого столу (розміщення значків, шпалери).
Налаштування меню та панелі задач.

2 Встановлення програмного забезпечення

Пошук та встановлення необхідних програм (офісні пакети, браузери, графічні редактори).
Налаштування автозавантаження програм.

3 Персоналізація

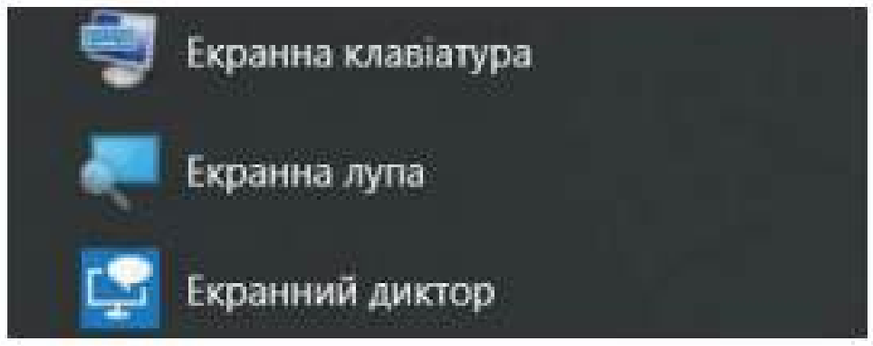
Зміна тем оформлення та шрифтів.
Налаштування гарячих клавіш та жестів.

4

Безпека та конфіденційність

Налаштування антивірусної програми.
Налаштування брандмауера та інших засобів захисту.
Налаштування резервного копіювання даних.

Кожна операційна система має вбудовані інструменти для підвищення доступності, що дозволяє налаштувати її відповідно до потреб користувачів з особливими потребами.



Екранний диктор

Екранний диктор — це інструмент для читання тексту, який відображається на екрані комп'ютера, призначений для людей з вадами зору. Він озвучує текст, сповіщення та інші елементи інтерфейсу, що допомагає користувачам взаємодіяти з комп'ютером без необхідності читати текст самостійно.

Основні налаштування

- *Голос:* оберіть голос диктора зі списку доступних.
- *Швидкість мовлення:* налаштуйте швидкість озвучення тексту.
- *Гучність:* налаштуйте гучність диктора.
- *Вибір клавіш*: виберіть зручні для вас клавіші керування.

Екранна клавіатура (On-Screen Keyboard)

Екранна клавіатура (On-Screen Keyboard) — це віртуальна клавіатура, яка відображається на екрані комп'ютера і дозволяє вводити текст та взаємодіяти з системою без фізичної клавіатури. Вона особливо корисна для користувачів з обмеженими можливостями або при пошкодженні фізичної клавіатури.

Основні функції

- *Введення тексту:* використовуйте мишу або сенсорний екран для введення тексту.
- *Спеціальні клавіші:* підтримка спеціальних клавіш, таких як **Shift**, **Ctrl**, **Alt** та інших.
- *Налаштування:* можливість налаштування зовнішнього вигляду та функціональності через меню параметрів.

Екранна лупа (Screen Magnifier)

Екранна лупа (Screen Magnifier) — інструмент, який дозволяє збільшувати частини екрана, полегшуючи перегляд тексту, зображень та інших елементів. Вона особливо корисна для користувачів з вадами зору.

Основні функції

- *Режими збільшення:* повноекранний режим, лупа на лінзі (lens mode), прив'язка до курсору миші.
- *Рівень збільшення:* регулюється клавішами **Windows + +** або **Windows + -**.
- *Налаштування:* відкрийте параметри лупи для налаштування рівня збільшення, інвертування кольорів та інших параметрів.

Оновлення операційної системи та програмного забезпечення

Регулярне оновлення операційної системи та програмного забезпечення є важливим для підтримання безпеки та продуктивності вашого комп'ютера.



Автоматичне оновлення

- Відкрийте **Параметри**: Натисніть **Windows + I**.
- Перейдіть до **Оновлення та безпека**: Виберіть **Update & Security**.
- Виберіть **Windows Update**: Натисніть **Check for updates** для перевірки наявності оновлень.
- Увімкніть автоматичні оновлення: Перейдіть до **Advanced options** і переконайтеся, що параметр **Automatic (recommended)** увімкнено.

Ручне оновлення

- Відкрийте **Параметри**: Натисніть **Windows + I**.
- Перейдіть до **Оновлення та безпека**: Виберіть **Update & Security**.
- Виберіть **Windows Update**: Натисніть **Check for updates** для перевірки наявності оновлень. Якщо оновлення доступні, натисніть **Download and install**.

Паролі та окремі облікові записи допомагають захищати та зберігати конфіденційність інформації на комп'ютері.

Типи облікових записів користувачів у Windows

Стандартний обліковий запис	Обліковий запис адміністратора
• Може бути локальним або підключеним до **Microsoft** обліковим записом користувача. • Зміна, редагування та видалення пароля. • Зміна зображення облікового запису. • Відсутня можливість встановлювати або відкривати нові програми.	• Може бути локальним або підключеним до **Microsoft** обліковим записом користувача. • Створення облікових записів. • Установлення та зміна прав. • Зміна паролів для облікових записів користувачів. • Встановлення та відкриття певних програм.

Переключення облікових записів дає змогу переходити між різними обліковими записами користувачів на одному пристрої, не виходячи з облікового запису повністю. Це може бути корисно, якщо кілька людей використовують один пристрій або у вас є окремі облікові записи для роботи та особистого використання.

Встановлення, видалення та автозавантаження програм

З офіційного магазину (наприклад, Microsoft Store)

- Відкрийте **Microsoft Store**.
- Знайдіть потрібну програму та натисніть **Встановити** (**Install**).

З інтернету (офіційних сайтів)

- Завантажте інсталяційний файл (**.exe**) з офіційного сайту програми.
- Відкрийте файл та дотримуйтесь інструкцій для встановлення.

З диску (наприклад, драйвери до пристроїв)

- Вставте диск у привід.
- Відкрийте диск і знайдіть файл **setup.exe**.
- Запустіть файл та дотримуйтесь інструкцій.



Практична робота №4.

Колективна робота

Ознайомитися з функціями екранної лупи, екранної клавіатури та екранного диктора в операційній системі **Windows**. Навчитися налаштовувати та використовувати ці функції для забезпечення доступності. Об'єднуємося в три команди та проводимо дослідницьку роботу.

1 група — досліджуємо екранну лупу

Дослідження режимів екранної лупи:

- Перевірте три режими роботи лупи: **Повноекранний режим** (**Full-screen mode**), **Лупа на лінзі** (**Lens mode**), **Прив'язка до курсору миші** (**Docked mode**).
- Використайте кожен режим для перегляду робочого столу та текстових документів.

Налаштування екранної лупи:

- Налаштуйте рівень збільшення.
- Увімкніть/вимкніть інвертування кольорів.
- Зробіть скріншоти з налаштуваннями лупи.

Практичне використання:

- Використовуйте екранну лупу для читання тексту на вебсторінках і в текстових документах.

Зробіть висновки про зручність використання кожного режиму. Результат оформте в **Google Презентацію**. Надайте доступ своєму педагогу.

2 група — екранна клавіатура

Дослідження функцій екранної клавіатури:

- Відкрийте текстовий редактор (наприклад, **Notepad**).
- Введіть текст за допомогою екранної клавіатури.
- Використовуйте спеціальні клавіші, такі як **Shift**, **Ctrl**, **Alt**.

Налаштування екранної клавіатури:

- Перейдіть до параметрів екранної клавіатури.
- Налаштуйте вигляд, розмір та інші параметри.
- Зробіть скріншоти з налаштуваннями екранної клавіатури.

Практичне використання:

- Використовуйте екранну клавіатуру для роботи з документами та навігації по вебсторінках.

Оцініть зручність використання екранної клавіатури.
Результат оформте в **Google Презентацію**.
Надайте доступ своєму педагогу.

3 група — екранний диктор

Дослідження функцій екранного диктора:

- Виберіть голос диктора зі списку доступних голосів.

- Налаштуйте швидкість мовлення, гучність та інші параметри.

Практичне використання:

- Використовуйте екранний диктор для навігації по робочому столу та роботи з текстовими документами.
- Відкрийте веббраузер і використовуйте диктора для читання вебсторінок.
- Зробіть висновки про зручність використання екранного диктора.

Перевірка роботи диктора:

- Відкрийте декілька програм та перевірте, як диктор читає текст у кожній з них.
- Зробіть скріншоти налаштувань диктора.
- Підготуйте звіт про виконану роботу у вигляді **Google Презентації**, включаючи скріншоти та описи всіх виконаних завдань.

Оцініть зручність використання кожної з досліджених функцій для користувачів з особливими потребами.

Поясніть, як ці функції можуть бути корисними у різних ситуаціях.

Індивідуальна пошукова робота

Організувати пошукову роботу. Матеріали організувати в **Canva Документі** та надати доступ педагогу! Оберіть 3–4 теми для пошукової роботи та організуйте пошукову діяльність!

Теми для пошукової роботи

- Як еволюціонували операційні системи від їх виникнення до сучасності?
- Які ключові інновації вплинули на розвиток операційних систем?
- Яка роль драйверів у роботі операційної системи?
- Які операційні системи використовуються на смартфонах?
- Що таке графічний інтерфейс користувача (GUI)?
- Що таке буфер обміну і як він використовується в операційній системі?
- Які основні відмінності між операційними системами **Windows**, **macOS** та **Linux**?
- Як можна налаштувати операційну систему під власні потреби (наприклад, зміна теми, налаштування робочого столу, вибір мови)?
- Які налаштування операційної системи можна змінити для поліпшення її продуктивності?
- Як можна зменшити кількість програм, що автоматично запускаються при завантаженні операційної системи, для прискорення її роботи?

Авторське право на програмне забезпечення. Ліцензії та їхні види. Поняття ліцензійного ключа

Тема 5

Ключові поняття

Ліцензія на програмне забезпечення
Безкоштовні ліцензії
Типи комерційних ліцензій
Технічні засоби захисту
Реєстрація авторського права

Згідно із Законом України «Про авторське право і суміжні права», програмне забезпечення розглядається як об'єкт авторського права, що охороняється аналогічно до літературних творів.

Це означає, що творці програмного забезпечення мають виключне право контролювати використання, копіювання, розповсюдження та модифікацію свого продукту.

Згідно з українським законодавством, автор програмного забезпечення має такі права:

Право на розповсюдження

автор має виключне право визначати, як його програмне забезпечення буде розповсюджуватися – через продаж, ліцензування або будь-яким іншим способом.

Право на модифікацію

тільки автор має право змінювати, доповнювати, або в будь-який спосіб модифікувати свій код.



Комерційні права

автор має право отримувати винагороду за використання його програмного забезпечення іншими особами, наприклад, через продаж ліцензій або отримання роялті.



Крім того, автор має немайнові права, такі як:

Право на авторство

Право бути визнаним автором програмного забезпечення та забороняти іншим особам привласнювати його авторство.

Право на недоторканність твору

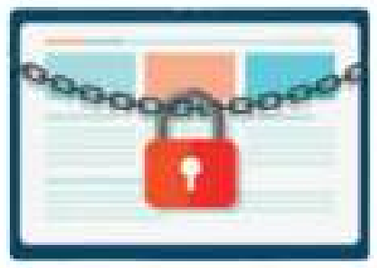
Права автора програмного забезпечення в Україні надійно захищені законодавством. Порушення цих прав тягне за собою адміністративну та кримінальну відповідальність.

Захист від несанкціонованого використання

Захист авторських прав на програмне забезпечення від несанкціонованого використання — це комплекс заходів, спрямованих на запобігання незаконному копіюванню, розповсюдженню та використанню вашого програмного забезпечення.

Ліцензування	Ліцензії чітко визначають права та обмеження щодо використання програми, запобігаючи її незаконному розповсюдженню.
Технічні засоби захисту	Використання спеціальних технологій, таких як шифрування, захист від копіювання, водяні знаки та інші, що ускладнюють несанкціонований доступ до коду та його використання.
Договори про нерозголошення	Укладання договорів з партнерами, співробітниками та іншими особами, які мають доступ до вашого програмного забезпечення, що зобов'язує їх зберігати конфіденційність та не розголошувати інформацію про програму третім особам.
Реєстрація авторського права	Реєстрація авторського права надає вашому програмному забезпеченню додатковий захист та полегшує процес захисту ваших прав у разі виникнення спорів.

Порушення авторського права — якщо програмне забезпечення використовується, копіюється, розповсюджується або змінюється без дозволу власника авторських прав.



- **Піратство програмного забезпечення.** Це несанкціоноване копіювання, використання або розповсюдження програмного забезпечення. Піратство може включати завантаження програм з інтернету без ліцензії або використання нелегальних копій програм.

- **Копіювання програмного забезпечення.** Створення копій програмного забезпечення без дозволу власника авторських прав.
- **Незаконне розповсюдження.** Продаж або розповсюдження програмного забезпечення без належної ліцензії.
- **Незаконна модифікація**. Зміна програмного забезпечення без дозволу власника авторських прав, включаючи створення та розповсюдження «зламаних» версій програм.

Ліцензія на програмне забезпечення

Ліцензія на програмне забезпечення — це юридична угода, яка надає користувачам певні права на використання програмного забезпечення відповідно до умов, встановлених розробником або власником авторських прав.

Комерційні

Користувачі повинні сплатити за ліцензію, щоб отримати право використовувати програмне забезпечення. Вартість може бути одноразовою або періодичною (наприклад, щорічна передплата).

Некомерційні

Програмне забезпечення надається безкоштовно, але може мати обмеження на комерційне використання.

Безкоштовні ліцензії

Freeware



Програмне забезпечення, яке можна використовувати безкоштовно, але з обмеженнями на модифікацію або розповсюдження. Прикладом є **Skype** або **Adobe Acrobat Reader**.

Freemium



Програмне забезпечення, яке надається безкоштовно з обмеженими функціями. Додаткові функції або контент доступні за плату. Прикладом є **Spotify** або **Dropbox**.

Одноразова ліцензія

Користувач сплачує одноразовий платіж за постійне використання програмного забезпечення. Після цього він може використовувати програму без додаткових витрат, але оновлення можуть вимагати окремої оплати.

Приклад: **Microsoft Office 2019**.

Ліцензія за передплатою

Користувач сплачує регулярні платежі (щомісячні або щорічні) за право використовувати програмне забезпечення. Всі оновлення та підтримка включені у вартість передплати.

Приклад: **Adobe Creative Cloud**, **Microsoft 365**.



Ліцензія за кількістю користувачів

Ліцензія обмежує кількість користувачів або пристроїв, які можуть використовувати програмне забезпечення одночасно. Це часто використовується в корпоративних середовищах.

Приклад: ліцензії для антивірусного програмного забезпечення.

Пробна ліцензія (Trial License)

Безкоштовне використання програмного забезпечення протягом обмеженого періоду часу. Після закінчення пробного періоду користувач має придбати повну ліцензію.

Приклад: безкоштовні пробні версії програм, як-от антивіруси або програми для редагування відео.

Ліцензія на час використання (Pay-Per-Use)

Користувачі сплачують за фактичний час або кількість використання програмного забезпечення. Це зручно для програм, які не використовуються постійно.

Приклад: деякі хмарні сервіси та онлайн-інструменти.

Переваги та недоліки комерційних ліцензій

- **Гарантія якості та підтримки.** Користувачі отримують офіційну технічну підтримку та регулярні оновлення, що забезпечують стабільну роботу програмного забезпечення.
- **Безпека.** Комерційні програми часто мають вищий рівень захисту від вірусів і зловмисного програмного забезпечення завдяки регулярним оновленням і виправленням.
- **Розширені можливості.** Комерційні ліцензії часто включають доступ до повного функціоналу програми, додаткових модулів та інструментів.
- **Вартість.** Використання комерційного програмного забезпечення може бути дорогим, особливо для малих підприємств або індивідуальних користувачів.
- **Обмеження на використання.** Обмеження на кількість пристроїв, обсяг функцій та умови використання можуть бути незручними для деяких користувачів.

Комерційні ліцензії часто обмежують кількість пристроїв, на яких можна встановлювати програмне забезпечення. Наприклад, ліцензія може дозволяти встановлення програми лише на одному комп'ютері або на певній кількості пристроїв.

Відкриті ліцензії (Open Source)

GNU General Public License (GPL)	Дозволяє використання, зміну і розповсюдження, зобов'язує робити вихідний код доступним.
MIT License	Дозволяє використання, копіювання, зміну і розповсюдження без обмежень, за умови збереження авторських прав.
Apache License	Дозволяє використання, зміну і розповсюдження, зобов'язує вказувати авторів і зберігати ліцензійні повідомлення.

Спільні ліцензії (Shared Source)

Community Source	Дозволяє використання вихідного коду для некомерційних цілей з обмеженнями на розповсюдження.
Proprietary Source	Код доступний тільки для перегляду або внесення змін певними користувачами з ліцензією.

Ліцензійний ключ

Ліцензійний ключ (або серійний номер) — це набір символів (букв і цифр), який надається разом з комерційним програмним забезпеченням.



Функції ліцензійного ключа

- **Активація програмного забезпечення.** Ліцензійний ключ вводиться під час встановлення або першого запуску програми для її активації.
- **Перевірка правомірності використання.** Ліцензійний ключ дозволяє виробнику програмного забезпечення перевірити, чи була програма придбана легально.
- **Обмеження використання.** За допомогою ліцензійного ключа можуть бути встановлені обмеження на використання програмного забезпечення, наприклад, на кількість пристроїв або термін дії ліцензії.

Процес активації програмного забезпечення за допомогою ліцензійного ключа

- **Введення ключа:** користувач вводить ліцензійний ключ під час встановлення або першого запуску програми.
- **Перевірка ключа:** програма перевіряє ключ на сервері виробника або за допомогою вбудованого алгоритму.
- **Активація:** якщо ключ є дійсним, програмне забезпечення активується і готове до використання.



Ліцензійний ключ є важливим інструментом для захисту програмного забезпечення від несанкціонованого використання і забезпечення дотримання ліцензійних умов. Він дозволяє виробникам контролювати використання своїх продуктів і забезпечує користувачам доступ до оновлень та підтримки.

Практична робота №5.

Колективна робота

1. Підготуйте короткий виступ для однокласників, де ви представите свою колективну інфографіку.
   Сервіс — **Canva**, інфографіка та документ для представлення.
   Кількість учасників в групі — 3–4 учні.
2. Оберіть тип ліцензії, про який будете готувати інформацію.
3. Зосередьтеся на комерційних ліцензіях, безкоштовних ліцензіях (freeware), ліцензіях з відкритим кодом (open-source), спільних ліцензіях (shared source) та публічних доменах (public domain).
4. Зберіть приклади програмного забезпечення, які використовують ці різні типи ліцензій.
5. Додайте графіку та ілюстрації для візуалізації інформації (наприклад, логотипи програмного забезпечення, діаграми, іконки).
6. Використовуйте інфографіку з мінімальною кількістю тексту для забезпечення зрозумілості та концентрації уваги на основних моментах.
7. Напишіть короткий сценарій виступу в **Canva Docs**, включаючи ключові моменти, які ви будете обговорювати на кожному слайді.
8 Підготуйтеся відповідати на можливі запитання однокласників про різні типи ліцензій.

Пошукова робота

1. Оберіть 3 теми та організуйте пошукову діяльність по обраних темах.
   - Що таке ліцензійна угода кінцевого користувача (EULA)?
   - Які основні умови зазвичай включають у ліцензійну угоду?
   - Які основні ліцензії з відкритим кодом існують?
   - Які переваги та недоліки має програмне забезпечення з відкритим кодом?
   - Які обмеження можуть бути встановлені в комерційній ліцензії?
   - Чому важливо розуміти умови ліцензій на програмне забезпечення перед його використанням?
   - Які наслідки можуть бути у випадку використання неліцензійного програмного забезпечення?
   - Що таке ліцензійна угода (EULA) і які її основні елементи?
   - Які основні відмінності між ліцензійними угодами на комерційне, безкоштовне та open-source програмне забезпечення?
   - Які наслідки можуть бути у випадку порушення ліцензійних умов?
   - Що таке справедливе використання (Fair Use) програмного забезпечення?
2. Результат оформте в **Canva Docs** та надайте доступ педагогу!

Тема 6 Сумісність програмного забезпечення. Автозавантаження. Оновлення програмного забезпечення

Ключові поняття

Розділ диска
Дефрагментація дисків
Сканування дисків
Форматування диска
Автозавантаження
Оновлення програмного забезпечення

Файлова система — це метод організації та управління даними на накопичувачі. Вона визначає, як дані зберігаються, доступні та керуються на диску.

Основні типи файлових систем

NTFS (New Technology File System)

Використовується в операційних системах **Windows**. Підтримує великі обсяги даних, розширені права доступу та шифрування.

FAT32 (File Allocation Table 32)

Поширена файлова система, сумісна з багатьма пристроями. Має обмеження на розмір файлів до 4 ГБ.

exFAT (Extended File Allocation Table)

Підтримує більші файли, ніж **FAT32**, та сумісна з багатьма операційними системами.



ext4 (Fourth Extended File System)

Використовується в операційних системах **Linux**. Підтримує великі обсяги даних та забезпечує високу швидкість доступу до файлів.

Форматування диска

Форматування диска — це процес підготовки накопичувача для зберігання даних шляхом створення файлової системи. Форматування видаляє всі дані на диску та налаштовує його для використання операційною системою.

Швидке форматування

Видаляє структуру файлової системи, але не перезаписує весь диск. Це займає менше часу, але не гарантує повне видалення даних.

Повне форматування

Видаляє всі дані та перевіряє диск на наявність пошкоджених секторів. Це більш тривалий процес, але забезпечує більш надійне видалення даних.

Розділ диска

Розділ диска — це логічна частина фізичного диска, яка функціонує як окремий диск. Створення розділів дозволяє ефективніше організовувати дані та розділяти операційні системи або типи даних.

Керування дисками

1. Відкрийте **Керування дисками** (натисніть правою кнопкою миші на **Цей ПК** і виберіть **Керування**).
2. Виберіть нерозподілений простір або диск, на якому потрібно створити розділ.
3. Натисніть правою кнопкою миші та виберіть **Створити простий том**.
4. Дотримуйтесь вказівок майстра створення розділів.



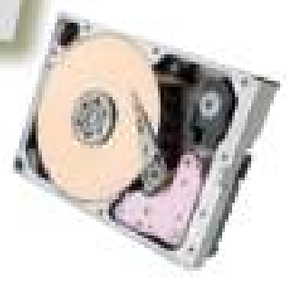
Для об'єднання або видалення розділів можна використовувати ті ж утиліти, що й для їх створення. Важливо створити резервну копію даних перед будь-якими операціями, що можуть вплинути на існуючі дані.

Сканування дисків

Сканування дисків — це процес перевірки файлової системи на наявність помилок та пошкоджених секторів. Сканування може допомогти виявити та виправити помилки, що можуть вплинути на продуктивність та надійність диска.

Інструменти для сканування дисків

Windows	Використовується утиліта **Check Disk (chkdsk)**.
macOS	Використовується утиліта **Disk Utility**.



Дефрагментація дисків

Дефрагментація дисків — це процес реорганізації фрагментованих даних на диску, щоб зменшити час доступу до файлів і підвищити загальну продуктивність системи. Фрагментація виникає, коли файли зберігаються у розрізнених блоках по всьому диску.

Інструменти для дефрагментації дисків

Windows	Використовується вбудована утиліта **Дефрагментація диска.**
macOS	Система автоматично виконує дефрагментацію, додаткові утиліти не потрібні.

Налаштування автоматичного обслуговування дозволяє системі автоматично виконувати завдання з обслуговування дисків, такі як сканування та дефрагментація, без втручання користувача.

Налаштування в Windows

1. Відкрийте **Планувальник завдань.**
2. Створіть нове завдання або налаштуйте існуюче для автоматичного сканування та дефрагментації дисків.
3. Вкажіть частоту виконання завдань (наприклад, щотижня або щомісяця).



Система автоматично виконує більшість завдань з обслуговування. Додатково можна налаштувати розклад перевірок за допомогою вбудованих утиліт.

Проблема нагромадження великих обсягів даних на власному пристрої

- Зменшення продуктивності пристрою через заповнення дискового простору.
- Зниження швидкості доступу до файлів.
- Підвищений ризик втрати даних через переповнення диска.

Зі збільшенням обсягів даних, що зберігаються на власних пристроях і в глобальному масштабі, виникає потреба у розв'язанні проблем, пов'язаних з управлінням цими даними. Великий обсяг даних може впливати на продуктивність пристроїв, а також мати екологічні наслідки через споживання енергії.



Методи розв'язання

1. **Стиснення файлів.** Використання утиліт для стиснення файлів з метою зменшення їхнього розміру (наприклад, **WinRAR**, **7-Zip**).
2. **Архівування.** Зберігання маловикористовуваних файлів у архівах.
3. **Видалення непотрібних файлів.** Регулярне видалення тимчасових файлів, кешу браузера, старих резервних копій та інших непотрібних даних.
4. **Зовнішні накопичувачі.** Використання зовнішніх жорстких дисків або USB-накопичувачів для зберігання великих обсягів даних.
5. **Хмарне зберігання.** Використання хмарних сервісів (**Google Drive**, **Dropbox**, **OneDrive**) для зберігання даних онлайн.

Практичні поради при нагромадженні великих обсягів даних на власному пристрої

Оптимізація використання дискового простору

1. Регулярно перевіряйте обсяг використаного дискового простору.

   Перевіряйте, скільки місця займають ваші файли, програми та інші дані на жорсткому диску або SSD. Це можна зробити за допомогою вбудованих утиліт операційної системи, таких як **Мій комп'ютер** у **Windows** або **Finder** у **macOS**.

2. Видаляйте непотрібні файли та програми.
   - Регулярно видаляйте тимчасові файли, кеш браузера, старі резервні копії та інші непотрібні файли.
   - Переглядайте встановлені програми та видаляйте ті, якими ви більше не користуєтесь. У **Windows** це можна зробити через **Панель керування** → **Програми та компоненти**, у **macOS** — через **Finder** → **Applications**.
3. Переносьте маловикористовувані файли на зовнішні накопичувачі або в хмару.
   - Зберігайте великі файли, які ви рідко використовуєте (наприклад, архіви фотографій або відео), на зовнішніх жорстких дисках або USB-накопичувачах.
   - Використовуйте хмарні сервіси (**Google Drive**, **Dropbox**, **OneDrive**) для зберігання документів, фотографій та інших даних онлайн. Це дозволяє звільнити місце на вашому комп'ютері та мати доступ до файлів з будь-якого пристрою з інтернетом.

Переваги хмарного зберігання

- Зберігайте важливі дані в хмарі для захисту від втрати.
- Використовуйте хмарні сервіси для зберігання важливих документів, фотографій, відео та інших даних. Це забезпечує захист від втрати даних через поломку пристрою або інші проблеми.

- Встановіть програми хмарних сервісів на ваш комп'ютер і синхронізуйте файли автоматично.

Використовуйте хмарні сервіси для спільної роботи та обміну файлами.

- Використовуйте хмарні сервіси для спільної роботи над документами, проєктами та іншими файлами. Це дозволяє легко ділитися файлами з колегами, друзями або родиною.
- Надавайте доступ до файлів або папок через спеціальні посилання або налаштовуйте права доступу для інших користувачів.

Сумісність програмного забезпечення, автозавантаження та оновлення програмного забезпечення є важливими аспектами управління комп'ютером. Ці процеси забезпечують ефективну роботу програм, їх стабільність та безпеку.

Сумісність програмного забезпечення означає, що програма може працювати на конкретній операційній системі або апаратній платформі без проблем. Це включає підтримку різних версій операційних систем, драйверів та інших програмних компонентів.

Типи сумісності

Апаратна сумісність

Програма повинна працювати на певному апаратному забезпеченні (процесор, оперативна пам'ять, графічна карта тощо).

Сумісність з операційною системою

Програма повинна працювати на певній версії операційної системи (**Windows, macOS, Linux**).

Сумісність з іншими програмами

Програма повинна взаємодіяти з іншими програмами без конфліктів.

Автозавантаження

Автозавантаження — це процес автоматичного запуску програм під час запуску операційної системи. Це дозволяє користувачам швидко отримати доступ до важливих програм відразу після увімкнення комп'ютера.

Переваги: швидкий доступ до часто використовуваних програм, економія часу на їхній запуск.

Недоліки: збільшений час завантаження операційної системи, підвищене навантаження на систему при запуску.

Як керувати автозавантаженням програм?



1. Відкрийте **Диспетчер завдань** (**Ctrl + Shift + Esc**).
2. Перейдіть на вкладку **Автозавантаження**.
3. Виберіть програму та натисніть **Вимкнути** або **Увімкнути.**
4. Відкрийте **System Preferences**.
5. Перейдіть до **Users & Groups**.
6. Виберіть ваш обліковий запис та перейдіть на вкладку **Login Items**.
7. Додайте або видаліть програми зі списку автозавантаження.

Оновлення програмного забезпечення

Оновлення програмного забезпечення — це процес встановлення нових версій програм, що включають виправлення помилок, поліпшення безпеки, нові функції та підвищення продуктивності.

Процес оновлення програмного забезпечення:

1. **Налаштування робочого середовища.** Перевірка наявності нових версій програмного забезпечення. Це можна зробити вручну (на офіційних сайтах виробників ПЗ) або налаштувати автоматичну перевірку наявності оновлень.
2. **Встановлення програмного забезпечення.** Завантаження необхідних файлів з офіційних джерел або серверів виробника.
3. **Персоналізація.** Інсталяція нових версій програм з виправленням помилок, додаванням нових функцій та поліпшенням безпеки.

Як керувати оновленнями програмного забезпечення?

Використання Windows Update

1. Відкрийте **Параметри → Оновлення та безпека**.
2. Натисніть **Перевірити наявність оновлень**.
3. Після перевірки натисніть **Встановити** для встановлення доступних оновлень.
4. Налаштуйте автоматичне оновлення для регулярного отримання останніх версій програм.



Оновлення програм через Microsoft Store

1. Відкрийте **Microsoft Store**.
2. Перейдіть до розділу **Завантаження та оновлення.**
3. Натисніть **Перевірити наявність оновлень** і встановіть доступні оновлення.

Практична робота №6.

Завдання 1. Керування автозавантаженням програм

1. Перевірка списку автозавантаження.
   - Відкрийте утиліту керування автозавантаженням на вашій операційній системі: **Windows**: **Диспетчер завдань** (**Ctrl + Shift + Esc**) → вкладка **Автозавантаження**.
2. Аналіз списку автозавантаження.
   - Перегляньте список програм, які запускаються при старті операційної системи.
   - Виберіть програми, які не потрібні для автозавантаження, і вимкніть їх.
3. Додавання нових програм до автозавантаження.
   - Додайте програму, яку ви часто використовуєте, до списку автозавантаження.
   - Перевірте, чи запускається ця програма автоматично при перезавантаженні комп'ютера.

Завдання 2. Оновлення програмного забезпечення

1. Перевірка наявності **оновлень.**
   - Перевірте наявність оновлень для операційної системи та встановлених програм **Windows**: **Параметри** → **Оновлення та безпека** → **Перевірити наявність оновлень**.
2. Встановлення оновлень.
   - Завантажте та встановіть доступні оновлення для операційної системи та програмного забезпечення.

Завдання 3. Дефрагментація диска

1. Запуск утиліти дефрагментації.
   - Відкрийте меню **Пуск** і введіть **Дефрагментація** у рядок пошуку.
   - Виберіть **Оптимізувати диски** з результатів пошуку.
2. Аналіз стану диска.
   - У вікні **Оптимізація дисків** виберіть диск, який ви хочете дефрагментувати.
   - Натисніть **Аналізувати**, щоб перевірити, наскільки фрагментованим є диск.
3. Дефрагментація диска.
   - Після аналізу стану диска натисніть **Оптимізувати** для запуску процесу дефрагментації.
   - Дочекайтеся завершення процесу. Це може зайняти деякий час залежно від розміру диска та ступеня його фрагментації.

4. Перевірка результатів дефрагментації.
   - Після завершення дефрагментації утиліта покаже результати. Перевірте, наскільки зменшився рівень фрагментації.

Індивідуальна пошукова робота

1. Оберіть 3 теми та організуйте пошукову діяльність по обраних темах.
2. Результат оформте в **Canva Docs** та надайте доступ педагогу.
   - Чи потрібно виконувати дефрагментацію на SSD-дисках? Чому?
   - Яка різниця між дефрагментацією HDD і SSD?
   - Як впливає використання різних файлових систем (наприклад, NTFS, FAT32) на процес дефрагментації?
   - Що слід зробити перед початком дефрагментації (наприклад, резервне копіювання даних)?
   - Які симптоми можуть вказувати на необхідність дефрагментації диска?
   - Як перевірити сумісність програмного забезпечення з операційною системою?
   - Які проблеми можуть виникнути при використанні несумісного програмного забезпечення?
   - Як перевірити системні вимоги програмного забезпечення перед його встановленням?
   - Які інструменти або вебсайти допомагають визначити сумісність програмного забезпечення?
   - Які переваги оновлення програмного забезпечення?
   - Які ризики можуть виникнути при оновленні програмного забезпечення?
   - Що таке резервне копіювання і чому його важливо виконувати перед оновленням програмного забезпечення?

Тема 7 **Проблеми зберігання даних та як їм запобігти. Втрата даних: причини, способи запобігання. Резервне копіювання даних на цифрових пристроях та у хмарних сервісах. Архівування даних. Поняття про методи стиснення даних**

Ключові поняття

Резервне копіювання
Оновлення програмного забезпечення
Стиснення даних
Архівування даних

Причини втрати даних

Апаратні збої	Поломки жорстких дисків, збій живлення, пошкодження пристроїв зберігання.
Програмні збої	Віруси, помилки в програмному забезпеченні, некоректне видалення даних.
Людський фактор	Випадкове видалення файлів, форматування диска без резервного копіювання.
Зловмисні дії	Хакерські атаки, шкідливі програми, крадіжка пристроїв.

Способи запобігання втраті даних

- **Резервне копіювання**: регулярне створення резервних копій важливих даних на зовнішніх носіях або в хмарних сервісах.
- **Захист комп'ютера від вірусів та шкідливих програм.**
- **Налаштування прав доступу:** обмеження прав доступу до важливих даних, використання паролів.
- **Регулярне оновлення програмного забезпечення:** встановлення останніх оновлень для операційної системи та програм.
- **Системи безперебійного живлення (UPS):** захист від раптових вимкнень електроенергії.

Збереження даних є важливою складовою роботи з комп'ютером. Вчасне виявлення та усунення проблем зберігання даних допомагає уникнути втрати важливої інформації та забезпечити стабільну роботу системи.

Резервне копіювання — це процес створення копій даних, які зберігаються окремо від оригіналів, для гарантування їхньої безпеки та можливості відновлення в разі втрати або пошкодження основних даних.



Резервне копіювання даних є критично важливим для захисту інформації від втрати через апаратні або програмні збої, випадкові видалення чи зловмисні дії.

Повне резервне копіювання (FULL BACKUP)

Створюється повна копія всіх даних.
Перевага: легке відновлення даних.
Недолік: займає багато часу та місця на носії.

Диференційне резервне копіювання (DIFFERENTIAL BACKUP)

Зберігаються лише зміни, що відбулися з моменту останнього повного резервного копіювання.
Перевага: швидше та економніше за повне резервне копіювання.
Недолік: відновлення вимагає останньої повної копії та всіх копій.

Методи резервного копіювання



Резервне копіювання на цифрові пристрої

Зберігання резервних копій на зовнішніх жорстких дисках, USB-накопичувачах.
Перевага: фізичний контроль над носіями, швидкий доступ до даних.
Недолік: ризик фізичного пошкодження або втрати носіїв.

Резервне копіювання у хмарні сервіси

Використання хмарних сервісів для зберігання резервних копій (**Google Drive**, **Dropbox**, **OneDrive**, **iCloud**).
Перевага: віддалений доступ до даних з будь-якого місця, автоматичне створення резервних копій.
Недолік: залежність від інтернет-з'єднання, можливі проблеми з конфіденційністю.

Етапи створення резервної копії

1 Створення резервної копії на цифровий пристрій

Використовуйте вбудовану утиліту **Резервне копіювання та відновлення**:

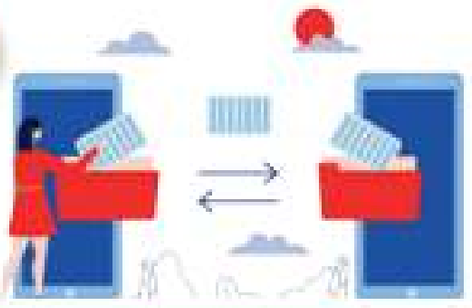
1. Відкрийте **Панель керування**.
2. Виберіть **Система та обслуговування** → **Резервне копіювання та відновлення**.
3. Налаштуйте резервне копіювання і виберіть місце для зберігання копії (зовнішній диск, мережевий диск).

2 Створення резервної копії у хмарний сервіс

1. Зареєструйтеся в обраному хмарному сервісі (**Google Drive**, **Dropbox**, **OneDrive**, **iCloud**).
2. Встановіть відповідну програму на комп'ютер або мобільний пристрій.
3. Налаштуйте автоматичну синхронізацію важливих файлів з хмарним сервісом.

3 Перевірка створеної резервної копії

Переконайтеся, що резервна копія успішно створена і всі необхідні файли збережені.



4 Автоматизація процесу резервного копіювання

Налаштуйте автоматичне створення резервних копій відповідно до обраної частоти (щодня, щотижня, щомісяця).

Стиснення даних

Стиснення даних — це процес зменшення розміру файлів шляхом усунення надлишкової інформації або оптимізації структури даних. Стиснення дозволяє зекономити місце на носіях даних та полегшує передачу файлів через мережу.



Архівування та стиснення даних є важливими інструментами для ефективного зберігання та передавання інформації. Використання різних методів стиснення дозволяє зменшити розмір файлів та гарантувати їхнє безпечне збереження.

Без втрат (LOSSLESS COMPRESSION)

Стиснення без втрат дозволяє відновити оригінальні дані з архіву без будь-якої втрати інформації.

Використовується для текстових файлів, програмного забезпечення, баз даних, зображень у форматах PNG та TIFF.

Приклади алгоритмів: DEFLATE (використовується в ZIP), LZMA (використовується в 7Z), BZIP2, GZIP.

Із втратами (LOSSY COMPRESSION)

Стиснення із втратами видаляє деяку частину даних, що не є критичною для сприйняття, для значного зменшення розміру файлів.

Використовується для мультимедійних файлів, таких як зображення (JPEG), аудіо (MP3), відео (MP4).

Приклади алгоритмів: JPEG, MP3, H.264.

Архівування даних

Архівування даних — це процес створення архівних копій файлів та папок для їх збереження в компактному вигляді. Архівні файли можуть містити один або кілька файлів і папок, які були стиснуті для зменшення їхнього розміру.



Архівування даних є важливою частиною управління файлами, що дозволяє зменшити розмір файлів для зберігання або передавання та забезпечити захист інформації.

Основні формати архівів

ZIP Один з найпопулярніших форматів архівування.
Підтримується більшістю операційних систем.
Підтримує стиснення та зберігання кількох файлів у одному архіві.

RAR Формат, відомий високим ступенем стиснення.
Потребує спеціального програмного забезпечення (**WinRAR**).
Підтримує створення багатотомних архівів.



7Z Формат, відомий високим ступенем стиснення та гнучкістю налаштувань.
Підтримує **AES-256** шифрування для забезпечення безпеки даних.
Використовує програмне забезпечення **7-Zip**.

Основні функції архівування

- **Стиснення даних.** Зменшення розміру файлів для економії місця на носіях даних та зменшення часу передачі файлів через мережу.
- **Об'єднання файлів.** Об'єднання кількох файлів та папок в один архівний файл для спрощення управління та передачі.
- **Захист даних.** Захист архівів за допомогою паролів та шифрування для забезпечення конфіденційності та безпеки даних.
- **Збереження структури.** Збереження структури директорій та атрибутів файлів при архівуванні та розархівуванні.

Архівування даних є важливим інструментом для ефективного управління файлами. Правильний вибір форматів та методів архівування дозволить зекономити місце на носіях даних, полегшити передачу файлів та забезпечити захист інформації.

Процес створення архівів з використанням WinRar

Встановлення WinRAR

- Завантажте інсталяційний файл з офіційного сайту **WinRAR** (**https://www.win-rar.com/**).
- Встановіть програму, дотримуючись інструкцій на екрані.

Створення архіву

- Виберіть файли та папки, які потрібно заархівувати.
- Натисніть правою кнопкою миші на вибрані файли та виберіть **Add to archive...**.
- У вікні налаштувань виберіть формат архіву (наприклад, **RAR** або **ZIP**), рівень стиснення та інші параметри.
- Натисніть **OK** для створення архіву.

Додавання пізніше файлів

- Перетягніть файли та папки, які потрібно додати, до вікна архіву **WinRAR**.
- Програма автоматично додасть вибрані файли до архіву.

Процес розархівування даних з використанням WinRar



1. Виберіть архів.
2. Відкрийте програму **WinRAR**.
3. Знайдіть архів, який потрібно розархівувати.
4. Розархівування: натисніть правою кнопкою миші на архів та виберіть **Extract Here** для розархівування в поточну папку або **Extract to ...** для вибору іншої папки.
5. Програма розархівує файли у вибране місце.

Практична робота №7.

Колективна робота

Завдання 1. Резервне копіювання даних

1. Вибір файлів для резервного копіювання.

Виберіть важливі файли та папки, які потрібно зберегти (заздалегідь підготовлена педагогом папка).

2. Створення резервної копії на цифровий пристрій.

- **Windows**: використовуйте вбудовану утиліту **Резервне копіювання та відновлення**:
- Відкрийте **Панель керування**.
- Виберіть **Система та обслуговування** → **Резервне копіювання та відновлення**.
- Налаштуйте резервне копіювання та виберіть місце для зберігання копії (зовнішній диск, мережевий диск).

Завдання 2. Архівування даних

1. Вибір програмного забезпечення для архівування.

Завантажте та встановіть програму для архівування, таку як **7-Zip** (або іншу на ваш вибір).

2. Створення архіву.

- Виберіть файли та папки, які потрібно заархівувати (заздалегідь підготовлена педагогом папка).
- Використовуйте встановлену програму для створення архіву.

3. Додавання даних до архіву.

- Відкрийте створений архів за допомогою програми архівування.
- Перетягніть додаткові файли до вікна архіву, програма автоматично додасть їх до архіву.

4. Розархівування даних.

- Відкрийте програму архівування.
- Знайдіть архів, який потрібно розархівувати, та виберіть його.

Індивідуальна пошукова робота

Оберіть 3 теми та організуйте пошукову діяльність по обраних темах. Результат оформте в **Canva Docs** та надайте доступ педагогу.

- Як програмні збої можуть призвести до втрати даних?
- Які заходи можна вжити для захисту даних від вірусів та шкідливих програм?
- Як захистити дані від випадкового видалення або форматування?

- Як налаштувати автоматичне резервне копіювання даних на **Windows, macOS** та **Linux**?
- Які переваги та недоліки резервного копіювання даних на зовнішні носії?
- Які переваги та недоліки використання хмарних сервісів для резервного копіювання?
- Як гарантувати безпеку даних, що зберігаються у хмарних сервісах?
- Як вибрати програму для архівування даних? Які популярні програми для архівування існують?
- Яка різниця між стисненням без втрат і стисненням із втратами?
- У яких випадках застосовуються методи стиснення без втрат, а в яких — стиснення із втратами?
- Які алгоритми стиснення даних використовуються у популярних форматах архівів (**ZIP**, **RAR**, **7Z**)?
- Які фактори впливають на ефективність стиснення даних?

Віруси та антивірусні програми. Брандмауер. Захист власного цифрового середовища від зловмисних дій

Тема 8

Ключові поняття

Шкідливе програмне забезпечення
Комп'ютерний вірус
Електронний фішинг
Вішинг (Vishing)
Смішинг (Smishing)
Брандмауер

Шкідливе програмне забезпечення

Шкідливе програмне забезпечення — це будь-яке програмне забезпечення, створене для пошкодження, проникнення або викрадення даних з комп'ютерних систем та мереж.



Шкідливе програмне забезпечення може впливати на роботу комп'ютерів, викрадати конфіденційну інформацію, вимагати викуп або використовувати ресурси комп'ютера для своїх цілей. Шкідливе ПЗ може мати різні форми та способи дії.

Основні способи поширення вірусів та шкідливого ПЗ

Електронна пошта

- *Вкладення:* віруси можуть бути вкладені у файли-вкладення електронних листів. Відкривши такий файл, користувач активує вірус.
- *Фішингові листи:* шахраї надсилають листи, які виглядають як повідомлення від легітимних організацій, з метою обманом змусити користувачів завантажити шкідливе ПЗ або надати особисту інформацію.

Завантаження з інтернету

- *Ненадійні джерела:* віруси часто ховаються у файлах, завантажених з ненадійних сайтів, піратських ресурсів або сумнівних джерел.
- *Фальшиві оновлення:* шкідливе ПЗ може маскуватися під оновлення для популярного програмного забезпечення.

Зовнішні носії даних

USB-накопичувачі: заражені USB-накопичувачі можуть переносити віруси з одного комп'ютера на інший. Вірус автоматично активується при підключенні до комп'ютера.

Мережі та інтернет

- *Мережеві черв'яки:* ці віруси можуть самостійно поширюватися через комп'ютерні мережі, використовуючи вразливості у програмному забезпеченні.
- *Зловмисні вебсайти:* відвідування сайтів, які містять шкідливий код, може призвести до автоматичного завантаження та встановлення вірусу на комп'ютері користувача.

Комп'ютерний вірус — це вид шкідливого програмного забезпечення, яке розроблене для того, щоб самовідтворюватися і поширюватися з одного комп'ютера на інший.

Комп'ютерні віруси можуть викликати різні проблеми, включаючи пошкодження або знищення даних, уповільнення роботи комп'ютера, викрадення особистої інформації та інші шкідливі дії.

Основні характеристики комп'ютерних вірусів

Самовідтворення	віруси мають здатність створювати свої копії та поширювати їх на інші файли або системи.
Прихованість	віруси можуть приховувати свою присутність у системі, щоб уникнути виявлення.
Шкідливість	віруси можуть виконувати різні шкідливі дії, такі як пошкодження файлів, видалення даних або збір конфіденційної інформації.



Шкідливе програмне забезпечення включає різні види шкідливих програм, які виконують небажані або шкідливі дії на комп'ютері.

Основні характеристики комп'ютерних вірусів та шкідливих ПЗ

Рекламна програма

Рекламна програма — це відносно небезпечний тип шкідливого програмного забезпечення, призначений радше для заробляння грошей, ніж для вчинення шкоди вашому комп'ютеру.

Ознаки зараження можуть бути різними, але основна з них — це спливаючі рекламні оголошення. Вони можуть з'являтися на робочому столі, в програмах, на вебсайтах або в місцях, де їх раніше не було.

Зловмисні програми з вимогою викупу

Зловмисні програми з вимогою викупу – це програми, створені зловмисниками, які погрожують жертві знищити або заблокувати доступ до важливих даних, доки вона не сплатить викуп.

Якщо користувач сплачує викуп, зловмисники можуть (але не завжди) надіслати ключ для розшифровки файлів. Однак немає гарантії, що після оплати доступ до даних буде відновлений.

Програма-вимагач може потрапити на комп'ютер через шкідливі вкладення електронної пошти, заражені вебсайти, завантаження з ненадійних джерел або через уразливості в програмному забезпеченні.

Хробак

Хробаки (Worms) – це вид шкідливого програмного забезпечення, яке самостійно поширюється через комп'ютерні мережі, використовуючи вразливості в програмному забезпеченні або операційних системах.

На відміну від вірусів, хробаки не потребують зараження інших файлів для розмноження і можуть поширюватися без участі користувача.

Хробаки здебільшого містяться у вкладеннях електронної пошти, текстових повідомленнях, програмах для обміну файлами, соціальних мережах, мережевих папках і на знімних дисках, а також поширюються мережею, використовуючи вразливості та копіюючи себе.

Шпигунські програми

Шпигунські програми можуть збирати паролі, банківські реквізити й електронні адреси, а також реєструвати натискання клавіш або змінювати налаштування безпеки. Уся зібрана інформація надсилається віддаленому користувачеві. Шпигунські програми також можна використовувати для завантаження та встановлення інших шкідливих програм без вашого дозволу.

Шпигунське ПЗ працює непомітно, часто без відома користувача, і може бути складним для виявлення. Збирає різноманітну інформацію, таку як натискання клавіш (кейлогери), історію вебпереглядів, паролі, електронні листи та інші особисті дані.

Руткіти (Rootkits)

Руткіти приховують присутність шкідливого ПЗ в системі, дозволяючи зловмисникам отримати адміністративний доступ до комп'ютера. Вони можуть бути дуже складними для виявлення та видалення.

- *Завантаження з інтернету:* може бути встановлений разом з іншими шкідливими програмами.
- *Фішингові атаки:* обдурені або довірливі користувачі завантажують чи встановлюють руткіти через підроблені листи або сайти.

Боти та ботнети (Bots and Botnets)

Боти – це шкідливі програми, які автоматично виконують команди, отримані від зловмисника.

Ботнет – мережа з багатьох заражених комп'ютерів (ботів), які координуються для виконання масових атак, таких як розподілені атаки на відмову в обслуговуванні (**DDoS**).

Як ураження відбувається:

- *Завантаження з інтернету:* встановлюються разом з іншими шкідливими програмами.
- *Мережеві вразливості:* використання уразливостей у мережевих протоколах для поширення.

Троянські коні (Trojan Horses)

Троянські коні – це шкідливі програми, які маскуються під легітимні або корисні програми. Після запуску вони виконують шкідливі дії, такі як викрадення даних або встановлення іншого шкідливого ПЗ.

Як ураження відбувається:

- *Фальшиве програмне забезпечення:* трояни часто маскуються під корисні програми або файли.
- *Фішингові атаки:* довірливі користувачі завантажують чи встановлюють троянських коней через підроблені листи або сайти.

Захист від вірусів та шкідливого ПЗ

Використання антивірусного програмного забезпечення

- *Оновлені антивірусні програми:* використовуйте актуальні антивірусні програми, які регулярно оновлюються для виявлення нових

загроз. Антивірусні програми можуть сканувати систему в реальному часі, виявляти та видаляти шкідливе ПЗ.

- *Повні сканування системи:* регулярно виконуйте повні сканування системи для виявлення прихованого шкідливого ПЗ.

Оновлення програмного забезпечення

- *Патчі та оновлення:* регулярно встановлюйте оновлення для операційної системи, браузерів та іншого програмного забезпечення. Оновлення часто містять виправлення безпеки, які закривають відомі вразливості.
- *Автоматичні оновлення:* увімкніть автоматичні оновлення, якщо це можливо, щоб завжди мати останні виправлення безпеки.

Обережність при роботі з електронною поштою та вебсайтами

- *Фішингові листи:* будьте обережні з листами від невідомих відправників. Не відкривайте вкладення та не клацайте на посилання в підозрілих листах.
- *Безпечні вебсайти:* відвідуйте лише відомі та надійні вебсайти. Перевіряйте наявність SSL-сертифікату (адреса починається з **https://**) перед введенням конфіденційної інформації.

Використання брандмауера

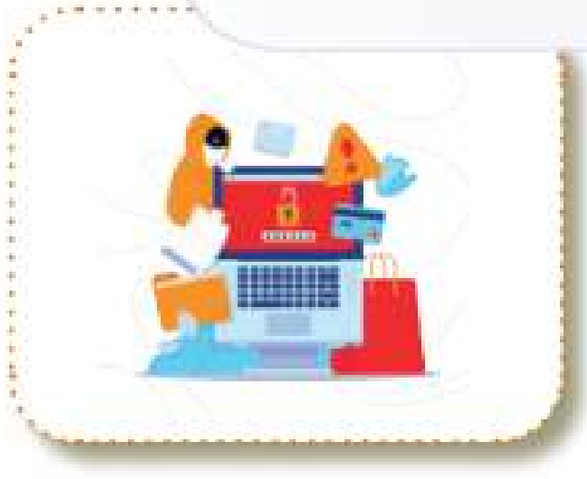
Мережевий брандмауер: налаштуйте та використовуйте мережевий брандмауер для контролю мережевого трафіку та блокування несанкціонованого доступу до комп'ютера.

Резервне копіювання даних

- *Регулярні резервні копії:* регулярно створюйте резервні копії важливих файлів на зовнішніх носіях або в хмарних сховищах. Це дозволить відновити дані у випадку зараження.
- *Захист резервних копій:* зберігайте резервні копії у безпечному місці та гарантуйте їхній захист паролями або шифруванням.

Електронний фішинг

Фішинг – це вид шахрайства, при якому зловмисники намагаються отримати конфіденційну інформацію, таку як паролі, номери кредитних карток, особисті дані та інші важливі відомості, обманом змушуючи користувачів надати ці дані.



Фішинг часто здійснюється через електронні листи, підроблені вебсайти або повідомлення в соціальних мережах, які виглядають як повідомлення від легітимних організацій.

Види фішингу

1 **Електронний фішинг**

- *Електронні листи:* найпоширеніший вид фішингу. Зловмисники надсилають електронні листи, які виглядають як повідомлення від банків, платіжних систем, соціальних мереж або інших довірених джерел. Листи можуть містити посилання на підроблені вебсайти або вкладення зі шкідливим ПЗ.
- *Підроблені вебсайти:* сайти, які виглядають як легітимні, але створені для викрадення даних користувачів. Відвідувачів просять ввести особисту інформацію, яка потім передається зловмисникам.

2 **Фішинг у соціальних мережах**

- *Фальшиві повідомлення:* повідомлення, які надсилаються через соціальні мережі, можуть містити посилання на підроблені вебсайти або заклики надати особисту інформацію.
- *Фальшиві профілі:* зловмисники створюють фальшиві профілі, видаючи себе за відомих людей або представників організацій, щоб втертися в довіру до користувачів і отримати їхні дані.

3 **Вішинг (Vishing)**

Голосові дзвінки: фішингові атаки, які здійснюються через телефонні дзвінки. Зловмисники можуть видавати себе за представників банків, технічну підтримку або інших довірених осіб, щоб виманити конфіденційну інформацію.

4 **Смішинг (Smishing)**

Текстові повідомлення: фішингові атаки, які здійснюються через SMS. Повідомлення можуть містити посилання на підроблені вебсайти або заклики надати особисту інформацію.

Як розпізнати фішинг

❑ **Підозрілі електронні листи та повідомлення**

- *Незвичні відправники:* листи від невідомих або підозрілих відправників.
- *Терміновість та погрози:* листи з повідомленнями про термінові дії або погрози блокування облікового запису.
- *Граматичні помилки:* листи з граматичними або орфографічними помилками.



- **Підозрілі посилання та вкладення**
  - *Перевірка посилань*: перед натисканням на посилання наведіть на нього курсор миші, щоб побачити URL-адресу. Якщо адреса виглядає підозріло або не відповідає офіційному сайту, не натискайте на покликання.
  - *Підозрілі вкладення:* не відкривайте вкладення від невідомих відправників або вкладення з підозрілими розширеннями файлів.
- **Фішингові вебсайти та сервіси**
  - *Перевірка URL-адреси:* переконайтеся, що адреса вебсайту починається з **https://** і містить правильний домен.
  - *Зовнішній вигляд:* якщо вебсайт виглядає незвично або містить помилки, можливо, це фішинговий сайт.

Дії у випадку фішингової атаки

Не взаємодійте з підозрілим повідомленням

Не натискайте на посилання. Якщо ви отримали підозрілий електронний лист або повідомлення, не натискайте на жодні покликання і не відкривайте вкладення.

Перевірте свої облікові записи

- *Змініть паролі.* Якщо ви підозрюєте, що ваші облікові дані могли бути скомпрометовані, негайно змініть паролі до всіх облікових записів. Використовуйте складні паролі і, якщо можливо, увімкніть двофакторну аутентифікацію (2FA).
- *Перевірте активність.* Перевірте свої облікові записи на наявність підозрілої активності. Зверніть увагу на невідомі входи або зміни, яких ви не робили.

Виконайте сканування системи

- *Перевірка URL-адреси.* Переконайтеся, що адреса вебсайту починається з **https://** і містить правильний домен.
- *Зовнішній вигляд.* Якщо вебсайт має незвичний вигляд або містить помилки, можливо, це підроблений сайт.

Використовуйте додаткові заходи безпеки

- *Двофакторна аутентифікація (2FA).* Увімкніть двофакторну аутентифікацію на всіх облікових записах, де це можливо. Це забезпечить додатковий рівень захисту.
- *Оновлення програмного забезпечення.* Регулярно оновлюйте операційну систему, браузери та інше програмне забезпечення, щоб захистити вашу систему від вразливостей.
- *Резервне копіювання даних.* Регулярно створюйте резервні копії важливих файлів, щоб мати можливість відновити дані у випадку їх втрати або пошкодження.

Ознаки фішингових повідомлень

1 Незвичний відправник

- *Невідомий відправник:* повідомлення від невідомих або підозрілих адрес.
- *Підроблені адреси*: адреси, що схожі на офіційні, але містять незначні відмінності, наприклад, додаткові літери або цифри.

2 Загальні привітання

Замість персоналізації: фішингові повідомлення часто починаються з загальних привітань, таких як «Шановний клієнте» або «Шановний користувачу», замість використання вашого імені.

3 Відчуття терміновості або загрози

- *Термінові повідомлення:* вимоги негайно виконати якусь дію, наприклад, оновити інформацію про обліковий запис або змінити пароль.
- *Погрози:* повідомлення про те, що ваш обліковий запис буде заблокований або з нього знімуть кошти, якщо ви негайно не вживете заходів.

4 Граматичні та орфографічні помилки

Наявність граматичних, орфографічних або стилістичних помилок у тексті повідомлення.

5 Підозрілі посилання та вкладення

- *Перевірка посилань:* посилання, які ведуть на невідомі або підозрілі вебсайти. Наведіть курсор на посилання, щоб перевірити URL-адресу, перш ніж натискати.
- *Небажані вкладення:* вкладення з підозрілими або несподіваними розширеннями файлів (**.exe**, **.zip**, **.rar**).

6 Запити на конфіденційну інформацію

- *Запит паролів:* легітимні організації ніколи не запитують ваші паролі або інші конфіденційні дані через електронну пошту.
- *Особиста інформація:* запити на надання особистих або фінансових даних.

7 Підозрілі вкладення

- *Неочікувані вкладення:* листи, що містять вкладення, яких ви не очікували отримати.
- *Небезпечні формати:* вкладення з розширеннями **.exe**, **.scr**, **.bat** або іншими виконуваними файлами.

Брандмауер

Брандмауер *(англ. firewall)* – це система захисту комп'ютерних мереж, яка контролює і фільтрує вхідний та вихідний мережевий трафік на основі встановлених правил безпеки.



Основна мета брандмауера — захистити внутрішню мережу від несанкціонованого доступу, шкідливих атак і вторгнень.

Основні функції брандмауерів

1. **Фільтрація трафіку**
Перевірка всіх вхідних і вихідних пакетів даних на відповідність встановленим правилам безпеки.

2. **Захист від несанкціонованого доступу**
Блокування спроб несанкціонованого проникнення в мережу зловмисників.

3. **Моніторинг активності**
Запис і аналіз мережевого трафіку для виявлення підозрілої активності.

4. **Запобігання DoS/DDoS-атакам**
Виявлення і блокування розподілених атак на відмову в обслуговуванні (DDoS), що спрямовані на перевантаження ресурсів мережі.

Мережевий трафік — це обсяг даних, які передаються через комп'ютерну мережу в обидві сторони, тобто від відправника до одержувача і навпаки.



Мережевий трафік включає всі види цифрової інформації, яка передається між пристроями в мережі, такими як комп'ютери, сервери, маршрутизатори, смартфони та інші пристрої, підключені до інтернету або локальної мережі.

Захист власного цифрового середовища від зловмисних дій

Використання антивірусного програмного забезпечення

- *Оновлення антивірусного ПЗ:* завжди використовуйте актуальне антивірусне програмне забезпечення, яке регулярно оновлюється для виявлення нових загроз.
- *Сканування системи:* виконуйте регулярне повне сканування вашого комп'ютера або інших пристроїв для виявлення та видалення шкідливого ПЗ.

Оновлення програмного забезпечення

- *Оновлення ОС та програм:* регулярно оновлюйте операційну систему, веббраузери та інші програми для закриття вразливостей, які можуть бути використані зловмисниками.
- *Автоматичні оновлення:* вмикайте автоматичні оновлення для важливих програм, щоб завжди мати останні виправлення безпеки.

Використання сильних паролів

- *Створення складних паролів:* використовуйте складні паролі, які містять великі та малі літери, цифри та спеціальні символи. Уникайте використання легко вгадуваних паролів.
- *Унікальні паролі:* використовуйте різні паролі для різних облікових записів, щоб у разі компрометації одного пароля інші залишалися захищеними.
- *Менеджери паролів:* використовуйте менеджери паролів для зберігання та генерування складних паролів.

Двофакторна аутентифікація (2FA)

- *Включення 2FA:* вмикайте двофакторну аутентифікацію для всіх важливих облікових записів. Це забезпечує додатковий рівень захисту, вимагаючи додатковий код, окрім пароля.
- *Методи 2FA:* використовуйте різні методи 2FA, такі як SMS-коди, додатки для аутентифікації (наприклад, **Google Authenticator**) або апаратні токени.

Захист мережі

- *Брандмауери:* використовуйте брандмауери для контролю мережевого трафіку і захисту від несанкціонованого доступу. Налаштуйте як апаратний, так і програмний брандмауер для оптимального захисту.
- *Шифрування Wi-Fi:* використовуйте сучасні протоколи шифрування для вашої Wi-Fi мережі, такі як **WPA3**. Змінюйте паролі за замовчуванням для маршрутизаторів.
- *VPN:* використовуйте віртуальні приватні мережі (VPN) для захисту даних під час їхньої передачі через інтернет, особливо при використанні публічних Wi-Fi мереж.

Використання безпечних налаштувань браузера

1. Використовуйте режим приватного перегляду (інкогніто) для запобігання зберіганню історії вебпереглядів та файлів *cookie*.
2. Увімкніть функцію блокування спливаючих вікон, щоб запобігти появі шкідливих або фішингових спливаючих повідомлень.
3. Налаштування конфіденційності та безпеки.
   - *Захист від фішингу та шкідливих програм:* увімкніть функції браузера, які попереджають про небезпечні вебсайти та блокують завантаження шкідливих програм.
   - *Розширення для блокування реклами:* використовуйте розширення для блокування реклами, щоб запобігти показу небажаної та потенційно шкідливої реклами.
4. Завжди використовуйте останню версію браузера. Оновлення часто містять виправлення безпеки, які закривають відомі вразливості.
5. Безпечні розширення та плагіни.
   - *Вибір розширень:* встановлюйте тільки ті розширення, які вам дійсно потрібні та які мають високу репутацію й хороші відгуки.
   - *Видалення непотрібних розширень:* видаляйте розширення, які більше не використовуються, щоб зменшити поверхню атаки.

Захист своїх смартфонів та планшетів

- **Встановлення пароля або ПІН-коду.** Встановіть пароль або ПІН-код для блокування вашого пристрою. Це перший шар захисту в разі втрати чи крадіжки.
- **Графічний патерн (Pattern Lock).** Замість пароля чи ПІН-коду ви можете використовувати графічний патерн, який потрібно намалювати на екрані для розблокування.
- **Розпізнавання відбитка пальця (Fingerprint Recognition).** Більшість сучасних смартфонів та планшетів підтримують розпізнавання відбитка пальця, що забезпечує швидкий та безпечний доступ до пристрою.
- **Віддалене відслідковування (Remote Tracking).** Увімкніть службу віддаленого відслідковування (наприклад, **Find My iPhone** для **Apple** або **Find My Device** для **Android**), яка дозволяє вам відстежувати місцезнаходження та навіть видалено блокувати чи видаляти дані на пристрої у випадку втрати.
- **Віддалене видалення даних (Remote Wipe).** В разі втрати пристрою ви можете видалити всі ваші особисті дані віддалено, щоб уникнути несанкціонованого доступу.
- **Встановлення додатків з надійних джерел.** Завжди завантажуйте додатки лише з офіційних джерел, таких як **Google Play Store** або **Apple App Store**, щоб уникнути встановлення шкідливого програмного забезпечення.

Антивірусна програма – це програмне забезпечення, розроблене для виявлення, запобігання і видалення шкідливих програм (вірусів), які можуть завдати шкоди комп'ютеру або знищити дані користувача.

Антивірусне ПЗ є важливим елементом кібербезпеки, оскільки воно захищає від широкого спектру загроз, включаючи віруси, трояни, черв'яки, шпигунське ПЗ, рекламне ПЗ та інше шкідливе ПЗ.

Основні функції антивірусної програми

Виявлення шкідливого ПЗ

Сигнатурний аналіз: антивірус сканує файли і порівнює їх зі своєю базою даних відомих шкідливих програм. Якщо знайдена сигнатура збігається з вірусною сигнатурою, файл позначається як шкідливий.

Евристичний аналіз: використовується для виявлення нових або змінених вірусів шляхом аналізу поведінки файлів.

Видалення шкідливого ПЗ

Видалення заражених файлів: антивірусні програми можуть видаляти або ізолювати заражені файли, щоб запобігти подальшому поширенню інфекції.

Відновлення файлів: деякі антивірусні програми можуть відновлювати заражені файли до їх чистого стану.

Захист мережі

Брандмауер: деякі антивірусні програми містять вбудований брандмауер, який контролює вхідний і вихідний мережевий трафік і блокує підозрілі з'єднання.

Захист від фішингу: виявлення і блокування фішингових вебсайтів, які намагаються викрасти особисту інформацію користувачів.

Сканування системи

Повне сканування: виконує повне сканування всієї системи, включаючи всі файли, папки і програми, щоб виявити та видалити шкідливе ПЗ.

Швидке сканування: сканує найбільш вразливі частини системи, такі як системні файли і ключові області реєстру, для швидкого виявлення загроз.

Заплановане сканування: користувачі можуть налаштувати антивірусну програму на виконання автоматичних сканувань у визначений час, наприклад, щотижня або щомісяця.

Захист конфіденційної інформації

Менеджер паролів: деякі антивірусні програми містять менеджери паролів, які допомагають створювати, зберігати і керувати складними паролями.

Захист вебкамери та мікрофона: виявлення і блокування несанкціонованого доступу до вебкамери і мікрофона для запобігання шпигунству.

Вибір антивірусної програми

- **Визначення потреб.** Якщо антивірус потрібен для домашнього комп'ютера, вибирайте програми, які забезпечують базовий захист від вірусів, шкідливих програм та фішингових атак.
- **Функціональність**

Реальний захист: переконайтесь, що програма забезпечує захист у реальному часі, сканує завантаження та активність у мережі.

Захист від фішингу: важлива функція для захисту від шкідливих сайтів та електронних листів.

Додаткові інструменти: фаєрвол, VPN, менеджер паролів, захист вебкамери та мікрофона.

- **Сумісність.** Перевірте сумісність антивірусної програми з вашою операційною системою та іншими програмами, які використовуєте.
- **Оцінка користувачів та відгуки.** Прочитайте відгуки інших користувачів та незалежні огляди, щоб оцінити надійність та ефективність антивірусної програми.
- **Вартість.** Порівняйте ціни та функціональність різних антивірусних програм. Безкоштовні версії можуть забезпечувати базовий захист, але платні версії часто мають додаткові функції та кращу підтримку.

Налаштування антивірусної програми

1. Встановлення антивірусної програми

Завантажте антивірусну програму з офіційного сайту і дотримуйтесь інструкцій для встановлення. Під час встановлення оберіть повну установку, щоб задіяти всі необхідні компоненти.

2. Початкове сканування

Після встановлення виконайте повне сканування системи для виявлення та видалення можливих загроз.

3. Реальний захист

Увімкніть функцію захисту в реальному часі, щоб антивірусна програма постійно моніторила ваш комп'ютер на наявність шкідливого ПЗ.

4. Планування регулярних сканувань

Налаштуйте регулярні автоматичні сканування системи, наприклад, щотижня або щомісяця, для забезпечення постійного захисту.

5. Захист веббраузера

Встановіть розширення для браузера, якщо воно доступне, для захисту від шкідливих вебсайтів та фішингових атак.

6. Додаткові функції

Якщо ваша антивірусна програма має додаткові функції, такі як VPN, менеджер паролів або батьківський контроль, налаштуйте їх відповідно до ваших потреб.

Практична робота №8.

Колективна робота

Завдання. Колективна дослідницька робота про різновиди та можливості антивірусів

1. Об'єднуємо учнів в 4-6 груп.
2. Оберіть антивірусну програму.

- Дослідіть можливості та обмеження різних антивірусних програм. Використовуйте інтернет для збирання інформації.
- Занесіть результати у **Google Таблицю** за таким зразком:
  Лист №1 — загальна робота класу (створює та надає доступ педагог).
  Критерії для Google Таблиці.
  - Антивірусна програма (назва)
  - Можливості
  - Обмеження
  - Вартість
  - Сумісність з ОС
  - Частота оновлень
  - Додаткові функції
- Підготуйте коротку **Google Презентацію** результатів дослідження вашої групи.
  Структура презентації
  Вступ:
  - Представте свою групу.
  - Назвіть антивірусну програму, яку ви досліджували (наприклад, **Norton**, **Kaspersky**, **Bitdefender**, **Avast** тощо).

  Опис антивірусної програми:
  Коротка історія антивірусної програми (розробник, рік випуску, версії).
  Основні можливості:
  – Виявлення вірусів та шкідливого ПЗ.
  – Запобігання зараженню.
  – Видалення шкідливого ПЗ.
  – Додаткові функції:
  Брандмауер, VPN, менеджер паролів, захист вебкамери, шифрування даних тощо.
  Аналіз можливостей та обмежень
  Можливості:
  - Описати функції та інструменти, які надає антивірусна програма.

  Наприклад, реальний захист, евристичний аналіз, регулярні оновлення.

Обмеження:

- Вказати на можливі недоліки або проблеми.

Наприклад, вплив на продуктивність системи, висока вартість, обмежена сумісність з певними ОС.

Висновки та рекомендації:

- Підбиття підсумків дослідження.
- Рекомендації щодо використання антивірусної програми для різних користувачів (домашні користувачі, бізнес-користувачі).

Питання та відповіді:

Кожен член групи повинен взяти участь у презентації.
Говоріть чітко і зрозуміло, підтримуйте зоровий контакт з аудиторією.
Будьте готові відповідати на питання однокласників та вчителя.

Індивідуальна робота

1. Розробити та впровадити комплексний план захисту власного цифрового середовища від зловмисних дій.
2. Створіть документ у **Google Docs** або **Microsoft Word**.

Структура документа

Назва документа:
Комплексний план захисту цифрового середовища.
Зміст:

- Вступ
- Встановлення та налаштування антивірусної програми
- Налаштування брандмауера
- Захист облікових записів
- Резервне копіювання даних
- Захист від фішингових атак
- Висновки та рекомендації

3. Розпиши покроково план захисту свого власного цифрового середовища.
4. Додайте скриншоти кожного етапу впровадження плану.
5. Надішліть педагогу посилання на документ з відповідними правами доступу.





Розділ 2

ЗБІР І ОПРАЦЮВАННЯ ДАНИХ

Тема 9 Збір даних. Персональні дані. Опитування, анкета, тест. Сервіси для збору даних

Ключові поняття

Збір даних
Персональні дані
Опитування
Анкета
Тест

Збір даних

У сучасному світі ми постійно збираємо та обробляємо дані, наприклад результати навчальних тестів, відповіді на анкети чи дані про використання вебсайтів.



Збір даних — це процес отримання та збереження інформації для її подальшого аналізу чи використання.

Персональні дані

Персональні дані — це будь-яка інформація, що стосується конкретної особи і може використовуватися для її ідентифікації. Це, зокрема, ім'я, дата народження, адреса, електронна пошта, а також фото або відео.



Збір персональних даних є особливо важливим процесом, оскільки вони можуть використовуватися для різної мети, тому важливо гарантувати їхню безпеку.

1. Дані повинні бути захищеними від несанкціонованого доступу.
2. Дані повинні використовуватися лише з визначеною метою.
3. Персональні дані можна збирати лише з дозволу особи.

Опитування. Анкета. Тест

Опитування — це метод збору інформації від групи людей через запитання, які потребують від них розгорнутої відповіді.



Мета опитування — отримати думку чи знання респондентів з певної теми.

Анкета — це структурований набір запитань, в яких респонденти обирають з переліку відповідь або вносять свій варіант.



Анкети використовуються для отримання детальної інформації на конкретну тему.

Тест — це форма перевірки знань або навичок, яка включає питання з визначеними правильними відповідями.

Тести можуть бути використані для оцінювання знань учнів, діагностики рівня навчання або визначення компетенцій.

Google Форми — це безкоштовний інструмент від Google, який дозволяє створювати опитування, анкети, вікторини та тести.



Він інтегрований з **Google Диском** і є частиною пакета **Google Workspace**, що дозволяє легко збирати та аналізувати дані. Користувачі можуть створювати форми для збирання різних типів інформації, а відповіді автоматично зберігаються в **Google Таблицях** для подальшого аналізу.

Типи запитань

Google Форми мають широкий набір функцій, які дозволяють створювати різні типи запитань та налаштовувати дизайн.

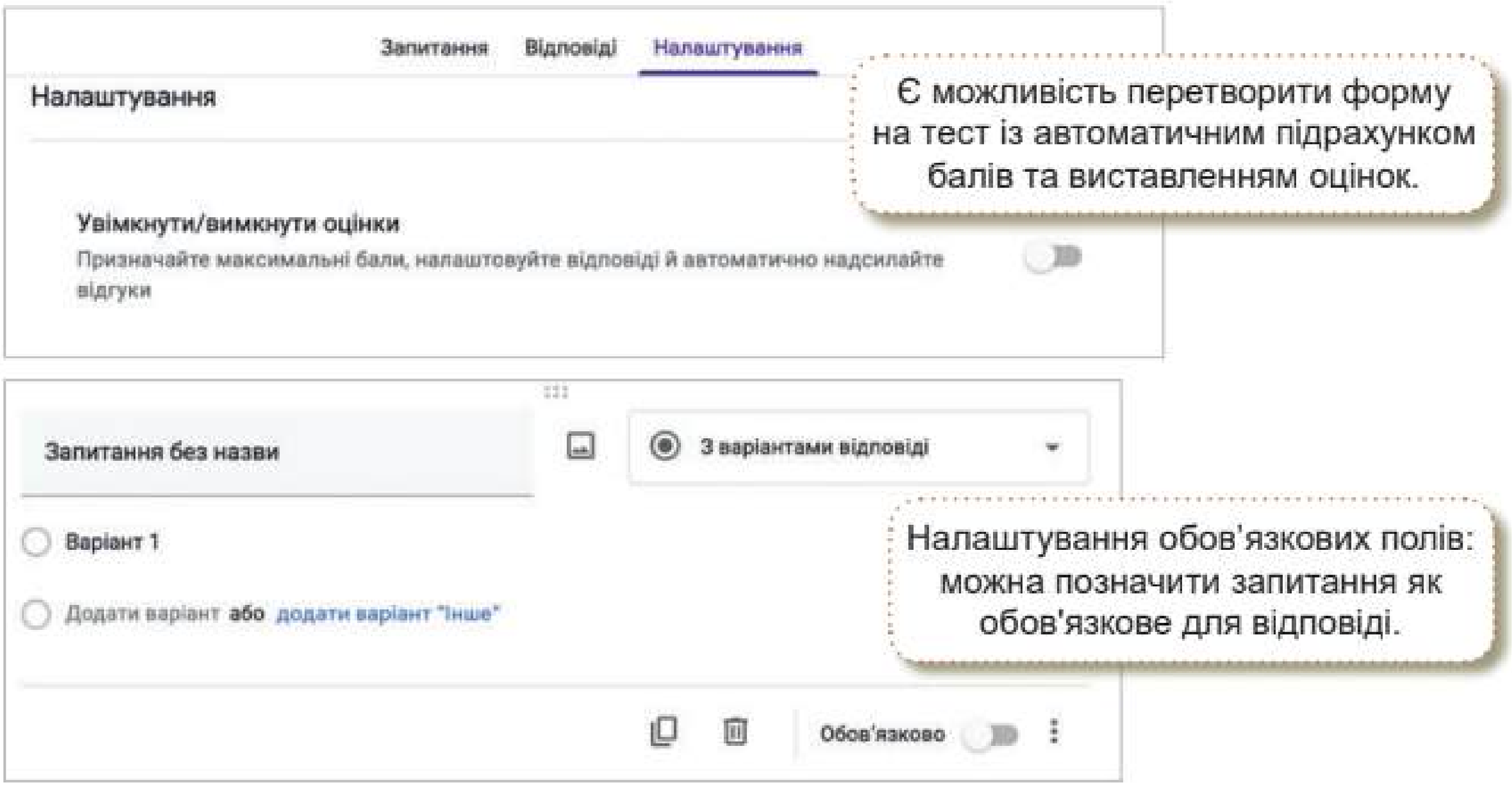
- **Налаштування обов'язкових полів.** Можна позначити запитання як обов'язкове для відповіді.

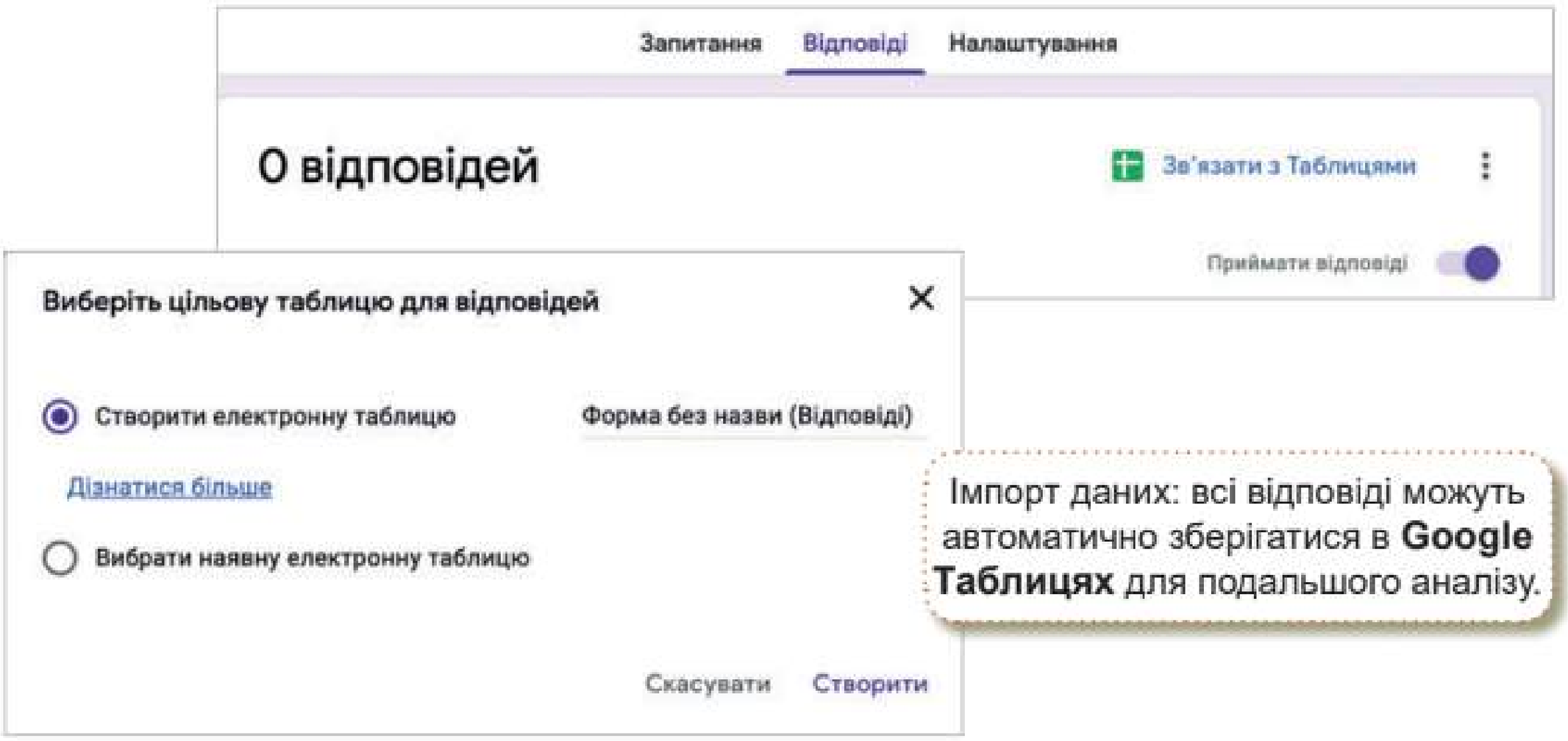
- **Вбудовані тести.** Є можливість перетворити форму на тест із автоматичним підрахунком балів та виставленням оцінок.
- **Файли для завантаження.** Можливість дозволити користувачам завантажувати файли, наприклад, домашні завдання або проєкти.
- **Імпорт даних.** Всі відповіді можуть автоматично зберігатися в **Google Таблицях** для подальшого аналізу.

1. Для коротких відповідей, наприклад, імен чи електронних адрес.
2. Для довших відповідей у формі абзацу.
3. Вибір одного з декількох варіантів.
4. Вибір кількох варіантів з наданого списку.
5. Вибір одного варіанта зі списку.
6. Файли для завантаження: можливість дозволити користувачам завантажувати файли, наприклад, домашні завдання або проєкти.
7. Рейтингові питання, де респондент може оцінити щось за шкалою від 1 до 10.
8. Запитання у формі таблиці, де респондент вибирає відповіді для кількох рядків і стовпців.
9. Запитання, що вимагають вказати дату або час.



Особливості





Переваги Google Форми

Google Форми без проблем працюють з іншими інструментами **Google**, такими як **Google Таблиці**, **Google Календар** і **Google Диск**. Наприклад, ви можете автоматично збирати дані та переглядати їх у таблицях або організовувати події в **Google Календарі**.

Вбудовані тести дозволяють **Google Формам** автоматично підраховувати бали та надавати зворотний зв'язок після завершення тесту. Це економить час і полегшує оцінювання.

Google Форми повністю адаптивні до мобільних пристроїв. Це означає, що учні можуть легко відповідати на запитання або проходити тести зі своїх смартфонів і планшетів.

1. Безкоштовно — усі функції доступні без додаткових витрат.
2. Ви можете поділитися формою з іншими користувачами для спільного редагування.
3. Вбудовані функції аналізу, можливість експорту в таблиці з можливістю створити графіки.

Практична робота №9.

Індивідуальна робота

Створення особистого тесту або опитування

1. Створіть **Google Форму** на обрану вами тему. Тема має бути актуальною для ваших однокласників. Наприклад, це може бути:
   - Тест на знання улюбленого шкільного предмету (математика, історія, біологія).
   - Опитування про хобі та захоплення однокласників.
   - Опитування про здоровий спосіб життя серед учнів.
2. Використайте не менш ніж три різні типи запитань.
3. Надішліть форму однокласникам, щоб зібрати мінімум 5 відповідей.
4. Проаналізуйте результати у **Google Таблицях** та створіть діаграму для візуалізації результатів опитувань.
5. Опишіть, що ви дізнались із отриманих даних (наприклад, «Більшість моїх однокласників займаються спортом тричі на тиждень», або «Виявилось, що найулюбленішим предметом у класі є біологія»).

Колективна робота

Створення **Google Форми** та аналіз даних

Об'єднуємо учнів в групи по 3–4 особи в кожній групі.

- Кожна група обирає одну з запропонованих тем для створення **Google Форми**.
- Група розробляє анкету у **Google Формах**, яка має містити різні типи запитань — мінімум 9 позицій.
- Завдання. Надіслати форму однокласникам, зібрати мінімум 10 відповідей.
- Відкрийте зібрані відповіді в **Google Таблицях**, створіть графіки для аналізу відповідей.
- Коротенько представник з групи розповідає про результати із діаграмами та основними висновками зібраних даних.

Теми для створення Google Форми

- Вплив гаджетів на навчання *(Скільки годин проводите за смартфоном? Як часто використовуєте інтернет для навчання?)*
- Вибір кар'єри *(Які професії вам цікаві? Чи задумувались над вибором майбутньої професії?)*
- Які соцмережі найпопулярніші у вашому класі? *(Якими соціальними мережами користуєтесь щодня? Скільки часу на день витрачаєте на соціальні мережі?)*
- Цифровий детокс *(Чи намагались зробити цифровий детокс — відмовитись від гаджетів на певний час? Як довго можете бути без смартфона?)*

Візуалізація даних (презентація, інфографіка, карти, діаграми тощо)

Ключові поняття

Google Таблиця

Візуалізація даних

Google Таблиця

Google Таблиці — це потужний інструмент для візуалізації даних, який дозволяє легко створювати різні типи діаграм і графіків, а також використовувати ці дані для презентацій та інфографіки.

Google Таблиці пропонують різноманітні типи діаграм, зокрема стовпчасті, кругові, лінійні, точкові діаграми та інші, що дозволяє візуалізувати дані в зручний і зрозумілий спосіб.

Процес створення діаграм дуже простий. Ви можете легко вибрати дані, а потім одним натисканням додати потрібну діаграму.

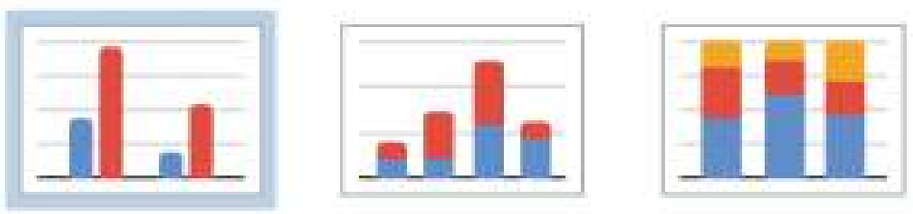
Якщо ви змінюєте дані у таблиці, діаграми автоматично оновлюються відповідно до нових значень. Це забезпечує актуальність візуалізацій без потреби їх вручну переробляти.

Ви можете надати доступ до своїх таблиць та діаграм іншим користувачам, які можуть в реальному часі переглядати, редагувати дані або створювати власні візуалізації. Це зручно для групових проєктів.

Діаграми, створені в **Google Таблицях**, можна експортувати у різних форматах (PNG, SVG, PDF), що дозволяє використовувати їх у зовнішніх документах або презентаціях.

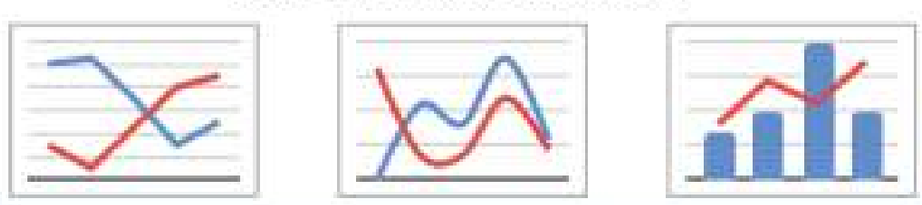
Стовпчаста діаграма

Використовується для порівняння кількості або частот для різних категорій.



Лінійна діаграма

Показує зміну даних з часом або залежність однієї змінної від іншої. Дані зображуються у вигляді крапок, які з'єднані лініями.



Кожен тип діаграми має своє застосування. Наприклад, стовпчаста діаграма добре підходить для порівняння кількісних даних, а кругова — для демонстрації часток у загальній структурі.

Колір діаграми повинен бути обраний так, щоб виділяти різні категорії та допомагати легше сприймати інформацію. Варто уникати занадто яскравих або контрастних кольорів, які можуть відволікати.

Важливо дотримуватись простоти та уникати перевантаження діаграм зайвою інформацією або складними графічними елементами.

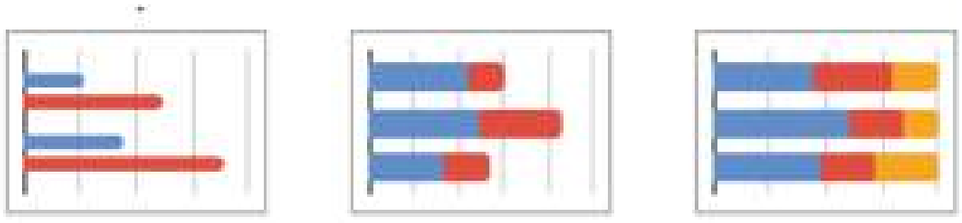
Площинна діаграма

Це варіант лінійної діаграми, але область під лініями заповнена кольором для наочності.



Гістограма

Подібна до стовпчастої діаграми, але дані відображаються горизонтальними стовпцями. Це зручний спосіб порівняння довгих текстових назв або категорій.



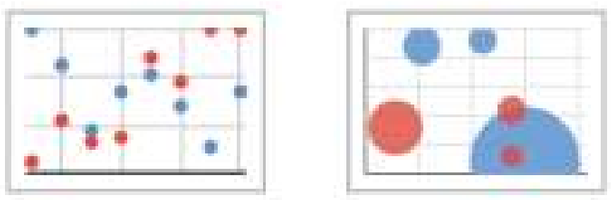
Підписи осей допомагають користувачам краще зрозуміти, що зображено на діаграмі. Зазначайте одиниці виміру, якщо це необхідно.

Для роботи з великими обсягами даних у **Google Таблицях** можна використовувати фільтри та зведені таблиці, щоб зосередитись на ключових аспектах та вивести на діаграму тільки релевантні дані.

Можна поєднувати декілька типів діаграм, наприклад, стовпчасті та лінійні, щоб порівняти різні типи даних на одному графіку.

Точкова діаграма

Показує залежність між двома змінними у вигляді точок на площині. Використовується для аналізу розподілу даних або пошуку взаємозв'язків між ними.



Секторна діаграма

Використовується для показу пропорцій або часток у загальній кількості. Дані відображаються у вигляді секторів кола.

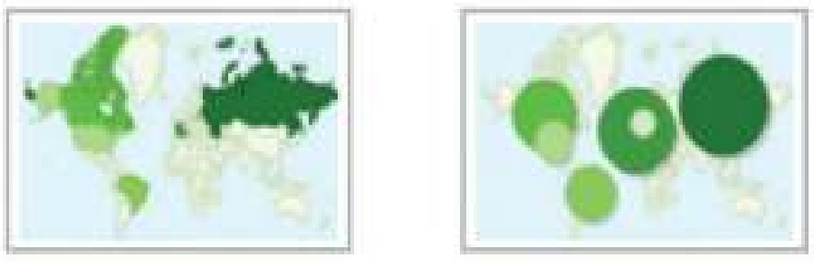


Google Таблиці дозволяють змінювати кольори, підписи, стилі ліній, мітки даних та інші параметри, щоб поліпшити вигляд і сприйняття діаграми.

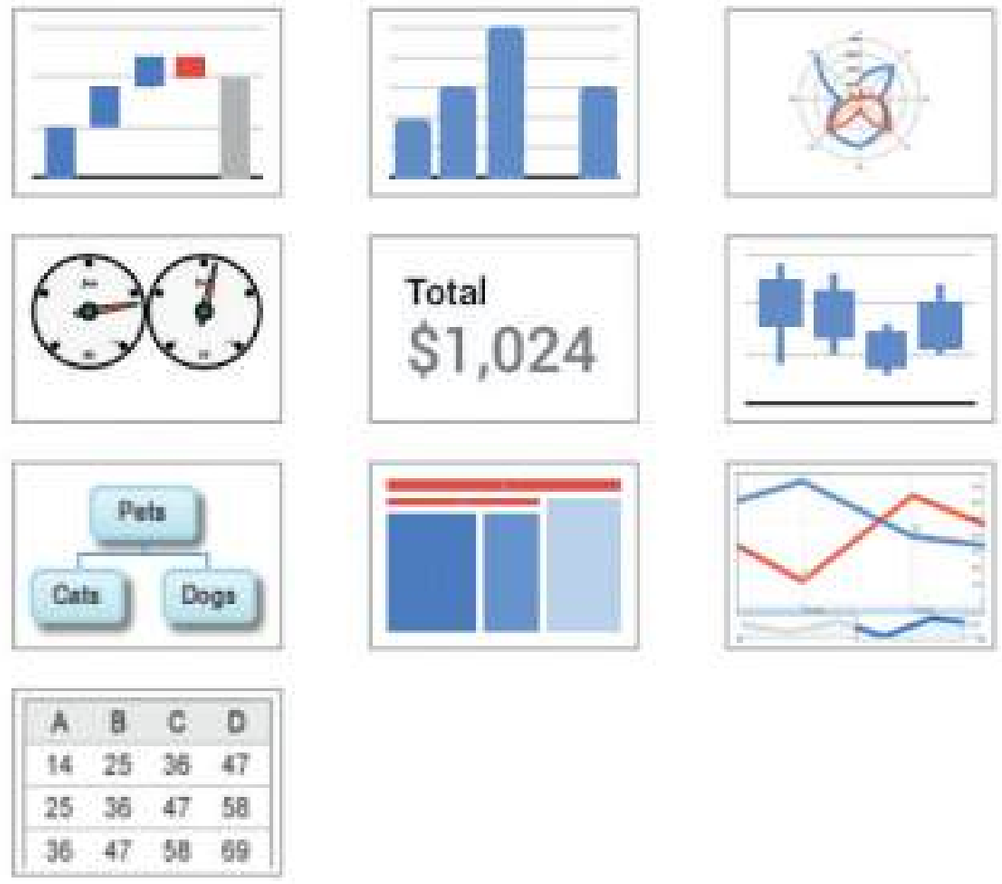
Діаграми, створені в **Google Таблицях**, можна експортувати у різних форматах (PNG, SVG, PDF), що дозволяє використовувати їх у зовнішніх документах або презентаціях.

Картодіаграма

Відображає дані на карті, використовуючи кольори або мітки для відображення інтенсивності чи кількості.



Інші діаграми



Візуалізація даних

Візуалізація даних — це спосіб представлення чисел або текстів у вигляді графіків і діаграм.





Практична робота №10.

Індивідуальна практична робота

Відкрити **Google Таблицю** (скопіювати окремо для кожного учня)

Завдання №1

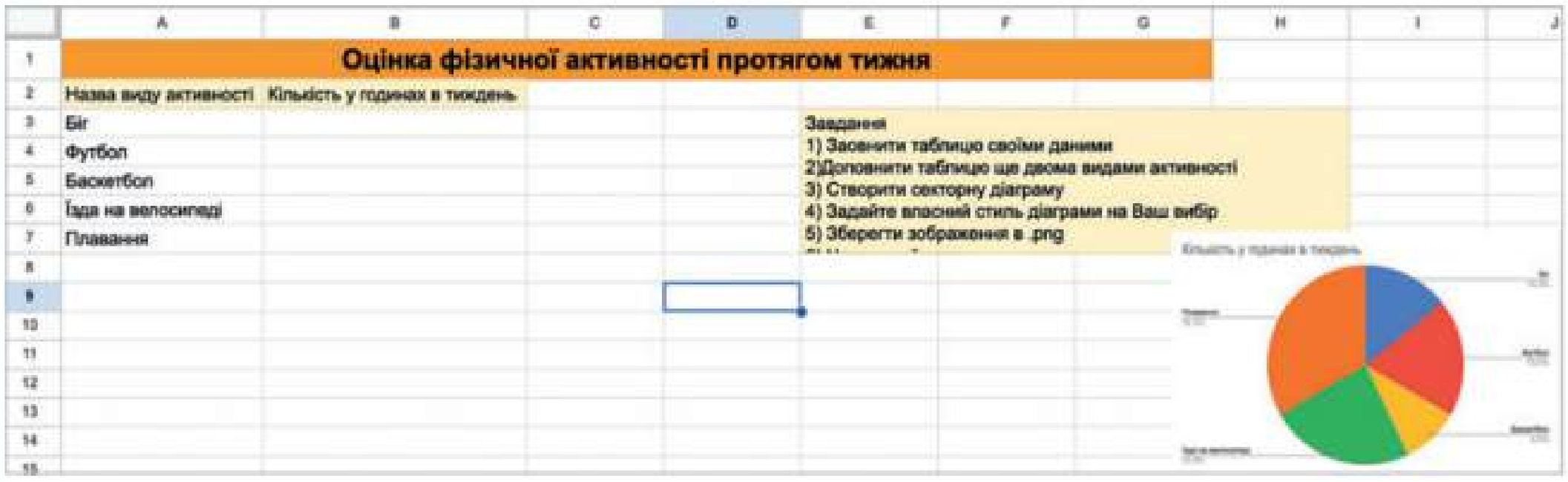
1. Заповніть таблицю своїми даними.

2. Доповніть таблицю ще двома видами активності.

3. Створіть секторну діаграму.

4. Задайте власний стиль діаграми на ваш вибір.

5. Збережіть зображення в **.png**.



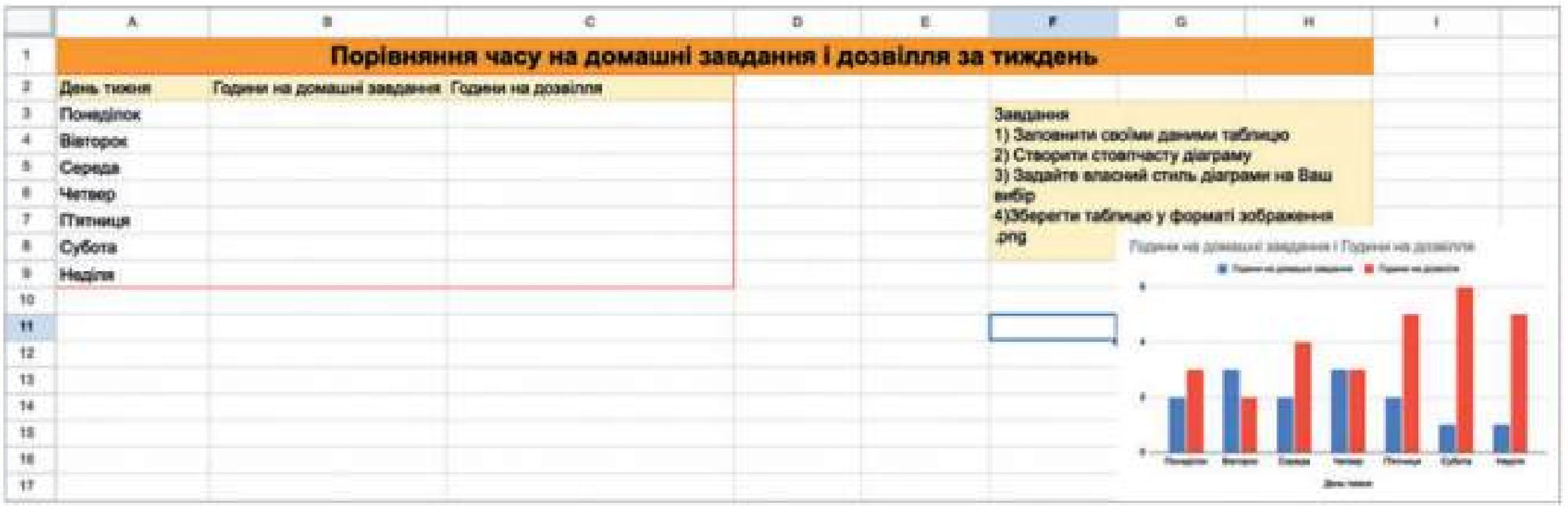
Завдання №2

1. Заповніть своїми даними таблицю.

2. Створіть стовпчасту діаграму.

3. Задайте власний стиль діаграми на ваш вибір.

4. Збережіть таблицю у форматі зображення **.png**.



Завдання №3

1. Заповніть таблицю своїми даними.

2. Створіть лінійну діаграму.

3. Задайте власний стиль діаграми на ваш вибір.

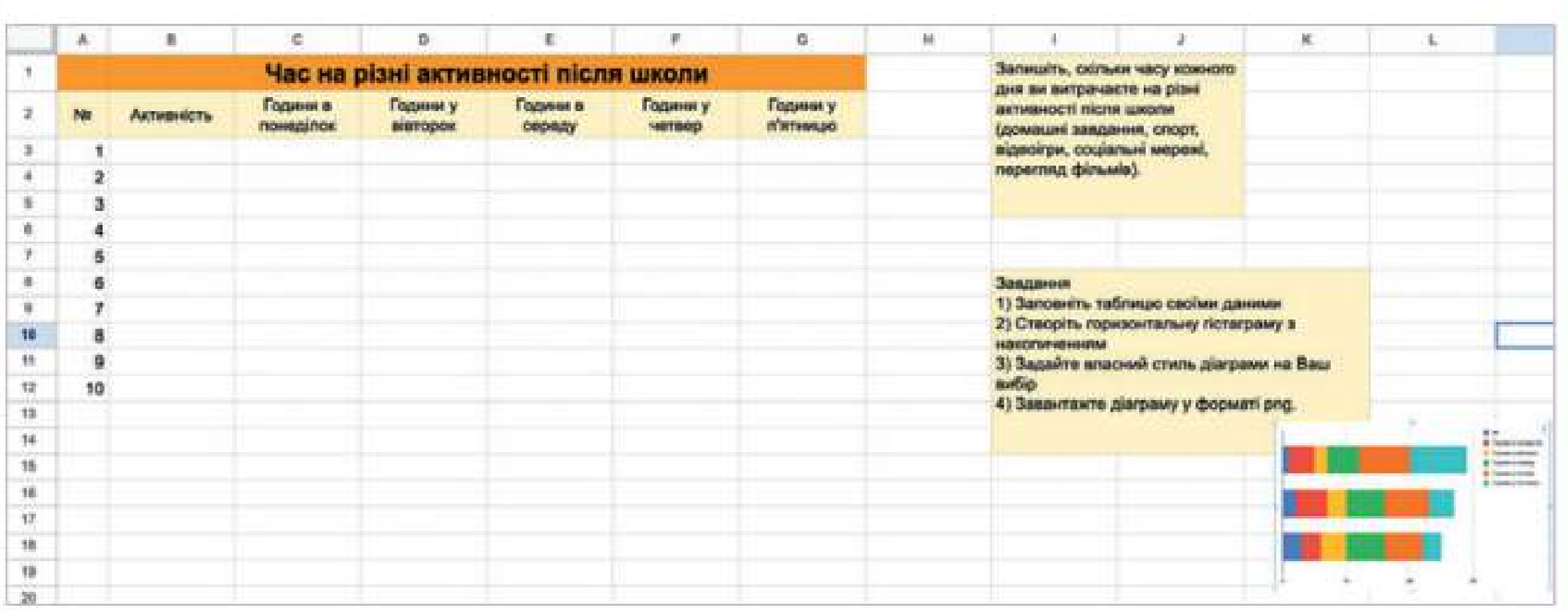
4. Збережіть діаграму в форматі зображення **.png.**



Завдання №4

Запишіть, скільки часу щодня витрачаєте на різні активності після школи (домашні завдання, спорт, відеоігри, соціальні мережі, перегляд фільмів).

1. Заповніть таблицю своїми даними.
2. Створіть горизонтальну гістаграму з накопиченням.
3. Задайте власний стиль діаграми на ваш вибір.
4. Завантажте діаграму у форматі **png**.

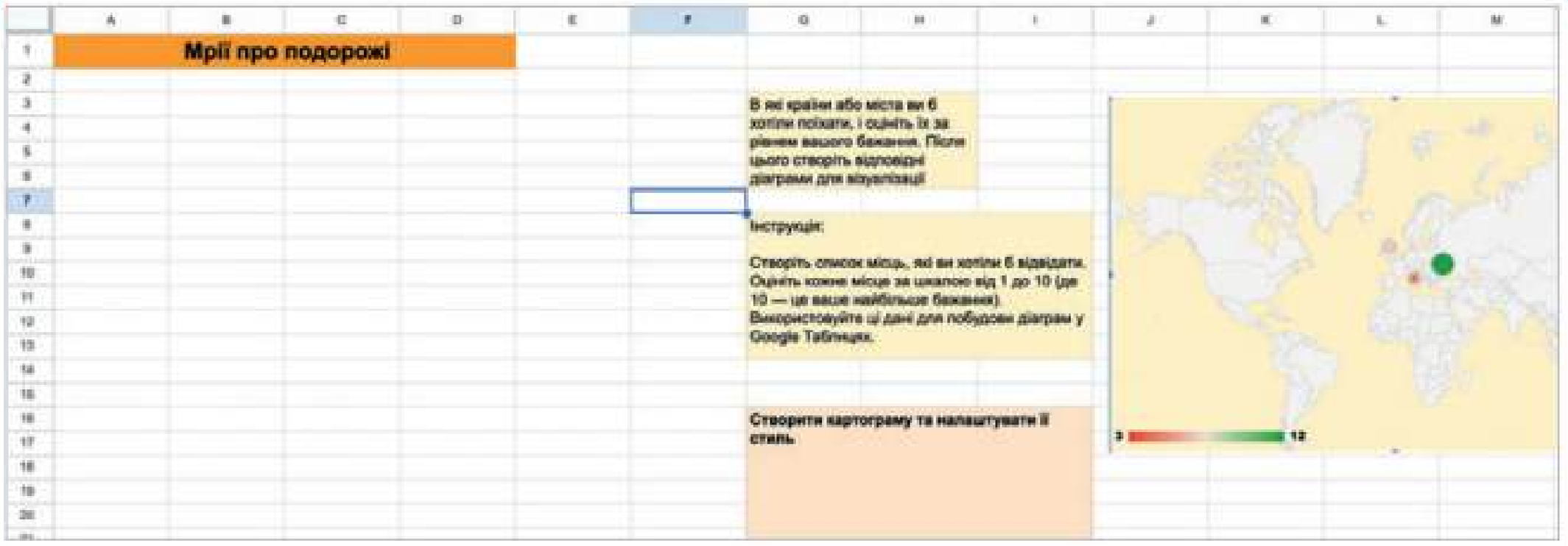


Завдання №5

В які країни або міста ви б хотіли поїхати? Оцініть їх за рівнем вашого бажання. Після цього створіть відповідні діаграми для візуалізації даних.

Інструкція: оцініть кожне місце, яке ви б хотіли відвідати, за шкалою від 1 до 10 (де 10 — це ваше найбільше бажання). Використовуйте ці дані для побудови діаграм у **Google Таблицях**.

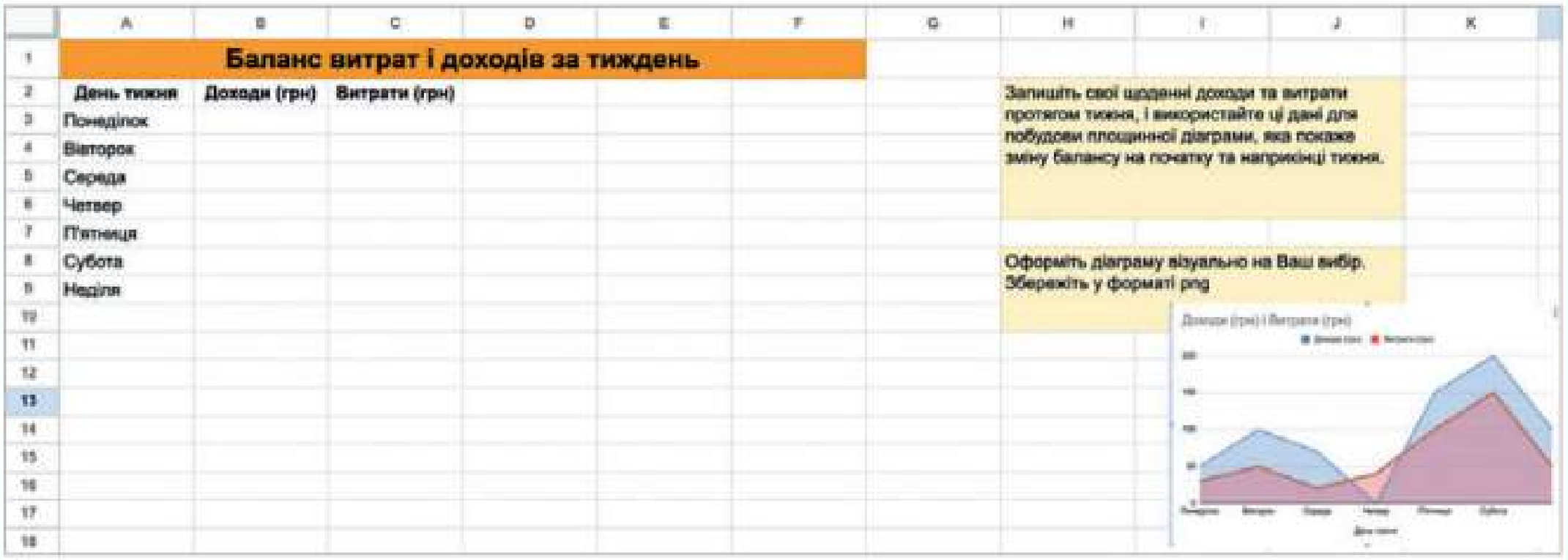
Створіть картограму та налаштуйте її стиль.



Завдання №6

Запишіть свої щоденні доходи та витрати протягом тижня і використайте ці дані для побудови площинної діаграми, яка покаже зміну балансу на початку та наприкінці тижня.

Оформте діаграму візуально на ваш вибір. Збережіть у форматі **png**.



Завдання №7

Визначте свою фінансову ціль (наприклад, купівля смартфона за 10 000 грн) і запишіть свої щотижневі доходи та витрати. Ваше завдання — накопичити необхідну суму за визначений час, враховуючи доходи та витрати.

Встановіть фінансову ціль: оберіть суму (наприклад, 10 000 грн), яку хочете накопичити на певну покупку. *Заплануйте щотижневі заощадження:* розрахуйте, скільки вам потрібно відкладати щотижня, щоб досягти своєї мети. *Записуйте реальні заощадження:* щотижня записуйте, скільки ви реально заощадили. *Оновлюйте прогрес:* обчисліть, скільки грошей вже накопичено і скільки ще потрібно для досягнення мети. *Аналізуйте прогрес:* порівняйте свої реальні заощадження з планом. Якщо відстаєте, подумайте, як можна скоротити витрати або збільшити доходи.



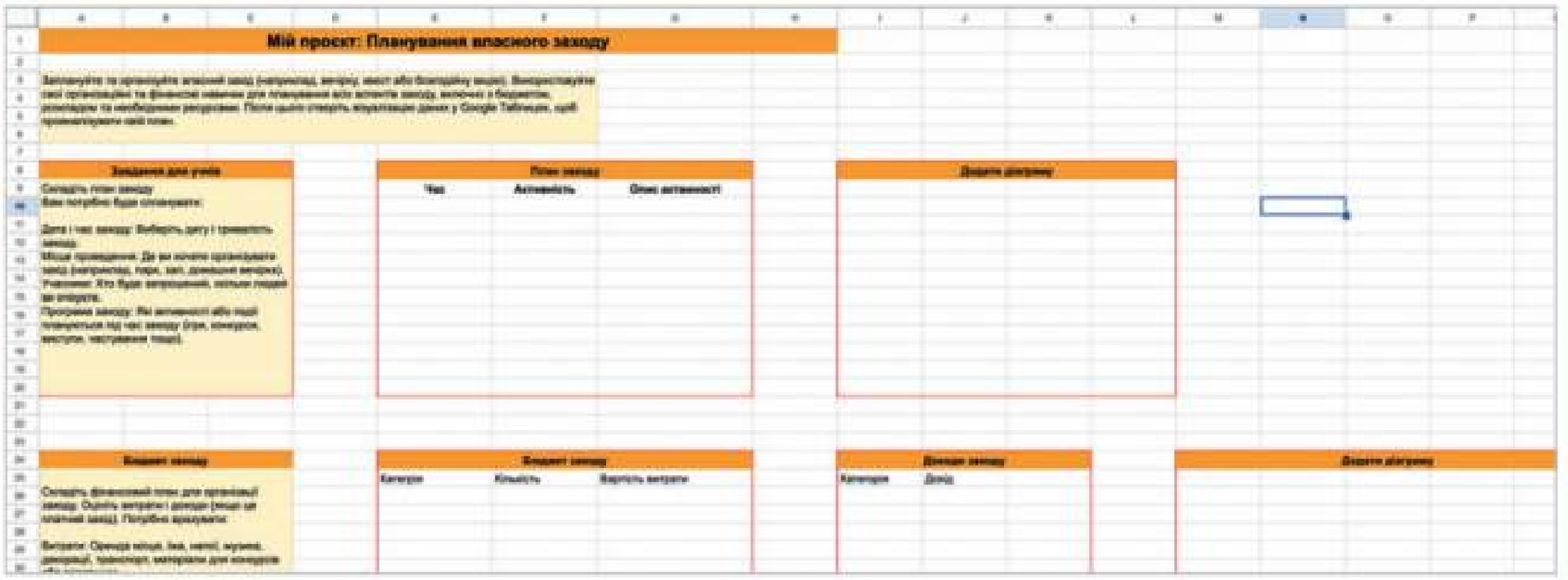
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	**Фінансова ціль та план її досягнення**														
2	Визначте свою фінансову ціль (наприклад, купівля смартфона за 10 000 грн) і запишіть щотижневі доходи та витрати. Ваше завдання — накопичити необхідну суму за визначений час, враховуючи свої доходи та витрати.					**Тиждень**	**План заощаджень (грн)**	**Реальні заощадження (грн)**	**Накопичено (грн)**	**Залишилось до цілі (грн)**			**Діаграма**		
3						Тиждень 1									
4						Тиждень 2									
5						Тиждень 3									
6						Тиждень 4									
7	**Завдання для учня:**					Тиждень 5									
8	Встановіть фінансову ціль. Оберіть суму (наприклад, 10 000 грн), яку хочете накопичити на певну покупку.					Тиждень 6									
9	Заплануйте щотижневі заощадження. Розрахуйте, скільки вам потрібно відкладати кожен тиждень, щоб досягти своєї цілі.														
10	Запишіть реальні заощадження. Кожного тижня записуйте, скільки ви реально заощадили.														
11	*Оновлюйте прогрес.* Обчисліть, скільки грошей вам накопичено і скільки ще потрібно для досягнення мети.														
12	*Аналізуйте прогрес.* Порівняйте свої реальні заощадження з планом. Якщо ви відстаєте, подумайте, як можна скоротити витрати або збільшити доходи.														
13															
14															
15															
16															
17															
18															

Завдання №8

Заплануйте та організуйте власний захід (наприклад, вечірку, квест або благодійну акцію). Використовуйте свої організаційні та фінансові навички для планування всіх аспектів заходу, включно з бюджетом, розкладом та необхідними ресурсами. Після цього створіть візуалізацію даних у **Google Таблицях**, щоб проаналізувати свій план.

Складіть план заходу. *Вам потрібно буде спланувати:* дата і час заходу. Виберіть дату і тривалість заходу. *Місце проведення:* де ви хочете організувати захід (наприклад, парк, зал, домашня вечірка). *Учасники:* хто буде запрошений, скільки людей ви очікуєте. *Програма заходу:* які активності або події плануються під час заходу (ігри, конкурси, виступи, частування тощо).

Складіть фінансовий план для організації заходу. Оцініть витрати і доходи (якщо це платний захід). *Потрібно врахувати:* витрати — оренда місця, їжа, напої, музика, декорації, транспорт, матеріали для конкурсів або подарунки; доходи *(Чи плануєте збирати внески від учасників або заробити гроші через благодійні збори?)*

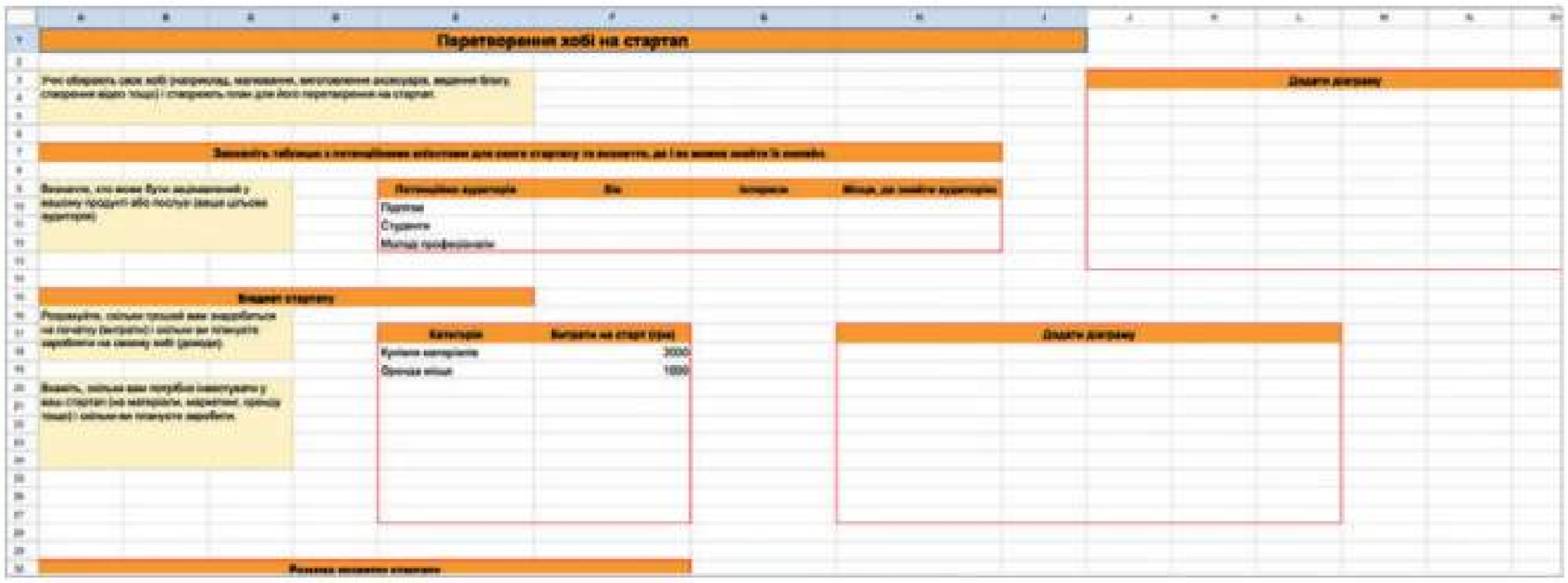


	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	**Мій проєкт: Планування власного заходу**															
2	Заплануйте та організуйте власний захід (наприклад, вечірку, квест або благодійну акцію). Використовуйте свої організаційні та фінансові навички для планування всіх аспектів заходу, включно з бюджетом, розкладом та необхідними ресурсами. Після цього створіть візуалізацію даних у Google Таблицях, щоб проаналізувати свій план.															
3																
4																
5																
6																
7																
8	**Завдання для учня**					**План заходу**				**Діаграма доходу**						
9	Складіть план заходу					**Час**	**Активність**	**Опис активності**								
10	Вам потрібно буде спланувати:															
11	Дата і час заходу: Виберіть дату і тривалість заходу.															
12	Місце проведення: Де ви хочете організувати захід (наприклад, парк, зал, домашня вечірка).															
13	Учасники: Хто буде запрошений, скільки людей ви очікуєте.															
14	Програма заходу: Які активності або події плануються під час заходу (ігри, конкурси, виступи, частування тощо).															
15																
16																
17																
18																
19	**Бюджет заходу**					**Бюджет заходу**				**Доходи заходу**			**Діаграма витрат**			
20						**Категорія**	**Кількість**	**Вартість витрати**		**Категорія**	**Дохід**					
21	Складіть фінансовий план для організації заходу. Оцініть витрати і доходи (якщо це платний захід). Потрібно врахувати:															
22	Витрати: Оренда місця, їжа, напої, музика, декорації, транспорт, матеріали для конкурсів															

Завдання №9

Оберіть якесь хобі (наприклад, малювання, виготовлення аксесуарів, ведення блогу, створення відео тощо) і створіть план для його перетворення на стартап.

- Визначте, хто може бути зацікавлений у вашому продукті або послузі (ваша цільова аудиторія).
- Розрахуйте, скільки вам знадобиться на початку (витрати) і скільки плануєте заробляти на своєму хобі (доходи).
- Вкажіть, скільки вам потрібно інвестувати у ваш стартап (на матеріали, маркетинг, оренду тощо) і скільки плануєте заробити.



Колективна робота

Планування подорожі та організація розваг

Ваша команда повинна спланувати подорож для групи друзів (або класу) на кілька днів, враховуючи витрати на транспорт, проживання, їжу та розваги. Завдання передбачає складання бюджету подорожі та візуалізацію витрат за допомогою діаграми в **Google Таблицях.**

Організаційні моменти

- Працюємо в **Google Таблиці** (надаємо доступ до колективної таблиці).
- В групі 3–4 учні.
- Подорож обговорюємо колективно.
- Розподіл обов'язків робимо колективно без педагога.

Частина 1

Вибір місця подорожі:

- Визначте місце, куди ви хочете поїхати з друзями. Це може бути:
  - Подорож до іншого міста (наприклад, поїздка до Києва, Львова або Одеси).
  - Відпочинок на природі (наприклад, похід у гори чи кемпінг біля озера).
  - Відвідування культурних чи історичних пам'яток.
- Визначте кількість днів, які ви плануєте провести у подорожі (наприклад, 3–5 днів).

Частина 2

Розробка бюджету подорожі:

- Визначте основні категорії витрат, які потрібно врахувати під час подорожі:
  - Транспорт (поїзд, автобус, бензин, якщо ви подорожуєте на машині).
  - Проживання (готель або кемпінг).
  - Їжа (витрати на харчування у ресторанах або закупівля продуктів).
  - Розваги та екскурсії (квитки в музеї, парки розваг, спортивні активності тощо).
- Визначте, скільки грошей потрібно для кожної категорії, та запишіть ці дані у таблицю.

Частина 3

Створення діаграм:

- Після того, як заповнили таблицю з витратами, вам потрібно створити діаграму для візуалізації даних:
  - *Кругова діаграма:* покажіть, яка частина бюджету витрачається на кожну категорію (транспорт, проживання, їжа, розваги).
  - *Стовпчаста діаграма:* якщо ви порівнюєте витрати на кілька днів подорожі або на різні види розваг, створіть стовпчасту діаграму, щоб побачити, як розподіляються витрати між днями або категоріями.

Частина 4

Складання плану подорожі:

- Визначте розклад подорожі: що ви будете робити кожного дня, які місця відвідувати і як плануєте витрачати час.
- Запишіть основні активності на кожен день у вигляді таблиці.

Частина 5

Презентація плану подорожі:

- Підготуйте коротку презентацію для класу, де ви представите свій план подорожі:
  - Розкажіть, куди хочете поїхати, що плануєте робити кожного дня.
  - Представте бюджет подорожі та поясніть, скільки грошей піде на кожен її аспект.
  - Продемонструйте діаграму, яка візуалізує витрати подорожі.

Теми 12–13 **Текстові функції. Математичні та логічні функції. Статистичні функції. Фінансові функції. Процес дослідження. Проблема та мета дослідження. Гіпотеза дослідження**

Ключові поняття

Функція
Процентна ставка
Щомісячний платіж
Термін
Амортизація
Гіпотеза

Функція

Функція — це спеціальна формула, яка виконує обчислення або операції на даних у клітинках таблиці.

Функції допомагають швидко здійснювати різні розрахунки, опрацьовувати текст, виконувати логічні перевірки або аналізувати статистичні дані.

Функції використовуються для автоматизації процесів у таблицях. Наприклад, ви можете підрахувати суму чисел у діапазоні клітинок, порівняти значення, об'єднати текст, зробити фінансові розрахунки або статистичний аналіз.

Текстові функції — Це функції, які працюють із текстовими даними. Вони дозволяють об'єднувати текст, змінювати регістр букв, витягувати частини тексту або рахувати кількість символів у тексті.

Математичні функції — Використовуються для виконання арифметичних розрахунків у таблицях.

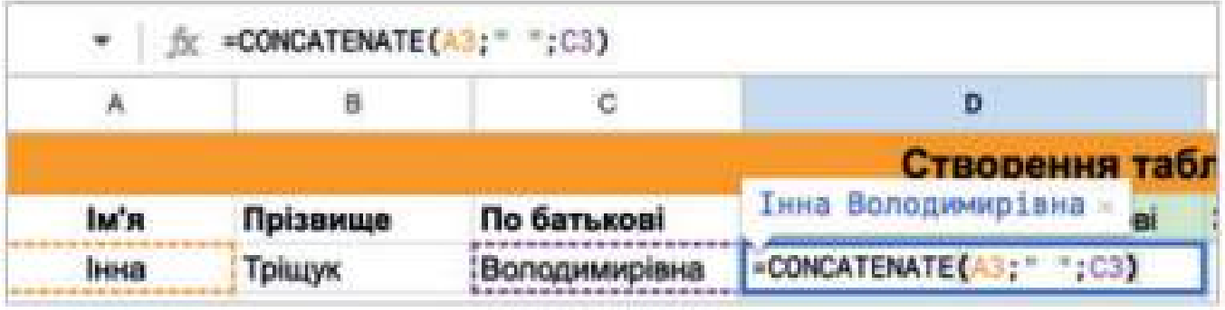
Логічні функції — Перевіряють істинність або хибність певної умови та дозволяють виконати дії залежно від результату перевірки.

Статистичні функції — Використовуються для аналізу та обчислення статистичних даних, таких як кількість значень у діапазоні, мінімум і максимум.

Фінансові функції — Допомагають розраховувати фінансові показники, такі як виплати за кредитом, майбутню вартість інвестицій або чисту приведену вартість інвестицій.

Основні текстові функції

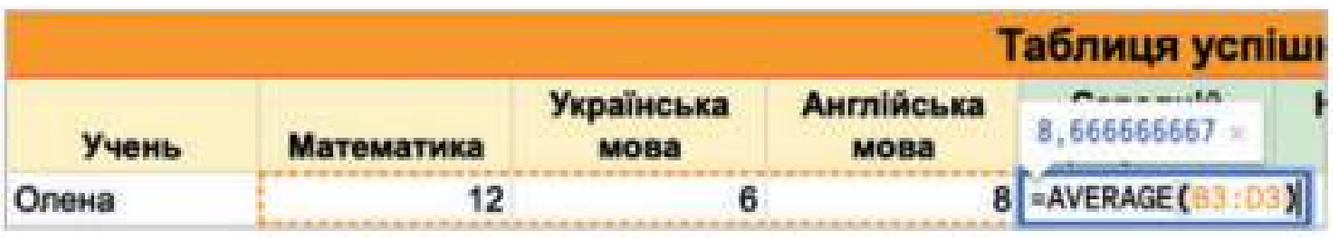
CONCATENATE	об'єднує кілька текстових рядків в один
UPPER	перетворює текст у верхній регістр
LOWER	перетворює текст у нижній регістр
LEN	рахує кількість символів у тексті

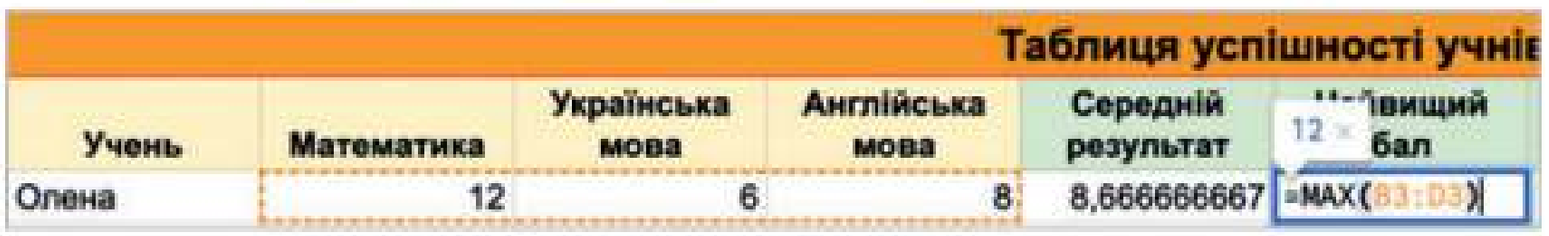




Математичні функції

SUM	підсумовує числа в діапазоні клітинок
AVERAGE	обчислює середнє арифметичне чисел у діапазоні
MIN і MAX	знаходить мінімальне і максимальне значення в діапазоні клітинок
ROUND	округлює число до заданої кількості десяткових знаків
POWER	підносить число до степеня
SQRT	обчислює квадратний корінь числа







Логічні функції

IF	перевіряє умову і повертає одне значення, якщо умова істинна, й інше, якщо хибна
AND	повертає істину, якщо всі умови істинні, і хибність, якщо хоча б одна умова хибна
OR	повертає істину, якщо хоча б одна умова істинна
NOT	повертає істину, якщо умова хибна, і хибність, якщо умова істинна
IFERROR	повертає задане значення у випадку помилки

Таблиця успішності учнів

Учень	Математика	Українська мова	Англійська мова	Середній результат	Найвищий бал	Найнижчий бал
Олена	12	6	8	8,666666667	12	=MIN(B3:D3)

Фінансовий аналіз підприємства

Місяць	Доходи (грн)	Витрати (грн)	Чистий прибуток (грн)	Прибутковий місяць (так/ні)	Збитковий місяць (так/ні)
ень	99000	80 000,00	19 000,00	=IF(D3>0;"Так";"Ні")	
ютий	98000	95 000,00			
резень	12000	85 000,00			

Оцінка успішності учнів з використанням логічних ф

Учень	Математика	Фізика	Хімія	Перевірка результатів за допомогою функції IF	Перевірка успішності за всіма предметами за допомогою функції AND	Перевірка успі предметом за
Олена	85	78	90		=IF(AND(B3>60;C3>60;D3>60);"Склав усі";"Не склав усі")	

Статичні функції

COUNT	рахує кількість клітинок із числовими значеннями
MODE	визначає моду (найчастіше буває значення у вибірці)
MEDIAN	обчислює медіану (середнє значення вибірки, яке ділить дані на дві рівні частини)
STDEV	обчислює стандартне відхилення, що показує, наскільки значення у вибірці відхиляються від середнього значення

Обчисліть середній результат бігу спортсменів за допомогою функції AVERAGE

```
=AVERAGE(B3:B10)
```

Знайдіть медіану результатів за допомогою функції MEDIAN

```
=MEDIAN(B3:B10)
```

Визначте найбільш поширений результат за допомогою функції MODE

```
=MODE(B3:B10)
```

Обчисліть стандартне відхилення результатів за допомогою функції STDEV.

```
=STDEV(B3:B10)
```

Знайдіть найкращий результат (мінімальне значення) за допомогою функції MIN.

```
=MIN(B3:B10)
```

Знайдіть найгірший результат (максимальне значення) за допомогою функції MAX.

```
=MAX(B3:B10)
```

Ключові терміни та пояснення

Процентна ставка

Це відсоток, який треба платити за використання грошей (наприклад, кредиту), або відсоток, який ви отримуєте за інвестицію. Якщо ви берете кредит під 5% річних, це означає: щороку повинні сплачувати 5% від суми, яку позичили.

Щомісячний платіж

Це сума, яку потрібно сплачувати кожен місяць за кредитом. Вона включає в себе частину основної суми (гроші, які ви позичили) і відсотки.



Термін

Це кількість місяців або років, протягом яких ви сплачуєте кредит чи інвестуєте гроші. Наприклад, термін може бути 36 місяців (3 роки) або 5 років.



Основна сума

Це початкова сума грошей, яку ви взяли в кредит або інвестували. Наприклад, якщо ви взяли кредит на 10 000 грн, це і є основна сума.

Відсотки

Це гроші, які ви повинні сплатити за використання позичених грошей. Відсотки додаються до основної суми кредиту. Наприклад, якщо процентна ставка становить 5%, то за рік ви будете сплачувати 5% від суми кредиту як відсотки.

Амортизація

Це процес поступового погашення кредиту через регулярні платежі. Кожен платіж включає частину основної суми і відсотки.

Фінансові функції

PMT	обчислює щомісячний платіж за позикою на основі постійних платежів і постійної процентної ставки	Приклад: =PMT(6%/12, 36, -10000) – обчислює щомісячний платіж за позикою у 10 000 грн на 36 місяців під 6% річних.
CUMIPMT	обчислює загальну суму відсотків, що виплачуються за конкретний період для позики	Приклад: =CUMIPMT(6%/12, 36, 10000, 1, 12, 0) – обчислює суму відсотків, сплачених за перший рік для позики у 10 000 грн на 36 місяців під 6% річних.
PV	обчислює теперішню вартість інвестиції або позики на основі майбутніх виплат і процентної ставки	Приклад: =PV(5%/12, 60, -200) – обчислює теперішню вартість позики з щомісячним платежем 200 грн на 60 місяців під 5% річних.
FV	обчислює майбутню вартість інвестиції на основі регулярних внесків, процентної ставки та кількості періодів	Приклад: =FV(5%/12, 12*5, -500, 0) – обчислює майбутню вартість інвестиції з регулярними внесками 500 грн на 5 років під 5% річних.

Показник	Значення
Сума кредиту (грн)	10000
Процентна ставка (річна)	6%
Термін (місяці)	36
Щомісячний платіж (грн)	=PMT(B4/12;B5;B3)

Показник	Значення
Сума кредиту (грн)	10000
Процентна ставка (річна)	6%
Термін (місяці)	36
Щомісячний платіж (грн)	-304,22 грн.
Загальна сума виплат (грн)	=B6*B5

Показник	Значення	
Сума кредиту (грн)	10000	
Процентна ставка (річна)	6%	
Термін (місяці)	36	
Щомісячний платіж (грн)	-304,22 грн.	Обчисл Викори Формул
Загальна сума виплат (грн)	-10 951,90 грн.	Обчисл Загалы Формул
Сума виплачених відсотків (грн)	=CUMIPMT(B4/12;B5;B3;1;B5;0)	

Показник	Значення
Сума кредиту (грн)	10000
Процентна ставка (річна)	6%
Термін (місяці)	36
Порахуйте теперішню вартість позики за допомогою функції PV	=PV(B4/12;B5;-B6)

PMT (Платіж за кредитом)

Функція PMT допомагає розрахувати, скільки потрібно платити кожен місяць за кредитом, враховуючи відсотки.

Функція PMT розраховує щомісячний платіж, який вам потрібно буде сплачувати протягом 36 місяців, щоб повністю погасити кредит у розмірі 10 000 грн під 6% річних.

Як працює формула:

6%/12 – це місячна процентна ставка (0,5% на місяць).

36 – це кількість місяців, протягом яких ви будете погашати кредит (3 роки).

-10 000 – це сума кредиту (взята в борг), де знак "-" показує, що це гроші, які потрібно повернути.

Показник	Значення
Сума кредиту (грн)	10000
Процентна ставка (річна)	6%
Термін (місяці)	36
Щомісячний платіж (грн)	=PMT(B4/12;B5;B3)

FV (Майбутня вартість)

Функція FV використовується для обчислення того, скільки грошей ви отримаєте в майбутньому, якщо будете регулярно інвестувати певну суму.

Функція FV у цьому прикладі обчислює майбутню вартість інвестиції, якщо ви інвестуєте 500 грн щомісяця протягом 5 років під річну процентну ставку 5%.

Вона покаже, скільки грошей у вас буде на рахунку через 5 років.

Як працює формула:
5%/12 – місячна процентна ставка.
12*5 – кількість місяців (5 років = 60 місяців).
-500 – регулярний щомісячний внесок (від'ємний, бо це витрати).
0 – початкова сума інвестиції.

Показник	Значення
Сума інвестиції (грн)	500
Процентна ставка (річна)	5%
Кількість років	5
Майбутня вартість (грн)	=FV(B17/12;12*B18;-B16;0)

PV (Теперішня вартість)

Функція PV допомагає обчислити теперішню вартість майбутніх платежів або кредиту.

У фінансових функціях у таблицях завжди вважається, що гроші, які отримуєте (доходи), мають позитивне значення, а гроші, які витрачаєте (платежі), мають негативне значення.

У нашому прикладі:

6%/12 – це місячна процентна ставка (якщо 6% – це річна ставка).

36 – це кількість місяців (термін кредиту).

-B6 – це щомісячний платіж, який ви сплачуєте за кредит (зазначений у клітинці B6). Оскільки це ваші витрати, сума введена з мінусом.

Порахуйте теперішню вартість позики за допомогою функції PV

=PV(B4/12;B5;-B6)

Показник	Значення
Сума кредиту (грн)	10000
Процентна ставка (річна)	6%
Термін (місяці)	36

Функція CUMIPMT (Cumulative Interest Payment) використовується для обчислення загальної суми відсотків, виплачених за певний період для позики. Вона дозволяє дізнатися, скільки відсотків ви сплатите за весь термін кредиту або за певний проміжок часу.

Функція CUMIPMT обчислює загальну суму відсотків, які ви заплатите банку за весь період кредитування (36 місяців) за кредит у розмірі 10 000 грн під річну процентну ставку 6%.

Як працює формула:

6%/12 – місячна процентна ставка (0,5% на місяць).

36 – кількість місяців (термін кредиту).

10 000 грн – це сума кредиту.

1 – перший місяць, з якого починається розрахунок відсотків.

36 – останній місяць, до якого проводиться розрахунок відсотків (весь термін).

0 – платежі здійснюються в кінці кожного місяця.

Показник	Значення
Сума кредиту (грн)	10000
Процентна ставка (річна)	6%
Термін (місяці)	36
Щомісячний платіж (грн)	-304,22 грн.
Загальна сума виплат (грн)	-10 951,90 грн.
Сума виплачених відсотків (грн)	=CUMIPMT(B4/12;B5;B3;1;B5;0)

Дослідження — процес систематичного збирання, аналізу та інтерпретації інформації для розв'язання певної проблеми або питання.

Етапи дослідження



Гіпотеза

Гіпотеза дослідження припущення, яке дослідник висуває на початку дослідження.

Гіпотеза — це передбачуваний результат, який дослідник перевіряє протягом дослідження. Вона може бути підтверджена або спростована на основі даних та аналізу.



Ціль дослідження — конкретний результат, який ви хочете отримати під час його виконання. Цілі допомагають зрозуміти, як саме ви збираєтесь досліджувати проблему та які аспекти вас цікавлять.

Головна ціль – це загальна мета, якої дослідник намагається досягти.

Окремі завдання – це конкретні етапи, які допомагають досягти головної мети.



Розуміння ролей у процесі дослідження

Практична робота №11.

Відкрити **Google Таблицю** (скопіювати для себе).

Частина 1

Завдання №1. Заповніть таблицю, використовуючи текстову функцію CONCATENATE
Не забуваємо додавати пропуски.



	A	B	C	D	E	F	G
1	Створення таблиці з текстовими даними						
2	Ім'я	Прізвище	По батькові	З'єднати ім'я та по батькові	З'єднати ім'я та прізвище	З'єднати прізвище та ім'я	З'єднати прізвище, ім'я та по батькові
3	Інна	Тріщук	Володимирівна				
4	Олександр	Лазарець	Юрійович				
5	Марина	Корпан	Максимівна				
6	Оксана	Трубілко	Миколаївна				
7	Михайло	Назарчук	Сергіївна				
8	Ярослав	Макарицький	Юрійович				
9							
10							
11	Заповнити дану таблицю використовуючи текстову функцію CONCATENATE						
12							

Завдання №2. Використайте функції UPPER та LOWER, щоб змінити регістр тексту. Перетворіть ім'я на великий регістр, а прізвище — на малий. Формула для імені: =UPPER(A2) Формула для прізвища: =LOWER(B2)

Зміна регістру тексту				
Ім'я	Прізвище	Ім'я	Прізвище	
Інна	Тріщук	ІННА		
Олександр	Лазарець			
Марина	Корпан			
Оксана	Трубілко			
Михайло	Назарчук			
Ярослав	Макарицький			

Завдання №3. Використайте функцію LEN, щоб підрахувати кількість символів у прізвищі кожної людини. Формула: =LEN(B2)

Підрахунок кількості символів				
Прізвище	кількість символів			
Тріщук				
Лазарець				
Корпан				
Трубілко				
Назарчук				
Макарицький				

Завдання №4. Навчіться об'єднувати текстові рядки для створення персоналізованих повідомлень або листів.

				Підрахунок кількості символів
Прізвище	кількість символів			
Тріщук				
Лазарець				
Корпан				
Трубілко				
Назарчук				
Макарицький				

Завдання №5. Доповніть таблицю! Навчіться використовувати текстові функції для створення індивідуальних запрошень на захід (наприклад, день народження, випускний, спортивне змагання).

Персоналізовані запрошення на захід				
Ім'я	**Прізвище**	**Захід**	**Дата**	**Місце**
Олена	Іваненко	День народження	12.10.2024	Кафе "Квітка"
Максим	Петров	Спортивні змагання	01.11.2024	Спортзал школи №12

	Місце для персоналізованих запрошень
Олена	Іваненко
Максим	Петров

Завдання №6. Навчіться створювати автоматичні нагадування для учнів або колег про важливі завдання або дедлайни.

				Автоматичне нагадування про завдання
Ім'я	**Завдання**	**Дедлайн**	**Email**	Доповнити таблицю
Олена	Написати статтю	15.10.2024	olena@example.com	
Максим	Підготувати звіт	20.10.2024	maksym@example.com	

	Місце для персоналізованих запрошень
Олена	
Максим	

Частина 2 — математичні функції

Завдання №1. Таблиця успішності учнів

1. Заповніть таблицю.

2. Визначте за допомогою функції середній бал, найвищий бал та найнижчий бал.

3. Побудуйте діаграму на ваш вибір.

Таблиця успішності учнів

Учень	Математика	Українська мова	Англійська мова	Середній результат	Найвищий бал	Найнижчий бал
Олена	12					
Максим	6					
Анастасія	8					

1) Заповнити таблицю
2) Визначте за допомогою функції середній бал, найвищий бал та найнижчий бал.
3) Подубуйте діаграму на Ваш вибір

Завдання №2. Проаналізуйте доходи та витрати сім'ї за п'ять місяців, обчисліть заощадження за кожен місяць і підсумуйте загальні накопичення за рік — ви навчитеся використовувати математичні функції у **Google Таблицях** для виконання фінансових розрахунків і планування бюджету.

Фінансове планування сім'ї

Частина 1: Доходи
Створіть таблицю з доходами сім'ї за місяць.

Місяць	Зарплата (грн)	Фріланс (грн)	Інші доходи (грн)	Загальний дохід (грн)
Січень	25 000	5 000	2 000	
Лютий				
Березень				
Квітень				
Травень				

Обчисліть загальний дохід кожного місяця.
Використайте функцію SUM, щоб підсумувати всі джерела доходів кожного місяця.

Обчисліть середній місячний дохід.
Використайте функцію AVERAGE для підрахунку середнього доходу за місяць.

Знайдіть максимальний дохід за місяць.
Використайте функцію MAX, щоб знайти місяць з найбільшим доходом.

Частина 2: Витрати
Створіть таблицю з витратами сім'ї за місяць.

Місяць	Продукти (грн)	Оренда (грн)	Комунальні послуги (грн)	Інші витрати (грн)	Загальні витрати (грн)
Січень	7 000	8 000	3 000	2 000	
Лютий					
Березень					
Квітень					
Травень					

Обчисліть загальні витрати кожного місяця.
Використайте функцію SUM, щоб підсумувати всі витрати кожного місяця.

Обчисліть середні місячні витрати.
Використайте функцію AVERAGE для підрахунку середніх витрат за місяць.

Знайдіть максимальні витрати за місяць.
Використайте функцію MAX, щоб знайти місяць з найбільшими витратами.

Частина 3 — логічні функції

Завдання №1. Використайте логічні функції для перевірки умов та прийняття рішень.

- Заповніть таблицю з оцінками учнів.

- Введіть дані про оцінки за три предмети для кожного учня.
- Використайте функцію **IF**, щоб визначити, чи склав учень іспити (середня оцінка понад 60).
- Використайте функцію **AND**, щоб перевірити, чи склав учень усі іспити.
- Використайте функцію **OR**, щоб визначити, чи склав учень хоча б один іспит.
- Використайте функцію **IFERROR**, щоб обробляти можливі помилки у розрахунках.

Оцінка успішності учнів з використанням логічних функцій							
Учень	Математика	Фізика	Хімія	Перевірка результатів за допомогою функції IF	Перевірка успішності за всіма предметами за допомогою функції AND	Перевірка успішності хоча б за одним предметом за допомогою функції OR	Обробка помилок за допомогою функції IFERROR
Олена	65	78	90				
Максим	45	65	55				
Анастасія	82	89	88				
Дмитро	70	50	40				
				Перевірка результатів за допомогою функції IF: Ви повинні перевірити, чи склав учень іспит. Якщо середня оцінка за три предмети більша 60, результат буде "Склав", якщо менше – "Не склав". Формула для стовпця "Підсумок": =IF(AVERAGE(B2:D2)>60, "Склав", "Не склав").	Використайте функцію AND, щоб перевірити, чи склав учень іспити з усіх предметів (усі оцінки більше 60). Якщо учень набрав більше 60 за кожен предмет, результат буде "Склав усі". Формула: =IF(AND(B2>60, C2>60, D2>60), "Склав усі", "Не склав усі").	Перевірка успішності хоча б за одним предметом за допомогою функції OR: Використайте функцію OR, щоб перевірити, чи склав учень хоча б один іспит (хоча б одна оцінка більше 60). Формула: =IF(OR(B2>60, C2>60, D2>60), "Склав один", "Не склав жоден").	При обчисленні середнього результату може виникнути помилка, наприклад, якщо одна з оцінок є нульовою або якщо є ділення на нуль. Використайте функцію IFERROR, щоб замість помилки показати "Помилка у даних". Формула: =IFERROR(AVERAGE(B2:D2), "Помилка у даних").

Завдання №2. Навчіться використовувати логічні функції для фінансового аналізу підприємства.

- Заповніть таблицю, внісши дані про доходи та витрати підприємства за три місяці.
- Введіть дані про доходи та витрати для кожного місяця.
- Використайте функцію **SUM**, щоб розрахувати чистий прибуток за кожен місяць.
- Перевірте прибутковість кожного місяця за допомогою функції **IF**, визначте збиткові місяці та оцініть фінансову стабільність підприємства.

Фінансовий аналіз підприємства					
Місяць	Доходи (грн)	Витрати (грн)	Чистий прибуток (грн)	Прибутковий місяць (так/ні)	Збитковий місяць (так/ні)
Січень	100 000	80 000			
Лютий	90 000	95 000			
Березень	120 000	85 000			
			Обчисліть чистий прибуток за кожен місяць. Чистий прибуток = Доходи - Витрати. Формула: =B2-C2.	Перевірте, чи є місяць прибутковим за допомогою функції IF: Використайте функцію IF, щоб перевірити, чи є чистий прибуток більшим за 0. Якщо так, результат "Так", якщо ні – "Ні". Формула: =IF(D2>0, "Так", "Ні").	Визначте, чи є місяць збитковим за допомогою функції IF: Використайте функцію IF, щоб перевірити, чи є місяць збитковим (чистий прибуток менший за 0). Якщо так, результат "Так", якщо ні – "Ні". Формула: =IF(D2<0, "Так", "Ні").

Частина 4 — статичні функції

Завдання №1. Навчіться використовувати статистичні функції для аналізу спортивних результатів, обчислення середнього результату, медіани, стандартного відхилення та визначення найкращого спортсмена.

Аналіз результатів спортивних змагань							
Спортсмен	Результат (секунди)	Обчисліть середній результат бігу спортсменів за допомогою функції AVERAGE	Знайдіть медіану результатів за допомогою функції MEDIAN	Визначте найбільш поширений результат за допомогою функції MODE	Обчисліть стандартне відхилення результатів за допомогою функції STDEV.	Знайдіть найкращий результат (мінімальне значення) за допомогою функції MIN.	Знайдіть найгірший результат (максимальне значення) за допомогою функції MAX.
Олена	12.5						
Максим	11.8						
Анастасія	13.2						
Дмитро	10.9						
Олексій	11.5						
Софія	12.8						
Іван	13.0						
Наталія	11.2						
Зробіть кілька діаграм на Ваш вибір							

Завдання №2. Навчіться аналізувати дані про улюблені предмети своїх однокласників, знаходити найпопулярніший предмет, середню кількість голосів та інші статистичні показники.

Аналіз опитування учнів щодо улюблених предметів							
Предмет	Кількість голосів	Обчисліть середню кількість голосів за допомогою функції AVERAGE.	Знайдіть медіану кількості голосів за допомогою функції MEDIAN	Визначте найпопулярніший предмет за допомогою функції MODE.	Порахуйте загальну кількість учнів, які взяли участь в опитуванні, за допомогою функції SUM.	Знайдіть предмет з найбільшою кількістю голосів за допомогою функції MAX.	Знайдіть предмет з найменшою кількістю голосів за допомогою функції MIN.
Математика	15						
Фізика	12						
Історія	18						
Біологія	10						
Хімія	8						
Українська мова	20						
Література	17						
Література	18						
Зробіть кілька діаграм на Ваш вибір							

Частина 5 — фінансові функції

Завдання №1. Навчіться використовувати фінансові функції для обчислення щомісячних платежів за кредитом, процентних ставок, теперішньої та майбутньої вартості інвестицій.

Фінансові функції для обчислення майбутньої вартості інвестицій та аналізу різних фінансових показників на основі процентних ставок, регулярних внесків та терміну інвестиції.

Аналіз майбутньої вартості інвестиції		
Показник	Значення	
Сума інвестиції (грн)	500	
Процентна ставка (річна)	5%	
Кількість років	5	
Майбутня вартість (грн)		Обчисліть майбутню вартість інвестиції за допомогою функції FV. Використайте функцію FV, щоб обчислити майбутню вартість інвестиції через 5 років з регулярними внесками по 500 грн на місяць і річною процентною ставкою 5%. Формула для обчислення майбутньої вартості через 5 років: excel =FV(5%/12, 12*5, -500, 0) Ця формула обчислює, скільки грошей ви накопичите через 5 років, якщо щомісяця будете вкладати по 500 грн під 5% річних.
Майбутня вартість (через 10 років)		Змініть кількість років на 10 і обчисліть нову майбутню вартість інвестиції: Тепер ви повинні обчислити майбутню вартість, якщо ви будете інвестувати 10 років. Формула для обчислення майбутньої вартості через 10 років: excel =FV(5%/12, 12*10, -500, 0) Ця формула покаже, скільки грошей ви накопичите через 10 років за тих самих умов (щомісячний внесок 500 грн, річна процентна ставка 5%).

Колективна практична робота

1. Створення інвестиційного плану для досягнення фінансової мети

Організаційні моменти:

- Учні об'єднуються в групи по 3–4 особи.
- Створюють одну колективну **Google Таблицю** та надають доступ педагогу.
- Працюють колективно над одним планом фінансової мети.

Оберіть фінансову мету для інвестиції:

Кожна група обирає одну фінансову мету, яку вони хотіли б досягти через певний період часу. Наприклад:

- Відпочинок за кордоном через 5 років.
- Купівля нового ноутбука через 3 роки.
- Збирання коштів на благодійний проєкт через 2 роки.

2. Визначте початкові умови:

Кожна група повинна вирішити:

- Яку суму вони можуть вкладати щомісяця (наприклад, 500 грн).
- Процентну ставку, під яку можуть вкладати гроші (наприклад, 5% або 7% річних).
- Термін інвестиції (скільки років планують збирати гроші).

Зразок **Google Таблиці**:

Показник	Значення
Мета інвестиції	(Назва)
Сума щомісячного внеску (грн)	500
Процентна ставка (річна)	5%
Кількість років	5
Майбутня вартість інвестиції	

Обчисліть майбутню вартість інвестиції:

- Використайте функцію **FV** для обчислення майбутньої вартості інвестиції.

Ця формула допоможе зрозуміти, скільки грошей буде через 5 років за щомісячного внеску 500 грн під 5% річних.





Розділ 3

САЙТ ТА ЙОГО ВЕРСТКА

Тема 14 Поняття мови гіпертекстової розмітки. Тег та його атрибути. Валідація сторінок. Теги створення елементів вебсторінки: зображень, списків, таблиць

Ключові поняття

Основні компоненти вебсторінки
Редактори кодів
Мова гіпертекстової розмітки (HTML)
Теги та їхні атрибути
Теги для створення зображень, списків, таблиць
Валідація сторінок

Інтернет став невід'ємною частиною сучасного світу, відкриваючи безліч можливостей для комунікації, бізнесу, розваг та навчання.

І саме вебсайти є основними елементами інтернету, надаючи користувачам доступ до інформації та послуг.



Розглянемо основні поняття:

Вебсторінка

Вебсайт

Вебсторінка — це окремий документ, написаний мовою розмітки **HTML**, який призначений для перегляду електронної інформації. Вебсторінка може містити різні види контенту, такі як текст, зображення, аудіо- або відеоматеріали, а також гіперпосилання на інші вебсторінки.

Вебсайт — це набір взаємопов'язаних вебсторінок, які об'єднані спільною темою або метою і розміщені на одному вебсервері. Кожен вебсайт має унікальну адресу (**URL**), яка використовується для доступу до нього.

Основні компоненти вебсторінки



Майже все, що ви бачите онлайн – від блогів та інтернет-магазинів до соціальних мереж та інформаційних порталів, створено з використанням **HTML**, **CSS** та **JavaScript**.

HTML (HyperText Markup Language) — мова розмітки, яка використовується для створення структури і змісту вебсторінок.

CSS (Cascading Style Sheets) — мова стилів, яка використовується для оформлення зовнішнього вигляду вебсторінок.

JavaScript — мова програмування, яка додає інтерактивність до вебсторінок.



Розпочнемо знайомство з мови гіпертекстової розмітки **HTML**.

Вебсторінки зберігаються як текстові файли з розширенням **htm** або **html**. І такі файли відображають вміст у його відформатованому вигляді за допомогою браузера.

Для створення HTML-файлу можна використовувати текстовий редактор **Блокнот**. Проте для зручності написання використовують редактори кодів.

Розглянемо найпопулярніші з них:

Notepad ++ — розширена версія звичайного блокнота, в якому html-теги підсвічуються кольоровим шрифтом, і розробник може візуально відрізнити код від основного тексту сторінки.

VS Code є одним з найпопулярніших текстових редакторів серед розробників завдяки своїй функціональності, продуктивності та зручності.

Sublime Text 3 відомий своєю швидкістю і легкістю. Цей текстовий редактор має інтуїтивно зрозумілий інтерфейс, ефективні функції підсвічування синтаксису і множинного вибору, що робить процес редагування коду швидким і зручним.

Також html-код можна писати в онлайн-редакторах та на смартфонах.

HTML-конструювання — онлайн-сервіс, який містить в собі редактор коду та навчальну інформацію.

W3Schools — онлайн-платформа для вивчення вебтехнологій, яка містить редактор коду.

Web Code — мобільний додаток для написання та тестування вебкоду на смартфонах.

Потрібно зауважити, що онлайн-редактори та мобільні додатки можуть обмежувати функціональність.



Якщо уявити сайт у вигляді будинку, то HTML задає розташування кімнат та різних об'єктів, проте не займається оформленням.

HTML задає браузеру, де має бути заголовок, текст, малюнок, посилання тощо. Це каркас, на основі якого будуються всі сайти.

Розглянемо приклад простого html-файлу, в якому міститься текст про особу, що розповідає про себе.

Проте, відкривши такий файл у браузері, побачимо вебсторінку, в якій інформація буде розміщена інакше: увесь текст уміщено в одному абзаці, без відступів.

Щоб такого не було, саме HTML допомагає сайтам бути зрозумілими та зручними для користувачів, застосовуючи спеціальні команди.

```
<!DOCTYPE html>
<html lang="ukr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Назва веб-сторінки</title>
</head>
<body>
    Тут міститься контент

    Мене звати Микита. Мені 12 років
    Я хочу стати веб-розробником.

    Це моя перша сторінка. Я тут можу писати будь який те
        аудіо, відео, таблиці, списки, гіперпосилання та
    Я зможу розмістити даний файл на сервері, і мою сторі
        різних місцях світу.
</body>
</html>
```



Тут міститься контент Мене звати Микита. Мені 12 років Я хочу стати веб-розробником. Це моя перша сторінка. Я тут можу писати будь який текст. вставляти зображення. аудіо. відео. таблиці. списки. гіперпосилання та багато іншого. Я зможу розмістити даний файл на сервері. і мою сторінку зможуть побачити в різних місцях світу.

Теги та їхні атрибути

Тег HTML

Тег HTML — це спеціальний елемент, який використовується для визначення структури та вмісту вебсторінки.



Парні теги

Парні теги складаються з відкривального (`<тег>`) та закривального (`</тег>`) елементів. Між ними розташовується вміст — текст, зображення, посилання тощо.

```
<тег> вміст </тег>
```

Непарні теги

Деякі теги є поодинокими, або самозакривальними.

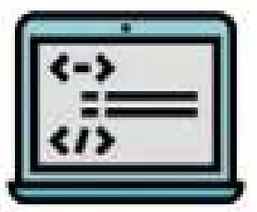
```
<тег/>
```

Також всередині тегів можна прописати **атрибути HTML**. Це щось на кшталт додаткових налаштувань для розширення можливостей.

Атрибути записують після назви тегу в такому вигляді: назва атрибуту, потім знак рівності та значення атрибуту, записане в лапках. Атрибутів для одного тегу може бути кілька.

```
<тег атрибут1="значення1" атрибут2="значення2"> вміст </тег>
```

```
<a href="https://google.com">Посилання</a>
    <img src="image.jpg" alt="Опис зображення">
    <p id="intro" class="text" style="color:red">Текст</p>
```



HTML-документ починається з елемента **<!DOCTYPE html>**, який визначає версію HTML.

Зауважимо, що **DOKTYPE** пишеться великими літерами, і це не є тег, а інструкція браузера про те, що використовується остання версія розмітки HTML5.

```
<!DOCTYPE html>
<html >
        <head>
                <title>Назва веб-сторінки</title>
        </head>
        <body>
                Тут міститься контент
        </body>
</html>
```

Початок і кінець документа визначається тегами **<html>** і **</html>**, які вказують браузеру, що це HTML-документ, і містять всередині два основні блоки: «голову» сайту — **head** і «тіло» сайту — **body**.

Блок **<head>...</head>** містить різну інформацію, яка не відображається в тілі документа, а саме: мета-дані про документ, такі як заголовок, посилання на стилі та скрипти, визначення кодування сторінки та інше.

Блок **<body>...</body>** містить весь видимий контент вебсторінки: текст, зображення, відео, посилання, форми та інші елементи.

Теги та атрибути блоку<Head>

Назва тегу	Опис
<title>Назва сторінки</title>	Вказує заголовок сторінки, який відображається на вкладці браузера
<meta>	Використовується для вказівки мета-даних, таких як кодування символів, опис сторінки тощо
<link>	Використовується для підключення зовнішніх ресурсів, таких як стильових файлів CSS.

```
<meta charset="UTF-8">
```

Атрибут **charset** задає кодування сторінки, щоб браузер міг відображати текст на сторінці. Тепер найпоширенішим є кодування UTF-8. Якщо буде обране неправильне кодування, то на сторінці замість тексту відображатимуться ієрогліфи.

```
<link rel="stylesheet" href="style.css">
```

Атрибут **rel** вказує на відношення між поточним документом і ресурсом, що підключається. Значення **stylesheet** означає, що підключається файл стилів.

Атрибут **href** вказує на шлях до файлу, який підключається. В даному випадку це файл з назвою **styles.css**, який повинен перебувати в тій же директорії, що і HTML-документ.

```
<link rel="icon" type="image/png" href="favicon.png">
```

Значення **icon** означає, що підключається фавікона — маленьке зображення, яке відображається поруч із заголовком сторінки на вкладці браузера.

Атрибут **type** вказує на тип файлу, який підключається. В даному випадку це PNG-зображення.

Теги та атрибути блоку <BODY>

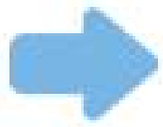
Як ви уже знаєте, тег **<body>** відповідає за видиму частину сторінки. Тому розглянемо основні теги та атрибути цього блоку.

Теги заголовків

Теги заголовків у HTML використовуються для визначення заголовків різних рівнів на вебсторінці. Вони грають важливу роль в організації контенту, допомагаючи як користувачам, так і пошуковим системам зрозуміти структуру сторінки. Заголовки використовуються для логічного поділу контенту на розділи та підрозділи.

В HTML існує шість рівнів заголовків: від **<h1>** до **<h6>**. Кожен наступний рівень заголовка має меншу важливість і розмір тексту.

```
<h1>Заголовок 1 рівня</h1>
    <h2>Заголовок 2 рівня</h2>
    <h3>Заголовок 3 рівня</h3>
    <h4>Заголовок 4 рівня</h4>
    <h5>Заголовок 5 рівня</h5>
    <h6>Заголовок 6 рівня</h6>
```





Теги абзацу та форматування тексту

Назва тегу	Опис
…	Визначає абзац (параграф) тексту. Це звичайний текст загального форматування
…	Виділяє текст курсивом (візуальний тег)
…	Виділяє текст курсивом (семантичний тег, означає наголос)
…	Виділяє текст жирним шрифтом (візуальний тег)
…	Виділяє текст жирним шрифтом (семантичний тег, означає важливість)
<u>…</u>	Підкреслює <u>текст</u>
<mark>…</mark>	Підсвічує <mark>текст</mark>
_…	Нижній _{індекс}
[…]	Верхній ^{індекс}
 	Вставляє розрив рядка там, де він розміщений у тексті
<hr>	Створює горизонтальну лінію на вебсторінці, яка використовується для візуального розділення контенту

Визначає абзац (параграф) тексту. Це звичайний текст загального форматування

Виділяє текст *курсивом (візуальний тег)*

Виділяє текст *курсивом (семантичний тег, означає наголос)*

Виділяє текст **жирним шрифтом (візуальний тег)**

Виділяє текст **жирним шрифтом (семантичний тег, означає важливість)**

Підкреслює текст

Підсвічує текст

Нижній $_{\text{індекс}}$

Верхній $^{\text{індекс}}$

Атрибути вирівнювання

```
<p align="left"> Текст зліва</p>
<p align="right"> Текст справа</p>
<p align="center"> Текст по центру</p>
<p align="justify"> Текст по ширині</p>
```

Текст зліва

Текст по центру

Текст справа

Текст по ширині

Тег для вставлення зображення

Тег **<img>** використовується в HTML для вставлення зображень на вебсторінку. Це самозакривальний тег, який не потребує закривального тегу.

```
<img src="image.jpg" alt="Опис зображення">
```

Основні атрибути

src="" — вказує шлях до файлу зображення або URL-адресу зображення в інтернеті.

alt="" — альтернативний текст для зображення, який відображається, якщо зображення не може бути завантажене.

Додаткові атрибути

width="" і height="" — встановлюють ширину та висоту зображення (у пікселях або відсотках).

title="" — текст підказки, який відображається при наведенні курсора на зображення.

```
<img src="nature.jpeg" alt="Природа" width="300"
height="200" title="Міст" align="center">
```



Теги для створення списків

Списки в HTML використовуються для організації і представлення елементів у структурованому вигляді. Існує три основні типи списків: марковані (невпорядковані), нумеровані (впорядковані) та описові (списки визначень).

- Щоб створити *маркований список*, використовують тег **
...
**. Для створення елемента списку — тег **
...
**.
- Щоб створити *нумерований список*, використовують тег **
...
**.

```
<ul>
  <li>Зима</li>
  <li>Весна</li>
  <li>Літо</li>
  <li>Осінь</li>
</ul>
```

- Зима
- Весна
- Літо
- Осінь

```
<ol>
  <li>Зима</li>
  <li>Весна</li>
  <li>Літо</li>
  <li>Осінь</li>
</ol>
```

1. Зима
2. Весна
3. Літо
4. Осінь

- Щоб створити *описовий список*, використовують тег **<dl>...</dl>**. Для задання терміна використовують тег **<dt>...</dt>**, а для визначення — тег **<dd>...</dd>**:

```
<dl>
  <dt>HTML</dt>
    <dd>Мова розмітки гіпертексту</dd>
    <dt>CSS</dt>
    <dd>Каскадні таблиці стилів</dd>
  </dl>
```

HTML
Мова розмітки гіпертексту
CSS
Каскадні таблиці стилів

Списки можуть бути вкладеними один в один, що дозволяє створювати складніші структури.

Атрибути списків

type="A" — визначає тип маркерів: для нумерованого списку (1, A, a, I, i) та маркованого (disc, circle, square);

start="5" — визначає початковий номер нумерованого списку;

reversed — визначає зворотний порядок нумерованого списку.

Теги для створення таблиць

Таблиці в HTML використовуються для організації даних у вигляді рядків і стовпців. Вони дозволяють структуровано представляти інформацію, таку як текстові дані, числові значення, зображення тощо.

Назва тегу	Опис
</table>... </table>	Визначає початок і кінець таблиці
</tr>...</tr>	Визначає рядок у таблиці
<td>...</td>	Визначає комірку таблиці
<th>...</th>	Визначає заголовок комірки. Зазвичай використовується для першого рядка або першого стовпця таблиці

```
<table>
        <tr>
            <th>Заголовок 1</th>
             <th>Заголовок 2</th>
            <th>Заголовок 3</th>
        </tr>
        <tr>
            <td>Комірка 1.1</td>
            <td>Комірка 1.2</td>
            <td>Комірка 1.3</td>
        </tr>
        <tr>
            <td>Комірка 2.1</td>
            <td>Комірка 2.2</td>
            <td>Комірка 2.3</td>
        </tr>
</table>
```

Атрибути таблиць

• Для тегу **<table>**:

border="1" — встановлює товщину рамки навколо таблиці та комірок

cellpadding="10" — визначає відступ всередині комірок

cellspacing="5" — визначає відстань між комірками

• Для тегів **<td>, <th>**:

colspan="2" — об'єднує кілька комірок по горизонталі, в даному випадку 2 колонки

rowspan="2" — об'єднує кілька комірок по вертикалі, в даному випадку 2 рядки

```
<table border="1" cellpadding="10" cellspacing="5">
        <tr>
            <th colspan="3">Заголовок</th>

        </tr>
        <tr>
            <td>Комірка 1.1</td>
            <td>Комірка 1.2</td>
            <td rowspan="2">Комірка 1.3</td>
        </tr>
        <tr>
            <td>Комірка 2.1</td>
            <td>Комірка 2.2</td>
        </tr>
</table>
```



Практична робота №12.

Завдання 1. Створити HTML-документ, присвячений вашим улюбленим книгам, використовуючи різноманітні HTML-теги, такі як заголовки, текст, списки, зображення та атрибути, щоб створити структуровану і привабливу вебсторінку.

Порядок виконання

1. Створення основного HTML-документа
   - Відкрийте текстовий редактор (наприклад, **Блокнот**, **Notepad++**, **Visual Studio Code** або будь-який інший).
   - Створіть новий файл у своїй робочій папці та збережіть його з розширенням **.html**, наприклад, **books.html**.
2. Структура HTML-документа
   - Додайте основні елементи HTML-документа: **<!DOCTYPE html>, <html>, <head>, <title>, <meta charset="UTF-8">, <body>**.

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <title>Мої улюблені книги</title>
</head>
<body>
```

3. Додавання заголовків і тексту
   - Використовуйте теги **<h1>**, **<h2>**, **<p>**, щоб створити заголовки різних рівнів і абзаци тексту, наприклад, як за зразком.

```
<h1>Мої улюблені книги</h1>

<h2>Чому я люблю читати</h2>
<p>Читання книг приносить мені велике задоволення. Це дає можливість зануритися в нові світи, дізнатися багато цікавого та розвивати свою уяву.</p>

<h2>Мої улюблені книги</h2>

<h2>Жанри, які я люблю</h2>

<h2>Обкладинки книг</h2>

<h2>Де купити ці книги</h2>

<h2>Інформація про книги</h2>
```

4. Додавання списків

- Після заголовка «Мої улюблені книги» створіть впорядкований (нумерований) список для переліку ваших улюблених книг за допомогою тегів **<ol>** і **<li>**. Заповніть своїми улюбленими книгами.

```
<ol>
   <li>Володар перснів - Дж. Р. Р. Толкін</li>
   <li>Гаррі Поттер і філософський камінь - Дж. К. Ролінг</li>
   <li>1984 - Джордж Орвелл</li>
</ol>
```

- Після заголовка «Жанри, які я люблю» створіть невпорядкований (маркований) список для переліку жанрів книг за допомогою тегів **<ul>** і **<li>**. Заповніть своїми улюбленими жанрами.

```
<ul>
   <li>Фентезі</li>
   <li>Наукова фантастика</li>
   <li>Детективи</li>
</ul>
```

5. Додавання зображень

- Знайдіть в інтернеті зображення обкладинок ваших улюблених книг. Частину з них збережіть у папку з вашим файлом **books.html**.
- Після заголовка «Мої улюблені книги» вставте обкладинки ваших улюблених книг, використовуючи тег **<img>**. Додайте атрибути **src**, **alt**, **width** і **height**. Деякі зображення вставте, використовуючи шлях до зображення у вашій папці, а частину через URL-адресу:

```
<img src="https://upload.wikimedia.org/wikipedia/uk/a/a3/The_Return_
of_the_King.jpg" alt="Володар перснів" width="200" height="300">
<img src="harry_potter.jpg" alt="Гаррі Поттер" width="200"
height="300">
<img src="1984.jpg" alt="1984" width="200" height="300">
```

6. Додавання таблиць

- Після заголовка «Інформація про книги» створіть таблицю з інформацією про книги за допомогою тегів **<table>**, **<tr>**, **<th>**, **<td>**. Додайте атрибути **border**, **cellpadding** і **cellspacing**.

```
<table border="1" cellpadding="10" cellspacing="0">
      <tr>
         <th>Назва книги</th>
         <th>Автор</th>
         <th>Жанр</th>
      </tr>
      <tr>
         <td>Володар перснів</td>
         <td>Дж. Р. Р. Толкін</td>
         <td>Фентезі</td>
      </tr>
      <tr>
```

```
            <td>Гаррі Поттер і філософський камінь</td>
            <td>Дж. К. Ролінг</td>
            <td>Фентезі</td>
        </tr>
        <tr>
            <td>1984</td>
            <td>Джордж Орвелл</td>
            <td>Наукова фантастика</td>
        </tr>
    </table>
```

7. Додатково

- Додайте до тегів заголовків атрибути вирівнювання «по центру».
- За допомогою html-тегів виділіть курсивом слова та словосполучення «Читання книг», «дізнатися».
- За допомогою html-тегів виділіть напівжирним шрифтом слова та словосполучення «нові світи», «уяву».
- За допомогою html-тегів підсвітіть слово «розвивати».
- Кожен розділ розділіть горизонтальною лінією, використовуючи тег `<hr>`.

Завдання 2. Створити HTML-документ на одну з перелічених тем, використовуючи різноманітні HTML-теги, такі як заголовки, текст, списки, зображення та атрибути, щоб створити структуровану і привабливу вебсторінку.

Тема 15 **Покликання та форми. Мультимедіа на вебсторінках. Поняття таблиці каскадних стилів. Інструменти розробника. Стильове оформлення сторінки. Підключення шрифтів**

Ключові поняття

Покликання
Форма та її елементи
Мультимедіа на вебсторінках
Таблиці каскадних стилів (CSS)
Селектори
Інструменти розробника

Покликання

Покликання (гіперпосилання) є одним з основних елементів **HTML**, що дозволяють створювати інтерактивні та навігаційні елементи на вебсторінках.

Покликання є на всіх вебсайтах. Саме вони дозволяють користувачам переходити з однієї сторінки на іншу і не тільки.

Покликання, по-іншому гіперпокликання, гіперпосилання, вставляють за допомогою парного тегу **<a>...**. Найважливішим атрибутом є **href** атрибут, який вказує на призначення посилання або URL-адресу, на яку веде покликання.

```
<a href="адреса">Покликання</a>
```

Найчастіше покликання роблять за текстовою частиною чи елементами списку. При цьому, клацнувши на текст покликання, вас буде перенаправлено за вказаною URL-адресою.

Проте покликання можна робити навіть за малюнком, клацаючи по ньому. При цьому замість тексту в тезі **<a>** потрібно вставити тег із зображенням.

```
<a href="https://www.google.com/">
Перейти на головну сторінку Google</a>
```

Перейти на головну сторінку Google

```
<a href="https://www.google.com/">
   <img src="nature.jpeg" alt="Природа">
</a>
```



Атрибути тегу покликання

href="#" — вказує URL-адресу, на яку веде покликання. Якщо замість адреси поставити **#** — отримаємо порожнє покликання, яке не переміщує користувача на іншу сторінку.

target="_blank" — визначає, де відкрити покликання (наприклад, **_self** — відкриває у тому ж вікні (за замовчуванням), **_blank** — відкриває у новій вкладці або вікні, **_parent** — відкриває у батьківському фреймі, **_top** — відкриває у повному вікні браузера).

```
<a href="https://www.google.com/" target="_blank">Відкрити у новій вкладці</a>
```

Види покликань

Внутрішні покликання

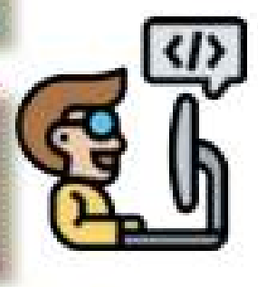
Ведуть на інші сторінки того ж сайту

```
<a href="about.html">Про нас</a>
```

Зовнішні покликання

Ведуть на інші сайти

```
<a href="https://youtube.com">YouTube</a>
```



Якорі (анкорні покликання)

Ведуть до певного місця на тій самій сторінці або іншій сторінці.

```
<a href="#section2">Перейти до секції 2</a>
<a href="page.html#section2"> Перейти до секції 2 на іншій сторінці</a>
```

Проте потрібно встановити якір в тому ж місці, куди потрібно зробити покликання. Для цього використовують атрибут **id=""**

```
<h2 id="section2">Секція 2</h2>
```

Поштові покликання (mailto)

Відкривають програму електронної пошти для надсилання листа за вказаною адресою.

```
<a href="mailto:someone@gmail.com">Надіслати електронний лист</a>
```

Телефонні покликання (tel)

Використовуються для здійснення телефонного дзвінка з мобільного пристрою або через VoIP.

```
<a href="tel:+1234567890">Зателефонувати</a>
```

Форма та її елементи

Форми в **HTML** використовуються для збирання та відправлення даних користувачів на сервер. Вони складаються з різних елементів, таких як текстові поля, кнопки, прапорці тощо, які дозволяють користувачам вводити та надсилати інформацію.

Для створення контейнера форми використовують тег **<form>**, всередині якого вже розміщують елементи форми.

```
<form>
    елементи форми
</form>
```

Для тегу **<form>** використовують атрибут **action=""**, що визначає URL, на який будуть відправлені ці форми.



Елементи форми

<input type="">

Тег **<input >** є найбільш важливим та вживаним елементом форми. Залежно від значення атрибута **type** залежатиме тип цього елемента. Розглянемо основні з них.

```
<input type="text" placeholder="Введіть ваше ім'я">
```

type="text: створює текстове поле для введення тексту.

placeholder="Введіть ваше ім'я": задає текст підказки, який відображається всередині текстового поля до моменту, поки користувач не почне вводити текст.

```
<input type="email">
```

type="email": створює поле для введення електронної пошти.

```
<input type="password" required>
```

type="password": створює поле для введення пароля. Введені символи будуть відображатися як крапки або зірочки.

required: робить поле обов'язковим для заповнення.

```
<input type="radio" name="name">
```

type="radio": створює радіокнопку. Користувач може вибрати лише одну з декількох опцій, що мають однакове значення атрибута *name*.

name="": всі радіокнопки з однаковим значенням атрибута *name* об'єднуються в одну групу, і користувач може вибрати тільки одну з них.

```
<input type="checkbox" name="name">
```

type="checkbox": створює прапорець, який користувач може відзначити або зняти відмітку. Користувач може вибрати декілька опцій, що мають однакове значення атрибута *name*.

name="": ім'я, яке використовується для групування певних прапорців при відправленні форми.

```
<input type="number" min="1" max="10">
```

type="number": створює поле введення, яке дозволяє користувачам вводити лише числові значення.

min, max: встановлюють мінімальні та максимальні межі.

```
<button>Кнопка</button>
```

Тег **<button>** використовується для створення кнопки в HTML. Він дозволяє створювати кнопки з різним вмістом, таким як текст, зображення або обидва. Тег **<button>** можна використовувати всередині форм або поза ними для виконання різних дій, таких як надсилання даних форми або запуск **JavaScript**.

```
<button type="submit">Надіслати</button>
```

type="submit": створює кнопку відправки форми. При натисканні кнопки дані форми відправляються на сервер.

```
<button type="reset">Скинути</button>
```

type="reset": створює кнопку скидання форми, яка очищує всі поля форми до їхніх початкових значень.

```
<button type="button">Кнопка</button>
```

type="button": створює звичайну кнопку, яка не відправляє форми. Використовується для виконання дій, таких як запуск **JavaScript**.

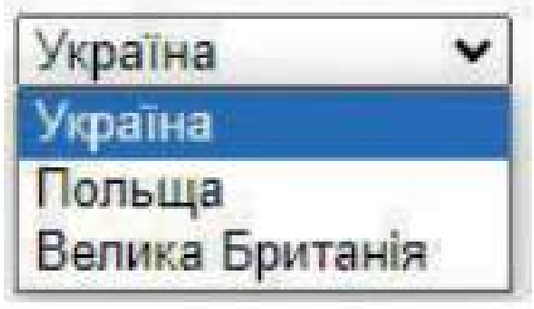
```
<textarea></textarea>
```

Тег **<textarea>** використовується для введення великого обсягу тексту.

```
<select></select>
```

Тег **<select>** використовується для створення випадаючого списку з варіантами. Для створення самих варіантів використовують тег **<option>**.

```
<select>
  <option value="ua">Україна</option>
  <option value="pl">Польща</option>
  <option value="uk">Велика Британія</option>
<select>
```




```
<label for="type">текст:</label>
```

Тег **<label>** в **HTML** використовується для визначення мітки та підпису елементів форми. Він дозволяє зробити форму більш доступною та зручною для користувачів, особливо для тих, хто користується екранними читачами.

Цей тег асоціюється з конкретним елементом форми через атрибут **for**, який пов'язує мітку з елементом за його **id**. Наприклад:

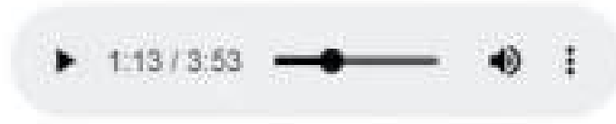
```
<label for="username">Ім'я користувача:</label>
<input type="text" id="username">
```



Мультимедіа на вебсторінках

Окрім зображень, мультимедійними елементами на вебсторінках можуть бути також аудіо та відео.

```
<audio src="audio.mp3" controls></audio>
```



Для вставки аудіофайлів на вебсторінки використовують тег **<audio>**. При цьому використовують такі атрибути:

src="" — вказує шлях до аудіофайлу;
controls — відображає елементи керування відтворенням;
autoplay — автоматично відтворює аудіо при завантаженні сторінки;
loop — повторно відтворює аудіо.

```
<video src="video.mp4" controls></video>
```

Для вставки відеофайлів використовують тег **<video>**. Окрім атрибутів, як для тегу **<audio>**, таких як **src**, **controls**, **autoplay** та **loop**, використовують такі:

width="" height="" — задають ширину та висоту відеоплеєра;
poster="" — задає шлях до зображення-заставки для відео.

Таблиці каскадних стилів (CSS)

Таблиця каскадних стилів (CSS) — це мова стилів, що використовується для опису зовнішнього вигляду і форматування HTML-документів.



CSS дозволяє відокремити зміст сторінки від її зовнішнього вигляду, що значно полегшує підтримку та оновлення вебсайтів. Якщо для прикладу взяти будинок, то **HTML** відповідав би за каркас, а **CSS** — за його стиль та зовнішній вигляд.

Тобто на вебсторінці відбувається те саме: за допомогою правил **CSS** на ній з'являються кольори, відступи, змінюється стиль шрифтів, розташування і розміри елементів, створюється цілий, завершений стиль.

CSS взаємодіє з **HTML**, використовуючи особливі **css**-правила, які визначають, до яких компонентів і як прикручувати стилі.

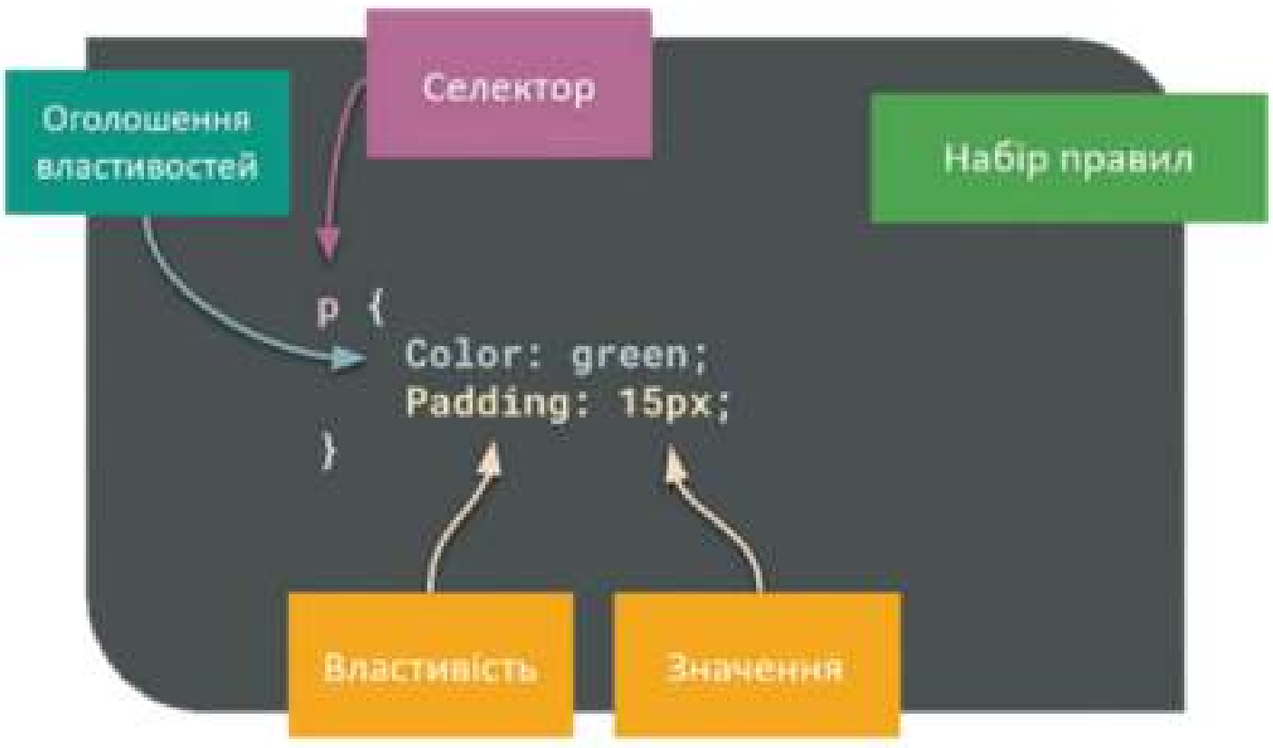
CSS — правило складається із селектора та блоку оголошень, розташованого у фігурних дужках. Всередині блоку є властивість та її значення.

селектор { властивість : значення }

Селектори визначають, до яких елементів **HTML** застосовуватимуться стилі. Селекторами можуть бути назви тегів, **id** або класи елементів.

Властивості визначають характеристики елементів, які потрібно змінити (колір, шрифт, розмір тощо).

Значення визначають конкретні значення властивостей.



Способи підключення CSS

Коли браузер читає таблицю стилів, він форматує **HTML**-документ відповідно до інформації в таблиці стилів. Розглянемо способи підключення стилів до HTML-документа.

Вбудовані стилі (Inline CSS)

Щоб використовувати вбудовані стилі, потрібно додати атрибут **style** до відповідного елементу (тегу). Атрибут може містити кілька властивостей, розділених крапкою з комою, фігурні дужки не використовуються. Вбудований стиль необхідно використовувати якомога рідше, адже він перевантажує **HTML**-код та збільшує загальний обсяг вебсторінки.

```
<p style="color:red; font-size:20px">Текст червоного кольору</p>
```

Текст червоного кольору

Внутрішні стилі (Embedded CSS)

Внутрішня таблиця стилів може використовуватися, якщо одна окрема **HTML**-сторінка має унікальний стиль, котрий більше ніде не використовується на інших сторінках. Стилі додаються всередину тегу **<style>** у розділі **<head>** HTML-документа.

```
<style>
   h1 {
       background-color: #f0f0f0;
         color: blue;
    }
</style>
```

Текст синього кольору

Зовнішні стилі (External CSS)

Підключення файлу зовнішньої таблиці стилів є найбільш розповсюдженим способом. Стилі зберігаються у зовнішньому файлі, який підключається до HTML-документа за допомогою тегу **<link>**. Файл можна написати в будь-якому текстовому редакторі (наприклад, як в стандартному **Блокноті**, так і у **VsCode**). Файл не повинен містити жодних HTML-тегів. Файл таблиці стилів має бути збережено з розширенням **.css**.

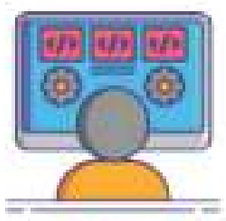
```
<head>
   <link rel="stylesheet" href="style.css">
</head>

<body>
   <p>Текст зеленого кольору</p>
</body>
```

Текст зеленого кольору

Інструменти розробника

Інструменти розробника (developer tools) — це набір інструментів, вбудованих у веббраузери, які допомагають розробникам створювати, налагоджувати та тестувати вебсайти та вебдодатки.



ФУНКЦІЇ

1. Перегляд і редагування **HTML** та **CSS**
2. Аналіз продуктивності сторінки
3. Налагодження **JavaScript**
4. Відстеження мережевих запитів

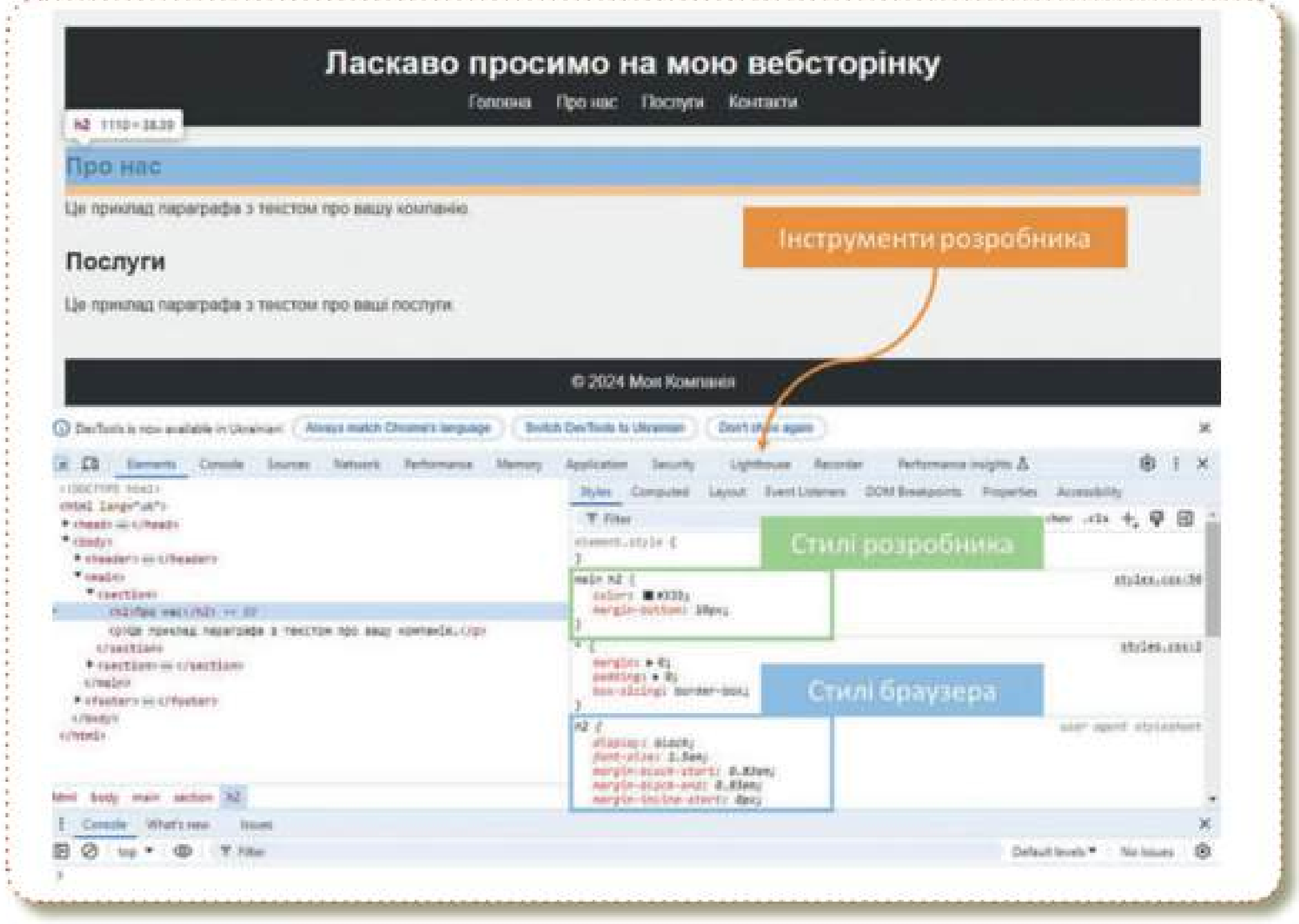
Функція **Перегляд і редагування HTML** та **CSS** дозволяє переглядати та редагувати **HTML** і **CSS** у реальному часі. Зручна для налагодження верстки, перевірки властивостей стилів та взаємодії елементів на сторінці.

Перегляд HTML

1. Відкрийте інструменти розробника (**F12** або **Ctrl+Shift+I**).
2. Виберіть вкладку **Elements** (або **HTML**).
3. Наведіть на елемент, щоб переглянути його деталі.

Редагування CSS

1. Виберіть вкладку **Styles** в інструментах розробника.
2. Внесіть зміни до CSS-стилів та миттєво перегляньте результати на сторінці. Після оновлення сторінки зміни не зберігаються.



Селектори

CSS Селектори

Оскільки селектор дозволяє звернутися до одного або кількох елементів, тому є кілька способів задання селекторів, залежно від того, як саме ми хочемо стилізувати сторінку.

1 **Селектор елемента**

Селектор елемента (тегу) стилізує усі елементи певного типу, які є на сторінці.

```
<p>Перший абзац</p>
<p>Другий абзац</p>
<p>Третій абзац</p>
```

HTML

```
p {
   text-align: center;
    color: blue;
}
```

CSS

Перший абзац

Другий абзац

Третій абзац абзац

Вигляд

2 **Селектор ідентифікатора** — це унікальний селектор, який має одне-єдине ім'я в усьому документі. При цьому до елемента, який потрібно стилізувати, додають атрибут **id=""** з іменем. Щоб звернутися до ідентифікатора, використовують знак решітки **#**

```
<p id="blue-text">Перший абзац</p>
<p>Другий абзац</p>
<p>Третій абзац</p>
```

HTML

```
#blue-text {
   text-align: center;
    color: blue;
}
```

CSS



Вигляд

3 Селектор класу

Селектор ідентифікатора стилізує усі елементи, до яких додали атрибут **`class=""`**. Це зручний спосіб обрати один або кілька елементів, яким потрібно змінити властивості, задавши їм одне і те саме ім'я класу. Щоб звернутися до ідентифікатора, використовують крапку.

```
<p class="blue-text">Перший абзац</p>
<p>Другий абзац</p>
<p class="blue-text">Третій абзац</p>
```

HTML

```
.blue-text {
   text-align: center;
    color: blue;
}
```

CSS



Вигляд

Якщо є елементи, які мають деякі властивості однакові, то для того, щоб мінімізувати код, доцільно селектори згрупувати, записавши їх через кому.

```
h1, p, .blue-text {
   text-align: center;
   color: blue;
}
```

Колір та фон

Таблиці стилів містять у своєму переліку дуже велику кількість властивостей елементів та їхніх значень. Розглянемо основні з них.

Властивості, що налаштовують кольори тексту та фону елементів, а також зображення на фоні.

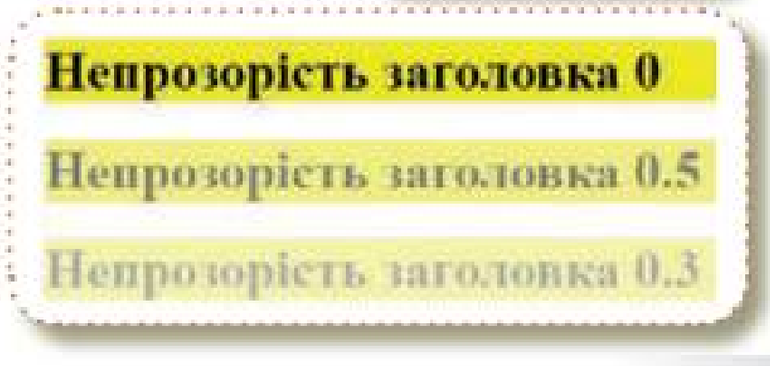
- **`color`** — визначає колір тексту

Колір вказується за допомогою:

- назва кольору — наприклад, `color: blue;`
- значення HEX — наприклад, `color: #9f9f9f;`
- значення RGB — наприклад, `color: rgb(255,122,45)`

- **`background-color`** — визначає колір фону елемента.





Колір фону елемента встановлюється таким чином: **background-color: yellow**.

Додаткова властивість **opacity: 0.5** визначає непрозорість елементу. Значення може набувати від 0.0 до 1.0. Що менше значення, то більша прозорість елемента.

Зображення на фоні

- **background-image** — задає зображення як фон елемента

Щоб на фон елемента поставити зображення, записують так: **background-image: url('background.png')**;

За замовчуванням зображення повторюється як по горизонталі, так і по вертикалі. Щоб керувати повторенням зображення, використовують властивість **background-repeat**, яка може мати такі властивості:

- **no-repeat** — без повтору;
- **repeat-x** — повторюється лише по горизонталі;
- **repeat-y** — повторюється лише по вертикалі.

Шрифти та текст

Властивості, що визначають шрифт, розмір, товщину, вирівнювання та прикраси тексту.

- **font-family** — визначає шрифт тексту

Щоб вказати шрифт тексту, використовують властивість **font-family: Arial, sans-serif**.

Якщо в назві шрифту є більше, ніж одне слово, вона має бути взята в лапки, наприклад, **font-family: "Times New Roman", serif**.

(**serif** — шрифт із засічками, **sans-serif** — шрифт без засічок)

- **font-size** — визначає розмір шрифту

Розмір тексту вказується за допомогою:

- пікселів — наприклад, **font-size: 24px**;
- відносно розміру шрифту батьківського елемента — наприклад, **font-size: 1em**;
- у відсотках відносно розміру шрифту батьківського елемента — наприклад, **font-size: 150%**;
- відносно вікна користувача (vw — 1% від ширини вікна, vh — 1% від висоти вікна) — наприклад, **font-size: 3vw**;

- **font-weight** — визначає товщину шрифту

Властивість **font-weight** визначає товщину шрифту:

- **font-weight: normal**;
- **font-weight: bold**;

- **font-style** — визначає стиль шрифту

Властивість **font-style** визначає стиль шрифту:

- **font-style: normal**;
- **font-style: italic**;
- **text-align** — вирівнює текст всередині елемента.

Властивість **text-align** використовується для встановлення горизонтального вирівнювання тексту.

Текст можна вирівняти:

- за лівим краєм **text-align: left**;
- за правим краєм **text-align: right**;
- по центру **text-align: center**;
- за шириною **text-align: justify**;
- **text-decoration** — визначає декорацію тексту.

Властивість **text-decoration** використовується для додавання та стилізації текстових декорацій, таких як підкреслення, надрядкові лінії та інше.

- видаляє всі декорації **text-decoration: none**;
- додає підкреслення під текстом **text-decoration: underline**;
- додає лінію над текстом **text-decoration: overline**;
- додає лінію через текст (перекреслення) **text-decoration: line-through**;

Є детальніші стилізації, такі як **text-decoration-color: red** (задає колір); **text-decoration-style: double** (визначає стиль текстових прикрас: **solid**, **double**, **dotted**, **dashed**, **wavy**).

Ще одна з властивостей є властивість **text-transform**, яка визначає, якими літерами буде відображено текст — ВЕЛИКИМИ (**uppercase**), малими (**lowercase**) або Кожне Слово З Великої Літери (**capitalize**).

Обрамлення та тіні

Властивості, що визначають рамки навколо елементів, заокруглення кутів та тіні.

border — визначає рамку навколо елемента

Властивість **border** в **CSS** використовується для додавання і стилізації рамок навколо елементів. Вона дозволяє налаштовувати ширину, стиль і колір рамки.

Основні властивості border

- **border-width:** визначає ширину рамки
- **border-style:** визначає стиль рамки (solid, dotted, dashed, double, groove, ridge, inset, outset)
- **border-color:** визначає колір рамки.

Короткий запис **border**, властивість **border** дозволяє в одному записі задати ширину, стиль і колір рамки: **border:** ширина, стиль, колір.

В CSS також є властивості для вказування кожної із меж (верхньої, правої, нижньої і лівої): **border-top**, **border-right**, **border-bottom**, **border-left**.

Рамка

```
border: 5px solid blue;
```

border-radius — визначає радіус заокруглення кутів рамки

Властивість **border-radius** використовується для заокруглення кутів рамок елементів. Вона дозволяє створювати елементи з округленими кутами, еліптичні форми або навіть кола.




```
border-radius: 10px;
```




```
border-radius: 25%;
```




```
border-radius: 10px 30px;
```

box-shadow — визначає тінь для елемента

Властивість **box-shadow** використовується для додавання тіней до елементів. Вона дозволяє налаштовувати положення, розмиття, розмір та колір тіні, а також створювати множинні тіні.

box-shadow: горизонтальне зміщення, вертикальне зміщення, розмиття, поширення, колір




```
box-shadow: 10px 10px 5px grey;
```




```
box-shadow: inset 10px 10px 20px grey;
```

Відступи та поля

Властивості, що визначають відстань всередині елемента (поля) та зовнішні відступи від інших елементів.

margin — визначає зовнішній відступ елемента

Властивість **margin** в **CSS** використовується для визначення зовнішніх відступів (поля) навколо елемента. Відступи створюють простір між елементом і сусідніми елементами.

Властивість **margin** дозволяє задавати відступи для всіх чотирьох сторін елемента в одному записі.

margin: 10px 20px 30px 40px — відступи у порядку: верхній, правий, нижній, лівий. Проте можна задавати відступи окремо до кожної зі сторін: **margin-top: 10px**; **margin-right: 20px**; **margin-bottom: 30px**; **margin-left: 40px**.

Для того, щоб відцентрувати елемент в контейнері, використовують значення **margin: 0 auto**.

padding — визначає внутрішній відступ елемента

Властивість **padding** в **CSS** використовується для визначення внутрішніх відступів (полів) всередині елемента. Відступи створюють простір між вмістом елемента і його межами.

Властивість **padding** дозволяє задавати відступи для всіх чотирьох сторін елемента в одному записі.

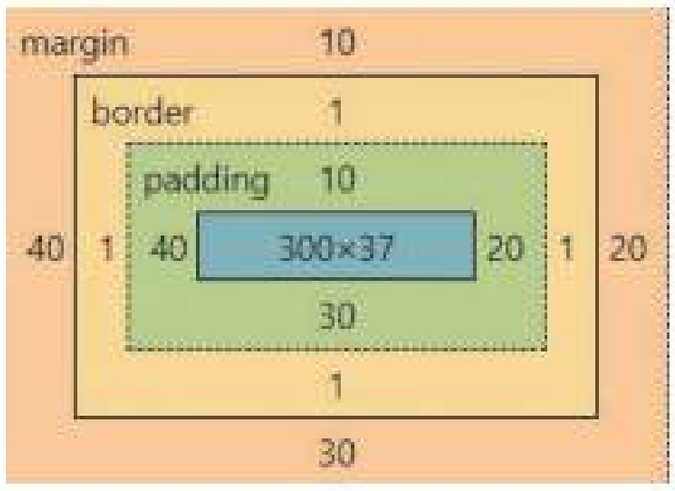
padding: 10px 20px 30px 40px — відступи у порядку: верхній, правий, нижній, лівий. Проте можна задавати відступи окремо до кожної зі сторін: **padding-top: 10px**; **padding-right: 20px**; **padding-bottom: 30px**; **padding-left: 40px**.

Для відступів використовують такі одиниці виміру: **px**, **em**, **rem**, **%**, **cm**, **mm** тощо.





У вікні розробника можна побачити, як впливають відступи на розміри контейнера. Тобто по межі контейнера можна побачити, що padding збільшує розміри даного об'єкта. Тому це потрібно враховувати для обчислення точності розмірів об'єкта на сторінці сайту.



Розміри та розташування

Властивості, що налаштовують ширину, висоту та розташування елементів на сторінці:

width — визначає ширину елемента;

height — визначає висоту елемента.

В **CSS** розміри використовуються для визначення висоти, ширини, відступів, полів та інших властивостей елементів. Використання правильних одиниць виміру допомагає створювати адаптивні та гармонійні дизайни.

Основні одиниці виміру:

1. Абсолютні одиниці:

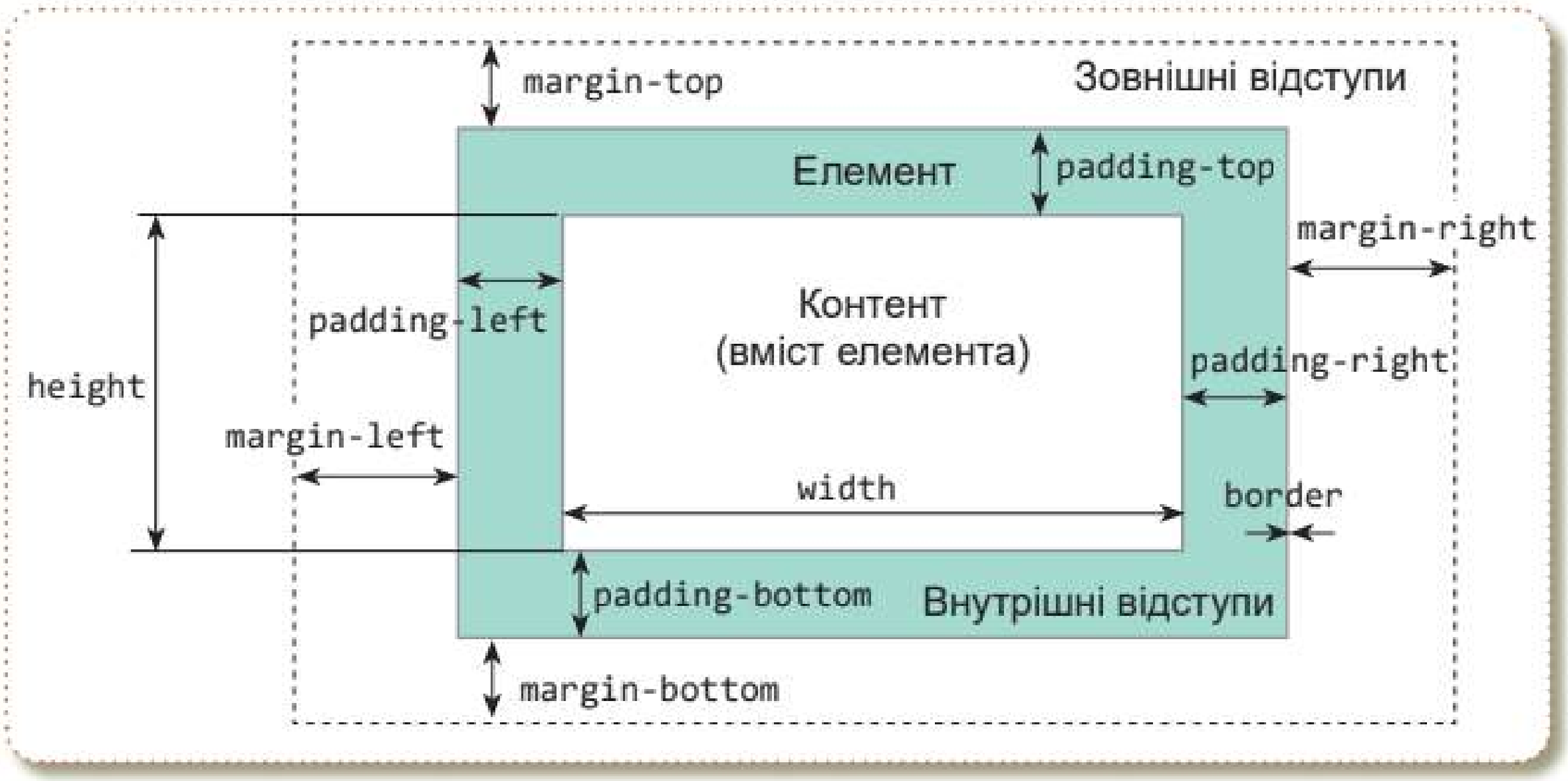
- **px** (пікселі) — найбільш використовувана одиниця виміру для точного контролю розміру;
- **m**, **mm**, **in**, **pc** (сантиметри, міліметри, дюйми, піки) — рідко використовуються у вебдизайні, але можуть бути корисні для спеціальних завдань.

2. Відносні одиниці:

- **%** (відсотки) — задають розмір як відсоток від розміру батьківського елемента;
- **em** — базується на розмірі шрифту батьківського елемента;
- **vw (viewport width)** — базується на ширині вікна браузера;
- **vh (viewport height)** — базується на висоті вікна браузера.

Властивості висоти і ширини не містять **padding** (відступи), **border** (межі) або **margin** (поля). Вони встановлюють висоту/ширину області всередині відступу, межі та поля елемента.

Для того, щоб задати максимальну та мінімальну ширину елемента, відповідно використовують властивості **min-width** та **max-width**.



Розміри та розташування

В **CSS** існують різні методи розташування елементів на вебсторінці. Основні методи включають: нормальний потік, плаваючі елементи, позиціонування та використання флексбоксів і грід-системи. Розглянемо деякі з них.

Нормальний потік

За замовчуванням елементи на вебсторінці розташовуються один за одним зверху вниз, як зазвичай, відповідно до їхньої природи (блочні або рядкові).



Позиціонування (position)

Властивість **position** дозволяє контролювати розташування елементів з допомогою координат.

Є п'ять різних значень **position** (позиції):

- **static** — статичний;
- **relative** — відносний;
- **fixed** — фіксований;
- **absolute** — абсолютний;
- **sticky** — закріплений.

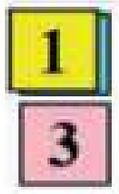




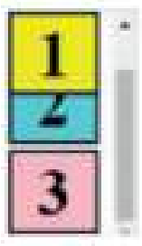

```
position: relative;
top: 10px;
left: 20px;
```




```
position: absolute;
top: 20px;
left: 80px;
```




```
position: fixed;
top: 0;
left: 0;
```

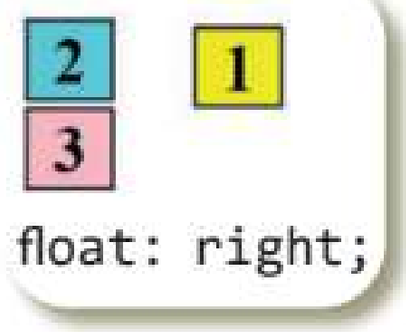



```
position: sticky;
top: 0;
```

Потім елементи позиціонуються з використанням властивостей **top**, **bottom**, **left** та **right** (вгорі, знизу, зліва, справа). Однак ці властивості не будуть працювати, якщо спочатку не встановлено властивість **position**. Вони також працюють по-різному залежно від значення **position**.

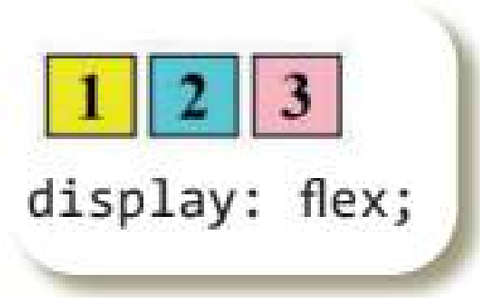
Плаваючі елементи (float)

Властивість **float** дозволяє розташовувати елементи ліворуч або праворуч від контейнера, обтікаючи їх текстом або іншими елементами. Може набувати значень: **left** (елемент плаває зліва від свого контейнера); **right** (елемент плаває справа від свого контейнера); **none** (елемент не плаває; це за замовчуванням). Найкраще **float** можна використовувати для обтікання текстом зображень.



Відображення (Display)

Властивість **display** визначає, як елементи будуть відображатися на вебсторінці. Вона є однією з ключових властивостей **CSS**, яка дозволяє контролювати, як елементи займають простір у документі та як вони впливають на розташування інших елементів.



Значення властивості **display**:

- **none** — елемент не відображається на сторінці;
- **block** — елемент відображається як блоковий елемент. Він займає всю ширину свого контейнера, і кожен новий блоковий елемент починається з нового рядка;
- **inline** — елемент відображається як стрічковий елемент. Він займає лише необхідну ширину і не починається з нового рядка;
- **flex** — елемент стає контейнером для *flex*-елементів, що дозволяє використовувати всі можливості *flexbox*-розмітки для вирівнювання та розподілу простору між дочірніми елементами.

Підключення шрифтів у CSS

Підключення користувацьких шрифтів у вебсторінках дозволяє застосовувати шрифти, які не встановлені на комп'ютерах користувачів. Це досягається за допомогою властивості **@font-face** та сервісів шрифтів, таких як **Google Fonts**.

Способи підключення шрифтів:

Підключення через Google Fonts

Google Fonts — це сервіс, який дозволяє легко підключати шрифти до вебсторінок через інтернет.

Кроки підключення

1. Вибір шрифту на сайті **Google Fonts**:

- Перейдіть на **Google Fonts**.
- Виберіть шрифт, який вам подобається.
- Налаштуйте стилі (вага шрифту, нахил тощо).

- Скопіюйте наданий код для підключення шрифту.

2. Додавання шрифту до HTML-документа:

- Додайте код підключення в розділ **<head>** вашого HTML-документа.

```
<link href="https://fonts.googleapis.com/css2?family=Roboto:
wght@400;700&display=swap" rel="stylesheet">
body {
      font-family: 'Roboto', sans-serif;
 }
```

Підключення через @font-face

Властивість **@font-face** дозволяє підключати власні шрифти, зберігаючи їх локально на сервері. Завантажувати шрифти також можна із сайту **Google Fonts**.

```
@font-face {
  font-family: 'Roboto';
  src: url('fonts/Roboto.woff2') format('woff2'),
       url('fonts/Roboto.woff') format('woff');
  font-weight: normal;
  font-style: normal;
}

body {
  font-family: 'Roboto', Arial, sans-serif;
}
```



Псевдокласи стану

Псевдокласи в CSS — це спеціальні ключові слова, які додаються до селекторів для вказівки особливого стану елемента.

Вони дозволяють стилізувати елементи на основі їхнього стану, позиції у документі або взаємодії з користувачем. Розглянемо деякі з них:

:hover — застосовується, коли користувач наводить курсор миші на елемент;

:active — застосовується у момент натискання на елемент;

:visited — застосовується до покликань, які вже відвідав користувач.

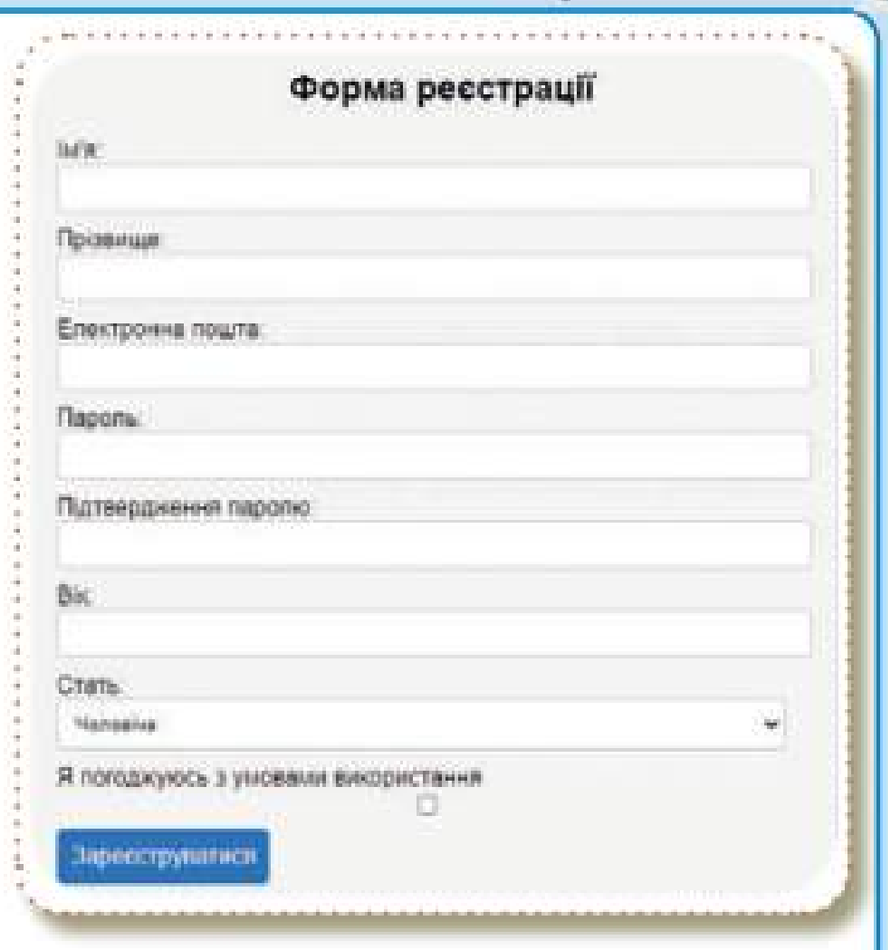
```
a:hover {
    color: red;
}
```

```
a:active {
    color: orange;
}
```

```
a:visited {
    color: purple;
}
```

Практична робота №13.

Завдання 1. Створити форму реєстрації користувача, використовуючи **HTML** та **CSS**. Форма повинна містити різні типи полів та стилізацію.



Порядок виконання

1. Створіть папку та в ній HTML-файл, наприклад, **index.html**.
2. Створіть базову структуру **HTML**:
   - Додайте основні елементи HTML-документа: **<!DOCTYPE html>**, **<html>, <head>, <title>, <meta charset="UTF-8">, <body>**
   - Додайте заголовок **<h2>Форма реєстрації</h2>**
   - Додайте тег для форми

```
<h2>Форма реєстрації</h2>
 <form>

 </form>
```

   - Створіть два поля для імені та прізвища

```
<label for="firstName">Ім'я:</label>
<input type="text" id="firstName">
```

   - Створіть поля для електронної пошти, пароля та підтвердження пароля

```
<label for="password">Пароль:</label>
<input type="password" id="password">

<label for="confirmPassword">Підтвердження пароля:</label>
<input type="password" id="confirmPassword">
```

   - Створіть числове поле для введення віку

```
<label for="age">Вік:</label>
<input type="number" id="age" min="1" max="100">
```

   - Створіть радіокнопки для вибору статі

```
<label for="gender">Стать:</label>
<select id="gender">
      <option value="male">Чоловіча</option>
      <option value="female">Жіноча</option>
      <option value="other">Інша</option>
</select>
```

- Створіть прапорець підтвердження

```
<label for="terms">
      Я погоджуюсь з умовами використання
      <input type="checkbox" id="terms">
</label>
```

- Створіть кнопку з іменем **Зареєструватися**

```
<button type="submit">Зареєструватися</button>
```

3. Стилізація форми з використанням **CSS**
   - Створіть **CSS**-файл, наприклад, **style.css**
   - Підключіть даний файл до html-файла

```
<link rel="stylesheet" href="styles.css">
```

   - Стилізуйте **body**, змінивши шрифт та задавши колір фону, наприклад:

```
body {
    font-family: Arial, sans-serif;
    background: #f7f7f7;
    margin-left: 20px;
}
```

   - Стилізуйте заголовок так, щоб був розміщений по центру
   - Задайте ширину форми, наприклад 500px

```
form {
    width:500px;
}
```

   - Стилізуйте поля, використовуючи відступи та рамки (для **input** та **select** задамо ширину 100% для того, щоб написи були не в одному рядку з елементами; в наступній темі розглянемо інші способи розміщення)

```
label {
    margin-bottom: 5px;
}
input, select {
    width: 100%;
    padding: 8px;
    margin-bottom: 10px;
    border: 1px solid #ccc;
    border-radius: 5px;
}
```

   - Стилізуйте кнопку, задавши їй колір, рамку, округлення тощо (**cursor:pointer** — властивість, яка дозволяє змінити курсор на "руку" при наведенні на елемент)

```
button {
    padding: 10px;
    background: #007bff;
    border: none;
    border-radius: 5px;
    color: #fff;
    font-size: 16px;
    cursor: pointer;
}
```

- Додатково. Змінити стиль кнопки при наведенні на неї мишкою

```
button:hover {
    background: #0056b3;
}
```

Завдання 2. Доповнити HTML-документ з теми 14, присвячений вашим улюбленим книгам, покликаннями, формою, мультимедіа. Стилізувати сторінку за допомогою таблиць каскадних стилів (**CSS**), інструментів розробника та підключення шрифтів.

Порядок виконання

1. Підготовка HTML-документа
   - Створіть папку з назвою **Books** та перемістіть туди ваш файл **books.html** з теми 14.
   - Відкрийте документ **books.html** в редакторі.
2. Додавання покликань
   - Після заголовка «Де купити ці книги» створіть гіперпокликання на сайти з рецензіями або купівлею книг, використовуючи тег **<a>**. Додайте атрибут **href**. Мінімум 3 покликання.

```
<p>Ви можете придбати ці книги на <a href="https://www.amazon.com/"
target="_blank">Amazon</a>.</p>
```

3. Додавання мультимедіа
   - Створіть заголовок «Відео про книги»
   - Знайдіть коротеньке відео про книги та розмістіть його в папці **Books**.
   - Вставте відео про книги за допомогою тегів **<video>** та **<source>**. Додайте атрибути **controls**, **width** і **height**

```
<h2>Відео про книги</h2>
 <video controls width="500" height="300">
     <source src="book_trailer.mp4" type="video/mp4">
     Ваш браузер не підтримує відтворення відео.
 </video>
```

4. Додавання форми
   - Створіть заголовок «Зворотний зв'язок»

- Створіть форму для зворотного зв'язку, використовуючи теги **<form>, <label>, <input>, <textarea>, <button>** так, щоб були поля для імені, електронної пошти, повідомлення та кнопка **Відправити**

```
<h2>Зворотний зв'язок</h2>
<form action="submit_form.php" method="post">
    <label for="name">Ім'я:</label>
    <input type="text" id="name" name="name" required>

    <label for="email">Електронна пошта:</label>
    <input type="email" id="email" name="email" required>

    <label for="age">Вік:</label>
    <input type="number" id="age" name="age" min="0" required>

    <label for="message">Повідомлення:</label>
    <textarea id="message" name="message" rows="4" required></
    textarea>

    <button type="submit">Відправити</button>
```

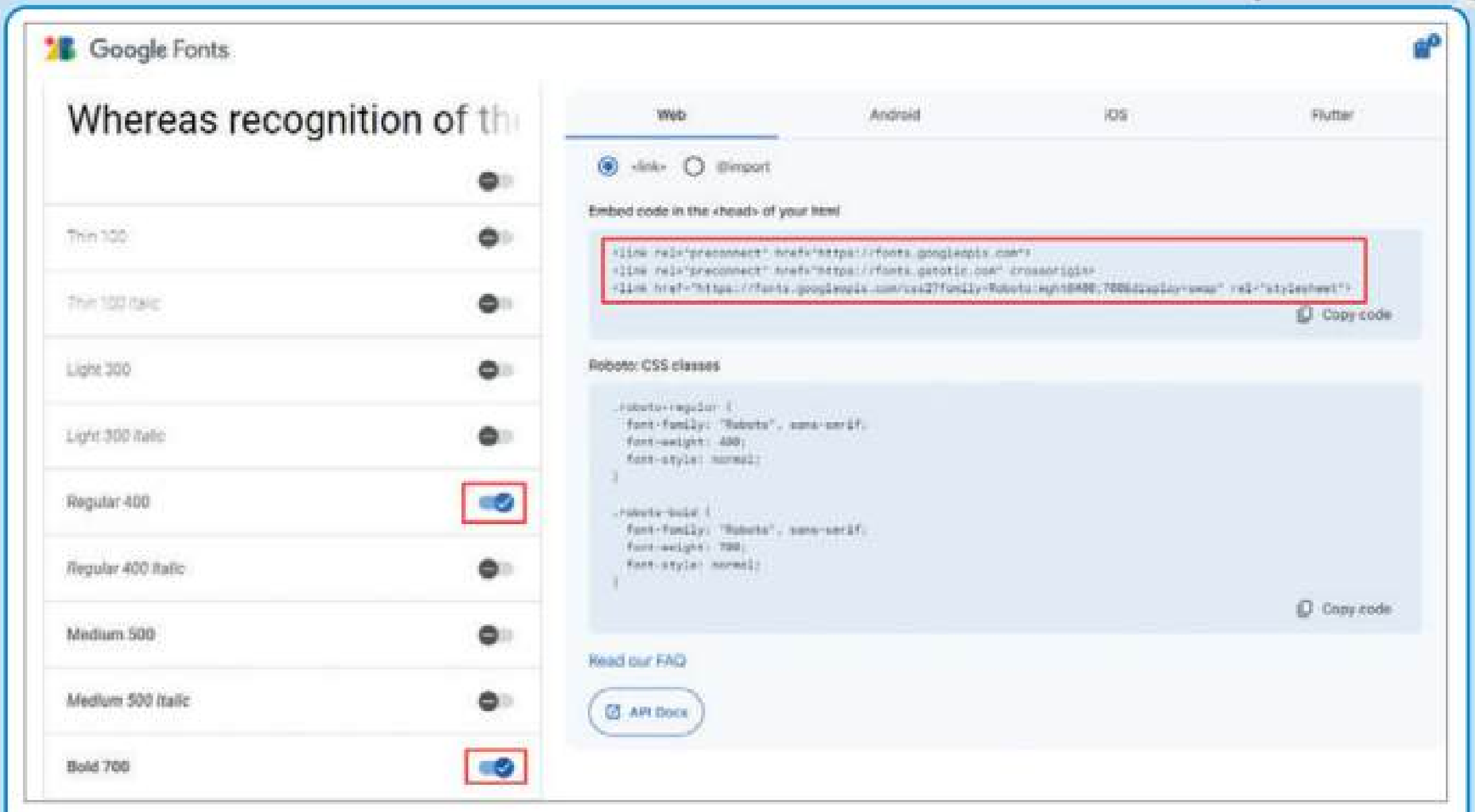
5. Підключення шрифтів
   - Перейдіть на сайт **Google Fonts**
   - Оберіть шрифт, наприклад, **Roboto**
   - Натисніть кнопку **Get Font**



   - Натисніть кнопку **Get embed code**



   - У фільтрах оберіть лише типи шрифтів 400 і 700, скопіюйте код та вставте його в **head** вашого документа



6. Додавання стильового оформлення сторінки

- Створіть файл **style.css** та підключіть його до html-файла
- Встановіть загальні стилі для сторінки, наприклад:

```
body {
    font-family: 'Roboto', sans-serif;
    margin: 0;
    padding: 20px;
    background-color: #f4f4f4;
    color: #333;
}
```

- Встановіть стилі для заголовків, наприклад:

```
h1 {
    color: #0056b3;
    text-align: center;
    margin-bottom: 20px;
}

h2 {
    color: #333;
    border-bottom: 2px solid #0056b3;
    padding-bottom: 10px;
    margin-top: 40px;
}
```

(**border-bottom: 2px solid #0056b3;** — додає рамку елемента знизу, щоб задати підкреслення заголовків)

- Встановіть стилі для абзаців, наприклад:

```
p {
    line-height: 1.6;
}
```

- Властивість **line-height** використовується для встановлення висоти рядка тексту (міжрядковий інтервал)
- Встановіть стилі для списків, наприклад:

```
ol, ul {
    margin-left: 20px;
    margin-bottom: 20px;
}

li {
    margin-bottom: 5px;
}
```

- Встановіть стилі для зображень, наприклад:

```
img {
    margin-right: 15px;
    border-radius: 5px;
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.5);
}
```

- Встановіть стилі для гіперпосилань, наприклад:

```
a {
    color: #0056b3;
    text-decoration: none;
}

a:hover {
    color: #003d80;
    text-decoration: underline;
}
```

- Встановіть стилі для таблиці, наприклад:

```
table {
    margin-bottom: 20px;
    background-color: #fff;
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.5);
}

th, td {
    padding: 15px;
    text-align: left;
    border-bottom: 1px solid #ddd;
}

th {
    background-color: #0056b3;
    color: #fff;
}
```

- Встановіть стилі для відео, наприклад:

```
video {
    display: block;
    margin: 0 auto;
    margin-bottom: 40px;
    border-radius: 5px;
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.5);
}
```

- Встановіть стилі для форми та її елементів, наприклад:

```
form {
    background-color: #fff;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 0 15px rgba(0, 0, 0, 0.5);
    margin-bottom: 40px;
}
label {
    display: block;
    margin-bottom: 10px;
    font-weight: bold;
}
input, textarea {
    width: 100%;
    padding: 10px;
    margin-bottom: 20px;
    border: 1px solid #ddd;
    border-radius: 5px;
    box-sizing: border-box;
}
button {
    background-color: #0056b3;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
    cursor: pointer;
    font-size: 16px;
}
button:hover {
    background-color: #003d80;
}
```

- За потреби обговоріть у групі властивості, які описані в прикладі.

Модуль гнучких блоків. Макет сайту. Цільова аудиторія та технічне завдання з розробки вебсайту. Аналіз відвідування сайту

Тема 16

Ключові поняття

Контейнери в HTML
Flexbox
Макет сайту
Цільова аудиторія
Технічне завдання

Контейнери в HTML

Контейнери в HTML використовуються для групування інших елементів на вебсторінці, що допомагає структурувати контент та застосовувати стилі. Вони дозволяють розробникам легше керувати макетом сторінки та її елементами.

`<div>...</div>` Найпоширеніший контейнер, використовується для групування блокових елементів.

Може містити будь-які інші елементи HTML.

Зазвичай використовується для створення секцій сторінки та застосування стилів.

```
<div>
   <p>Це контейнер, що містить параграф</p>
</div>
```

Це контейнер, що містить параграф.

`<span>...</span>` — контейнер для стрічкових елементів.

Використовується для групування стрічкових елементів або для стилізації частин тексту.

```
<p>Це <span class="highlight">важливий</span>
                текст.</p>
```

Це важливий текст.

Дочірні та батьківські елементи в HTML

Коли елементи вкладено один в одного, з'являються поняття батьківських та дочірніх елементів.

Батьківський елемент — елемент, який містить інші елементи всередині себе. Наприклад, у конструкції

```
<div>
   <p>Це контейнер, який містить параграф</p>
</div>,
<div> є батьківським елементом для <p>.
```

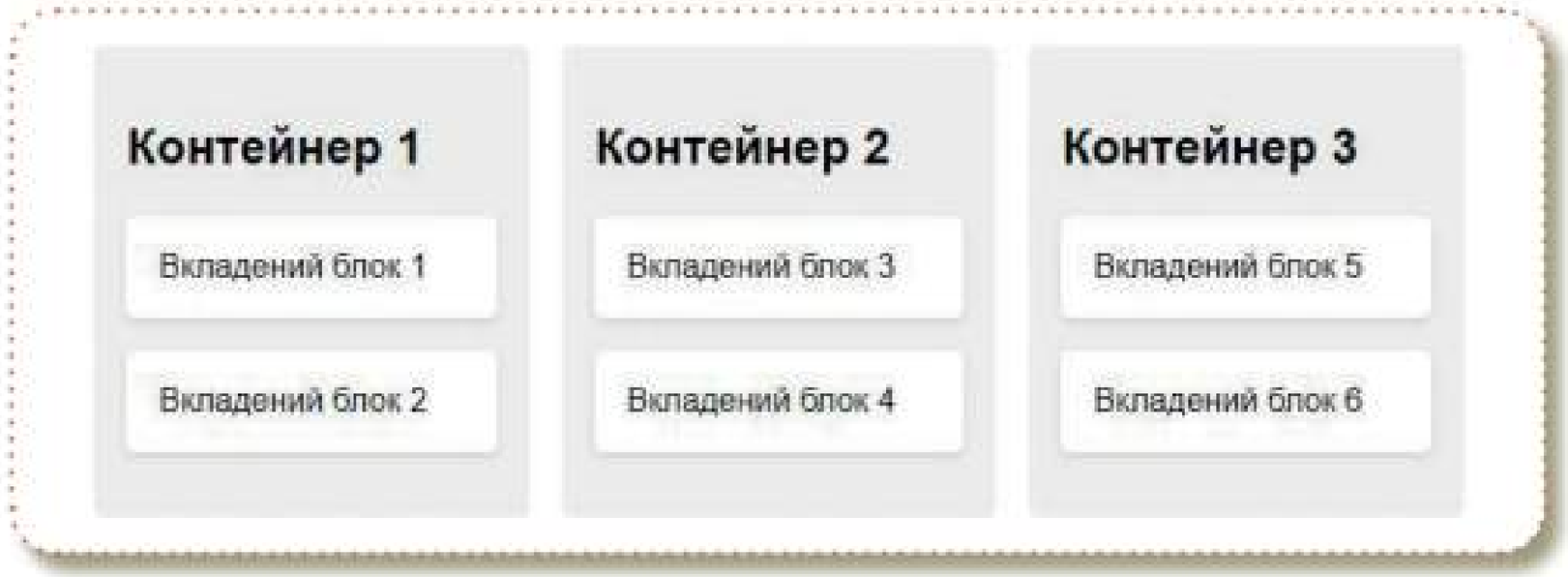
Дочірній елемент — елемент, який перебуває всередині іншого елемента. Наприклад, у конструкції:

```
<div>

   <p>Це контейнер, який містить параграф</p>

</div>,

<p> є дочірнім елементом для <div>.
```



Проте деякі блоки **div**, які визначають певні частини вебсторінки, **HTML5** пропонує замінити спеціальними семантичними елементами:

header **section** **main** **footer** **nav** **aside**

<header>...</heaader>

- Контейнер для верхньої частини сторінки або секції.
- Зазвичай містить логотип, заголовки, навігаційні елементи.

<footer>...</footer>

- Контейнер для нижньої частини сторінки або секції.
- Містить копірайт, контактну інформацію.

<section>...</section>

- Контейнер для розділів документа.
- Містить тематично пов'язаний контент.

<nav>...</nav>

- Контейнер для навігаційного меню.
- Містить посилання на інші сторінки або розділи.

<main>...</main>

- Контейнер для основного контенту сторінки.
- Містить основний унікальний контент сторінки.

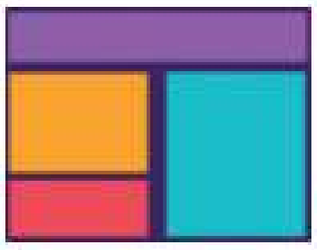
<aside>...</aside>

- Контейнер для бокового контенту, який доповнює основний.
- Може містити рекламу, додаткові посилання.

Модуль гнучких блоків (Flexbox)

Модуль гнучких блоків (Flexbox) — це CSS-технологія, яка дозволяє створювати гнучкі та адаптивні макети. *Flexbox* забезпечує ефективне вирівнювання і розподіл простору між елементами в контейнері, навіть коли їхні розміри невідомі або динамічно змінюються.

Основні концепції Flexbox

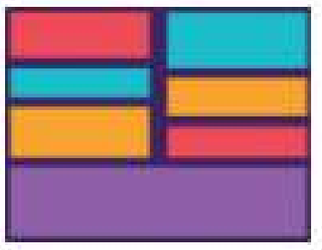


Flex Container

Елемент з властивістю **`display: flex`**. Він стає контейнером для своїх дочірніх елементів (flex items).

Flex Items

Дочірні елементи **flex** контейнера.



Основні властивості Flex Container

flex-direction

визначає, в якому напрямку контейнер хоче складати елементи (`row`, `row-reverse`, `column`, `column-reverse`)




```
display: flex;

flex-direction: column;
```

Розміщує елементи по вертикалі (згори вниз)

```
display: flex;

flex-direction: row;
```



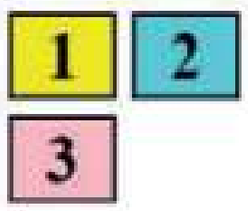
Розміщує елементи горизонтально (зліва направо)

flex-wrap

визначає, чи повинні елементи переноситися на новий рядок (`nowrap`, `wrap`, `wrap-reverse`)

```
display: flex;

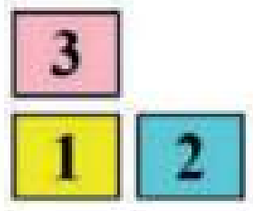
flex-wrap: wrap;
```



Визначає, що елементи будуть переноситись на новий рядок, якщо не поміщатимуться в одному рядку

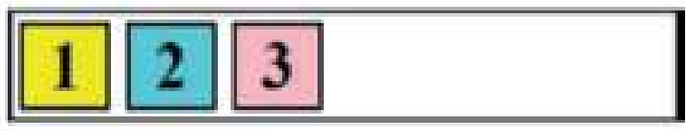
```
display: flex;

flex-wrap: wrap-reverse;
```



Визначає, що елементи будуть переноситись на новий рядок, якщо не поміщатимуться в одному рядку, тільки у зворотному порядку

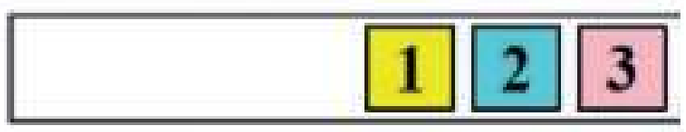
justify-content вирівнює елементи відносно головної осі (flex-start, flex-end, center, space-between, space-around)




```
display: flex;

justify-content: flex-start;
```

вирівнює елементи на початку контейнера (за замовчуванням)




```
display: flex;

justify-content: flex-end;
```

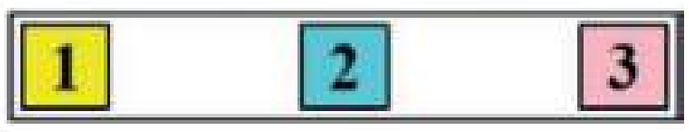
вирівнює елементи в кінці контейнера




```
display: flex;

justify-content: flex-center;
```

вирівнює елементи в центрі контейнера




```
display: flex;

justify-content: space-between;
```

вирівнює елементи по ширині контейнера з рівними пропусками між елементами

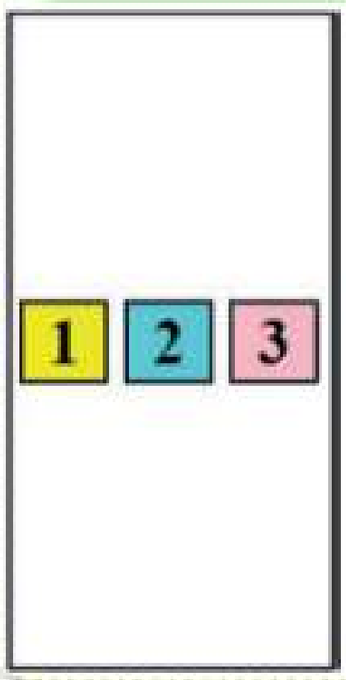



```
display: flex;

justify-content: space-around;
```

вирівнює елементи по ширині контейнера з пропусками між елементами та межами контейнера

align-items вирівнює елементи відносно перпендикулярної осі (stretch, flex-start, flex-end, center, baseline)




```
display: flex;

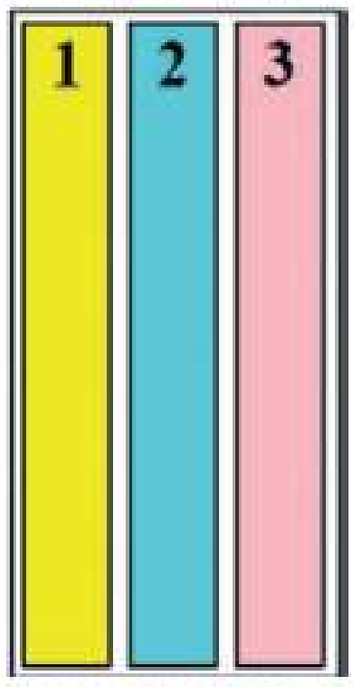
align-items: center;
```

Вирівнює елементи по середині контейнера

```
display: flex;

align-items: stretch;
```

Розтягує елементи так, щоб заповнити весь контейнер по висоті



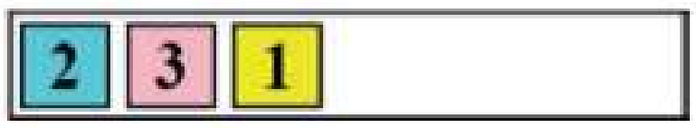
Основні властивості Flex Items

Якщо для контейнера задати властивість **display:flex**, то й самі елементи (дочірні) також стають гнучкими. Тому основні властивості **Flex Items** визначають, як кожен елемент поводиться всередині flex-контейнера.

order

Визначає порядок елемента у контейнері. Значення за замовчуванням: 0 (нумерація елементів починається з 0).

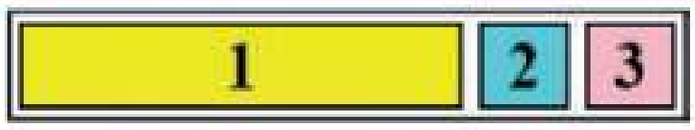
```
.first {
    order: 2;
}
```



Розміщає перший елемент на позицію 2.

flex-grow

Визначає здатність елемента розтягуватися, щоб зайняти вільний простір. Значення за замовчуванням: 0 (елемент не розтягується).

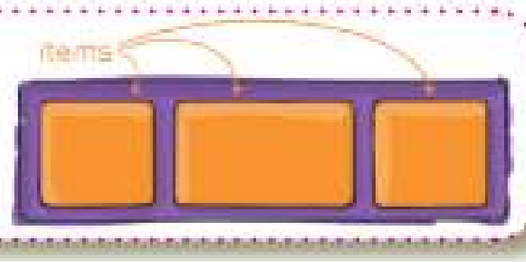



```
.first {
    flex-grow: 1;
}
```

Розтягує перший елемент так, щоб було заповнено всю ширину контейнера. Якщо ця властивість є у кількох елементів — вони порівну розтягуються всередині контейнера.

align-self

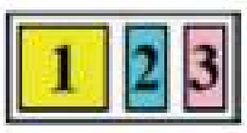
Перевизначає значення **align-items** для конкретного елемента



flex-shrink

Дозволяє елементові зменшуватися при недостатньому просторі. Значення за замовчуванням: 1 (елемент може зменшуватися).

```
.first {
    flex-shrink: 0;
}
```



При зменшенні контейнера — перший елемент не зменшується, а інші, навпаки, зменшують свою ширину.

flex-basis

Визначає початковий розмір елемента перед тим, як застосовуються властивості **flex-grow** і **flex-shrink**. Може бути заданий у пікселях, відсотках або інших одиницях.



Макет сайту

Макет сайту — це схема або план розташування елементів на вебсторінці. Від правильного макета залежить зручність користувача, естетичний вигляд та функціональність вебсайту.

Основні компоненти макету

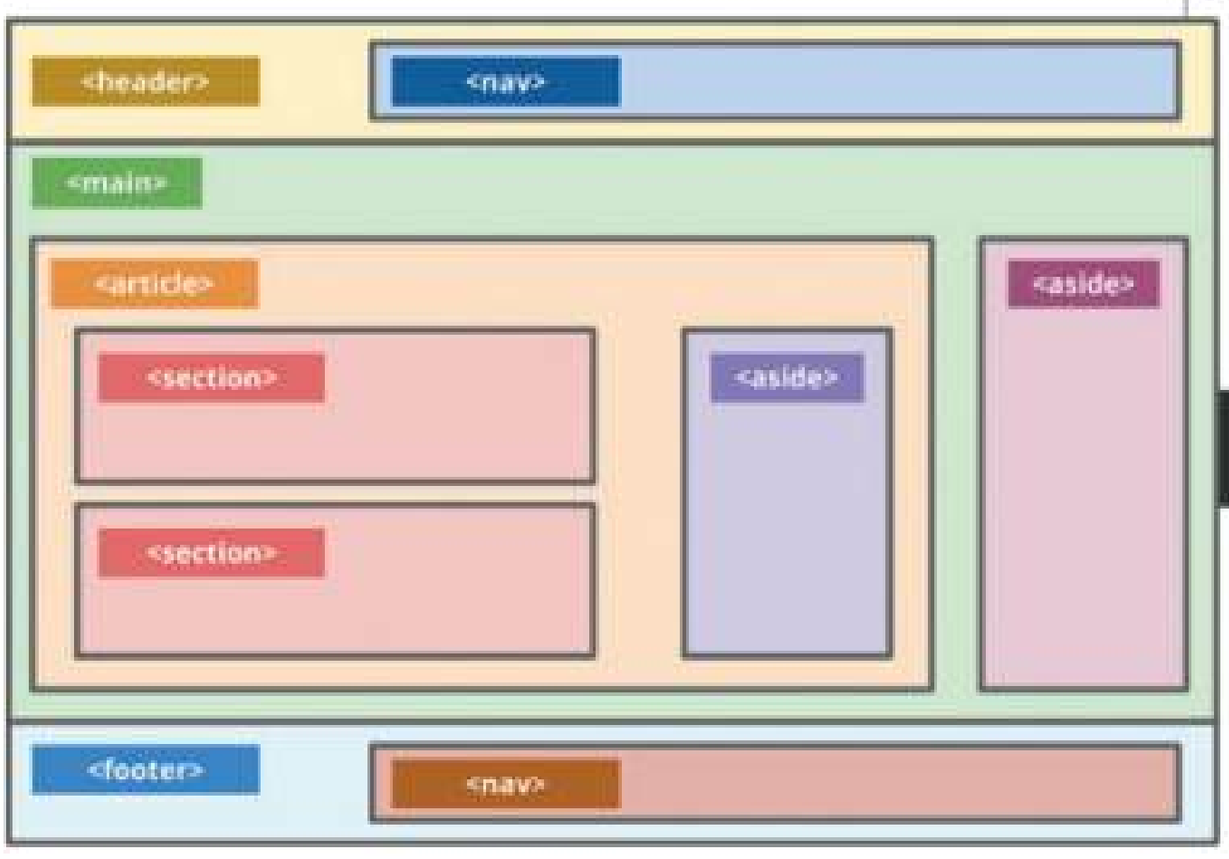
Header (Заголовок)
Містить логотип, назву сайту, навігаційне меню.
Розташовується у верхній частині сторінки.

Navigation (Навігація)
Містить покликання на основні розділи сайту.
Може бути частиною заголовка або розташовуватись окремо.

Main Content (Основний контент)
Містить основний вміст сторінки, такий як статті, зображення, відео.
Є центральною частиною сторінки.

Sidebar (Бокова панель)
Використовується для додаткового контенту, такого як новини, покликання, реклама.
Розташовується збоку від основного контенту.

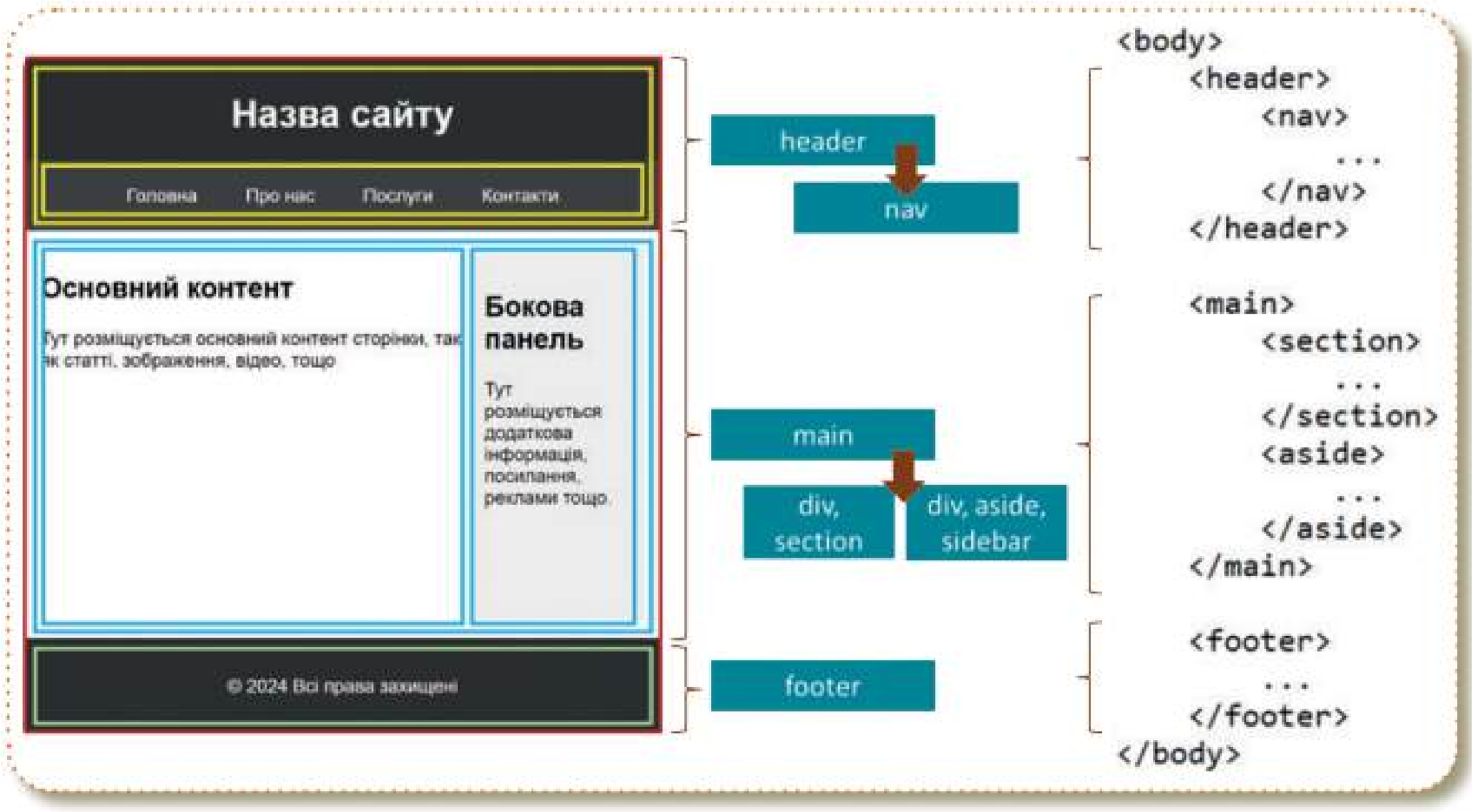
Footer (Підвал)
Містить інформацію про авторські права, контакти, додаткові покликання.
Розташовується внизу сторінки.





Як же розмістити блоки в такому порядку, як на прикладі?

Такі теги, як ** **, **<header>**, **<section>** тощо, та вкладеність цих блоків разом із гнучкими CSS-властивостями дозволяють створити різноманітні, організовані та естетично привабливі макети вебсторінок.



При цьому за допомогою властивостей **Flexbox** можна різноманітно розмістити блоки на вебсторінці.

```
body {
        display: flex;
        flex-direction: column;
    }
    main {
        display: flex;
    }
```

Цільова аудиторія

Цільова аудиторія — це група людей, на яку орієнтований вебсайт.



Визначення цільової аудиторії допомагає розробникам і дизайнерам створити контент і функціональність, які будуть корисні та привабливі для цих користувачів.

Ключові аспекти визначення цільової аудиторії

1. **Демографічні дані**
   Вік, стать, місце проживання, освіта, дохід
2. **Інтереси та потреби**
   Хобі, професія, проблеми, які потрібно розв'язати
3. **Поведінка користувачів**
   Частота відвідувань, улюблені пристрої, способи взаємодії з контентом

Технічне завдання

Технічне завдання (ТЗ) — це документ, який детально описує вимоги до розробки вебсайту.

ТЗ допомагає організувати роботу команди розробників і дизайнерів, а також забезпечити виконання всіх вимог замовника.

Основні компоненти технічного завдання

Функціональні вимоги
Опис функціональності сайту (форми, бази даних, інтеграція з іншими системами).

Мета і завдання
Опис мети створення сайту та завдань, які він має виконувати.

Дизайн і макет
Опис візуального стилю, кольорової гами, типографіки, макета сторінок.

Технічні вимоги
Опис платформи, мови програмування, системи управління контентом (CMS), хостингу.

Терміни та етапи розробки
Графік виконання робіт, контрольні точки, дедлайни.

Аналіз відвідування сайту

Аналіз відвідування сайту дозволяє оцінити ефективність роботи веб-ресурсу, зрозуміти поведінку користувачів і виявити слабкі місця для їх подальшого поліпшення.

Основні інструменти для аналізу

Google Analytics

найпопулярніший інструмент для аналізу трафіку вебсайтів

Hotjar

інструмент для аналізу поведінки користувачів, зокрема теплові карти, записи сесій

Основні метрики для аналізу

Відвідуваність

Кількість унікальних відвідувачів, кількість сеансів, кількість переглядів сторінок.

Джерела трафіку

Органічний пошук, прямі заходи, реферальний трафік, соціальні мережі.

Поведінка користувачів

Середній час на сайті, кількість сторінок за сеанс, показник відмов (bounce rate).

Конверсії

Кількість заповнених форм, кількість покупок, інші цільові дії.

Приклад аналізу відвідування сайту за допомогою Google Analytics

Налаштування облікового запису Google Analytics

1. Створіть обліковий запис на сайті **Google Analytics**.
2. Додайте сайт у свій обліковий запис.
3. Вставте отриманий трекінговий код на всі сторінки вашого сайту.

Аналіз метрик

1. Перейдіть до розділу **Аудиторія** для аналізу кількості відвідувачів та їхньої демографії.
2. Перейдіть до розділу **Атракція** для аналізу джерел трафіку.
3. Перейдіть до розділу **Поведінка** для аналізу поведінки користувачів на сайті.
4. Перейдіть до розділу **Конверсії** для аналізу виконаних цільових дій.

Практична робота №14.

Завдання 1. Вирівнювання елементів по центру. Створити контейнер, в якому розмістити три блоки в ряд по центру як вертикально, так і горизонтально.

Порядок виконання



1. Створіть блок **div** з класом **.container**, у якому розмістіть три блоки **div** з однаковим класом **.box**.
2. Для тегу **body** задайте такі властивості: висота **100vh**, вирівнювання дочірніх елементів по центру як по вертикалі, так і горизонталі:

```
body {
      height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
}
```

3. Для контейнера з класом **.container** задайте такі властивості: вирівнювання дочірніх елементів по центру як по вертикалі, так і горизонталі, ширина контейнера 50%, висота контейнера 30%, колір фону **lightblue**.
4. Для контейнерів з класом **.box** задайте такі властивості: вирівнювання дочірніх елементів по центру як по вертикалі, так і горизонталі, ширина та висота контейнера 50px, колір фону **coral**, зовнішні відступи 10px, розмір тексту 28px, колір тексту білий.

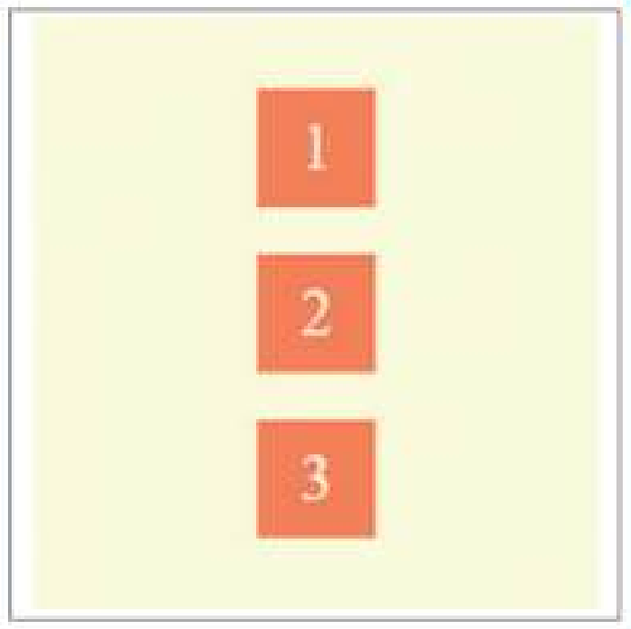
Завдання 2. Розміщення елементів з простором між ними. Створити контейнер, в якому три блоки будуть розміщені з рівним простором між ними.



Порядок виконання

1. Для тегу **body** та контейнерів з класом **.box** задайте властивості, аналогічні до властивостей цих контейнерів із завдання 1.
2. Для контейнера з класом **.container** задайте такі властивості: вирівнювання дочірніх елементів з рівними пропусками між ними по ширині, ширина контейнера 80%, колір фону **lightgreen**, внутрішні відступи 20px.

Завдання 3. Розміщення елементів у стовпчик. Створити контейнер, в якому три блоки будуть розміщені у стовпчик і вирівняні по центру.



Порядок виконання

1. Для тегу **body** та контейнерів з класом **.box** задайте властивості, аналогічні до властивостей цих контейнерів із завдання 1.
2. Для контейнера з класом **.container** задайте такі властивості: вирівнювання дочірніх елементів по центру та розміщених у стовпець, ширина контейнера 200px, колір фону **lightyellow**, внутрішні відступи 20px.

Завдання 4. Створення навігаційної панелі.



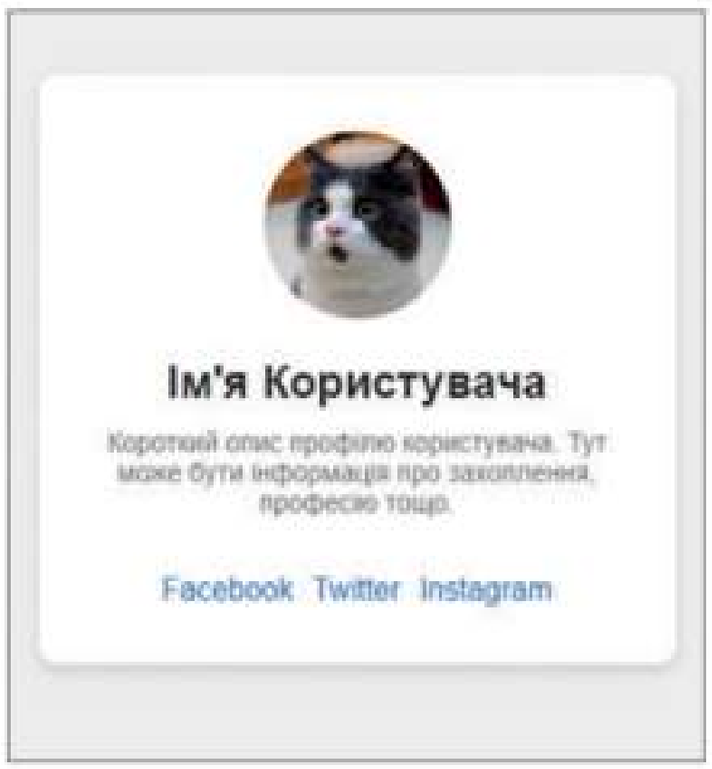
Порядок виконання

1. Створіть html-структуру таким чином:

```
<body>
    <nav class="nav">
        <div class="logo">Логотип</div>
        <div class="nav-links">
            <a class="nav-links-item" href="#">Головна</a>
            <a class="nav-links-item" href="#">Про нас</a>
            <a class="nav-links-item" href="#">Послуги</a>
            <a class="nav-links-item" href="#">Контакти</a>
        </div>
    </nav>
</body>
```

2. Для тегу **body** задайте такі властивості: обнуліть зовнішні відстані та задайте шрифт **Arial**.
3. Для контейнера з класом **.nav**, який містить всередині два блоки, задайте такі властивості: вирівнювання дочірніх елементів з рівними пропусками між ними по ширині та по центру по горизонталі, колір фону #333333, колір тексту білий та внутрішні відступи 20px.
4. Для елементів з класом **.nav-links-item** задайте такі властивості: забрати підкреслення, задати колір тексту білий та внутрішні відступи по 10px зверху та знизу та 15px справа і зліва.
5. Задайте властивості при наведенні на елементи навігації, а саме: зміна кольору фону на #575757, заокруглення елемента 5px.

Завдання 5. Створення картки профілю.



Порядок виконання

1. Оцініть фото завдання, розділивши його на блоки та елементи, з яких він складається, та задайте структуру у файлі **html**.




```
<div class="profile-card">
  <img src="cat.jpg" alt="Фото профілю" class="profile-image">
  <div class="profile-info">
    <h2 class="profile-name">Ім’я Користувача</h2>
    <p class="profile-description">Короткий опис профілю користувача.
        Тут може бути інформація про захоплення, професію тощо.</p>
    <div class="social-links">
        <a href="#" class="social-link">Facebook</a>
        <a href="#" class="social-link">Twitter</a>
        <a href="#" class="social-link">Instagram</a>
    </div>
  </div>
</div>
```

2. Для тегу **body** задайте такі властивості: висота 100vh, вирівнювання дочірніх елементів по центру як по вертикалі, так і горизонталі, задайте шрифт **Arial** та колір фону #f0f0f0.

3. Для контейнера з класом **.profile-card** задайте такі властивості: вирівнювання дочірніх елементів по центру та розміщених у стовпець, колір фону білий, колір тексту білий, внутрішні відступи 30px зверху та знизу і 20px зліва та справа, ширина контейнера 300px, заокруглення 10px та тінь рамки зі значенням 0 8px 16px rgba(0, 0, 0, 0.3).
4. Для зображення з класом **.profile-image** задайте такі властивості: встановити ширину та висоту зображення по 100px, заокруглення 50%, зовнішній відступ зверху 20px та запишіть властивість **object-fit: cover**; яка дозволяє розмістити зображення зі збереженими пропорціями.
5. Для контейнера з класом **.profile-info**, який містить заголовок, абзац та блок із соціальними мережами, встановіть такі властивості: вирівнювання дочірніх елементів по центру та розміщених у стовпець та вирівнювання тексту по центру.
6. Для заголовка з класом **.profile-name** задайте такі властивості: обнулити зовнішні відступи, розмір тексту 28px, колір #333333.

   Для абзацу з класом **.profile-description** задайте такі властивості: розмір тексту 14px, колір тексту #777777 та зовнішні відступи зверху та знизу по 10px і справа та зліва 0.
7. Для блоку із соціальними мережами з класом **.social-links** встановіть такі властивості: **display:flex** та зовнішній відступ зверху 20px.
8. Для елементів з класом **.social-link** задайте такі властивості: колір тексту #007bff, прибрати всі текстові декорації (підкреслення) і внутрішні відступи 0 зверху і знизу та 10px зліва та справа.
9. При наведенні на соціальну мережу задати текстову декорацію «підкреслення».











Тема 17 **Поняття хостингу. Інструменти координації роботи команди (Scrum, Kanban тощо). Система контролю версій**

Ключові поняття

Хостинг
Інструменти координації роботи команди
Методи керування проєктами
Система контролю версій

Хостинг

Хостинг — це послуга, яка дозволяє розміщувати ваш вебсайт в інтернеті, щоб його могли побачити інші користувачі.



Компанії, які надають ці послуги, називаються хостинг-провайдерами. Вони забезпечують простір на сервері, де зберігаються файли вебсайту, і доступ до інтернету для цих файлів.

Хостинг необхідний для того, щоб ваш сайт був доступний в інтернеті, працював безперебійно і швидко завантажувався. Це основа для будь-якого вебсайту, незалежно від його розміру і призначення.

Види хостингу

Віртуальний хостинг (Shared Hosting) — це тип вебхостингу, при якому кілька вебсайтів розміщуються на одному сервері.

Це один з найпопулярніших і найдоступніших способів хостингу, особливо для новачків і невеликих проєктів. Усі ресурси сервера, такі як оперативна пам'ять, процесор і дисковий простір, розподіляються між усіма користувачами.

Переваги:

- *Низька вартість.* Це найдешевший спосіб розміщення сайту в інтернеті.
- *Простота налаштування.* Простий процес налаштування з готовими рішеннями для швидкого запуску сайту.
- *Технічна підтримка.* Провайдери надають підтримку, що важливо для початківців.
- *Зручні панелі управління.* Інтуїтивно зрозумілі інтерфейси для управління сайтом, базами даних та іншими сервісами.

Недоліки:

- *Обмежені ресурси.* Спільне використання ресурсів сервера може призвести до зниження продуктивності сайту.
- *Вплив сусідів.* Продуктивність вашого сайту може залежати від навантаження інших сайтів на сервері.

- *Обмежені можливості налаштувань.* Обмежений доступ до серверних налаштувань і можливостей оптимізації.
- *Безпека.* Підвищений ризик безпеки через спільне використання серверних ресурсів.

Приклади провайдерів віртуального хостингу: Bluehost, HostGator, SiteGround, A2 Hosting, DreamHost.

Віртуальний приватний сервер (VPS Hosting) — це тип вебхостингу, при якому один фізичний сервер розділяється на кілька віртуальних серверів, кожен з яких має виділені ресурси і працює як окремий сервер. Це забезпечує більшу гнучкість, продуктивність і безпеку порівняно з віртуальним хостингом (**Shared Hosting**).

Переваги:

- *Вища продуктивність.* Виділені ресурси забезпечують стабільну та високу продуктивність навіть при високих навантаженнях.
- *Повний контроль.* Доступ до кореневих прав (**root access**) дозволяє встановлювати будь-яке програмне забезпечення і налаштовувати сервер під свої потреби.
- *Безпека.* Вища безпека завдяки ізоляції від інших користувачів на тому ж фізичному сервері.
- *Гнучкість та налаштування.* Можливість вибору операційної системи, серверного програмного забезпечення та конфігурації сервера.
- *Скалабельність.* Легке збільшення ресурсів сервера у разі потреби.

Недоліки:

- *Вища вартість.* Дорожчий у порівнянні з віртуальним хостингом, хоча і дешевший, ніж виділений сервер.
- *Необхідність технічних знань.* Вимагає більше технічних знань для налаштування та управління сервером.
- *Управління.* Потребує більше часу та зусиль для налаштування та підтримки порівняно з віртуальним хостингом.

Приклади провайдерів VPS Hosting: DigitalOcean, Vultr, Linode, Amazon Web Services (AWS), Google Cloud Platform (GCP), Hostinger.

Виділений сервер (Dedicated Hosting) — це тип вебхостингу, при якому користувач отримує в оренду цілий фізичний сервер.

На відміну від віртуального хостингу або VPS, виділений сервер надає всі ресурси сервера лише одному клієнту. Це забезпечує максимальну продуктивність, безпеку і контроль над сервером.

Переваги:

- *Висока продуктивність.* Серверні ресурси не діляться з іншими користувачами, що забезпечує стабільну продуктивність навіть при високих навантаженнях.
- *Повний контроль і гнучкість.* Можливість налаштовувати сервер під конкретні потреби, встановлювати будь-яке програмне забезпечення і контролювати всі аспекти роботи сервера.
- *Підвищена безпека.* Можливість реалізувати власні заходи безпеки, зокрема брандмауери, антивірусне ПЗ та інші засоби захисту.
- *Скалабельність.* Можливість додавати або замінювати апаратне забезпечення для збільшення ресурсів сервера відповідно до зростання потреб.

Недоліки:

- *Висока вартість.* Виділений сервер є найдорожчим типом хостингу через повну оренду фізичного сервера.
- *Необхідність технічних знань.* Управління виділеним сервером вимагає високого рівня технічних знань і досвіду.
- *Управління.* Потребує більше часу і ресурсів для адміністрування, обслуговування та підтримки сервера.

Приклади провайдерів виділеного хостингу: Bluehost, HostGator • SiteGround, InMotion Hosting, Liquid Web.

Хмарний хостинг (Cloud Hosting) — це тип вебхостингу, який використовує ресурси кількох серверів для розподілу навантаження і забезпечення максимальної продуктивності та надійності.

На відміну від традиційних типів хостингу, таких як віртуальний хостинг або виділений сервер, хмарний хостинг забезпечує масштабованість і гнучкість за рахунок використання обчислювальних потужностей хмарної інфраструктури.

Переваги:

- *Масштабованість.* Легке збільшення або зменшення ресурсів залежно від поточних потреб.
- *Висока доступність та надійність.* Відмова одного сервера не впливає на роботу сайту або додатка завдяки розподіленій інфраструктурі.
- *Оптимізація витрат.* Платите тільки за використані ресурси, що може бути більш економічно вигідним.
- *Продуктивність.* Висока швидкість і продуктивність завдяки розподіленим ресурсам.
- *Безпека.* Хмарні провайдери пропонують різні рівні безпеки, зокрема брандмауери, шифрування та інші заходи захисту.

Недоліки:

- *Складність управління.* Потребує технічних знань для налаштування та управління хмарним середовищем.
- *Залежність від інтернет-з'єднання.* Для доступу до хмарних ресурсів потрібне стабільне інтернет-з'єднання.
- *Безпека даних.* Хоча хмарні провайдери пропонують високу безпеку, дані все ще зберігаються на сторонніх серверах, що може викликати занепокоєння у деяких користувачів.

Приклади провайдерів хмарного хостингу: **Amazon Web Services** (AWS), **Google Cloud Platform** (GCP), **Microsoft Azure**, **DigitalOcean**, **Vultr**, **Linode**.

Хостинг для вебдодатків, відомий також як **Managed Hosting**, передбачає надання провайдером повного управління сервером і всіма супутніми послугами, що дозволяє клієнтам зосередитися на розробці та розгортанні додатків, не турбуючись про адміністрування та технічну підтримку інфраструктури.

Переваги:

- *Економія часу та зусиль.* Веброзробники можуть зосередитися на створенні та удосконаленні додатків, не відволікаючись на управління сервером.
- *Висока продуктивність.* Оптимізовані налаштування сервера та додаткові сервіси, такі як кешування і балансування навантаження, забезпечують швидку та стабільну роботу додатка.
- *Підвищена безпека.* Провайдери застосовують сучасні заходи безпеки для захисту даних і додатків від потенційних загроз.
- *Цілодобова підтримка.* Технічна підтримка доступна 24/7, що забезпечує швидке розв'язання будь-яких проблем.

Недоліки:

- *Вища вартість.* **Managed Hosting** зазвичай коштує дорожче, ніж інші види хостингу, через додаткові послуги управління і підтримки.
- *Обмежений контроль.* Менший контроль над сервером і конфігурацією, оскільки провайдер керує більшістю аспектів інфраструктури.
- *Залежність від провайдера.* Клієнт залежить від провайдера у питаннях управління і підтримки серверної інфраструктури.

Приклади провайдерів Managed Hosting WP Engine: Kinsta, Flywheel, SiteGround, Liquid Web.

Інструменти координації роботи команди

Координація роботи команди є важливою складовою успіху будь-якого проєкту. Інструменти координації роботи команди допомагають людям,

які працюють разом над спільним проєктом, ефективно організувати свою роботу, комунікувати та стежити за прогресом. Це особливо важливо, коли команда працює віддалено або розподілена по різних місцях.

Основні причини використання інструментів координації роботи команди

1. Організація завдань

Інструменти дозволяють розділити великий проєкт на менші завдання, призначити їх конкретним людям та стежити за їхнім виконанням.

2. Спілкування

Вони забезпечують швидкий і зручний спосіб спілкування між членами команди, щоб обговорювати питання, ділитися ідеями та усувати проблеми.

3. Відстеження прогресу

Інструменти допомагають стежити за тим, що вже зроблено, що ще потрібно зробити і хто за що відповідає. Це дозволяє уникнути плутанини і затримок.

4. Зберігання і обмін файлами

Команда може зберігати всі необхідні документи, файли і ресурси в одному місці, щоб усі мали до них доступ у будь-який час.

5. Планування

Інструменти дозволяють планувати робочий час, встановлювати дедлайни та організовувати зустрічі, щоб всі знали, коли і що потрібно зробити.

6. Спільна робота

Вони дозволяють декільком особам одночасно працювати над одним документом або проєктом, вносячи зміни в режимі реального часу.

Як це працює

- **Створення проєкту.** Команда створює проєкт в обраному інструменті й додає всі необхідні завдання.
- **Призначення завдань**. Завдання розподіляються між членами команди, і кожен отримує свою частину роботи.
- **Обговорення**. Учасники команди можуть обговорювати завдання, залишати коментарі та ділитися ідеями в спеціальних чатах або форумах.
- **Відстеження**. Всі можуть бачити, як просувається робота, які завдання виконані, а які ще потрібно зробити.
- **Обмін файлами.** Всі необхідні файли завантажуються в систему, щоб всі мали до них доступ.

Використання спеціальних інструментів дозволяє забезпечити ефективне спілкування, планування, відстеження прогресу та управління завданнями. Ось деякі популярні інструменти для координації роботи команди.

Trello

Візуально організує завдання у вигляді карток на дошках, дозволяючи легко відстежувати їх виконання.

Особливості:

- Візуальна організація завдань у вигляді карток на дошках.
- Легке переміщення завдань між колонками.
- Інтеграція з іншими сервісами, такими як **Slack**, **Google Drive**.
- Можливість призначення відповідальних, додавання дедлайнів, чеклістів та коментарів.

Використання: Підходить для малих та середніх команд для управління завданнями і проєктами.

Asana

Дозволяє структурувати проєкти, призначати завдання і стежити за прогресом команди.

Особливості:

- Структурування проєктів у вигляді списків, дошок або календарів.
- Відстеження прогресу завдань і проєктів.
- Інтеграція з багатьма іншими інструментами, такими як **Slack**, **Google Drive**, **Dropbox**.
- Можливість створення шаблонів для типових проєктів.
- Детальні звіти та аналітика.

Використання: Підходить для команд будь-якого розміру і складності проєктів.

Jira

Інструмент для управління проєктами, особливо популярний серед IT-команд і розробників.

Особливості:

- Підтримка методологій **Agile**, **Scrum** і **Kanban**.
- Управління задачами, беклогами, спринтами і релізами.
- Інтеграція з іншими інструментами **Atlassian**, такими як **Confluence і Bitbucket**.
- Потужні інструменти для звітності та аналітики.
- Можливість налаштування робочих процесів під конкретні потреби команди.

Використання: Підходить для великих команд, особливо розробників ПЗ.

Slack

Забезпечує миттєве спілкування через чати, де можна обговорювати робочі питання.



Особливості:

- Чати, канали та приватні повідомлення для різних обговорень.
- Інтеграція з багатьма іншими інструментами, такими як **Google Drive**, **Trello**, **Asana**.
- Можливість обміну файлами, зображеннями, документами.
- Підтримка відеоконференцій та дзвінків.
- Боти і автоматизація для полегшення рутинних задач.

Використання: Підходить для команд будь-якого розміру для організації ефективного спілкування.

Методи керування проєктами

Методи керування проєктами допомагають організувати й успішно завершити проєкти. Вони забезпечують чітку структуру і порядок, що дозволяє команді працювати ефективно та досягати поставлених цілей.

Основні причини використання методів керування проєктами

- **Планування роботи.** Методи допомагають детально спланувати всі етапи проєкту, визначити задачі, терміни виконання і ресурси.
- **Розподіл завдань.** Вони дозволяють чітко розподілити завдання між членами команди, щоб кожен знав, за що він відповідає.
- **Контроль за виконанням.** Методи допомагають відстежувати прогрес виконання завдань і вчасно виявляти проблеми.
- **Комунікація**. Вони сприяють ефективній комунікації всередині команди, що допомагає уникнути непорозумінь і затримок.
- **Якість роботи.** Методи керування проєктами забезпечують контроль якості на кожному етапі роботи.
- **Завершення проєкту.** Вони допомагають успішно завершити проєкт, досягнувши всіх поставлених цілей і забезпечивши виконання всіх вимог.

Як це працює

- **Початок проєкту.** Визначення цілей, завдань і ресурсів, необхідних для виконання проєкту.
- **Планування.** Детальний план робіт, розподіл завдань і встановлення термінів виконання.
- **Виконання.** Реалізація плану, виконання завдань і постійний контроль за прогресом.
- **Моніторинг.** Регулярна перевірка виконання завдань, виявлення та усунення проблем.
- **Завершення**. Завершення всіх робіт, перевірка якості та підбиття підсумків проєкту.

Існує кілька популярних методологій, кожна з яких має свої особливості та підходить для різних типів проєктів.

Waterfall (Каскадна модель)

Опис: Послідовний підхід до управління проєктами, де кожен етап повинен бути завершений перед початком наступного.

Основні етапи: Аналіз вимог, дизайн, розробка, тестування, впровадження, підтримка.



Переваги: Простота планування, чітка структура, добре підходить для проєктів з чітко визначеними вимогами.

Недоліки: Важко вносити зміни після завершення етапу, підходить тільки для проєктів з незмінними вимогами.

Agile (Гнучка методологія)



Опис: Ітеративний підхід, що дозволяє швидко адаптуватися до змін та зосереджуватися на постійному вдосконаленні продукту.

Основні принципи: Гнучкість, часті ітерації, взаємодія з клієнтом, крос-функціональні команди.

Переваги: Гнучкість та адаптивність до змін, регулярний зворотний зв'язок, висока якість продукту.

Недоліки: Може бути складним для великих команд, вимагає високого рівня співпраці та комунікації.

Scrum



Опис: Підмножина **Agile**, що фокусується на коротких ітераціях (спринтах) для швидкої розробки продукту.

Основні зустрічі: Планування спринту, Щоденний стендап, Огляд спринту, Ретроспектива спринту.

Переваги: Чітка структура і процес, регулярний зворотний зв'язок, швидка адаптація до змін.

Недоліки: Може бути складним для великих команд, потребує дисципліни і досвіду.

Kanban



Опис: Візуальний метод управління проєктами, що використовує дошку для відстеження прогресу задач.

Основні принципи: Візуалізація роботи, обмеження кількості задач у роботі, зосередження на потоці.

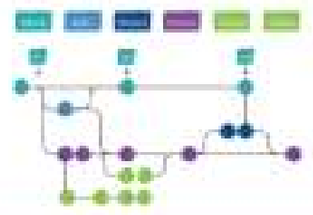
Основні елементи: Дошка **Kanban**, картки задач, колонки статусу.

Переваги: Простота і гнучкість, прозорість процесу, зосередження на постійному поліпшенні.

Недоліки: Може бути менш структурованим, ніж інші методології, потребує постійного моніторингу.

Система контролю версій

Система контролю версій (VCS) — це інструмент, який допомагає відстежувати зміни в коді або інших файлах проєкту.



Вона дозволяє зберігати історію змін, працювати над проєктом спільно з іншими і повернутися до попередніх версій файлів, якщо це необхідно.

Види систем контролю версій

Локальні VCS

Опис: Усі версії файлів зберігаються на локальному комп'ютері розробника.

Переваги: Простота використання.

Недоліки: Відсутність можливості співпраці, ризик втрати даних.

Централізовані VCS (CVCS)

Опис: Єдиний централізований сервер зберігає всі версії файлів. Розробники підключаються до цього сервера для отримання або надсилання змін.

Переваги: Легка співпраця, єдине джерело істини.

Недоліки: Вразливість до відмови сервера.

Розподілені VCS (DVCS)

Опис: Кожен розробник має повну копію репозиторію, зокрема всю історію змін.

Переваги: Висока надійність, незалежність від центрального сервера, можливість працювати офлайн.

Недоліки: Складніша конфігурація та управління.

Популярні системи контролю версій



Git

Найпопулярніша система контролю версій, яка використовується багатьма розробниками і командами по всьому світу. Вона забезпечує високу гнучкість і потужні інструменти для роботи з кодом.

Інструменти: GitHub, GitLab, Bitbucket.



Subversion (SVN)

Система контролю версій, яка використовується для збереження історії змін і спільної роботи над проєктами.



Mercurial

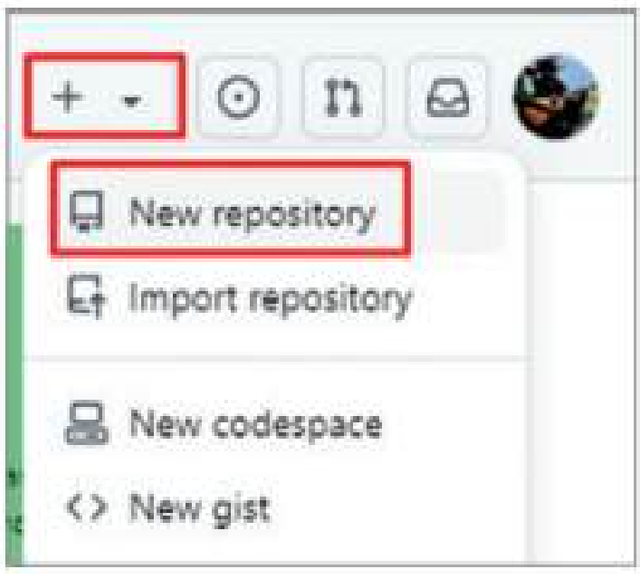
Децентралізована система контролю версій, яка надає простий та ефективний спосіб відстеження змін і спільної роботи.

Практична робота №15.

Завдання. Завантаження сайту-проєкту на **GitHub** через вебінтерфейс.

Порядок виконання

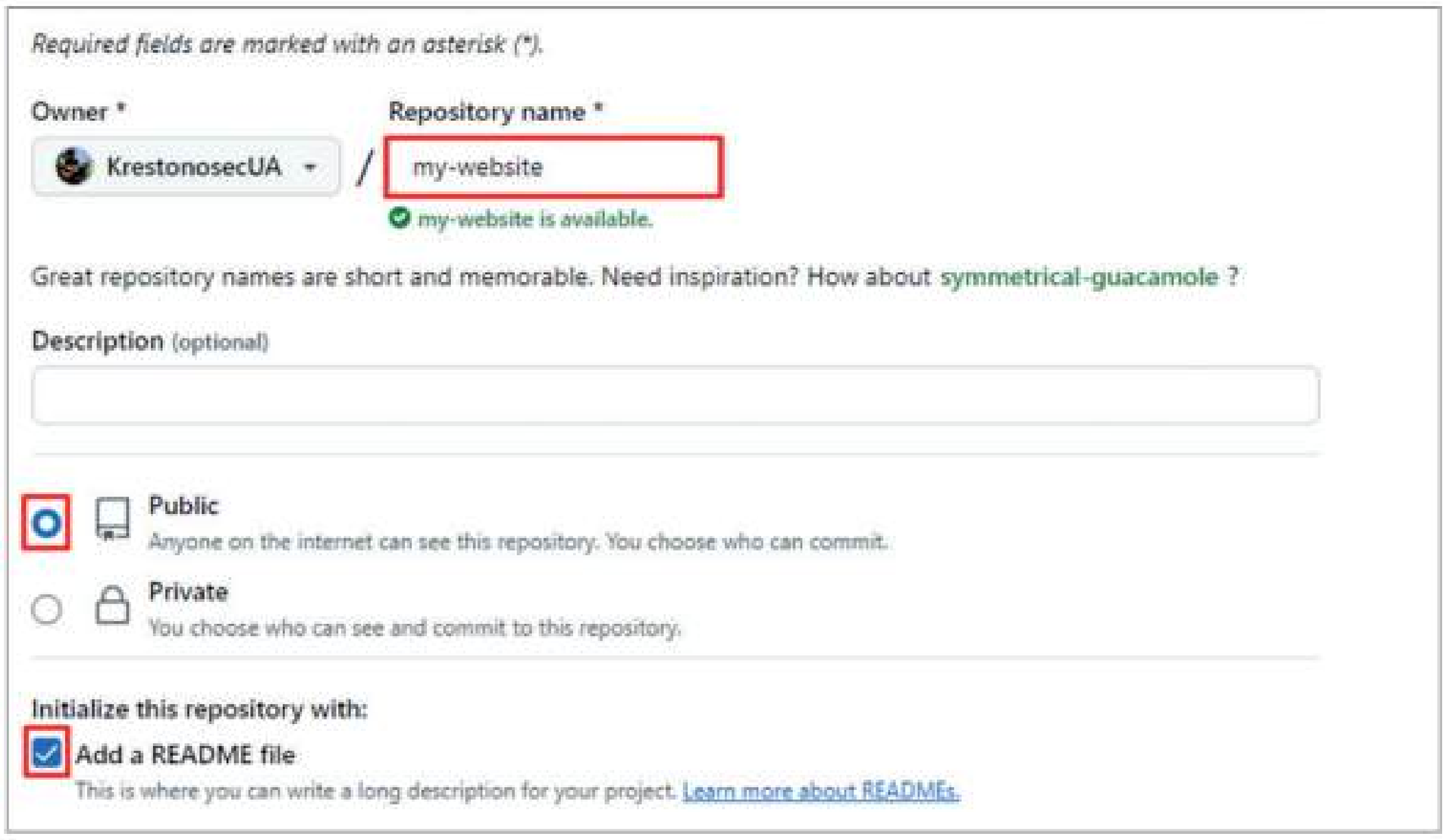
1. Відкрийте ваш браузер і перейдіть на сайт **GitHub**.
2. Увійдіть у свій обліковий запис або створіть новий, якщо його ще немає.
3. Натисніть на кнопку **New repository** (Новий репозиторій) у верхньому правому куті або натисніть на кнопку **New** у верхньому лівому кутку.



або



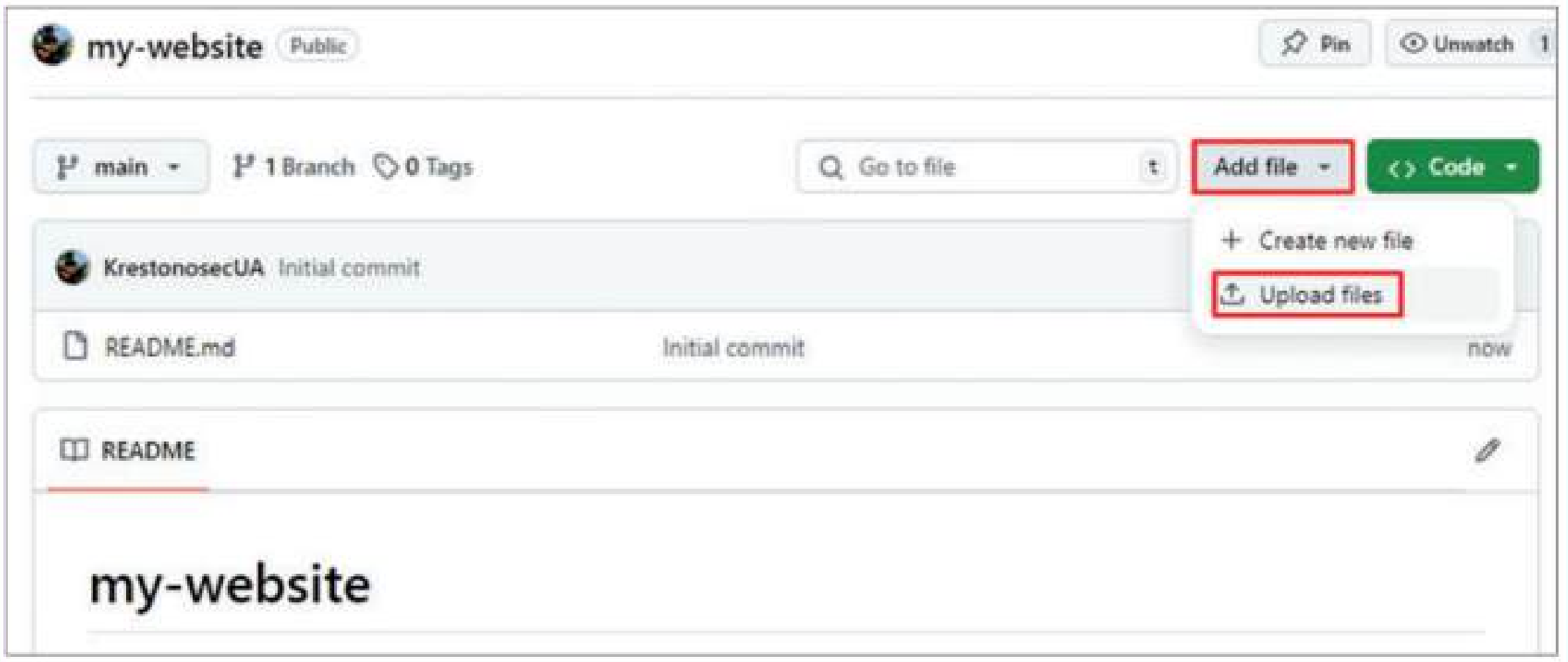
4. Заповніть інформацію про репозиторій:
   - **Repository name** (Назва репозиторію): Введіть ім'я для вашого проєкту, наприклад, **my-website**.
   - **Description** (Опис): Введіть короткий опис проєкту (необов'язково).
   - **Public**: Залиште репозиторій відкритим.
   - **Initialize this repository with a README** (Ініціалізувати репозиторій з README): Поставте галочку.



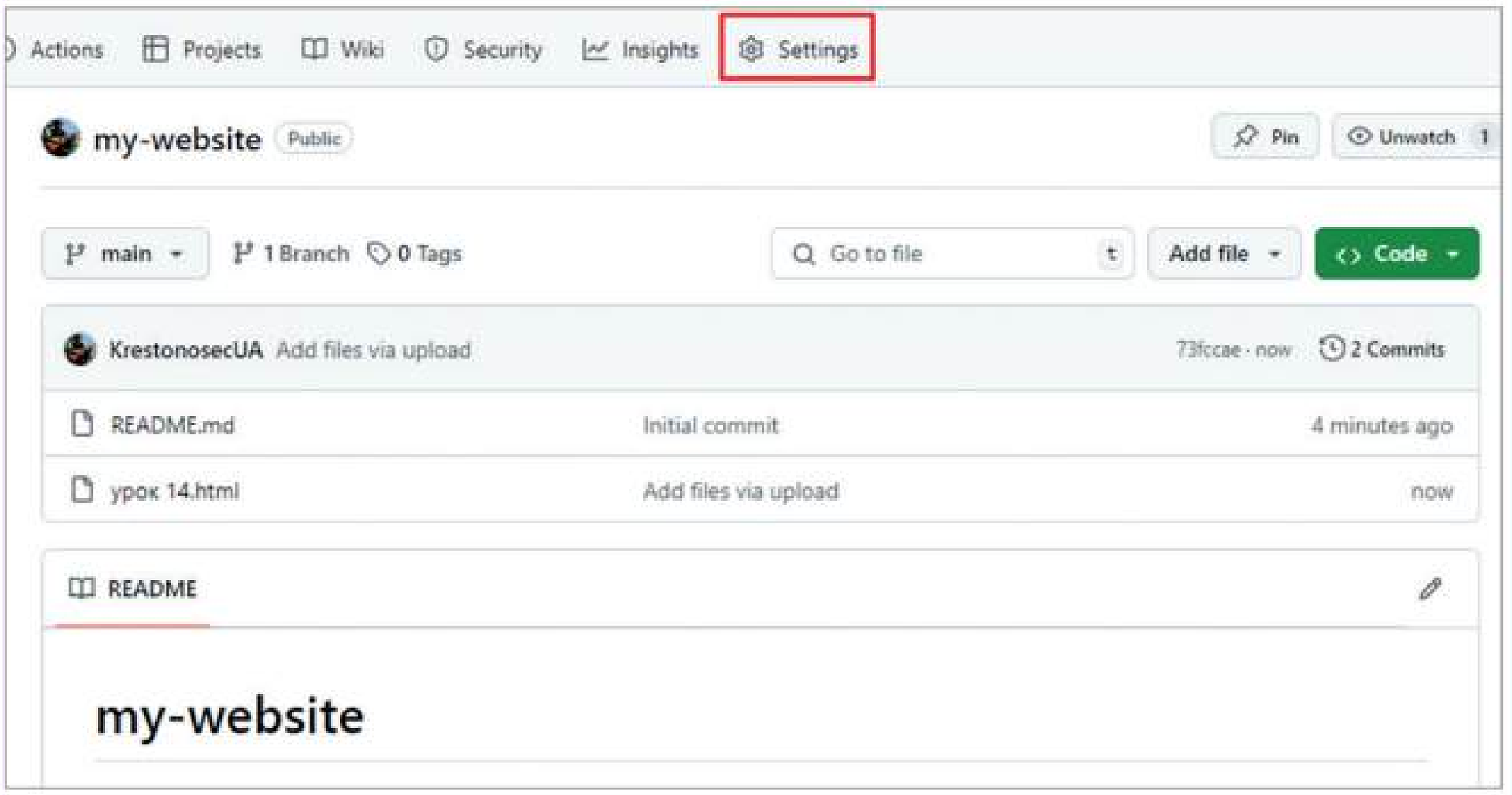
5. Натисніть на кнопку **Create repository** (Створити репозиторій).



6. Натисніть на кнопку **Add file** (Додати файл) і виберіть **Upload files** (Завантажити файли).

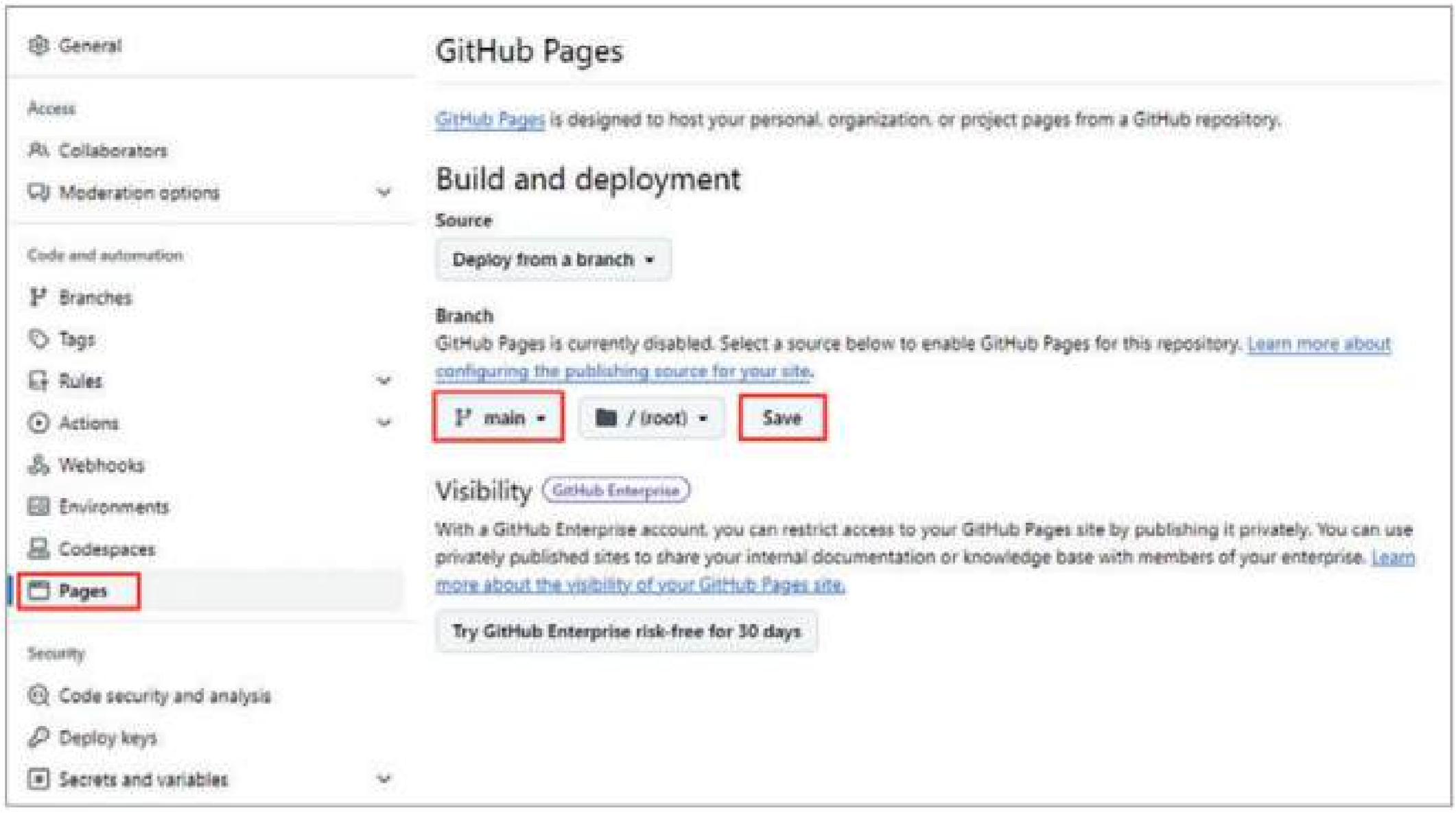


7. Перетягніть файли вашого сайту (наприклад, сайт **Books** або інший) до області завантаження або натисніть на кнопку **choose your files** (виберіть ваші файли) і виберіть файли на вашому комп'ютері.
8. Після завантаження файлів введіть коментар до коміту (наприклад, «Завантаження початкових файлів», необов'язково) у полі **Commit changes** (Коміт змін) та натисніть на кнопку **Commit changes** (Коміт змін).
9. Перейдіть на сторінку вашого репозиторію на **GitHub** та натисніть на вкладку **Settings** (Налаштування).

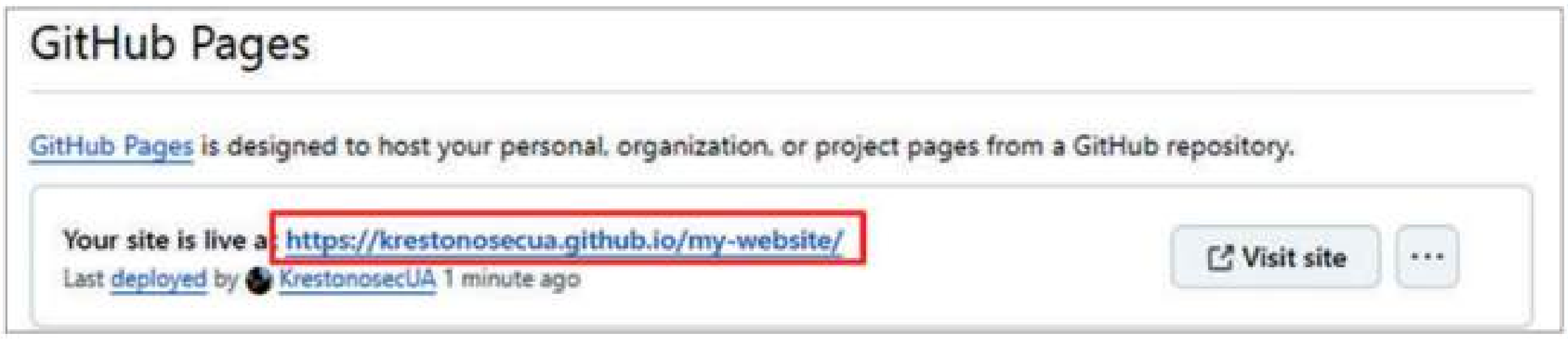


10. В лівому меню оберіть **Pages**.

11. У полі **Source** виберіть гілку **main** або **master** і натисніть **Save** (Зберегти).



12. Після збереження, зачекавши кілька хвилин, оновіть сторінку — і ви побачите **URL** вашого сайту в розділі **GitHub Pages**. Це буде покликання виду:



Ви успішно завантажили свій сайт на **GitHub** через вебінтерфейс і налаштували **GitHub Pages** для його перегляду. Тепер можете поділитися цим покликанням із вчителем та друзями, щоб вони могли переглянути ваш сайт.

Проєкт 1.
Створення односторінкового вебсайту (лендинг) «Книжковий Рай»

Теми 18–19

Мета проєкту: Створити привабливий та функціональний вебсайт для книжкової бібліотеки «Книжковий Рай», який дозволяє користувачам ознайомитися з бібліотекою, переглядати колекцію книг та підписатися на розсилку новин.

Зразок:



Етап 1. Аналіз вимог

1. Визначення цільової аудиторії:
   - Бібліофіли та любителі читання.
   - Учні, студенти та науковці, які шукають літературу для навчання.
   - Місцеві жителі, які шукають місце для читання та відпочинку.
2. Визначення основних функцій сайту:
   - Презентація бібліотеки.
   - Перегляд колекції книг.
   - Підписка на розсилку новин.
   - Контактна інформація та можливість зв'язатися з бібліотекою.

Етап 2. Проєктування

1. Створення макету або вивчення готового макету сайту:
   - Головна сторінка з логотипом, назвою та кнопкою для отримання додаткової інформації.
   - Секція з інформацією про бібліотеку та засновника.
   - Секція з рекомендованими книгами.
   - Футер з формою підписки на розсилку та контактною інформацією.
2. Визначення структури сторінки та розташування елементів:
   - Головна секція з логотипом та заголовком.
   - Секція з текстовим описом та зображеннями.
   - Секція з рекомендованими книгами.
   - Футер з формою підписки та контактами.

3. Вибір кольорової схеми та шрифтів:

- Основні кольори: зелений (#556b2f), світло-зелений (#8fbc8f), сірий (#666), білий (#fff).
- Шрифти: 'Lora' для заголовків, 'Roboto' для тексту.

Етап 3. Розробка

1. Підготовка до розробки:

- Створіть папку **Book heaven**, в якій буде папка **img** для зображень, файл **index.html** для розмітки сайту та файл **style.css** для стилів сайту.
- Знайдіть потрібні зображення для сайту та збережіть їх у папці **img** (або використовувати URL-адреси зображень) (або використати зображення із зразка у файлах до підручника).
- За потреби завантажте стилі **Roboto** та **Lora** (або використовувати посилання на шрифти без завантаження).

2. Написання HTML коду:

- Створення структури сторінки за допомогою HTML тегів.

Створіть стандартну структуру **<html>**, **<head>**, **<body>** та назвіть сайт «Книжковий Рай».

Оскільки макет складається з чотирьох секцій, створіть чотири блоки **main**, **section**, **section** та **footer**, відповідно з класами **section-main**, **second-section**, **third-section**, **footer**.

```
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Книжковий Рай</title>
</head>
<body>
    <main class="section-main">

    </main>

    <section class="second-section">

    </section>

    <section class="third-section">

    </section>

    <footer class="footer">

    </footer>
</body>
</html>
```

- Додавання контенту сайту (заголовки, текст, зображення та кнопки).

Блок **main**. Цей блок складається з чотирьох елементів: логотипу — зображення без фону з книжкою  , головного заголовку **h1** з текстом «Книжковий Рай», підзаголовка **h2** з текстом «Відкрий для себе наступну чудову книгу» та кнопки з текстом «Дізнатися більше». Для кожного елемента задайте класи, відповідно `section-main-logo`, `section-main-title`, `section-main-subtitle`, `section-main-button`.

```
<img class="section-main-logo" src="logo.png" alt="логотип">
        <h1 class="section-main-title">Книжковий Рай</h1>
        <h2 class="section-main-subtitle">Відкрий для себе наступну
чудову книгу</h2>
        <button class="section-main-button" type="button">Дізнатися
більше</button>
```

Блок **second-section**. Друга секція складається із зображення, яке презентує бібліотеку, та блоку **div** із інформацією. Задайте класи, відповідно **second-section-img** та **second-section-info**.

```
<img src="library.jpeg" alt="зображення бібліотеки"
class="second-section-img">
 <div class="second-section-info">
</div>
```

Блок з інформацією містить такі дочірні елементи: заголовок **h3** з текстом “ЛАСКАВО ПРОСИМО ДО КНИЖКОВОГО РАЮ” та класом **section-subtitle**; заголовок **h3** з текстом «Про нашу бібліотеку» та класом **section-title**; абзац з текстом «Ласкаво просимо до нашої бібліотеки, де ви знайдете широкий вибір книг для будь-якого віку та вподобань. Приходьте та насолоджуйтеся читанням у затишній атмосфері» та класом **second-section-text**; абзац з текстом «ІВАН ІВАНЕНКО — засновник Книжкового Раю» та класом **second-section-name**, причому текст ІВАН ІВАНЕНКО загорнути в тег **** з класом **second-section-name2**.

```
<p class="second-section-name"><span class="second-section-name2">
ІВАН ІВАНЕНКО</span> - засновник Книжкового Раю</p>
```

Блок **third-section**. Третя секція складається із двох заголовків **h3**, відповідно з текстами «РЕКОМЕНДОВАНІ КНИГИ» і «Наша колекція» з класами **section-subtitle** і **section-title** та блоку **div** з класом **third-section-books**.

```
    <h3 class="section-subtitle">РЕКОМЕНДОВАНІ КНИГИ</h3>
          <h3 class="section-title">Наша колекція</h3>
          <div class="third-section-books">

          </div>
```

Блок з книгами містить блоки-картки з класом **third-section-book** та інформацією про книжку. Кожна з карток складається із зображення з класом **book-img**, заголовка **h4** з класом **book-title** та двох абзаців з класами **book-author** та **book-price**. Заповніть 4 картки з інформацією про різні книги.

```
<div class="third-section-book">
                <img src="book1.png" alt="книга 1" class="book-img">
                <h4 class="book-title">Маленький принц</h4>
                <p class="book-author">Антуан де Сент Екзюпері</p>
                <p class="book-price">150 грн</p>
        </div>
```

Блок **footer**. Остання секція складається із заголовка **h3** з текстом «КНИЖКОВИЙ РАЙ» та класом **section-subtitle**, із заголовка **h3** з текстом «Приєднуйтесь до нашої спільноти» та двома класами **section-title footer-title** (деякі властивості візьмуться зі стилів уже вище описаного класу), з абзацу з текстом «Підпишіться на нашу розсилку, щоб отримувати останні новини про новинки, ексклюзивні пропозиції та багато іншого. Для запитів, будь ласка, зв'яжіться з нами за номером (+380) 123-456-789» та класом **footer-text**. Перше речення загорніть в тег **** з класом **footer-text-main**. Між реченнями поставити тег **
**.

```
<h3 class="section-title footer-title">Приєднуйтесь до нашої спільно-
ти</h3>
```

```
<h3 class="section-title footer-title">Приєднуйтесь до нашої спільно-
ти</h3>
        <p class="footer-text"><span class="footer-text-main">Підпи-
шіться на нашу розсилку, щоб отримувати останні новини про новинки,
ексклюзивні пропозиції та багато іншого.</span><br> Для запитів, будь
ласка, зв'яжіться з нами за номером (+380) 123-456-789</p>
```

Також у секції розташована форма з класом **footer-form**. Всередині форми містяться три поля для введення імені, електронної пошти і телефону з класом **form-input** та кнопка з текстом «Підписатися» та класом **footer-button**. Перед кожним полем є абзац з класом **form-title** для підпису цих полів відповідно назвами «Ім'я», «Електронна пошта», «Телефон».

```
<form action="" class="footer-form">
            <p class="form-title">Ім'я</p>
            <input type="text" id="name" class="form-input">
            <p class="form-title">Електронна пошта</p>
            <input type="email" id="email" class="form-input">
            <p class="form-title">Телефон</p>
            <input type="tel" id="phone" class="form-input">
            <button class="footer-button">Підписатися</button>
        </form>
```

3. Стилізація сторінки за допомогою CSS:

- Підключення файлу зі стилями до html-документу.
- Підключення в **head** шрифти **Roboto** та **Lora**.
- Встановлення загальних стилів для тіла сторінки.

Для тегу **body** встановіть такі властивості: обнулення зовнішніх відступів, шрифт **Roboto**, колір тексту #333, колір фону #eaeaea, розмір тексту 16px.

- Стилізація головної секції **section-main**.

Для класу **section-main** встановіть колір фону #556b2f, внутрішній відступ зверху 40px, внутрішній відступ знизу 120px та вирівняйте дочірні елементи по центру і в стовпець. Для логотипу встановіть ширину 160px. Для заголовка з класом **section-main-title** встановіть зовнішній відступ зверху 40px, зовнішній відступ знизу 20px, колір тексту #fff, шрифт 'Lora', розмір тексту 55px та напівжирний шрифт (font-weight: 700). Для підзаголовка з класом section-main-subtitle встановіть зовнішній відступ знизу 40px, колір тексту #fff, розмір тексту 22px. Для кнопки з класом section-main-button встановіть колір фону #8fbc8f, без рамки з округленням 25px, колір тексту #fff, напівжирний шрифт, всі літери великі, внутрішні відступи 12px зверху і знизу та 30px зліва і справа, при наведенні на кнопку курсор міняється на палець (cursor: pointer).

- Стилізація секції **second-section**.

Для класу **second-section** встановіть такі властивості: вирівнювання елементів по ширині контейнера з пропусками між елементами та межами контейнера та по центру по вертикалі, внутрішні відступи знизу та зверху по 100px. Для зображення встановіть ширину 500px, рамка 5px solid #556b2f та заокруглення 20px. Для блоку **second-section-info** встановіть зовнішній відступ зліва 30px та максимальну ширину 600 px (max-width: 600px). Для класу **section-subtitle** встановіть зовнішній відступ зверху 0 та зовнішній відступ знизу 10px, колір тексту #8fbc8f, розмір тексту 18px, напівжирний шрифт та всі літери великі. Для класу **section-title** встановіть зовнішній відступ зверху 0 та зовнішній відступ знизу 40px, колір тексту #556b2f, розмір тексту 32px, напівжирний шрифт та перші літери кожного слова великі. Для класу **second-section-text** встановіть зовнішній відступ 30px, колір тексту #666 та розмір тексту 18px. Для класу **second-section-name** встановіть колір тексту #556b2f та перші літери кожного слова великі. Для класу **second-section-name2** встановіть колір тексту #8fbc8f, напівжирний шрифт та всі літери великі.

- Стилізація секції **third-section**.

Для класу **third-section** встановіть вирівнювання дочірніх елементів по центру і в стовпець, колір фону #f5f5f5, внутрішні відступи зверху і знизу по 100px. Для блоку з класом **third-section-books** встановіть вирівнювання по центру, по горизонталі і по вертикалі, зовнішній відступ зверху 50px, та перенесення елементів на новий рядок, якщо вони не

поміщаються (**flex-wrap: wrap**). Для блоку **third-section-book** встановіть вирівнювання дочірніх елементів по центру і в стовпець і зовнішні відступи зверху і знизу по 10px та зліва і справа по 20px. Для зображень з класом **book-img** встановіть висоту 200px. Для класу **book-title** встановіть зовнішній відступ зверху 10px, знизу 5px, колір тексту #556b2f, розмір тексту 22px та напівжирний шрифт. Для класу **book-author** встановіть зовнішній відступ зверху і знизу по 2px та колір тексту #666. Для класу **book-price** обнуліть зовнішні відступи та встановіть колір тексту #8fbc8f, розмір тексту 20px та напівжирний шрифт.

- Стилізація секції **footer**.

Для класу **footer** встановіть вирівнювання дочірніх елементів по центру і в стовпець, колір фону #2f4f4f, внутрішні відступи зверху і знизу по 100px. Для класу **footer-title** встановіть колір тексту #fff. Для класу **footer-text** встановіть ширину елемента 600px, вирівнювання тексту по центру, зовнішній відступ зверху 0 та зовнішній відступ знизу 20px, колір тексту #8fbc8f та розмір тексту 20px. Для класу **footer-text-main** встановіть колір тексту #fff. Для форми з класом **footer-form** встановіть вирівнювання дочірніх елементів по центру і в стовпець. Для класу **form-title** встановіть зовнішній відступ зверху 20px, зовнішній відступ 10px, вирівнювання тексту по центру, колір тексту #fff, напівжирний шрифт та всі літери великі. Для полів з класом **form-input** встановіть ширину 400px, висоту 40px, зовнішній відступ знизу 10px, внутрішні відступи зверху і знизу 0 та зліва і справа 10px, колір фону #2f4f4f, колір тексту #fff та рамка 1px solid #fff із заокругленням 8px. Для кнопки з класом **footer-button** встановіть зовнішній відступ зверху 30px, внутрішні відступи зверху і знизу по 10px та зліва і справа 20px, колір фону #8fbc8f, колір тексту #fff з напівжирним шрифтом, без рамки із заокругленням 25px, при наведенні на кнопку курсор міняється на палець.

Етап 4. Тестування

1. Перевірка сумісності з різними браузерами:
   - Відкриття сайту в популярних браузерах (**Chrome**, **Opera**, **Edge**).
   - Перевірка коректного відображення всіх елементів.
2. Виправлення знайдених помилок:
   Аналіз та виправлення всіх знайдених помилок та неточностей.

Етап 5. Впровадження

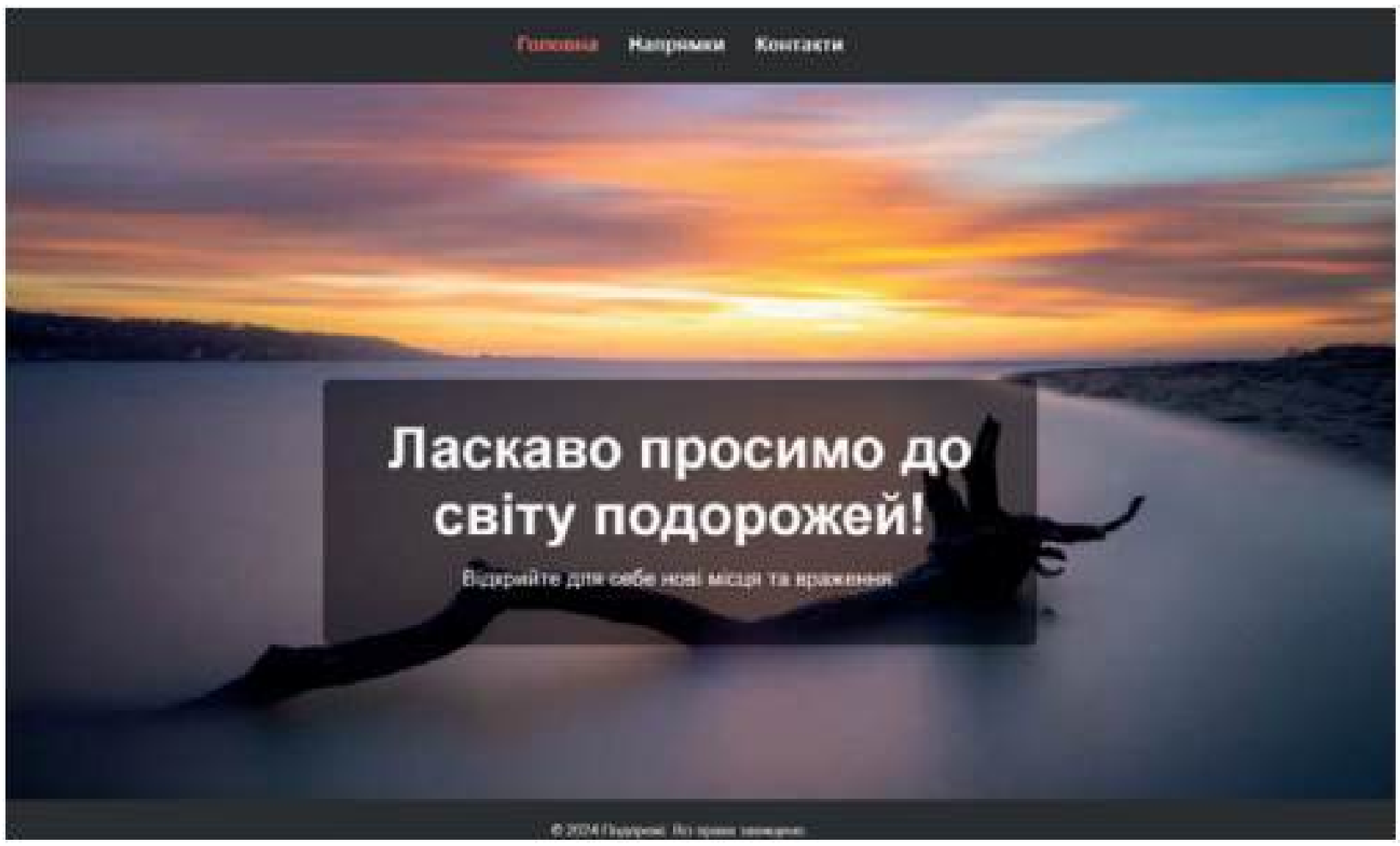
1. Розміщення сайту на **GitHub**:
   - Завантаження файлів сайту на сервер.
   - Надіслати **URL** опублікованого сайту вчителеві.

Проєкт 2.
Створення багатосторінкового вебсайту «Подорожі»

Теми 20–21

Мета проєкту: Створити багатосторінковий вебсайт на тему «Подорожі», який містить головну сторінку, сторінку напрямків та сторінку контактів. Сайт має бути інформативним, привабливим та зручним у використанні.

Зразок:



Етап 1. Аналіз вимог

1. Визначення цільової аудиторії:
   - Люди, які люблять подорожувати та шукають нові напрямки.
   - Туристичні агенції та організатори турів.
   - Люди, які шукають інформацію про популярні туристичні напрямки.
2. Визначення основних функцій сайту:
   - Презентація туристичних напрямків.
   - Надання контактної інформації.
   - Можливість підписки на розсилку новин та акцій.

Етап 2. Проєктування

1. Створення макету або вивчення готового макету сайту:
   - Головна сторінка з привітанням та інформацією про компанію.
   - Сторінка напрямків з популярними туристичними місцями.
   - Сторінка контактів з формою зворотного зв'язку.

2. Визначення структури сторінки та розташування елементів:

- Головна секція з логотипом та навігацією.
- Секція з привітанням.
- Секція з перевагами компанії.
- Секція напрямків з популярними туристичними місцями.
- Секція з формою зворотного зв'язку.
- Футер з контактною інформацією.

3. Вибір кольорової схеми та шрифтів:

- Основні кольори: темно-сірий (#333), білий (#fff), кораловий (#ff6f61), світло-сірий (#f4f4f4).
- Шрифти: 'Arial'.

Етап 3. Розробка

1. Підготовка до розробки:

- Створіть папку **Travels**, в якій буде папка **img** для зображень, файл **index.html** для розмітки головної сторінки, файл **destinations.html** для сторінки з напрямками популярних туристичних місць, файл **contact.html** для розмітки сторінки з формою зворотного зв'язку та файл **style.css** для стилів сайту.
- Знайдіть потрібні зображення для сайту та збережіть їх у папці **img** (або використовувати URL-адреси зображень) (або використати зображення із зразка у файлах до підручника).
- За потреби завантажте стилі **Helvetica** (або використовувати покликання на шрифти без завантаження).

2. Написання HTML-коду для головної сторінки:

Відкрийте файл **index.html** та створіть стандартну структуру **<html>**, **<head>, <body>** та назвіть сторінку «Подорожі — Головна».

Головна сторінка складається з чотирьох секцій. Тому створіть чотири блоки **header**, **section**, **section** та **footer** відповідно з класами **head**, **hero**, **intro**, **footer.**

```
<body>
    <header class="head">

    </header>
    <section class="hero">

    </section>
    <section class="intro">

    </section>
    <footer class="footer">

    </footer>
</body>
```

• Додавання контенту сторінки

Блок **head**. Цей блок складається із логотипу з класом **logo** та блоку меню **nav** з класом **menu**.

```
<img src="logo.png" class="logo" alt="логотип">
        <nav class="menu">

        </nav>
```

Блок меню містить список з класом **menu-links**, який складається трьох елементів з класом **menu-item**. Кожен елемент містить покликання з класом **menu-item-link**. Перший елемент покликається на сторінку **index.html** та має назву «Головна». Другий елемент покликається на сторінку **destinations.html** та має назву «Напрямки». Третій елемент покликається на сторінку **contact.html** та має назву «Контакти».

```
<ul class="menu-links">
                <li class="menu-item"><a href="index.html"
class="menu-item-link">Головна</a></li>
                <li class="menu-item"><a href="destinations.html"
class="menu-item-link">Напрямки</a></li>
                <li class="menu-item"><a href="contact.html"
class="menu-item-link">Контакти</a></li>
            </ul>
```

Блок **hero**. Друга секція складається із блоку з класом **hero-info**. Цей блок містить заголовок **h1** з класом **hero-title** та абзац з класом **hero-text**. Текст заголовка: «Ласкаво просимо до світу подорожей!» Текст абзацу: «Відкрийте для себе нові місця та враження!»

Блок **intro**. Третя секція складається із заголовка **h2**, з класом **intro-title** та текстом «Чому слід обрати нас?», а також з трьох блоків-карток з класом **card**. Кожна картка складається із зображення з класом **card-img** та блоку з класом **card-content**. Кожен блок **card-content** містить заголовок **h3** з класом **card-title** та список з класом **card-list** та елементами списку з класом **card-list-item**

Інформація для кожної з карток:

Найкращі напрямки для вашої подорожі:

- Екзотичні острови
- Історичні міста
- Культурні центри

Індивідуальні тури під ваші потреби:

- Сімейні тури
- Романтичні подорожі
- Екстремальні пригоди

Відмінний сервіс:

- 24/7 підтримка клієнтів
- Зручне бронювання
- Гарантована якість послуг

```
<div class="card">
        <img src="destinations.jpg" alt="Найкращі напрямки"
class="card-img">
        <div class="card-content">
            <h3 class="card-title">Найкращі напрямки для вашої подо-
рожі:</h3>
            <ul class="card-list">
                <li class="card-list-item">Екзотичні острови</li>
                <li class="card-list-item">Історичні міста</li>
                <li class="card-list-item">Культурні центри</li>
            </ul>
        </div>
    </div>
```

Блок **footer**. Цей блок містить лише один абзац з класом **footer-info** та текстом:

```
&copy; 2024 Подорожі. Автор: Іванов І.М. Всі права захищено.
```

3. Стилізація головної сторінки за допомогою CSS:

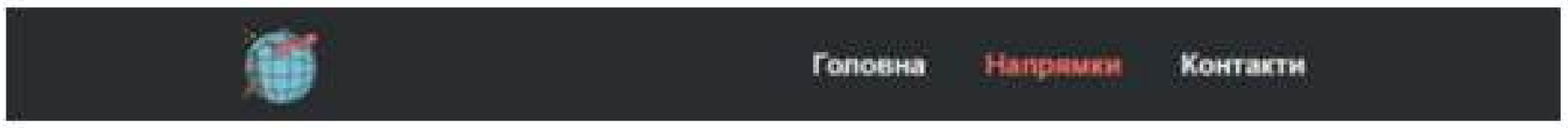
- Підключення файлу зі стилями до html-документу.
- Встановлення загальних стилів для тіла сторінки.

Для тегу **body** встановіть такі властивості: обнулення зовнішніх та внутрішніх відступів, шрифт `Arial`, колір тексту `#333`, колір фону `#f4f4f4`.

- Стилізація секції **head**.

Для класу **head** встановіть вирівнювання елементів по ширині контейнера з пропусками між елементами та межами контейнера та по центру по вертикалі, колір фону #333, колір тексту #fff, внутрішній відступ 20px знизу та зверху та 0 зліва та справа. Для логотипу встановіть ширину 50px. Для блоку з класом **menu-links** обнуліть зовнішні та внутрішні відступи та заберіть маркери в елементів списку (**list-style: none**). Елементи, які розташовані у списку з класом **menu-item**, розташуйте в рядок (**display: inline**) та задайте зовнішні відступи 0 зверху та знизу та 15px зліва та справа.

Для заголовка з класом **section-main-title** встановіть зовнішній відступ зверху 40px, зовнішній відступ знизу 20px, колір тексту #fff, шрифт **'Lora'**, розмір тексту 55px та напівжирний шрифт (**font-weight: 700**). Для підзаголовка з класом **section-main-subtitle** встановіть зовнішній відступ знизу 40px, колір тексту #fff, розмір тексту 22px. Для кнопки з класом **section-main-button** встановіть колір фону #8fbc8f, без рамки з округленням 25px, колір тексту #fff, напівжирний шрифт, всі літери великі, внутрішні відступи 12px зверху і знизу та 30px зліва і справа, при наведенні на кнопку курсор міняється на палець (**cursor: pointer**). Елементам з класом **menu-item-link** задайте колір тексту білий, приберіть підкреслення (**text-decoration: none**) та зробіть текст напівжирним. При наведенні на елемент, пункт меню, зробіть так, щоб колір тексту змінювався на #ff6f61



- Стилізація секції **hero**.

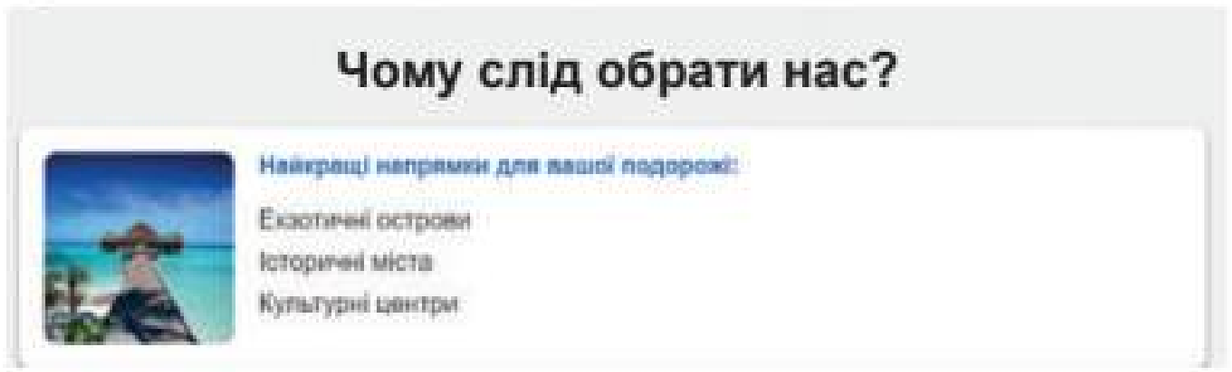
Для контейнера з класом **hero** встановіть такі властивості: вирівнювання елементів по центру як по горизонталі, так і по вертикалі, текст вирівняти по центру, на фон розмістити картинку на тематику подорожей, встановити висоту контейнера 100vh, колір тексту білий

```
.hero {
    display: flex;
    align-items: center;
    justify-content: center;
    text-align: center;
    background: url('heroback.jpg') no-repeat center center/
cover;
    height: 100vh;
    color: white;
}
```

Для блоку з класом **hero-info** встановіть чорний напівпрозорий колір фону (**background-color: rgba**(0, 0, 0, 0.5)), внутрішні відступи по 40px, заокруглення рамки 10px та ширина контейнера 720px. Для заголовку з класом **hero-title** встановіть розмір тексту 64px та обнуліть зовнішні відступи. Для тексту з класом **hero-text** встановіть розмір тексту 24px та зовнішні відступи 20px зверху і знизу та 0 зліва і справа.

- Стилізація секції **intro**.

Для контейнера з класом **intro** встановіть вирівнювання елементів по центру і в стовпець та внутрішні відступи 50px зверху та знизу і 20px зліва та справа. Для заголовка з класом **intro-title** встановіть розмір тексту 40px, колір тексту #333 та зовнішній відступ зверху 20px. Для блоку з класом **card** розмістіть дочірні елементи в рядок, зовнішні відступи зверху та знизу 20px і зліва та справа 0, внутрішні відступи по 20px, ширина блоку 900px, колір фону білий, заокруглення рамки 10px, тінь блоку box-shadow: 0 2px 4px rgba(0,0,0,0.5). Для зображення з класом card-img встановіть розміри 150px по ширині і по висоті, заокруглення 10px та зовнішній відступ справа 20px. Для блоку **card-content** вирівняйте текст по лівому краю (**text-align: left**). Для заголовка card-title обнуліть зовнішній відступ зверху та встановіть колір тексту color: #0073e6. Для списку з класом **card-list** обнуліть внутрішні відступи та заберіть маркери (**list-style: none**). Для елементів списку з класом **card-list-item** встановіть зовнішні відступи зверху і знизу 10px та 0 зліва і справа та розмір тексту 20px.



- Стилізація секції **footer**.

Для класу **footer** встановіть внутрішні відступи зверху та знизу 10px і 0 зліва та справа, колір фону #333, колір тексту білий та вирівнювання тексту по центру (**text-align: center**). Також зафіксуйте **footer**, встановивши такі властивості:

position: fixed;

bottom: 0;

width: 100%.

4. Написання HTML-коду для сторінки з напрямками популярних туристичних місць:

Відкрийте файл **destinations.html** та створіть стандартну структуру **<html>**, **<head>**, **<body>** та назвіть сторінку «Подорожі — Напрямки».

Ця сторінка складається з трьох секцій. Секції **header** та **footer** скопіюйте із файлу **index.html** та створіть ще одну секцію між ними з класом **destinations**.

Блок **destinations**. Секція з класом **destinations** містить заголовок **h2** з класом **destinations-title** і текстом «Популярні напрямки», а також блок з класом **destinations-container**.

```
<section class="destinations">
        <h2 class="destinations-title">Популярні напрямки</h2>
        <div class="destinations-container">

        </div>
  </section>
```

Контейнер **destinations-container**, своєю чергою, міститиме картки з напрямками. Тому створіть блок для картки з класом **destination**. Кожна картка складається із зображення з класом **destination-img**, заголовка **h3** з класом **destination-title** та абзацу з класом **destination-text**.

Інформація для кожної з карток (додатково можете створити ще свої картки із своїми країнами та місцями відпочинку):

1 картка:

- Зображення, яке описує Париж
- Париж, Франція
- Романтичні прогулянки та історичні пам'ятки

2 картка:

- Зображення, яке описує Нью-Йорк
- Нью-Йорк, США
- Місто, що ніколи не спить

3 картка:

- Зображення, яке описує Токіо
- Токіо, Японія
- Сучасна культура й традиційні цінності

```
<div class="destination">
                <img src="parig.jpg" alt="Напрямок 1"
class="destination-img">
                <h3 class="destination-title">Париж, Франція</h3>
                <p class="destination-text">Романтичні прогулянки та
історичні пам’ятки.</p>
            </div>
            </div> class="destination">
```

5. Стилізація сторінки з напрямками популярних туристичних місць за допомогою CSS:

- Підключення файлу зі стилями до html-документу.
- Стилізація секції **destinations**.



Для блоку з класом **destinations** встановіть вирівнювання елементів по центру і в стовпець та внутрішні відступи 50px зверху та знизу і 20px зліва та справа. (Якщо зауважили, такі ж стилі має клас **intro**, то щоб не було дубляжу, можна цей клас дописати до стилів класу **intro**). Для заголовка з класом **destinations-title** встановіть такі ж стилі, як і для класу **intro-title**. Для блоку з класом **destinations-container** встановіть вирівнювання елементів по ширині контейнера з пропусками між елементами та межами контейнера та перенесення елементів на новий рядок (**flex-wrap: wrap**). Для блоку **destination** встановіть зовнішні відступи 15px, внутрішні відступи 20px, колір фону білий, заокруглення 10px, ширина блоку 30% та тінь **box-shadow:** 0 2px 4px rgba(0,0,0,0.1). При наведенні курсором на цей блок встановити тінь **box-shadow:** 0 2px 4px rgba(0,0,0,0.5). Для зображення з класом **destination-img** встановіть ширину 100% та заокруглення 10px. За бажанням стилізуйте текст самостійно.

6. Написання HTML-коду для сторінки контактів з формою зворотного зв’язку:

Відкрийте файл **contact.html** та створіть стандартну структуру **<html>**, **<head>**, **<body>** та назвіть сторінку «Подорожі — Контакти».

Ця сторінка складається з трьох секцій. Секції **header** та **footer** скопіюйте із файлу **index.html** та створіть ще одну секцію між ними з класом **contact**.

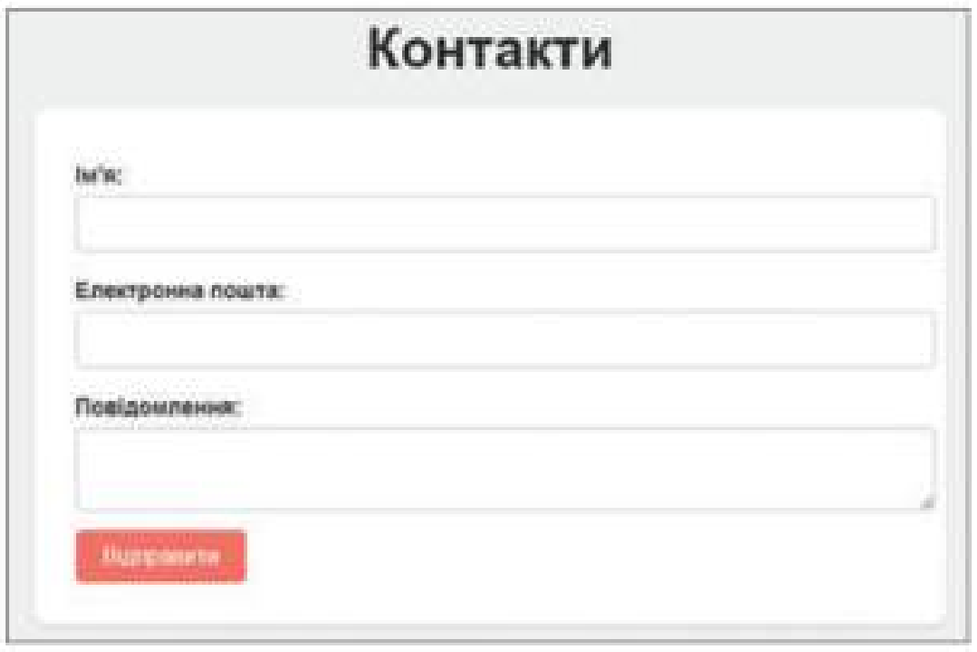
Блок **contact**. Секція з класом **contact** містить заголовок **h2** з класом **contact-title** і текстом «Контакти», а також форму з класом **form**.

Форма складається з таких елементів: **label Ім’я** та поле для введення імені, **label Електронна пошта** та поле для введення електронної

пошти, **label Повідомлення** та поле **textarea** для введення текстового повідомлення, кнопка з текстом «Відправити». Для кожного елемента задайте класи, відповідно **form-label**, **form-input**, **form-textarea**, **form-button**.

7. Стилізація сторінки контактів з формою зворотного зв'язку за допомогою CSS:
   - Підключення файлу зі стилями до html-документу.
   - Стилізація секції **contact**.

Для блоку з класом **contact** встановіть такі ж стилі, як і для класу **intro**. Для заголовка з класом **contact-title** встановіть такі ж стилі, як і для класу **intro-title**. Для форми з класом **form** встановіть колір фону білий, внутрішні відступи 30px, заокруглення 10px, тінь **box-shadow:** 0 2px 4px rgba(0,0,0,0.1), максимальна ширина форми 600px. Для міток з класом **form-label** встановіть **display: block**, напівжирний шрифт, зовнішні відступи зверху 10px, знизу 5px і 0 зліва та справа. Для класів **form-input** та **form-textarea** встановіть ширину 100%, зовнішній відступ знизу 10px, внутрішні відступи по 10px, рамку **border:** 1px solid #ccc, заокруглення 5px та розмір тексту 16px. Для кнопки з класом **form-button** встановіть внутрішні відступи 10px зверху і знизу та 20px справа і зліва, без рамки (**border: none**), заокруглення 5px, колір фону #ff6f61, колір тексту білий, розмір тексту 16px, **cursor: pointer**. При наведенні на кнопку колір фону змінюється на #e55a50.



Етап 4. Тестування

1. Перевірка сумісності з різними браузерами:
   - Відкриття сайту в популярних браузерах (**Chrome**, **Opera**, **Edge**).
   - Перевірка коректного відображення всіх елементів.
2. Виправлення знайдених помилок:
   - Аналіз та виправлення всіх знайдених помилок та неточностей.

Етап 5. Впровадження

Розміщення сайту на **GitHub**:

- Завантаження файлів сайту на сервер.
- Надіслати **URL** опублікованого сайту вчителеві.

Розділ 4

ПРОЄКТУВАННЯ ГРАФІЧНОГО ІНТЕРФЕЙСУ

Тема 22 **Графічний інтерфейс продукту. Дизайн UI/UX. Інтуїтивність та зручність використання, інклюзивність та доступність інформаційних продуктів**

Ключові поняття

UI (User Interface)
UX (User Experience)
Інклюзивний дизайн

UI (User Interface)

Що таке UI та UX?

UI (User Interface) — це графічний інтерфейс користувача, а саме кнопки, меню, шрифти, кольори та інші елементи, з якими взаємодіє користувач.



Досвід користувача визначається тим, наскільки легко чи складно взаємодіяти з елементами призначеного для нього інтерфейсу, створеного дизайнерами.

При цьому UX-дизайнери також дбають про користувальницький інтерфейс програми, і саме тому люди часто плутаються або ж взагалі не знаходять різниці між ними.

Якщо UI-дизайнери повинні вирішувати, як буде виглядати користувальницький інтерфейс, то UX-дизайнери відповідають за визначення того, як працює інтерфейс.

Вони визначають структуру інтерфейсу і функціональність, а також те, як всі частини програми організовані й пов'язані між собою. Загалом кажучи, вони проєктують роботу інтерфейсу. Якщо він працює добре, то у користувача буде хороший досвід після взаємодії з таким інтерфейсом. Але якщо навігація складна або інтуїтивно незрозуміла, то, швидше за все, користувач отримає негативний досвід при взаємодії з інтерфейсом. Основне завдання UX-дизайнера і полягає в тому, щоб уникнути другого сценарію.

UX (User Experience)

UX (User Experience) — це досвід користувача, який він отримує під час взаємодії з продуктом. UX передбачає зручність, логіку, швидкість роботи інтерфейсу та задоволення користувача.

Розділення ролей в UI/UX-дизайні

UX-дизайнер

1. Аналізує аудиторію.
2. Планує шляхи користувача (user journey).
3. Працює з інформаційною архітектурою (як організована інформація на сторінках).



UI-дизайнер

1. Відповідає за зовнішній вигляд.
2. Створює макети кнопок, меню, іконок, шрифтів.
3. Обирає кольори та забезпечує візуальну узгодженість.

Чому важливі UI/UX-дизайн у сучасних технологіях?

- **Інтерфейс** — це перше, що бачить користувач. Зручність використання часто визначає, чи буде продукт успішним.
- **Користувацький досвід**. Якщо продукт складний у використанні, людина ймовірніше перейде до іншого, більш інтуїтивного аналога.
- **Доступність для всіх**. UI/UX-дизайн допомагає зробити продукти доступними для людей з різними потребами.

Основні принципи UI/UX-дизайну

- *Простота:* інтерфейс має бути мінімалістичним і без зайвих деталей.
- *Інтуїтивність:* користувач повинен легко зрозуміти, як працює продукт.
- *Послідовність:* елементи дизайну (шрифти, кольори, кнопки) мають бути однакові на різних сторінках.
- *Доступність:* інформаційний продукт має бути зручним для людей з різними можливостями (з порушеннями зору, слуху тощо).
- *Швидкість:* інтерфейс має швидко реагувати на дії користувача.

Інтуїтивний інтерфейс

Інтуїтивний інтерфейс — це такий, яким користувач може користуватися без попередніх інструкцій.



- Логічно розташовані кнопки.
- Чіткі підказки (наприклад, **Завантажити файл**).
- Зрозуміла навігація (меню, пошуковий рядок).

Інтуїтивний дизайн передбачає, що користувач одразу розуміє, як працює продукт, завдяки логічності розташування елементів, зрозумілим підказкам та передбачуваній навігації.

Ключові принципи інтуїтивного дизайну

Логічне розташування елементів

- Елементи інтерфейсу розташовані так, як користувач інстинктивно очікує.
- Кнопка **Зберегти** зазвичай розміщена в правому нижньому куті.
- Меню навігації — у верхній частині сторінки або збоку.

Чому це важливо: користувач не витрачає часу на пошук потрібних елементів.

Чіткі підказки

- Інтерфейс повинен містити короткі, але зрозумілі інструкції. Наприклад: кнопка **Завантажити файл** замість абстрактного **Виконати**.
- Використання іконок із текстом: лупа для пошуку, кошик для покупок.

Чому це важливо: користувач одразу розуміє, які дії можна виконати.

Зрозуміла навігація

- Меню, пошуковий рядок та інші навігаційні елементи організовані логічно.
- Всі розділи мають зрозумілі назви (наприклад, **Головна**, **Про нас**, **Контакти**).
- Наявність кнопки **Назад** для повернення до попереднього екрану.

Чому це важливо: користувач завжди знає, де він перебуває і як повернутися до попередньої дії.

Кольори, шрифти та іконки підказують, де знайти ключові елементи. Наприклад, активна кнопка має яскравий колір, а неактивна — сірий.

Інтерфейс повинен давати користувачеві зрозумілий зворотний зв'язок. Наприклад, кнопка змінює колір при натисканні. Повідомлення **Файл успішно завантажено** після виконання дії.

Основи візуальної привабливості

Гармонія кольорів	• Використання не більше трьох основних кольорів. • Контраст для акцентів (наприклад, кнопки виділяються яскравим кольором).
Шрифти	• Зручні для читання (без декоративних деталей). • Мінімум два шрифти: для заголовків і тексту.
Іконки та графіка	• Використання стандартних іконок (лупа для пошуку, три риски для меню). • Простота, яка не перевантажує екран.

Інклюзивний дизайн

Інклюзивний дизайн — це створення продуктів, які можуть використовувати всі люди, незалежно від їхніх фізичних чи когнітивних можливостей.

- Використання контрастних кольорів для людей із порушеннями зору.
- Додавання текстових описів для зображень.
- Можливість керування інтерфейсом без миші (через клавіатуру).
- Підтримка екранних читачів для людей з вадами зору.

Інклюзивний дизайн враховує потреби різних користувачів, включаючи людей з інвалідністю.

Зорові обмеження: Контрастні кольори, великі шрифти, підтримка екранних читачів.

Слухові обмеження: Текстові субтитри для відео.

Моторні обмеження: Великі кнопки для зручного натискання.

Когнітивні обмеження: Простота структури, зрозумілі інструкції.

Приклади успішного та невдалого UI/UX

Успішний UI/UX

- Додаток, де ви можете купити квиток за три кроки (вибір маршруту, вибір дати, оплата).
- Вебсайт з логічною структурою меню, що дозволяє швидко знайти потрібний розділ.





Невдалий UI/UX

- Сайт, що повільно завантажується, має складну навігацію та плутане меню.
- Додаток із незрозумілими кнопками або текстом, що важко читається.

Патерни та стандарти в дизайні UI/UX

Патерни (patterns) Це повторювані рішення для типових завдань в інтерфейсі, які користувачі вже добре знають.

Стандарти Загальноприйняті правила оформлення елементів дизайну, які забезпечують передбачуваність і зручність.

Патерни та стандарти в дизайні UI/UX

Лупа для пошуку

- Іконка у вигляді візка або сумки символізує функцію додавання товарів у кошик.
- Часто розташовується у верхньому правому куті сторінки інтернет-магазинів.



Бургер-меню

- Іконка із трьох горизонтальних ліній, яка відкриває меню навігації.
- Використовується на мобільних пристроях або в адаптивному дизайні.

Кошик для покупок

- Іконка у вигляді візка або сумки символізує функцію додавання товарів у кошик.

- Часто розташовується у верхньому правому куті сторінки інтернет-магазинів.

Головна сторінка (Home)

- Іконка у вигляді будиночка повертає користувача на головну сторінку.
- Часто розташована в нижній панелі або в меню.

Кнопка Поділитися (Share)

- Іконка зі стрілкою, яка вказує назовні, або три сполучені крапки, що символізують спільне використання.
- Застосовується для поширення контенту через соціальні мережі або інші платформи.

Список дій (дропдаун меню)

- Розкривний список, позначений стрілкою вниз, використовується для вибору однієї з кількох опцій.
- Часто буває у формах або фільтрах.

Кнопка Підписатися (Subscribe)

- Виразна кнопка з текстом **Підписатися** або іконкою дзвіночка.
- Зазвичай виділяється кольором для привернення уваги.

Повідомлення про помилки

- Використання червоного кольору та чіткого тексту для вказівки на помилки (наприклад, **Поле не може бути порожнім**).
- Розташовується біля відповідного поля або у вигляді спливаючого повідомлення.

Індикатори завантаження

- Кільце, яке обертається, або прогрес-бар, що показує процес виконання операції.
- Використовується для зворотного зв'язку, щоб користувач знав, що операція триває.

Форми заповнення

- Поля для вводу мають стандартний вигляд: сірий фон із текстовою підказкою (**placeholder**), яка зникає при заповненні.
- Кнопка **Надіслати** розташовується після форми.

Активні та неактивні елементи

- Активні кнопки яскраві й привертають увагу, а неактивні — сірі або зменшеної яскравості.
- Це дозволяє користувачам зрозуміти, які функції доступні.

Навігаційна панель (Navbar)

- Фіксована панель у верхній або нижній частині екрана з основними розділами сайту чи додатка.
- Містить кнопки, такі як **Головна**, **Контакти**, **Про нас**.

Етапи розробки дизайну

1 Дослідження: визначення потреб аудиторії

На цьому етапі визначаються цільова аудиторія, її потреби та проблеми, які продукт повинен задовольнити. Проводяться опитування, інтерв'ю або аналіз конкурентів, щоб створити персоналії користувачів та сценарії їхньої взаємодії.

2 Прототипування: створення макета продукту

Прототипування включає створення скетчів і вайрфреймів, які показують структуру інтерфейсу. Це дозволяє продумати розташування елементів і функціональність, не зосереджуючись на візуальних деталях.

3 Тестування. Перевірка продукту на зручність та інтуїтивність

Меню, пошуковий рядок та інші навігаційні елементи організовані логічно.

Всі розділи мають зрозумілі назви (наприклад, **Головна**, **Про нас**, **Контакти**).

Наявність кнопки **Назад** для повернення до попереднього екрану.

Чому це важливо: Користувач завжди знає, де він перебуває і як повернутися до попередньої дії.

4 Впровадження. Завершення дизайну і передача для розробки

Фіналізується візуальний дизайн: додаються кольори, шрифти, іконки, ефекти. Створюється документація з описом елементів і передаються макети розробникам для реалізації.

Інструменти для UI/UX-дизайну

Figma

Онлайн-інструмент для створення прототипів.

Canva

Інструмент для дизайну з інтуїтивно зрозумілим інтерфейсом.

Adobe XD

Програмне забезпечення для створення UI/UX-дизайну.

Практична робота №16.

Навчити створювати частину дизайну сайту (головний екран) у **Canva**.

Створіть головний екран сайту в **Canva** для вигаданої теми. Наприклад, сайт вивчення англійської мови або іншого предмету.

1. Підготовка

- Увійдіть у свій акаунт **Canva**.
- Виберіть шаблон — сайт та оберіть дизайн, який вам подобається.

2. Створення макету сайту

- Додайте заголовок сайту:
  - Виберіть шрифт із розділу **Тексти** та напишіть назву сайту (наприклад, **Програмуємо разом**).
  - Вирівняйте текст по центру верхньої частини екрана.
  - Встановіть розмір шрифту так, щоб текст був головним акцентом екрана.
- Додайте навігаційну панель:
  - Намалюйте горизонтальний прямокутник у верхній частині екрана.
  - Додайте текстові блоки для кнопок навігації (наприклад: **Головна**, **Про нас**, **Контакти**).
  - Використайте простий і зрозумілий шрифт.

3. Створіть головний банер:

- Завантажте фонове зображення або виберіть із бібліотеки **Canva**.
- Накладіть на зображення прозорий прямокутник (білий або чорний із прозорістю 30-50%) для виділення тексту.
- Додайте текст із закликом до дії (наприклад, **Приєднуйтесь до нашого курсу вже сьогодні!**).

4. Розмістіть основну кнопку:

- Намалюйте кольоровий прямокутник (кнопка).
- Додайте текст усередині кнопки, наприклад: **Записатися** або **Дізнатися більше**.
- Виберіть яскравий колір, що контрастує з фоном.

5. Додайте візуальні акценти

- Іконки: вставте одну або кілька іконок, що відповідають тематиці сайту (наприклад, книги для навчального сайту).
- Графічні елементи: використовуйте лінії або крапки для візуального поділу секцій.

6. Завершення роботи

- Перевірте, чи зрозумілий ваш дизайн для користувача:
  - Чи легко побачити назву сайту та кнопку дії?

- Чи логічно розташовані елементи?

Збережіть свій макет: натисніть **Завантажити** та збережіть проєкт у форматі **PNG** або **JPG**.

Готовий дизайн головного екрана сайту, який містить:

- Назву сайту.
- Навігаційну панель.
- Головний банер із текстом і кнопкою.
- Декоративні графічні елементи.

Векторний графічний редактор. Комбінування та налаштування графічних примітивів. Робота з векторними контурами. Текст та типографіка у векторному графічному редакторі. Ефекти та фільтри

Теми 23–24

Ключові поняття

- Векторна графіка
- Графічні примітиви
- Контур
- Типографіка
- Шрифтові пари
- Ефекти та фільтри

Векторна графіка

Векторна графіка — створення зображення в комп'ютерній графіці з сукупності геометричних примітивів — крапок, ліній, кривих, тобто об'єктів, які можна описати математичними виразами.

Не втрачає якості при масштабуванні.

Наприклад, логотип можна збільшувати без втрати чіткості. Використовується для створення іконок, логотипів, ілюстрацій, а також дизайнів сайтів.

Приклади програм для векторної графіки: **Canva**, **Figma**, **Adobe Illustrator**, **CorelDRAW**.

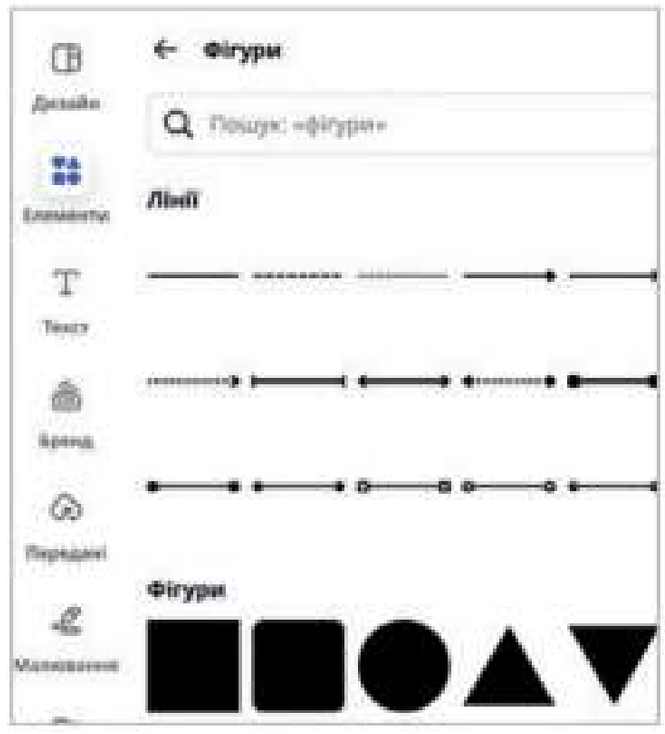


Графічні примітиви

Графічні примітиви — це базові форми, з яких складаються векторні зображення: лінії, прямокутники, кола, еліпси, багатокутники.

У **Canva** графічні примітиви можна знайти у вкладці **Елементи**:

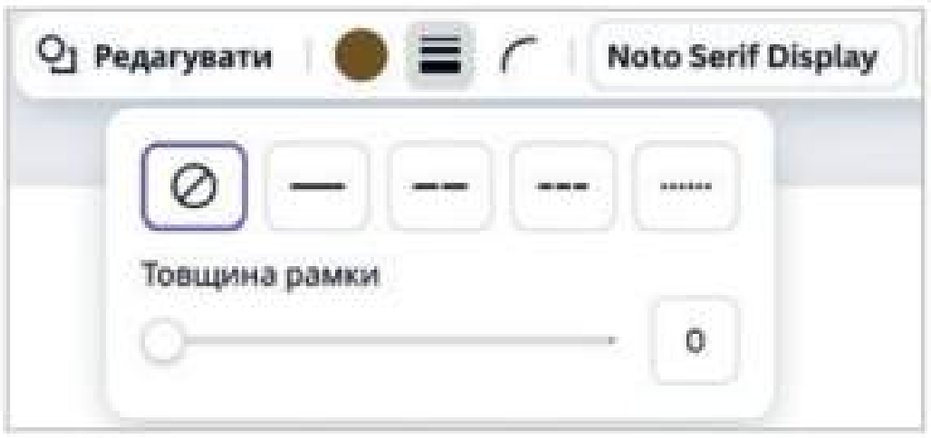
- Фігури (квадрати, кола).
- Лінії (прямі, хвилясті, пунктирні).
- Іконки (багатокутники, стрілки).

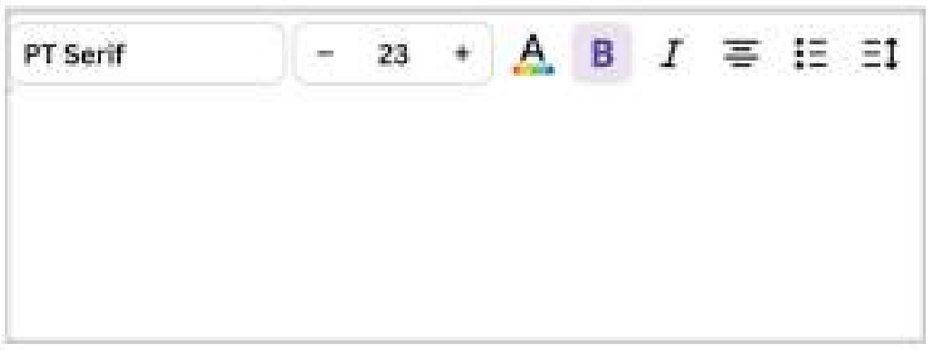




Робота з векторними контурами

Контур — це лінія, яка визначає форму об'єкта.





Редагування контуру:

- Змінюйте товщину лінії.
- Обирайте колір і стиль лінії (суцільна, пунктирна, хвиляста).

У **Canva** можна змінювати контури фігур у меню налаштувань елемента:

Обведення: додайте рамку до фігури, виберіть її товщину та колір.

Кути: зробіть кути гострими або заокругленими.

Типографіка — це мистецтво роботи з текстом, яке включає вибір шрифтів, їхній розмір, колір і розташування.

Шрифт — Вибирайте стиль, що відповідає темі дизайну (наприклад, строгий для офіційних сайтів, декоративний для творчих проєктів).

Розмір — Виділіть заголовки більшим розміром, підзаголовки — середнім, текст — меншим.

Колір — Використовуйте контрастний колір для читабельності.

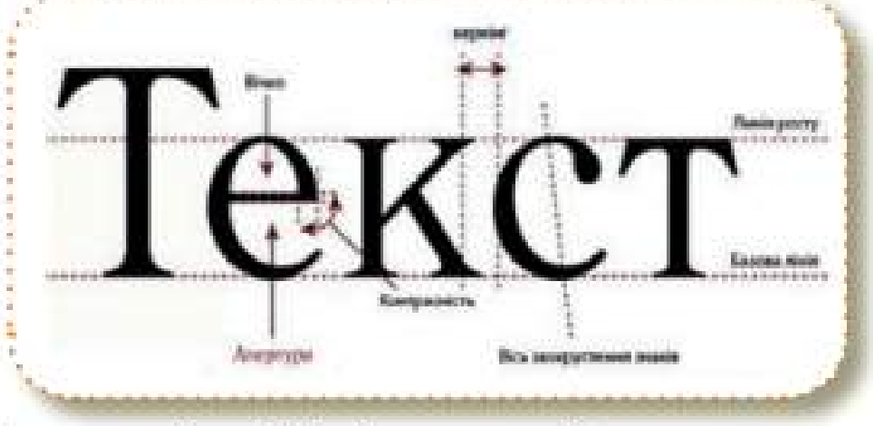
У **Canva** доступні інструменти для:

- вирівнювання тексту (по центру, ліворуч, праворуч).
- додавання ефектів до тексту (тіні, обведення, прозорість).



Правильна типографіка допомагає тексту бути зрозумілим, читабельним та візуально привабливим.

Головне завдання тексту — бути зрозумілим і читабельним.



Читабельність

Уникайте надто декоративних шрифтів для великих обсягів тексту — їх складно читати.

Для заголовків можна використовувати більш сміливі й декоративні шрифти.

Використання різних шрифтів або їхніх варіацій (жирний, курсивний, тонкий) допомагає виділити важливі частини тексту.

Шрифт формує перше враження користувача про сайт.

Наприклад, строгий шрифт із засічками підходить для юридичних сайтів, а рукописний — для творчих блогів.

Не використовуйте більше 2-3 шрифтів у одному дизайні.

Колір тексту

Використовуйте контрастні кольори для кращої видимості.

Наприклад, темний текст на світлому фоні або білий текст на темному фоні.

Розмір

- Заголовки — великі (20-36 pt).
- Основний текст — середній (14-18 pt).
- Примітки або підписи — маленький (10-12 pt).

Інтервали між літерами (кернінг)

Відстань між літерами впливає на читабельність.

У **Canva** можна збільшити або зменшити інтервали в меню налаштувань тексту (**Spacing**).

Розташування тексту

Вирівнюйте текст залежно від композиції:

- По центру для формального вигляду.
- По лівому краю для традиційного тексту.
- По правому краю — для декоративних композицій.

Міжрядковий інтервал (leading)

Це відстань між рядками тексту.

Рекомендації:

- зменшуйте інтервал для заголовків.
- збільшуйте для основного тексту, щоб полегшити читання.



Шрифтові пари

Шрифтові пари — це поєднання двох або більше шрифтів, які гармонійно працюють разом, роблячи текст у дизайні структурованим, зрозумілим і візуально привабливим.

Шрифти з зарубками	Шрифти без зарубок	Брускові шрифти	Заголовкові шрифти	Рукописні шрифти
Serif	Sans	Slab	FONT	*Script*
Інші назви: — Романські шрифти, — Антиквенні шрифти, — Антикви. **Назва англійською мовою:** — Serif fonts.	**Інші назви:** — Гротескні шрифти, — Гротески, — Рубані. **Назва англійською мовою:** — Sans-serif.	**Інші назви:** — Механістичні, — Єгипетські. **Назва англійською мовою:** — Slab-serif.	**Інші назви:** — Плакатні шрифти, — Акцидентні шрифти, — Декоративні шрифти. **Назва англійською мовою:** — Display.	**Інша назва:** — Каліграфічні шрифти. **Назва англійською мовою:** — Script, — Handwritten (іноді).

Контраст

- Використовуйте шрифти з різними стилями, щоб створити чітку різницю між заголовками та текстом.

Наприклад: для заголовків оберіть декоративний шрифт, а для основного тексту — простий та читабельний:

Abril Fatface (для заголовків) + **Lato** (для тексту).

Узгодженість стилю

- Вибирайте шрифти, які візуально доповнюють один одного.

Наприклад, обидва можуть мати м'які лінії або бути схожими за характером:

Lora (елегантний стиль) + **Poppins** (сучасний вигляд).

Варіативність розміру

- Використовуйте шрифти різного розміру, щоб створити акценти та виділити важливі елементи.

Наприклад: зробіть заголовок великим і яскравим, а текст — меншим і більш нейтральним:

Playfair Display для заголовків + **Open Sans** для тексту.

Мінімалізм

- Використовуйте не більше двох шрифтів у дизайні. Один шрифт — для заголовків, інший — для тексту. Для великих проєктів допустимо додати ще один шрифт для акцентів:
  Montserrat для заголовків + **Raleway** для основного тексту.
- Використовуйте виділення (жирний текст, курсив), щоб привернути увагу до важливих частин.

Один шрифт із варіаціями

- Використовуйте один шрифт із різними стилями (наприклад, жирний для заголовків, звичайний для тексту):
  Roboto Bold для заголовків + **Roboto Regular** для основного тексту.

Поради для роботи з текстом у UI/UX-дизайні

1. **Завжди думайте про користувача**
   Текст має бути не лише красивим, а й легким для сприйняття.

2. **Розташування**
   Розміщуйте текст там, де його очікує користувач (наприклад, заголовок у верхній частині сторінки).

3. **Короткі абзаци**
   Уникайте великих текстових блоків — вони важко читаються.

4. **Додавайте візуальні акценти**
   Використовуйте виділення (жирний текст, курсив), щоб привернути увагу до важливих частин.



Текст має коректно відображатися на різних екранах (комп'ютери, планшети, смартфони).

Сервіси для підбору шрифтових пар



Google Fonts — безкоштовна бібліотека шрифтів із готовими прикладами пар. Дозволяє тестувати текст у реальному часі.

Ефекти та фільтри

- **Прозорість.** Використовується для створення напівпрозорого фону, акцентів або додавання шарів до дизайну.
  Наприклад, зробіть текст видимим на яскравому зображенні, додавши прозорий білий прямокутник позаду.
- **Свічення.** Підкреслює текст або фігури, додаючи сяючий ефект.
  Наприклад, використовуйте для створення магічного або сучасного вигляду.
- **Тіні.** Додає об'єм та робить елементи помітнішими.
  Наприклад, використовуйте тіні для кнопок, щоб створити ефект натискання.
- **Обведення.** Створює чітку межу навколо тексту або фігури для виділення.
  Наприклад, обведення використовується для заголовків, щоб зробити їх більш помітними.
- **Фільтри.** Змінює вигляд зображень за допомогою кольорових ефектів.
  Наприклад, **Сепія** додає ретро-стиль, а **Чорно-білий** підкреслює мінімалізм.
- **Контраст.** Підсилює різницю між світлими та темними елементами на зображенні.
  Наприклад, покращує читабельність тексту на складному фоні.
- **Яскравість.** Робить зображення світлішим або темнішим, залежно від потреб дизайну.
  Наприклад, збільшіть яскравість, щоб додати життєрадісності.
- **Розмиття.** Робить зображення нечітким, щоб використовувати його як фон або акцентувати увагу на інших елементах.
  Наприклад, розмиття заднього фону для виділення тексту.

Практична робота №17.

Створення секції **Про нас** або **Наші послуги**

1. Підготовка

- Відкрийте **Canva** і ваш проєкт із головним екраном сайту.
- Додайте новий слайд для другої секції.
- Вирішіть, яку інформацію хочете розмістити.

Наприклад: розділ **Про нас**, що розповідає про компанію/сайт, або **Наші послуги**, що описує ключові можливості.

2. Створення секції

- Заголовок секції:
  - Додайте текстовий блок із заголовком: **Про нас** або **Наші послуги**.
  - Використовуйте шрифт, що узгоджується зі стилем заголовка на головному екрані.
  - Вирівняйте заголовок у верхній частині секції.
- Описова частина:
  - Додайте текстовий блок для короткого опису.

Наприклад: **Ми пропонуємо унікальні курси, які допоможуть вам досягти своєї мети швидко та ефективно.**

  - Використовуйте простий і читабельний шрифт.
- Іконки або графічні елементи:
  - У вкладці **Елементи** знайдіть іконки, які відповідають змісту (наприклад, книга, комп'ютер або людина).
  - Розташуйте іконки поруч із кожним текстовим блоком, щоб візуалізувати інформацію.
- Візуальний поділ:

Використовуйте горизонтальні лінії, тіні або кольорові блоки, щоб розділити елементи секції.

3. Налаштування композиції

- Розташування елементів:
  - Розділіть секцію на дві або три частини:
    - Ліва частина — заголовок і текст.
    - Права частина — іконки або зображення.
- Баланс:
  - Уникайте перевантаження секції текстом чи зображеннями.
  - Залиште простір між елементами для легкості сприйняття.

4. Застосування ефектів

- Фон секції.

Додайте фон: використовуйте легкий градієнт або кольоровий блок із прозорістю.

- Ефекти для тексту.

Додайте тіні або свічення до заголовка, щоб зробити його більш помітним.

- Іконки.

Налаштуйте колір і прозорість, щоб іконки гармонійно поєднувалися з текстом.

5. Завершення роботи

- Перевірте секцію:
  - Чи виглядає текст читабельним?
  - Чи логічно розташовані елементи?
  - Чи відповідає стиль нової секції головному екрану?
- Збережіть оновлений дизайн.

Готова секція **Про нас** або **Наші послуги**, яка містить:

- Заголовок.
- Короткий опис.
- Іконки або зображення.
- Гармонійний фон та елементи композиції.

Композиція та макет у дизайні. Створення прототипу. Документація інформаційного продукту

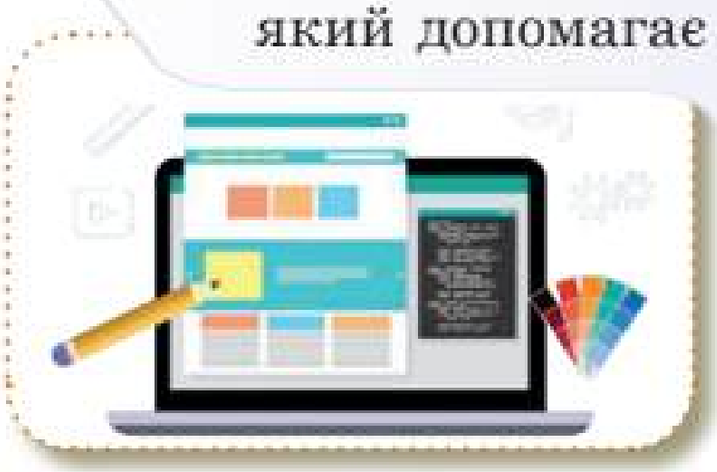
Теми 25–26

Ключові поняття

Композиція
Принципи композиції
Макет
Прототип

Композиція

Композиція — це спосіб організації елементів у дизайні, який допомагає створити гармонійний і зручний для користувача вигляд.



Правильно побудована композиція привертає увагу до важливих частин дизайну й полегшує сприйняття інформації.

Принципи композиції

1 **Баланс**

Елементи розташовуються рівномірно, щоб дизайн мав стійкий вигляд. Буває:

- *Симетричний баланс:* елементи розташовані рівномірно по центру.
- *Асиметричний баланс:* різні елементи врівноважують один одного.

2 **Ієрархія**

Визначає, які частини дизайну користувач має побачити першими.

Наприклад, заголовок більший за основний текст.

3 **Контраст**

Використовується для виділення важливих елементів. Контраст створюється через кольори, розміри або шрифти.

4 **Фокус**

Виділення найважливішого елемента за допомогою кольору, розміру або розташування.

Макет

Макет — це схема, за якою розташовуються елементи дизайну. Це план, який показує, як виглядатиме кінцевий продукт.



Односторінковий (One-page)

Усі елементи розташовані на одній сторінці.
Використовується для лендингів або невеликих сайтів.

Сітковий (Grid)

Дизайн розділений на сітку для точного розташування елементів.
Зручний для структурованих сайтів і додатків.

Модульний

Інформація розділена на блоки, які виконують різні функції (наприклад, галерея, контактна форма).



Прототип

Прототип — це попередній варіант дизайну, який показує, як виглядатиме продукт. Він допомагає зрозуміти структуру та функції продукту ще до його розробки.



Прототип показує структуру, розташування елементів і базову функціональність без повного технічного опрацювання.

Навіщо створювати прототип?

Візуалізація ідей

Прототип дозволяє побачити, як виглядатиме кінцевий продукт ще до початку розробки.

Тестування зручності

Допомагає оцінити, наскільки дизайн зрозумілий і зручний для користувачів.

Економія часу та ресурсів

Виявлення помилок на ранніх етапах проєктування значно дешевше, ніж після запуску продукту.

Особливості дизайну сайтів

Зрозуміла навігація

- Меню повинно бути інтуїтивним і легким у використанні.
- Використовуйте чіткі назви розділів, наприклад, **Про нас**, **Послуги**, **Контакти**.

Читабельність тексту

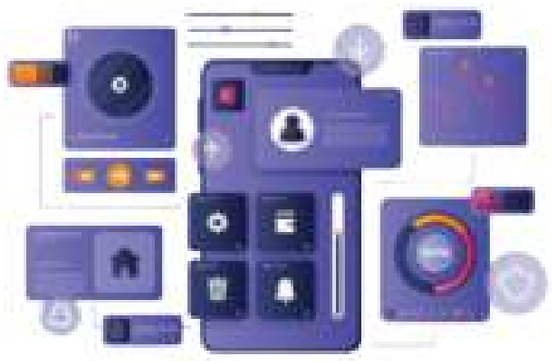
- Використовуйте зручні шрифти (наприклад, **Montserrat**, **Open Sans**).
- Оптимальний розмір тексту для читання — 16–18 px.

Оптимізація для швидкості

Сайт має швидко завантажуватись, навіть із повільним інтернетом.

Адаптивність

Більшість користувачів заходять на сайт зі смартфонів, тому адаптивність є обов'язковою.



Кнопки заклику до дії (CTA)

Кнопки мають бути видимими й закликати до дії (**Зареєструватися**, **Замовити**).

Типові помилки при формуванні структури сайту

Відсутність чіткої ієрархії сторінок.

- Меню повинно бути інтуїтивним і легким у використанні.
- Використовуйте чіткі назви розділів, наприклад, **Про нас**, **Послуги**, **Контакти**.

Неадаптивний дизайн для мобільних пристроїв.

- Використовуйте зручні шрифти (наприклад, **Montserrat**, **Open Sans**).
- Оптимальний розмір тексту для читання — 16–18 px.

Неправильна організація навігації та меню.

Сайт має швидко завантажуватись, навіть із повільним інтернетом.

Не враховані потреби користувачів.

Більшість користувачів заходять на сайт зі смартфонів, тому адаптивність є обов'язковою.

У нашому випадку документація створюється в **Canva** і містить всі графічні та текстові елементи дизайну.

Документація інформаційного продукту — це набір матеріалів, які описують, який вигляд має створений продукт, як він функціонує та які технічні й дизайнерські рішення використано.

1. **Стандарти стилю**

   Шрифти, кольори, іконки та їхнє використання в дизайні.

2. **Елементи брендингу**

   Логотипи, фірмові шрифти, кольорові схеми, які визначають унікальність продукту.

3. **Контентні вимоги**

   Приклади текстів, які мають бути використані в продукті.



4. **Інструкції з використання**

   Покрокова інструкція, як працювати з дизайном або продуктом.

Документація інформаційного продукту в **Canva** — це зручний і наочний спосіб організувати всю інформацію про проєкт.

Практична робота №18.

1. Завершення дизайну сайту

Додайте фінальні секції:

- Секція **Контакти**:

– Додайте текст із контактною інформацією (телефон, email).

– Вставте форму для зворотного зв'язку або кнопку **Написати нам**.

- Секція **Відгуки**: розмістіть блоки з текстовими відгуками клієнтів або учнів.

Оформте футер:

- Розташуйте логотип, посилання на соціальні мережі та копірайт.
- Використайте простий дизайн із нейтральними кольорами.

2. Створення інтерактивного прототипу

Додайте посилання:

- Виділіть кнопки на вашому дизайні (наприклад, **Записатися** або **Про нас**).
- Вкажіть слайди, на які перенаправлятимуть ці кнопки, використовуючи функцію **Посилання** в **Canva**.

Перевірте інтерактивність:

Увімкніть режим презентації та протестуйте переходи між сторінками.

3. Документація інформаційного продукту

Оформте сторінку з палітрою кольорів:

- Створіть таблицю з основними кольорами сайту (вказавши коди, наприклад, #F4F4F4).
- Опишіть, для чого використовується кожен колір.

Додайте розділ зі шрифтами:

- Вкажіть шрифти, які будете використовувати для заголовків, тексту та кнопок.
- Розмістіть приклади текстів із зазначеними стилями.

Опис іконок: вкажіть, які іконки використовували та їх призначення (наприклад, іконка телефону для контактів).

Інструкція з користування: опишіть, як взаємодіяти з прототипом (натиснути кнопку для переходу, заповнити форму тощо).

4. Завершення роботи

Остаточна перевірка:

- Чи всі секції сайту мають логічну структуру?
- Чи працюють інтерактивні кнопки?

Збережіть проєкт: завантажте його у форматі PDF або поділіться посиланням на **Canva**.

Розділ 5

АЛГОРИТМИ ТА ЇХНІ КОДИ

Тема 27 Середовища для написання коду. Транслятори. Програмний проєкт

Ключові поняття

Середовища для написання коду
Транслятори
Компілятори
Інтерпретатори
Програмний проєкт

У сучасному світі програмування стає важливою навичкою, яка допомагає створювати різні технології, від простих додатків до складних програм.



Щоб писати код і створювати програми, програмісти використовують спеціальні інструменти, такі як середовища для написання коду та транслятори.

Вивчаючи їх, ви зможете краще зрозуміти, як працюють комп'ютери та як створюють програмні продукти.

Середовища для написання коду

Середовище для написання коду (або **IDE** — інтегроване середовище розробки) — це програма або набір інструментів, що допомагає розробнику писати та редагувати код, тестувати і запускати його.



Це важлива складова у розробці програмного забезпечення, оскільки спрощує процес створення програм і надає зручні інструменти для роботи.

Мови програмування, такі як **Python**, мають різні середовища, які роблять процес написання коду простішим і зручнішим.

Основні можливості середовищ для написання коду

Редактор коду	Компілятор, або інтерпретатор	Дебагер (відладчик)	Система керування файлами і проєктами



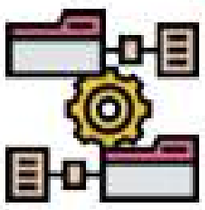
Редактор коду — основне місце, де програмісти пишуть свій код. Редактор підсвічує синтаксис, що допомагає краще орієнтуватися в коді та уникати помилок.



Компілятор, або **інтерпретатор** — програма, що перекладає код, написаний людиною, у код, зрозумілий комп'ютеру.



Дебагер (відладчик) — інструмент, що дозволяє відстежувати і виправляти помилки у програмі, даючи змогу програмісту побачити, як працює код на різних етапах виконання.



Система керування файлами і проєктами дозволяє легко організовувати файли у великому проєкті.

Приклади популярних середовищ для написання коду

Ось кілька найпопулярніших середовищ для написання коду (IDE), які використовуються розробниками для різних мов програмування та проєктів:



IDLE IntelliJ IDEA PyCharm — безкоштовне і потужне середовище, яке підтримує багато мов програмування.

Популярність: один із найпопулярніших редакторів серед розробників через свою легкість, швидкість і продуктивність.



PyCharm — середовище для розробки на мові **Python**.

Популярність: один з найкращих варіантів для Python-розробників, особливо для веброзробки та наукових обчислень.



IntelliJ IDEA — зручне середовище для різних мов програмування, насамперед **Java**.

Популярність: особливо популярний серед Java-розробників завдяки потужній підтримці мови.



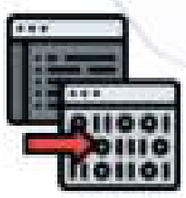
IDLE — просте середовище для написання програм на **Python**, що під силу навіть початківцям.

Популярність: використовується новачками та тими, хто вивчає **Python**, через простоту та інтеграцію з офіційним дистрибутивом **Python**.

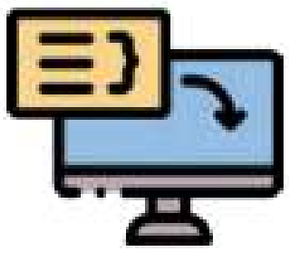
Транслятори



Транслятори — це програми, що перетворюють код, написаний програмістом, у форму, зрозумілу для комп'ютера.



Є два основні типи трансляторів:



Компілятори

Інтерпретатори



Компілятори

Компілятор — це програма, яка бере весь вихідний код програми, написаний на мові програмування, і перетворює його в машинний код або код низького рівня.

Це схоже на те, якби ви переклали книгу з однієї мови на іншу перед тим, як хтось почне її читати.

Як працює компілятор:

програміст пише код → компілятор переводить його в машинний код → цей код зчитується комп'ютером

Переваги: швидке виконання програм після компіляції, ефективність

Недоліки: важко відразу знайти помилки, оскільки програма компілюється повністю перед запуском

Інтерпретатори

Інтерпретатор виконує код покроково. Це означає, що кожна команда виконується відразу після того, як була написана та перевірена на наявність помилок.

Як працює компілятор:

програміст пише код → інтерпретатор виконує кожну команду одразу, переводячи її на зрозумілу для комп'ютера мову

Переваги: легше відстежувати помилки, зручніше для початківців

Недоліки: повільніше виконання програми, оскільки кожна команда перекладається під час виконання

Програмний проєкт

Програмний проєкт — це сукупність файлів і програмного коду, що створюється для розв'язання певного завдання.

Проєкт може бути невеликим, над яким працює один розробник, або масштабним, який розробляють команди з десятків чи сотень фахівців, кожен з яких відповідає за свою частину роботи. Успішний програмний проєкт вимагає ретельного планування, координації та управління ресурсами.

Основні етапи створення програмного проєкту

1. **Аналіз вимог**
Перед початком роботи важливо зрозуміти, яку проблему має розв'язати проєкт. Наприклад, програма може допомагати учням у розв'язанні математичних задач.

2. **Планування**
Розробники планують, який вигляд буде в програми, які функції вона матиме та як краще організувати її структуру.

3. **Розробка (написання коду)**
Програмісти пишуть код за допомогою середовищ для розробки та трансляторів.

4. **Тестування**
Після написання коду його потрібно протестувати на наявність помилок.

5. **Впровадження і підтримка**
Коли проєкт готовий, його передають користувачам. Якщо виникають проблеми або необхідно додати нові функції, програмісти підтримують і оновлюють проєкт.

Приклад створення програмного проєкту:

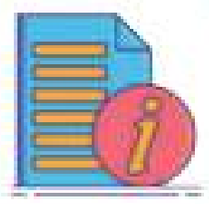
Уявімо, що ми розробляємо калькулятор для розв'язання рівнянь. Спочатку визначаємо, які рівняння буде розв'язувати програма. Потім програміст пише код, наприклад, на **Python**, використовуючи середовище

PyCharm. Після цього програму тестують, щоб переконатися, що вона правильно розв'язує всі рівняння, і нарешті калькулятор стає доступним для учнів.

Опис та інформація про проєкт

Коли над проєктом працюють кілька людей, доцільно робити опис та інформацію про цей проєкт. Для цього використовують спеціальний інформативний файл.

README.txt — це текстовий файл, що містить важливу інформацію про проєкт і зазвичай розміщується в кореневій папці проєкту.



Він допомагає новим користувачам та розробникам швидко зрозуміти призначення проєкту, його функціональні можливості, особливості встановлення та використання.

Як створювати README.txt

Файл **README.txt** створюють у звичайному текстовому редакторі (наприклад, **Notepad** або **Блокнот**). Зберігаючи його у форматі **.txt**, ви забезпечуєте сумісність на всіх платформах та можливість відкрити файл будь-яким текстовим редактором.

Чому важливий README.txt

- полегшує ознайомлення з проєктом
- ефективне спілкування з користувачами
- стандартизація інформації

README.txt використовується як стандартний спосіб представлення інформації про проєкт, які завдання розв'язує, як встановити чи запустити, як використовувати, приклади використання та доповненння, контактна інформація та авторство тощо.

Середовища для написання коду та транслятори — це важливі інструменти для програмістів, які допомагають писати, відлагоджувати та виконувати програми.

Знання про те, як вони працюють, допоможуть вам створювати власні програмні проєкти, працювати з різними мовами програмування та трансляторами.



Практична робота №19.

Завдання 1. Ознайомлення із середовищем розробки (**IDE**)

1. Відкрийте встановлене на вашому комп'ютері середовище розробки **Python**, наприклад, **IDLE**, **PyCharm** або **Visual Studio Code**.
2. Створіть новий проєкт і додайте перший файл, назвавши його **welcome.py**.
3. Напишіть у файлі простий код: `print("Hello, Python!")`

```
print("My name is Oleksandr")
```

4. Запустіть код, щоб побачити результат у вікні консолі.

Запитання для обговорення:

- Які частини інтерфейсу IDE допомогли вам створити і запустити програму?
- Що відображається у вікні консолі після запуску?

Завдання 2. Робота з інтерпретатором Python

1. У середовищі розробки створіть новий файл під назвою **`math_operations.py`**.
2. Напишіть програму, яка виконує кілька простих математичних операцій і виводить результати на екран.
3. Запустіть програму і перевірте, чи результати відображаються правильно.
4. Змініть значення чисел `a` та `b` та знову запустіть програму на виконання.

Запитання для обговорення:

- Які результати ви отримали?
- Що робить кожна з команд **`print()`** у програмі?

```
a = 12
b = 5
print("Сума:", a + b)
print("Різниця:", a - b)
print("Добуток:", a * b)
print("Частка:", a / b)
```

Завдання 3. Знайомство з онлайн-середовищем **OnlineGDB**

1. Відкрийте **OnlineGDB**
   - Перейдіть за покликанням **https://www.onlinegdb.com/**.
   - У верхньому лівому куті оберіть мову програмування **Python** (або **Python 3**). Ви побачите головний екран редактора коду.



2. Запишіть код простої програми

```
print("Hello, OnlineGDB!")
print("Я навчаюся програмувати мовою Python")
```

3. Запустіть програму

- Натисніть на кнопку **Run (Запустити)** у верхній частині екрану.



- Ви побачите результат у вікні **Output (Вивід)** під редактором коду.

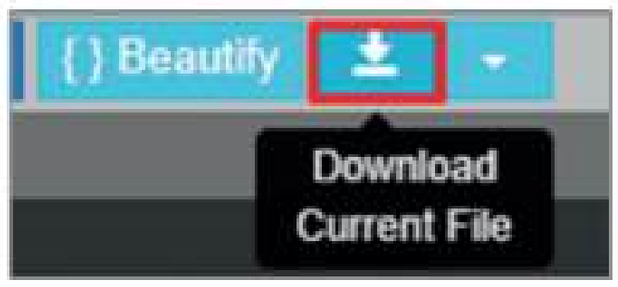
4. Додайте програму з введенням даних

- Напишіть нову програму, яка запитує ім'я користувача і виводить привітання:

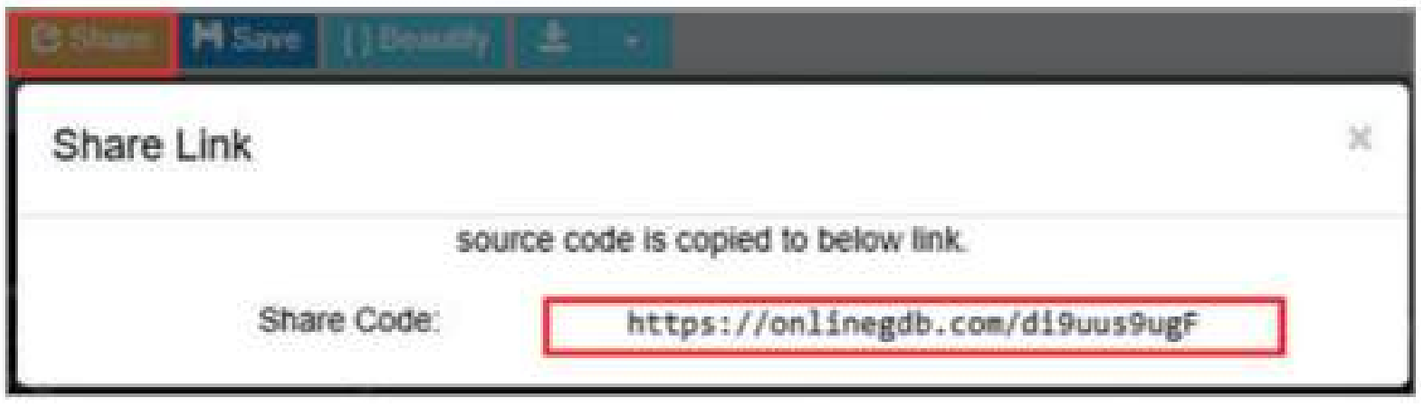
```
name = input("Введіть ваше ім'я: ")
print("Привіт,", name, "!")
```

- Натисніть **Run** та введіть своє ім'я у вікні введення, коли програма запросить це.

5. Завантаження та поширення коду



- Натисніть (**Завантажити**) у верхній частині вікна. Завантажений файл помістіть у свою робочу папку. Відкрийте цей файл у редакторі, який є на вашому персональному комп'ютері, та запустіть його на виконання.
- Щоб поділитися кодом, потрібно натиснути **Share** (**Поширити**) у верхній частині вікна. Скопіюйте покликання та поділіться з вчителем.



Запитання для обговорення:

- Що відображається у вікні виводу?
- Чи є різниця між запуском коду на комп'ютері та в онлайн-середовищі?
- Як можливість зберігати та ділитися кодом допомагає в командній роботі над проєктом?

Ключові поняття

Мова програмування
Переваги Python
Змінна
Типи даних
Бібліотека Tkinter

Мова програмування

Програмування — це спосіб «спілкування» з комп'ютером за допомогою спеціальних інструкцій.



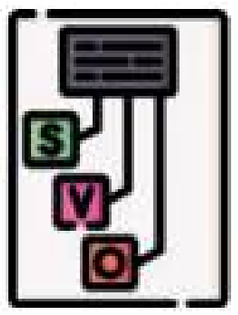
Такі інструкції для виконання завдань надає мова програмування. Є різні мови програмування: **Python**, **C++**, **Java**, **JavaScript** та інші. Кожна з них має свої особливості, але основні поняття, такі як змінні та типи даних, однакові для всіх мов. Із мовами **Scratch** та **Python** ви уже ознайомлювалися в попередніх класах.

Що таке мова програмування?

Мова програмування — це набір правил і команд, що дозволяють людям створювати програми для комп'ютера.

Вона перекладає наші інструкції на мову, зрозумілу комп'ютеру, який виконує програми.

Основні характеристики мов програмування:



Синтаксис | Семантика | Лексика

Синтаксис — набір правил для написання коду. Це своєрідна граматика мови, яка визначає, як правильно писати інструкції.

Семантика – значення команд і конструкцій. Важливо не тільки правильно написати код, а й зрозуміти, що він робить.

Лексика – це набір символів і ключових слів, з яких складаються програми. Кожна мова має свій набір ключових слів.

Класифікація мов програмування

Мови низького рівня

Мови високого рівня

Мови низького рівня: найближчі до машинного коду, який виконують комп'ютери. Приклад – **Assembler**. Кожна команда відповідає машинній інструкції процесора. Такі мови вимагають від програміста глибоких знань про архітектуру комп'ютера.

Мови високого рівня: легкі для розуміння людьми (наприклад, **Python, Java, C++**). Програміст може писати код, не переймаючись деталями роботи комп'ютера.

Програма, написана мовою програмування, виконує конкретні дії: обчислення, взаємодія з користувачем, робота з файлами тощо.

Мова програмування Python

У 8 класі ми будемо з вами продовжувати вивчати мову програмування **Python**.



Python — одна з найпопулярніших і найзрозуміліших мов програмування, особливо для новачків.

Вона використовується для навчання через свою простоту, але її можливості дозволяють застосовувати **Python** і для складних задач.

Переваги Python

Простий синтаксис

Python використовує прості інструкції, які легко зрозуміти навіть початківцям.

Широке застосування

Python використовується у веброзробці, науці про дані, штучному інтелекті, машинному навчанні, розробці ігор та інших галузях.

Кросплатформеність

Програми на **Python** працюють на різних операційних системах (**Windows**, **macOS**, **Linux**).

Велика спільнота

Багато розробників працюють з **Python**, тому для цієї мови існує безліч безкоштовних матеріалів для навчання та готових бібліотек для спрощення роботи.

Приклад простого коду на **Python**:

```
print("Привіт, світ!")
```



Ця програма виведе на екран фразу `"Привіт, світ!"`. Це класичний приклад для початківців у багатьох мовах програмування.

Змінна

Змінна в Python — це іменований контейнер для зберігання даних, який може бути використаний для подальших обчислень або операцій.

Змінна дозволяє програмі запам'ятовувати значення, щоб використовувати їх пізніше. Уявімо, що змінна — це коробка, в яку можна покласти значення (наприклад, число або текст), а потім знову використовувати його в програмі.

Python є мовою з динамічною типізацією, тобто не потрібно явно вказувати тип змінної при її оголошенні. Тип змінної визначається автоматично на основі присвоєного значення.

Змінна має дві важливі характеристики:

- **Ім'я (назва)**. Як коробка має етикетку, так і змінна має своє ім'я.
- **Значення.** Це те, що зберігається у змінній. Значенням можуть бути числа, текст, логічні значення тощо.

Правила іменування змінних

Імена змінних повинні відповідати певним правилам:

1. Ім'я може містити літери, цифри та символ підкреслення (_), але не може починатися з цифри.
2. Ім'я змінної чутливе до регістру: `myName`, `MyName` та `myname` — це різні змінні.
3. Ім'я не може збігатися з ключовими словами (зарезервованими словами) **Python**, наприклад: `if`, `while`, `for`, `import` тощо.
4. Ім'я повинно бути інформативним, щоб полегшити читання і розуміння коду.

Приклади правильних імен:

```
age = 25
user_name = "Inna"
total = 100.50
```

Приклади неправильних імен:

```
1name = "Bob"  # починається з цифри
my-name = "Eva" # містить тире
```

Як працюють змінні?

Коли ми створюємо змінну, комп'ютер виділяє місце в пам'яті, щоб зберігати її значення. Згодом ми можемо змінювати значення змінної.

```
x = 5    # Створюємо змінну x зі значенням 5
x = 10   # Тепер змінна x зберігає нове значення 10
```

Типи даних

Типи даних для змінних

У **Python**, як і в інших мовах програмування, є різні типи даних — це категорії, що визначають, які значення може зберігати змінна і які операції можна над ними виконувати. Розглянемо основні типи даних у **Python**.

Цілі числа (int)

Цілі числа (`int`): це числа без дробової частини.

Наприклад, `5, -3, 100`.

```
age = 30
```

Дробові числа (float)

Дробові числа (`float`): числа з дробовою частиною.

Наприклад, `3.14, 0.5, -7.89`.

```
temperature = 36.6
```

Рядки (str)

Рядки (`str`): це послідовність символів, що можуть містити букви, цифри, символи та пробіли. Рядки у **Python** беруться в одинарні або подвійні лапки.

```
name = "Аліна"
greeting = 'Привіт'
```

Логічний тип (bool)

Логічний тип (`bool`): має лише два значення: `True` (істина) і `False` (хиба). Цей тип використовується для перевірки умов.

```
is_active = True
```

Бібліотеки та модулі в Python

Пригадаємо, що **Бібліотека** в **Python** — це набір готових до використання модулів та функцій, які розширюють стандартні можливості мови. Замість того, щоб писати код з нуля для кожної задачі, програмісти можуть використовувати вже готові рішення, які містять бібліотеки. Це значно спрощує розробку програм та підвищує продуктивність.

Модуль — це файл з кодом **Python**, що може містити змінні, функції та класи. Модуль є одиницею, з якої складається бібліотека. Бібліотека може складатися з одного або декількох модулів.

В 7 класі ви уже могли ознайомитися з такими модулями, як **`math`**, **`random`** та **`turtle`**. Та в курсі 8 класу будемо працювати ще з одним модулем — **`tkinter`**.

Бібліотека Tkinter

Tkinter — це стандартна бібліотека **Python** для створення графічних інтерфейсів (GUI — Graphical User Interface).



Вона є інтерфейсом до Tcl/Tk (бібліотеки для створення графічних інтерфейсів), яка інтегрована з **Python**.

Tkinter дозволяє створювати вікна, кнопки, текстові поля та інші компоненти інтерфейсу користувача, необхідні для створення десктопних програм.

Tkinter входить до складу стандартної бібліотеки **Python**, тому не потребує додаткової інсталяції.

Створення першого вікна в Tkinter

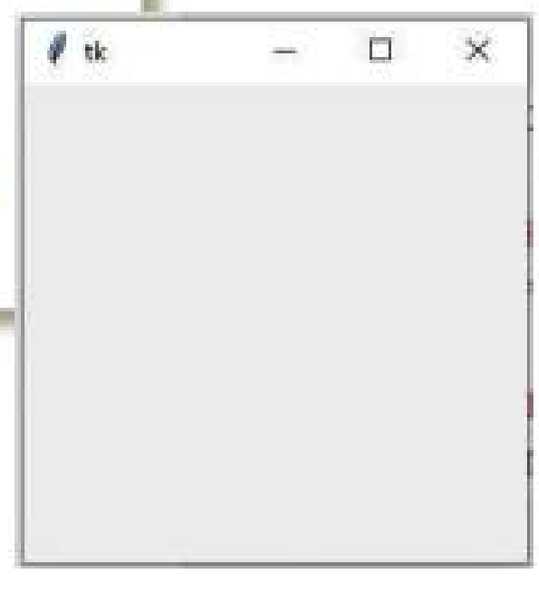
Щоб створити просте вікно, необхідно виконати кілька кроків:

1. Імпортувати бібліотеку **Tkinter**
2. Створити екземпляр головного вікна
3. Запустити головний цикл програми (обробка подій)

```
from tkinter import *  # Імпорт бібліотеки Tkinter

# Створюємо головне вікно
window = Tk()

# Запускаємо головний цикл програми
window.mainloop()
```



У цьому коді створюється просте вікно. Функція **`mainloop()`** запускає головний цикл подій, необхідний для підтримки роботи програми.

Властивості вікна

Tkinter дозволяє не лише створювати власні вікна, а й налаштовувати їхній зовнішній вигляд, розмір та функціональність. Ось ключові властивості вікна **Tkinter**:

1. Заголовок вікна (title)

Ця властивість дозволяє встановити текст у заголовку вікна.

`window.title("Моє перше вікно")`

2. Розмір вікна (geometry)

Встановлює початкові розміри вікна (ширину та висоту) у пікселях.

`window.geometry("400x300")` — встановлює розмір 400x300 пікселів

Можна також задати початкові координати вікна, наприклад:

`window.geometry("400x300+200+100")` — розмір 400x300 і позиція (200, 100) на екрані

3. Мінімальний та максимальний розмір вікна (minsize, maxsize)

Встановлює мінімальні та максимальні розміри вікна.

`window.minsize(200, 150)` — мінімальні розміри: 200x150 пікселів

`window.maxsize(800, 600)` — максимальні розміри: 800x600 пікселів

4. Зміна розміру вікна (resizable)

Ця властивість дозволяє або забороняє зміну розміру вікна по горизонталі та вертикалі.

`window.resizable(True, True)` — дозволяє змінювати розмір вікна

`window.resizable(False, False)` — забороняє змінювати розмір вікна

5. Іконка вікна (iconbitmap)

Можна встановити власну іконку для вікна, використовуючи **.ico** файл.

```
window.iconbitmap('my_icon.ico')
```

Є ще один варіант, який дозволяє встановити іконку не лише розширення **.ico**, а й, наприклад, **.png**.

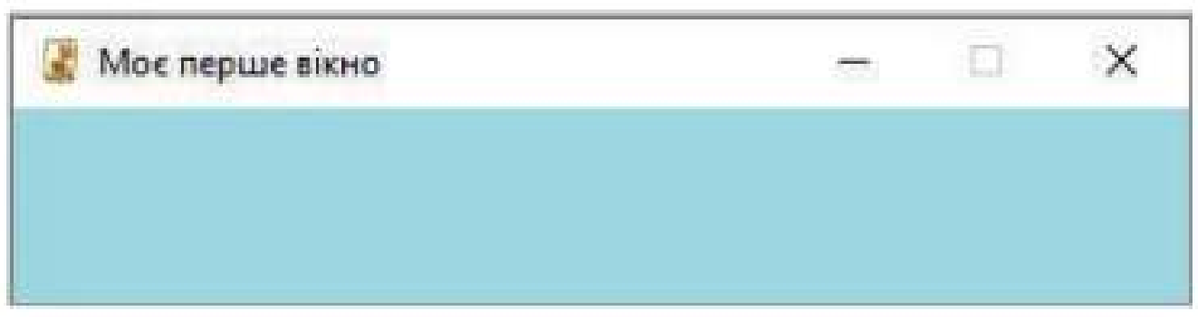
`icon = PhotoImage(file="my_icon.png")` — завантажує зображення для іконки

`window.iconphoto(False, icon)` — встановлює іконку

6. Фон вікна (configure)

Ця властивість дозволяє змінювати фон вікна.

`window.configure(bg='lightblue')` — змінює фон вікна на блакитний



Практична робота №20.

Завдання 1. Створення змінних та робота з типами даних.

Напишіть програму, яка створює чотири змінні різних типів даних:

- Строкову змінну з вашим ім'ям.
- Числову цілочисельну змінну з вашим віком.
- Числову змінну з дробовою частиною, яка відображає ваш зріст в метрах.
- Логічну змінну, яка вказує, чи любите ви програмування.

```
name = "Олександр"
age = 14
height = 1.62
loves_programming = True
```

Використайте команду **print()**, щоб вивести на екран значення кожної змінної.

```
print("Моє ім'я:", name)
print("Мені", age, "років")
print("Мій зріст:", height, "м")
print("Чи люблю програмувати:",
loves_programming)
```

Завдання 2. Обчислення периметра та площі.

Уявіть, що у вас є сад, який потрібно обгородити парканом та засіяти травою. Для цього потрібно дізнатися периметр та площу території. Створіть програму, яка запитує у користувача довжину і ширину садка. Після цього обчислити та вивести на екран площу та периметр саду.

Використайте команду **input()** для отримання даних від користувача.

```
a = float(input("Введіть довжину саду: "))
```

Для знаходження різних даних використовуйте математичні формули та операції.

```
P = (a + b) * 2
```

Додатково: дізнатися, скільки насіння трави потрібно придбати, якщо відомо, що на 1 м^2 йтиме 300 г насіння.

Завдання 3. Підрахунок грошей у гаманці.

Запитайте у користувача, скільки у нього монет і купюр різного номіналу (1, 2, 5 гривень). Порахуйте загальну суму грошей у гаманці.

Завдання 4. Створення вікон з різними властивостями.

Створіть вікно з **Tkinter**, яке матиме такі властивості:

а) розміри вікна 400x300 пікселів; заголовок вікна: `"Базове вікно"`; колір фону вікна — синій (`"blue"`); забороніть зміну розмірів вікна.

Підказка: використайте методи `title()`, `geometry()`, `configure()` та `resizable()`;

б) розміри вікна 500x500 пікселів; заголовок вікна: `"Обмеження розміру"`; колір фону вікна у форматі HEX (`"#FF5733"` — помаранчевий);

встановіть мінімальний розмір 300x300 та максимальний розмір 800x800 пікселів.

Підказка: використайте методи `minsize()`, `maxsize()` та `configure()`.

в) розміри вікна 450x350 пікселів; заголовок вікна: `"Вікно зі зміщенням"`; колір фону вікна бірюзовий (`"turquoise"`); розмістіть вікно на екрані зі зміщенням 400 пікселів від лівого краю та 200 пікселів від верхнього краю.

Підказка: використайте метод `geometry()` у форматі `geometry("ШиринахВисота+Зміщення_по_X+Зміщення_по_Y")`.

Завдання 5. Створення вікна з власною іконкою.

Створіть вікно розміром 300x200 пікселів. Встановіть заголовок `"Вікно з іконкою"`. Встановіть фоновий колір жовтий (`"yellow"`). Забороніть зміну розмірів вікна.

Встановіть іконку для вікна (використання файлу із розширенням **png**):

- знайдіть зображення невеликого розміру з розширенням **png** та збережіть його у папці з вашою програмою

Примітка: в пошуковому рядку скористатися ключовими словами логотип **png**, зображення **png**, піктограми **png** тощо.

- імпортуйте зображення у вашу програму `icon = PhotoImage(file="my-icon.png")`, де `my-icon` — назва вашого зображення.
- встановіть це зображення як іконку вашого вікна: `window.iconphoto(False, icon)`, де `window` — ім'я вашого вікна, `icon` — змінна, яка відповідає за імпортоване зображення.

Ключові поняття

Функція input()
Функція print()
Присвоєння
Віджет Label
Зображення у мітках
Віджет Entry

Введення та виведення даних у Python

Як ви уже знаєте з попередніх класів, у **Python** для цього використовуються дві основні функції: `input()` для введення даних і `print()` для виведення результатів на екран.

У будь-якій програмі важливо вміти взаємодіяти з користувачем: запитувати необхідні дані та виводити результат.

Функція input()

Функція input() — введення даних

Функція **`input()`** дозволяє програмі отримувати інформацію від користувача. Коли програма досягає цієї функції, вона зупиняється і чекає, поки користувач введе текст і натисне клавішу **Enter**.

Синтаксис:

```
змінна = input("Текст запиту для користувача")
```

Приклад:

```
name = input("Введіть своє ім'я: ")
print("Привіт,", name, "!")
```



Результат:

```
Введіть своє ім'я: Максим
Привіт, Максим !
```

Ця програма попросить користувача ввести ім'я і виведе привітання, наприклад: `"Привіт, Максим !"`.

`input()` завжди повертає значення як рядок (тип даних **`str`**), тому для роботи з числами введені дані потрібно перетворювати на інші типи (наприклад, на ціле число або число з дробом).

Оскільки `input()` завжди повертає значення як рядок (тип даних **`str`**), тому для роботи з числами введені дані потрібно перетворювати на інші типи (наприклад, на ціле число або число з дробом). Іншими словами, вказати тип даних, який потрібно буде ввести з клавіатури.

- Перетворення у ціле число: використовується функція **int()**

Приклад:

```
age = int(input("Введіть свій вік: "))
print("Вам", age, "років.")
```

Результат:

```
Введіть свій вік: 14
Вам 14 років.
```

- Перетворення у число з дробовою частиною: використовується функція **float()**

Приклад:

```
height = float(input("Введіть свій зріст у метрах: "))
print("Ваш зріст:", height, "метрів.")
```

Результат:

```
Введіть свій зріст у метрах: 1.62
Ваш зріст: 1.62 метрів.
```

Функція print()

Функція **print()** використовується для виведення інформації на екран. Вона може виводити текст, числа, змінні або навіть результат обчислень.

Синтаксис:

```
print(дані, які потрібно вивести)
```

print() може виводити кілька значень одночасно, розділяючи їх комами.

Автоматично додає новий рядок після кожного виклику, тому кожен наступний **print()** починається з нового рядка.

Приклад:

```
x = 5
y = 10
print("Значення x:", x)
print("Значення y:", y)
print("Сума x і y дорівнює", x + y)
```

Результат:

```
Значення x: 5
Значення y: 10
Сума x і y дорівнює 15
```

Присвоєння

Присвоєння — це процес збереження значення в змінну.
У **Python** оператор **=** використовується для присвоєння значення змінній.

```
x = 10  # змінна x тепер дорівнює 10
y = 3.14  # змінна y містить дробове число
name = "Аліна"  # змінна name містить рядок "Аліна"
```

Приклад:

```
number = 5
print("Значення змінної number:", number)
number = number + 2  # змінюємо значення змінної
print("Нове значення number:", number)
```

Результат:

```
Значення змінної number: 5
Нове значення number: 7
```

Виведення за допомогою f-рядків

F-рядки (formatted string literals) — це зручний спосіб форматування рядків, який з'явився в **Python 3.6**. Вони дозволяють легко вставляти значення змінних та виразів всередині рядка, що робить код коротшим і легшим для читання.

Як використовувати f-рядки:

```
name = "Анна"
age = 13
print(f"Привіт, {name}! Тобі {age} років.")
```

```
Привіт, Анна! Тобі 13 років.
```

Щоб створити f-рядок, потрібно почати рядок із символу **`f`** перед лапками (**`f"..."`** або **`f'...'`**).

Ви можете вставити значення змінних, виразів чи навіть виклики функцій безпосередньо в рядок, використовуючи фігурні дужки **`{}`**.

Переваги f-рядків:

- *Зручність та компактність:* не потрібно використовувати **`+`** для конкатенації або **`str()`** для перетворення чисел у рядки.
- *Читабельність:* код легко читається, а змінні всередині f-рядків одразу видно.
- *Гнучкість:* можна вставляти вирази відразу в рядок.

Звичайний вивід:

```
print('Сума',x,'і',y,'дорівнює',x + y,'.')
```

```
Сума 5 і 10 дорівнює 15.
```

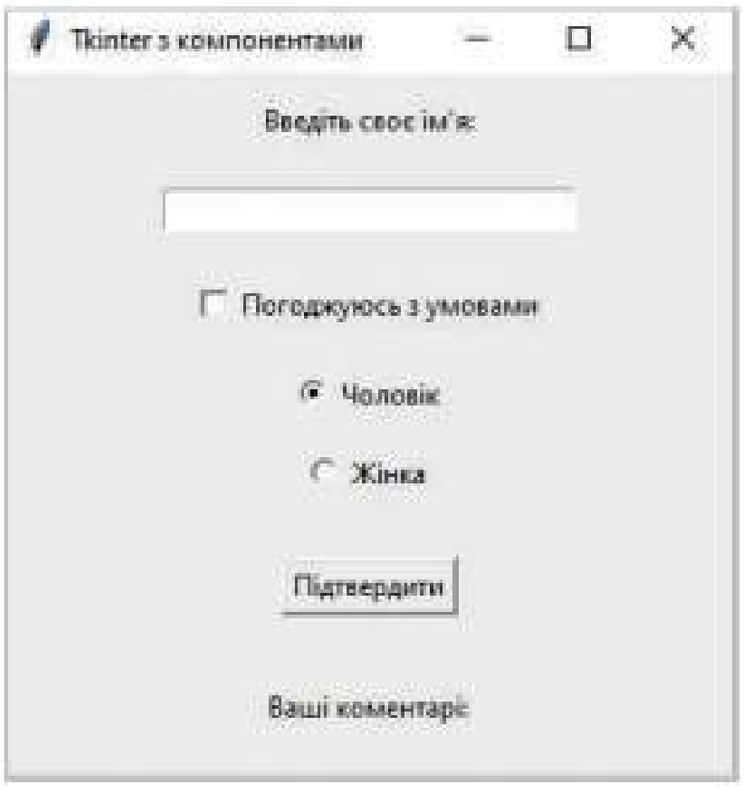
Вивід з використанням f-рядків:

```
x = 5
y = 10
print(f"Сума {x} і {y} дорівнює {x + y}.")
```

```
Сума 5 і 10 дорівнює 15.
```

Введення та виведення даних у Tkinter

Для взаємодії користувача з програмою **Tkinter** дозволяє використовувати багато різних компонентів (віджетів) для створення інтерфейсу. Ось основні з них:



Label — текстовий напис

Button — кнопка

Entry — однорядкове текстове поле

Text — багаторядкове текстове поле

Listbox — список для вибору одного або кількох елементів

Checkbutton — прапорець (checkbox)

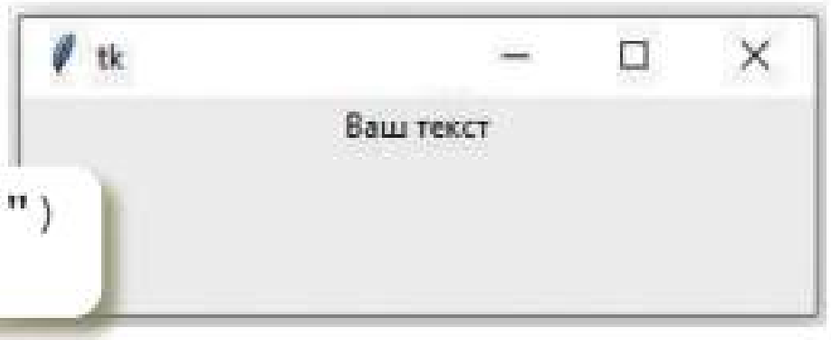
Radiobutton — перемикач (radio button)

Віджет Label

Віджет **Label** може відображати текст, налаштовувати шрифт, колір, вирівнювати текст, а також показувати зображення. Розгляньмо основні властивості та параметри, які можна використовувати для налаштування мітки.

Синтаксис:

```
label = Label(parent, text="Ваш текст")
label.pack()
```



Примітка! Для відображення компонента у вікні використовують метод **pack()**

parent — вказує на батьківський елемент - зазвичай це ім'я вікна або фрейм

text — текст, який потрібно відобразити в мітці

Основні параметри для Label

Розглянемо основні властивості мітки **Label**, які дозволять стилізувати її та розмістити в потрібному місці.

font — параметр для зміни шрифту та розміру тексту (наприклад, ("Arial", 14))

fg — (foreground), задає колір тексту

bg — (background), задає колір фону

width та height: — задають ширину та висоту мітки в символах

anchor — вирівнює текст у межах мітки (наприклад, anchor="center", anchor="w" для вирівнювання по лівому краю тощо).

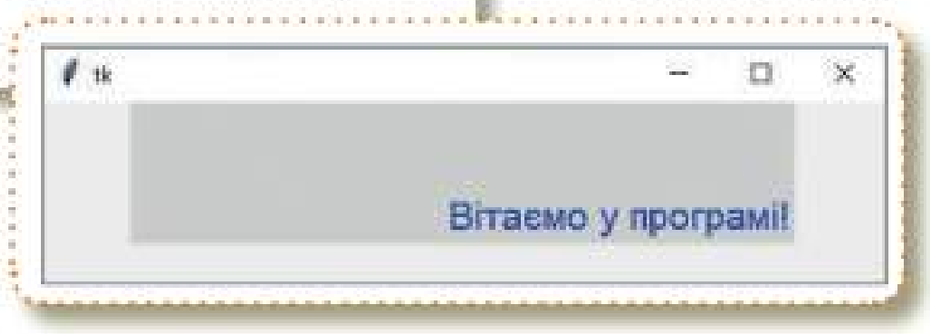
Додатково: Список можливих значень **anchor**:

n — вирівнює текст по верхньому краю по центру.
s — вирівнює текст по нижньому краю по центру.
e — вирівнює текст по правому краю по центру.
w — вирівнює текст по лівому краю по центру.
center — вирівнює текст по центру (значення за замовчуванням).
nw — вирівнює текст у верхньому лівому куті.
ne — вирівнює текст у верхньому правому куті.
sw — вирівнює текст у нижньому лівому куті.
se — вирівнює текст у нижньому правому куті.



```
label = Label(window, text="Вітаємо у програмі!",
              font=("Arial", 16), fg="blue",
              bg="lightgray",
              width=30, height=3, anchor="se")
label.pack()
```

Результат:



Зображення у мітках

Виведення зображень у мітках

Крім тексту **Label** також може відображати зображення. Для цього використовується клас **PhotoImage** у **Tkinter**, який підтримує лише формат **.png** та **.gif** .

```
img = PhotoImage(file="logo.png")
img = img.subsample(3,3)
image_label = Label(window, image=img)
image_label.pack()
```



Як працює програма

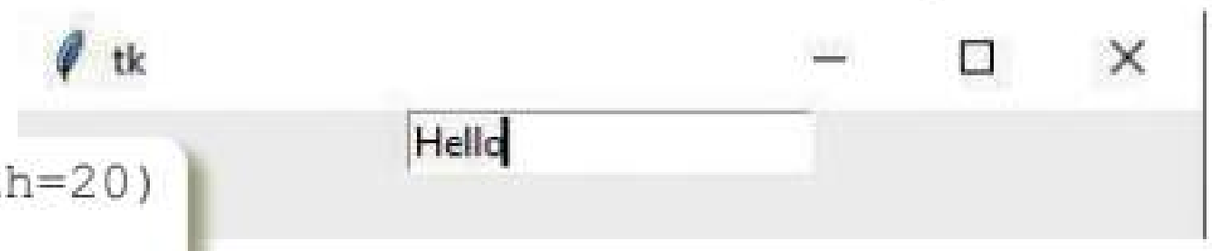
- Програма завантажує зображення **logo.png**.
- Метод **subsample()** зменшує розмір зображення у кілька разів. У цьому випадку зображення зменшується втричі як по ширині, так і по висоті.
- Віджет **Label** створюється для відображення цього зображення, яке ми передали через параметр **image=img, у вікні window**.
- Нарешті, віджет **Label** додається у вікно і автоматично розміщується.

Віджет Entry

Віджет **Entry** використовується для створення однорядкових полів введення, які дозволяють користувачеві вводити текст або числа, які потім можна використовувати в програмі. Це один із найпоширеніших віджетів для прийому інформації від користувача в інтерфейсах **Python**-програм.

Синтаксис:

```
entry = Entry(parent, width=20)
entry.pack()
```



Примітка! Для відображення компонента у вікні використовують метод **pack()**.

parent — вказує на батьківський елемент; зазвичай це ім'я вікна або фрейм
width — задає ширину поля введення в символах

Основні параметри для Entry

Під час створення віджета **Entry** можна використовувати кілька параметрів для налаштування його зовнішнього вигляду та поведінки.

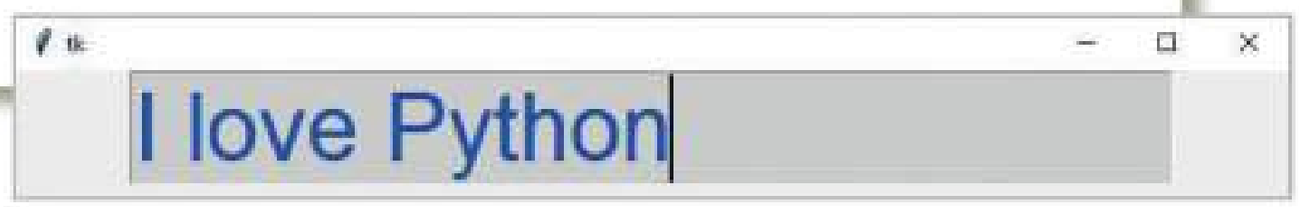
width — задає ширину поля введення в кількості символів
show — дозволяє приховати введений текст (наприклад, для полів введення паролів можна використовувати `show="*"`)
state — задає стан поля (`normal`, `disabled` або `readonly`)

Аналогічно, як і в **Label**, поле **Entry** також має параметри **font**, **fg**, **bg**.

Наприклад:

```
entry = Entry(window, width=20, font=("Arial",
40), fg="blue", bg="lightgray")

entry.pack()
```



Основні методи Entry

У цього віджета є кілька корисних методів, які допомагають взаємодіяти з текстом, вводити дані, отримувати введений текст, очищати поле та багато іншого. Розглянемо основні з них.

get() — отримує текст, який користувач ввів у поле.

```
entry_text = entry.get()
print(entry_text)
```

У результаті текст, який користувач ввів у поле **Entry**, буде збережений у змінній `entry_text` та виведений у консоль.

delete(start, end) — видаляє текст із поля **Entry**

```
entry.delete(0, "end")  # Видалити весь текст
entry.delete(5)  # Видалити символ на 5-й позиції
```

Можна видалити як окремі символи, так і весь текст.

Аргументи:

- **start** — початковий індекс для видалення.
- **end** — кінцевий індекс. Якщо його не вказати, буде видалено лише символ на позиції **start**.
- Якщо потрібно видалити весь текст, можна вказати "0" як початкову позицію і "**end**" як кінцеву.

insert(index, string) — вставляє текст у поле **Entry** на зазначену позицію

```
entry.insert(0, "Привіт")
```

Аргументи:

index — індекс (позиція) для вставки тексту. Наприклад, "0" вставить текст на початок, а "**end**" — в кінець.

string — текст, який потрібно вставити.

Проте використання методів поля **Entry** у поєднанні з кнопкою **Button** дозволяє ефективно обробляти введені користувачем дані в графічному інтерфейсі, забезпечуючи зручну взаємодію для введення інформації та виконання необхідних дій на основі отриманих даних.

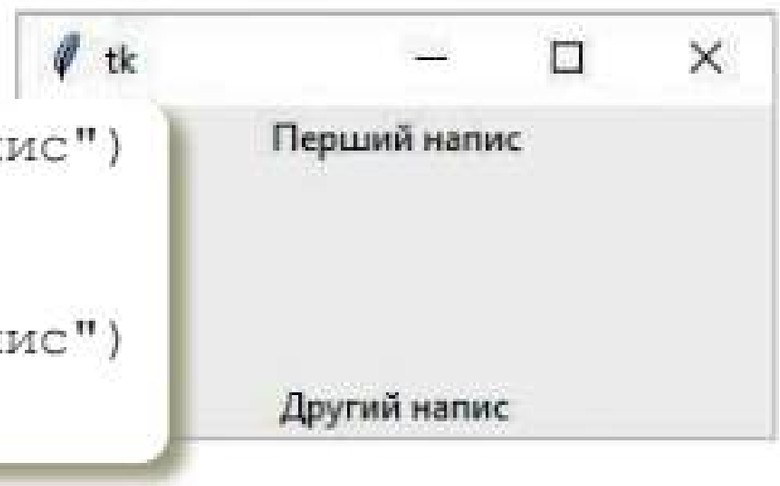
Розміщення віджетів у вікні

У **Tkinter** є кілька способів розміщення віджетів у вікні. Ці способи дозволяють вам контролювати, як і де будуть відображатися елементи інтерфейсу (віджети) у головному вікні.

pack() — автоматично розташовує віджети у вікні один за одним у вказаному напрямку

```
label1 = Label(window, text="Перший напис")
label1.pack(side="top")

label2 = Label(window, text="Другий напис")
label2.pack(side="bottom")
```

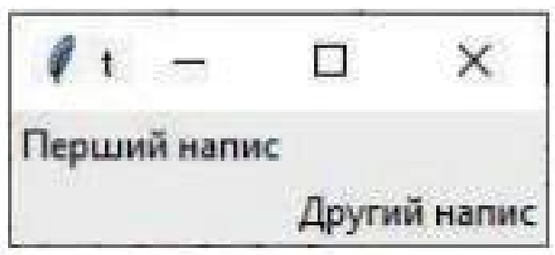


Метод **pack()** розміщує віджети один під одним у вікні або поряд. Віджети можуть бути вирівняні за напрямками: ліворуч (`left`), праворуч (`right`), зверху (`top`) або знизу (`bottom`). Якщо не вказувати напрям, то за замовчуванням елементи вирівнюються зверху.

grid() — розміщує віджети у вигляді сітки з рядків і стовпців

```
label1 = Label(window, text="Перший напис")
label1.grid(row=0, column=0)

label2 = Label(window, text="Другий напис")
label2.grid(row=1, column=1)
```



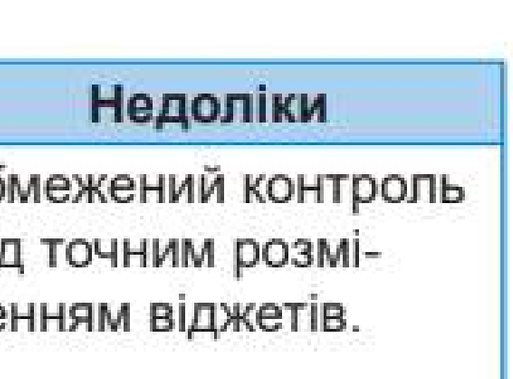
Основні параметри `grid()`:

- **`row`**: номер рядка, в якому буде розміщений віджет (починається з 0).
- **`column`**: номер стовпця, в якому буде розміщений віджет (починається з 0).
- **`sticky`**: вирівнювання віджета в межах комірки. Можливі значення:
  - `"n"` (*north*, зверху),
  - `"s"` (*south*, знизу),
  - `"e"` (*east*, праворуч),
  - `"w"` (*west*, ліворуч).

  Також можна комбінувати ці напрямки, наприклад, `"nsew"` — для розтягування віджета на всю комірку.
- **`padx і pady`**: встановлює відступи всередині комірки.
- **`columnspan`**: вказує, скільки стовпців буде займати віджет.
- **`rowspan`**: вказує, скільки рядків буде займати віджет.

`place()` — розміщує віджети за точними координатами

```
label = Label(window, text="Напис")
label.place(x=50, y=100)
```



Основні параметри `place()`:

- **`x і y`** — точні координати розміщення віджета (у пікселях).
- **`relx і rely`** — відносні координати (значення від 0 до 1), де 0 — це початок вікна, а 1 — кінець.
- **`anchor`** — визначає точку прив'язки віджета відносно координат (`n, s, e, w, ne, nw, se, sw` або `center`).

Порівняння методів

Метод	Опис	Переваги	Недоліки
`pack()`	Автоматично розміщує віджети один під одним або поруч.	Простий для використання в простих інтерфейсах.	Обмежений контроль над точним розміщенням віджетів.
`grid()`	Розміщує віджети у вигляді сітки з рядків і стовпців.	Зручно для форм і складних інтерфейсів.	Вимагає трохи більше налаштувань.
`place()`	Дозволяє точно вказати координати для віджетів.	Повний контроль над позицією.	Менш гнучкий, важче змінювати інтерфейс.

Підсумок:

- Для простих інтерфейсів найкраще використовувати **`pack()`**.
- Для форм або інтерфейсів з сітчастою структурою краще підходить **`grid()`**.
- Якщо потрібне точне позиціонування віджетів, використовуйте **`place()`**.

Віджет Frame

Якщо одночасно використовувати метод розміщення **grid()** з іншими методами — програма видаватиме помилку. Це відбувається через конфлікт між методами **.pack()** і **.grid()**, які не можна використовувати одночасно в одному батьківському контейнері (window).

Щоб усунути цю проблему, необхідно застосовувати лише один з цих методів для всіх елементів або використовувати окремий **Frame** для компонентів з **.grid()**.

Фрейм — це свого роду «вікно» всередині вікна. Тобто компонент, який використовується для групування та організації інших віджетів.

Він впорядковує компоненти у вікні програми, спрощує розташування елементів і забезпечує зручну структуру коду.

Основні властивості Frame:

bg — задає колір фону фрейму

width height — задають розміри фрейму

relief — визначає стиль рамки (flat, groove, raised, ridge, solid, sunken).

bd — задає ширину рамки

highlightbackground — колір рамки фрейму

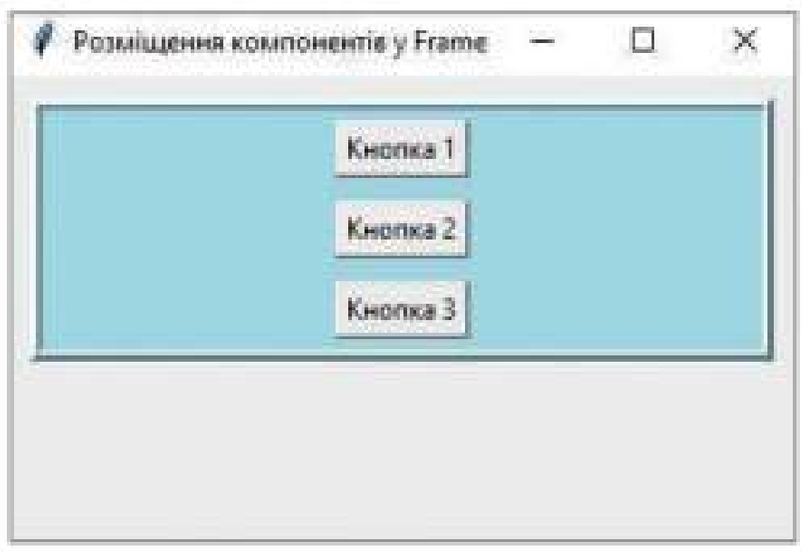
highlightthickness — товщина рамки фрейму

Створення Frame

Фрейм створюється за допомогою класу **Frame**. Його можна розташувати в основному вікні або всередині іншого фрейму.

Синтаксис:

```
frame = Frame(parent, options)
```



parent — батьківський елемент, до якого прив'язаний фрейм (зазвичай це основне вікно)

options — параметри стилю (колір, рамка, розміри тощо)

Практична робота №21.

Завдання 1. Вітання користувача

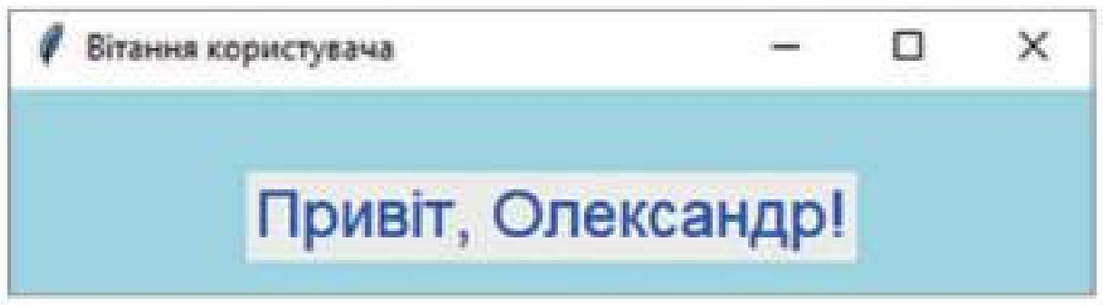
Створити програму за допомогою бібліотеки **Tkinter**, яка буде вітати користувача.

- імпортувати бібліотеку **tkinter**
- отримати ім'я користувача

```
name = input("Введіть своє ім'я: ")
```

- створити вікно з розмірами 400x100, заголовком "Вітання користувача" та кольором фону `lightblue`
- додати до вікна компонент **Label**, який буде із відступом 30 пікселів від верхньої межі вікна та матиме такі властивості: текст "Привіт, [ваше ім'я]!"; шрифт **Arial** з розміром 18; колір тексту синій.

```
label = Label(window, text=f"Привіт, {name}!",
              font=("Arial", 18), fg="blue")
label.pack(pady=30)
```



Завдання 2. Розрахунок калорій, спалених під час бігу

Користувач вводить вагу (в кг) та тривалість бігу (в хвилинах). Написати програму, яка виводитиме кількість калорій, спалених під час бігу (спалюється приблизно 10 калорій за 1 хвилину на 1 кг ваги).

Рекомендаційні дії:

- записати команди для вводу ваги та тривалості бігу з клавіатури
- обчислити кількість калорій
- вивести інформацію в консоль за допомогою f-рядків:

`print(f"За {time} хв ви спалите {calories} калорій")`, де `time` — тривалість бігу, а `caloris` — кількість калорій.

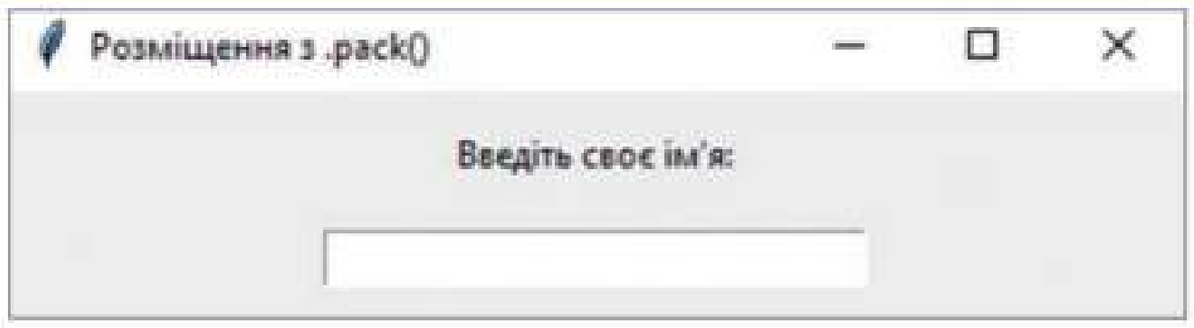
- запустити програму на виконання

Додатково: імпортувати модуль **Tkinter** та вивести інформацію в компонент **Label**.

Завдання 3. Розміщення компонентів у вікні

а) Створіть вікно розміром 400x100 пікселів із заголовком "Розміщення з **.pack()**". Додайте **Label** з текстом "Введіть своє ім'я:" у верхній частині вікна. Додайте **Entry** для введення тексту під **Label**.

Використайте метод **.pack()** з параметром `side=TOP` для обох компонентів.



Підказка: Використайте параметри `padx`, `pady` для додавання відступів.

Додатково: задати властивості вікна та компонентів на свій смак.

б) Створіть вікно розміром 400x200 пікселів із заголовком `"Розміщення з .place()"`. Додайте **Label** з текстом `"Введіть свій вік:"` у позиції `(x=50, y=80)`.

Додайте **Entry** для введення віку у позиції (`x=200, y=80`). Використайте метод `.place()` для точного позиціонування компонентів.



Підказка: Використайте параметри `width` та `height` для налаштування розмірів компонентів.

Додатково: задати властивості вікна та компонентів на свій смак.

в) Створіть вікно розміром 300x200 пікселів із заголовком `"Розміщення з .grid()"`. Додайте **Label** з текстом `"Ім'я користувача:"` у перший рядок першої колонки. Додайте **Entry** у перший рядок другої колонки для введення імені.

Додайте **Label** з текстом `"Пароль:"` у другий рядок першої колонки. Додайте **Entry** у другий рядок другої колонки для введення пароля (використайте параметр `show='*'`). Додайте **Label** з текстом `"Мітка, яка займає два стовпці"` в центрі третього рядка (об'єднайте дві колонки). Використайте метод `.grid()` з параметрами `row` та `column`.



Підказка: Використайте параметри `columnspan`, `sticky`, `padx` та `pady` для налаштування розташування компонентів.

`sticky="e"` вирівнює текст по правому краю

```
# Додання мітки "Ім'я користувача:"
username_label = Label(window, text="Ім'я користувача:")
username_label.grid(row=0, column=0, padx=10, pady=10, sticky="e")
# Додання мітки, що займає два стовпці
center_label = Label(window, text="Мітка, яка займає два стовпці")
center_label.grid(row=2, column=0, columnspan=2, pady=10)
```

Примітка! **sticky** використовується для вирівнювання (приклади: "n", "e", "s", "w" — зверху, справа, знизу, зліва)

Додатково: задати властивості вікна та компонентів на свій смак.

Завдання 4. Створення вікна із зображенням та комбіноване використання методів **.pack()**, **.place()** та **.grid()**. Створення компонента **Frame**.

Створіть вікно розміром 600x400 пікселів із заголовком "Комбіноване розміщення".

У верхній частині вікна (за допомогою **.pack()**) додайте **Label** з текстом "Форма реєстрації".

Додайте зображення до форми та розмістіть його, використовуючи метод **.pack()**

```
img = PhotoImage(file="photo.png") - імпорт зображення
img = image.subsample(4,4) - масштаб зображення в 4 рази менше по ширині і по висоті
image_label = Label(window, image=img) - створення мітки, в якій знаходитиметьсмя зображення
image_label.pack(pady=10)
```

Примітка:

Переконайтеся, що зображення **photo.png** міститься в одній папці з вашим файлом **.py**.

PhotoImage підтримує лише формати **.png**, **.gif**, **.ppm**, **.pgm**.

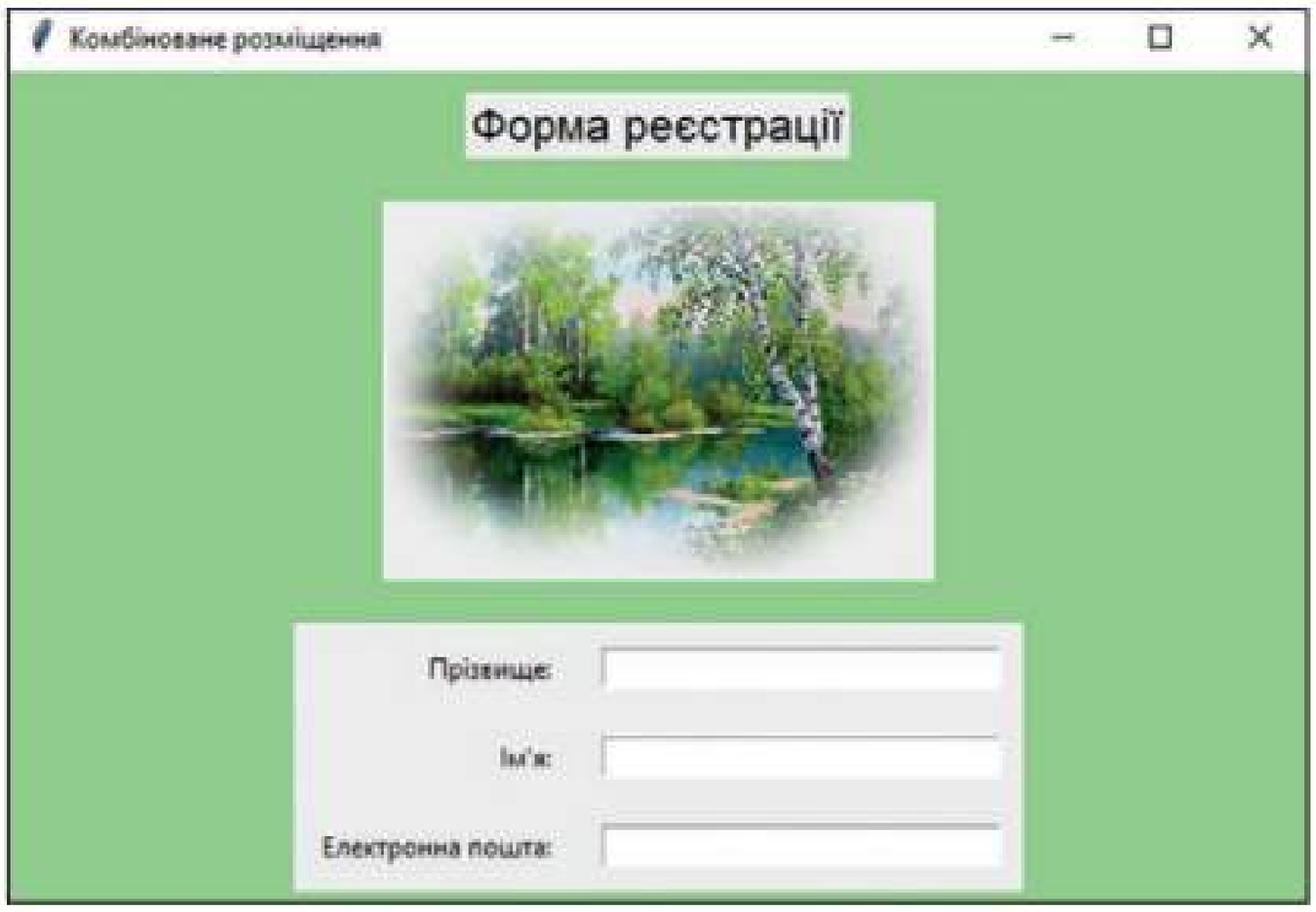
Додайте компонент **Frame** після зображення

```
form_frame = Frame(window)
form_frame.pack(pady=10)
```

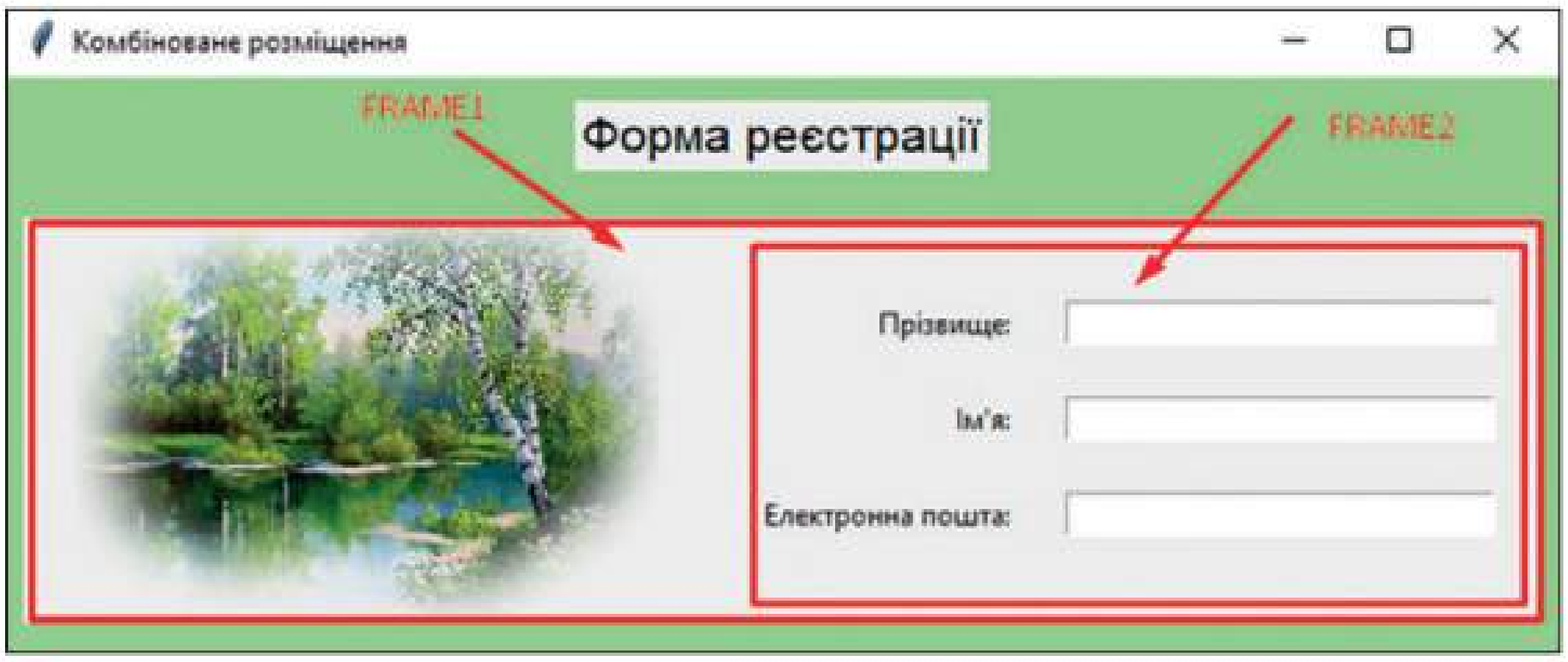
Використайте **.grid()** для створення форми, розмістивши пропоновані компоненти вже не у вікні **window**, а у фреймі **form_frame**:

Label "Прізвище:" та відповідне **Entry** (рядок 0). **Label** "Ім'я:" та відповідне **Entry** (рядок 1). **Label** "Електронна пошта:" та відповідне **Entry** (рядок 2).

```
surname_label = Label(form_frame, text="Прізвище:")
surname_label.grid(row=0, column=0, padx=10, pady=10, sticky="e")
surname_entry = Entry(form_frame, width=30)
surname_entry.grid(row=0, column=1, padx=10, pady=10)
```



Примітка! Для різноманітного розміщення компонентів різними методами можна використовувати кілька фреймів.



Ключові поняття

- Арифметичні операції
- Математичні функції
- Віджет Button
- Отримання даних з Entry
- Обробка подій

Арифметичні операції в Python

Арифметичні операції у Python



У 7 класі, вивчаючи мову програмування **Python**, ви могли ознайомитися та попрацювати з математичними операціями. Пригадаємо, який стандартний набір арифметичних операцій для базових математичних обчислень надає нам **Python** та розширимо знання про їхнє використання за допомогою бібліотеки **Tkinter**:

+	додавання
–	віднімання
*	множення
/	ділення (завжди повертає число з плаваючою крапкою)
//	цілочислове ділення (повертає цілу частину від ділення)
%	залишок від ділення
**	піднесення до степеня

Математичні функції

Для розширених математичних обчислень **Python** надає бібліотеку **math**, яка містить різні математичні функції, такі як обчислення кореня тощо.

`sqrt(x)`	корінь квадратний з числа x
`pow(x, y)`	піднесення числа x до степеня y
`factorial(x)`	факторіал числа x
`pi`	число π
`fabs(a)`	модуль числа

Функції у Python

У **Python** функції є основними будівельними блоками для організації коду.

Функція — це окремий блок коду, який виконує певну дію та може бути викликаний в інших частинах програми.

Функції допомагають зробити код більш структурованим, зменшити дублювання коду та полегшити його підтримку.

Оголошення функції: функції в **Python** визначаються за допомогою ключового слова **def**, після якого йде ім'я функції, дужки з параметрами (якщо є) та двокрапка. Всі команди, що входять до функції, повинні бути відступлені.

Синтаксис:

```
def імя_функції(параметри):
    Тіло функції
    Дії, які виконує функція
    return результат
```

Приклад простої функції:

```
def привітання():
    print("Привіт, Python!")
```

Щоб викликати функцію, достатньо звернутися до неї за ім'ям:

```
привітання()  # Виведе: Привіт, Python!
```

Примітка! У 8 класі ми лише поверхнево ознайомимося з функціями, а саме виклик функцій та методів за допомогою віджета **Button**. Більш детально з функціями ви працюватимете в старших класах.

Віджет Button

Віджет **Button** у бібліотеці **Tkinter** використовується для створення інтерактивних кнопок у графічному інтерфейсі користувача (GUI). Кнопки використовуються для виконання дій, коли користувач натискає на них. Віджет **Button** може бути налаштований для відображення тексту або зображень, а також для виклику функцій у відповідь на події (натискання кнопки).

Синтаксис:

```
button = Button(parent, text="Текст кнопки", command=function)
button.pack()
```

parent — вказує на батьківський елемент — зазвичай це ім'я вікна або фрейм

text — текст, який буде відображений на кнопці

command — функція, яка викликатиметься при натисканні на кнопку

Такі параметри, як **state**, **width** і **height**, **fg**, **bg**, також використовуються і для кнопки.



Приклад простої кнопки:

Створюємо функцію, яка викликається при натисканні на кнопку

```
def on_button_click():
    print("Кнопка натиснута!")

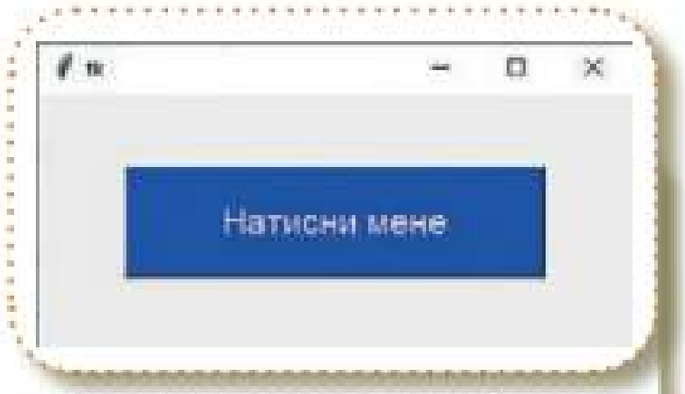
button = Button(window, text="Натисни мене", command=on_button_
click)
button.pack()
```

Кнопка з текстом **Натисни мене** викликає функцію **on_button_click()**, яка виводить повідомлення **Кнопка натиснута!** у консоль, коли користувач натискає на кнопку.

Кнопка натиснута!



Приклад кнопки із стилями:




```
def on_button_click():
    print("Кнопка натиснута!")

# Створюємо кнопку з налаштуванням вигляду
button = Button(window, text="Натисни мене", command=on_button_
click,
                font=("Arial", 16), fg="white", bg="blue",
width=20, height=2)
button.place(relx=0.5, rely=0.5, anchor="center")
```

Функція зміни властивостей компонентів

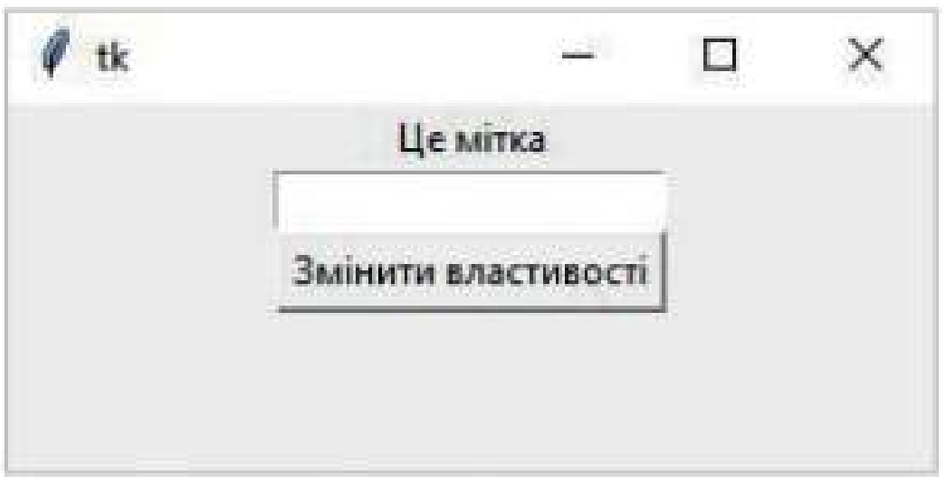
Ви можете змінювати властивості компонентів за допомогою методу **config()**. Цей метод дозволяє змінювати властивості, такі як колір фону, текст, шрифт та інші атрибути компонентів під час виконання програми.

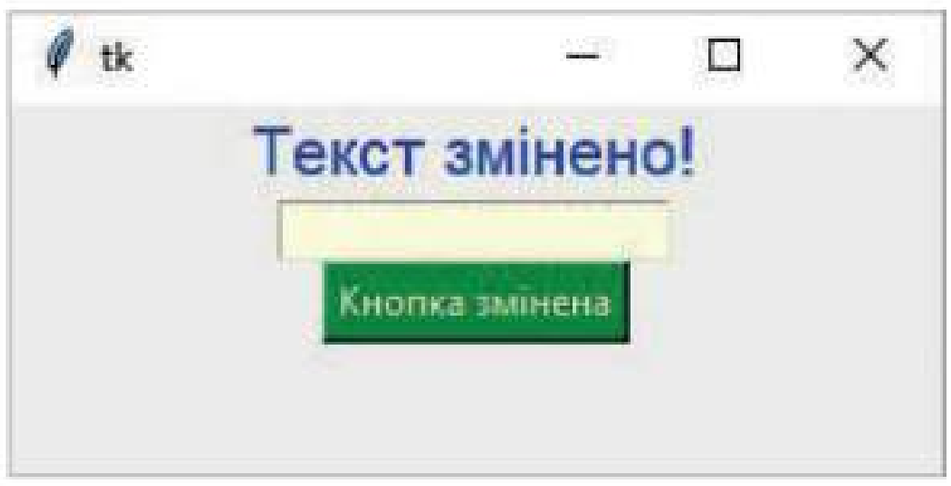
Ось приклад функції, яка змінює властивості компонентів **Tkinter**:

```
def change_properties():
   label.config(text="Текст змінено!", fg="blue", font=("Arial",
16))
    button.config(text="Кнопка змінена", bg="green", fg="white")
     entry.config(bg="lightyellow")

button = Button(window, text="Змінити властивості",
command=change_properties)
button.pack()
```

- Після натискання на кнопку текст мітки зміниться на **Текст змінено!**, текст стане синім, а шрифт — **Arial**, 16 пікселів.
- Текст на кнопці також зміниться на **Кнопка змінена**, фон кнопки стане зеленим, а текст на ній — білим.
- Поле вводу змінить колір фону на світло-жовтий.





Отримання даних з Entry

Щоб отримати числові дані з поля **Entry**, необхідно перетворити текст, введений користувачем, на числовий тип даних, наприклад, ціле число **(int)** або число з плаваючою комою **(float)**. Ось приклад функції для отримання числових даних з **Entry**:

```
def process ():
   # Отримання тексту з полів введення і перетворення його на
числа
    number1 = int(entry1.get())
    number2 = int(entry2.get())

    # Додавання чисел
    result = number1 + number2

    # Виведення результату в Label
    result_label.config(text=str(result))
```



Цей код є функцією для отримання числових даних з полів введення (**Entry**) в **Tkinter**, виконання їх додавання і виведення результату. Ось як він працює:

Отримання даних: функція **process()** використовує метод **.get()** для отримання тексту, введеного в поля **entry1** і **entry2**.

Перетворення: введені дані перетворюються на цілі числа за допомогою функції **int()**. Це означає, що користувач має вводити тільки цілі числа.

Обробка: числа додаються.

Виведення результату: результат перетворюється в рядок за допомогою **str()** і відображається у віджеті **Label**.

Як ви зрозуміли, для кожної кнопки буде описано функцію — частину задачі, яка викликатиметься натисканням на неї.

Обробка подій

Для найдопитливіших!

У **Tkinter** можна обробляти різні події, такі як натискання кнопок, введення тексту, пересування миші тощо. Для цього використовується метод **bind()**.

```
def on_key_press(event):
    print('Ви натиснули клавішу:',  event.char)


window.bind("<Key>", on_key_press)
```

on_key_press(event) — функція, що викликається під час натискання клавіші **event.char**, містить символ, відповідний натиснутій клавіші.

window.bind("<Key>", on_key_press) — прив'язує подію натискання клавіші (**<Key>**) до функції **on_key_press**.

```
def process(event):
    <код>

window.bind("<Return>", process)
```

Щоб програма виконувала обчислення при натисканні клавіші **Enter**, потрібно "зв'язати" подію натискання цієї клавіші з функцією обробки введених даних. Клавіша **Enter** позначається як **Return**.

Практична робота №22.

Завдання 1. Зміна кольору фону

Створити програму, яка буде змінювати колір фону вікна, коли натискатимуть на відповідну кнопку.

Програма повинна містити:

- Головне вікно розміром 400x300 пікселів із заголовком "Зміна кольору фону".
- Три кнопки, кожна з яких змінює колір фону вікна на певний колір, а також заголовок вікна.

Наприклад:

- Перша кнопка змінює фон вікна на червоний ("red") та заголовок вікна "Червоний".
- Друга кнопка змінює фон вікна на зелений ("green") та заголовок вікна "Зелений".
- Третя кнопка змінює фон вікна на синій ("blue") та заголовок вікна "Синій".

Використайте метод **configure()** для зміни кольору фону вікна.

Додатково: стилізувати кнопки, задавши їм індивідуальні значення властивостей, таких як розміри, колір фону, шрифт тощо.



Авторська частина коду, яка відповідає за функцію "зеленої" кнопки.

```
def change_to_green():
    window.configure(bg="green")
    window.title("Зелений")
```

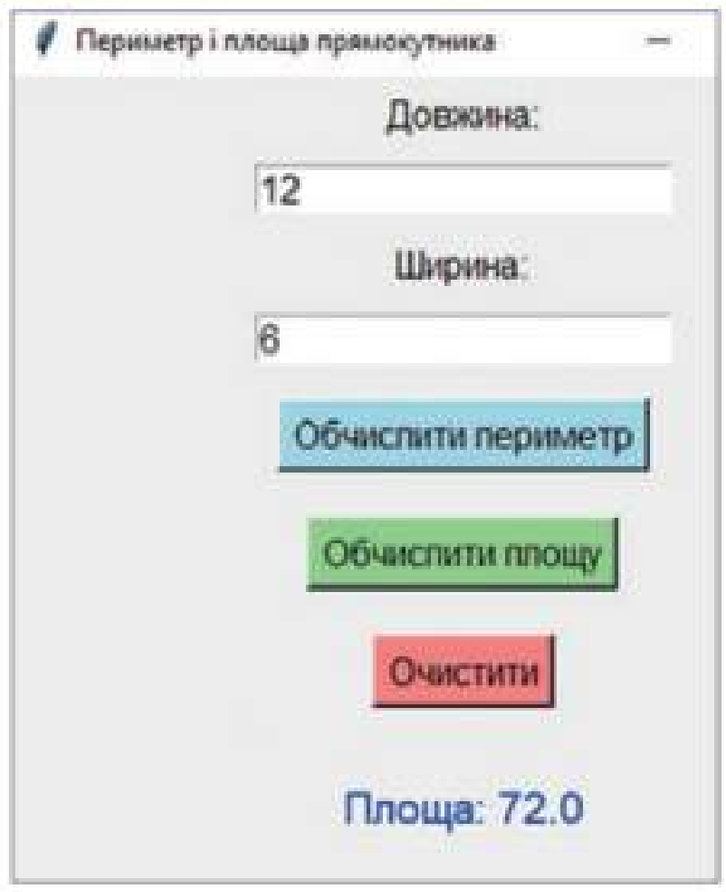
де, `change_to_green()` — функція, яка викликається натисканням на кнопку "Зелений".

Завдання 2. Знаходження площі та периметра прямокутника

Створіть програму з графічним інтерфейсом, яка буде обчислювати периметр та площу прямокутника.

Програма повинна містити:

- Головне вікно розміром 400x400 пікселів із заголовком "Периметр і площа прямокутника".
- Дві мітки (**Label**) та два текстові поля (**Entry**) для введення довжини та ширини прямокутника.



- Три кнопки:
- **Обчислити периметр** — виводить периметр прямокутника.
- **Обчислити площу** — виводить площу прямокутника.
- **Очистити** — очищає всі поля введення та результат.
- Текстовий елемент (**Label**) для відображення результатів обчислень.

Порядок дій та авторські частини коду:

- Імпортуйте бібліотеку **Tkinter**
- Створіть головне вікно програми з відповідними параметрами розміру та заголовком.
- Додайте два написи (**Label**) та два текстові поля (**Entry**) для введення довжини та ширини прямокутника.

```
# Поля введення для довжини і ширини
label_length = Label(window, text="Довжина:", font=("Arial", 12))
label_length.pack(pady=5)
entry_length = Entry(window, font=("Arial", 12))
entry_length.pack(pady=5)

label_width = Label(window, text="Ширина:", font=("Arial", 12))
label_width.pack(pady=5)
entry_width = Entry(window, font=("Arial", 12))
entry_width.pack(pady=5)
```

- Додайте кнопки для запуску відповідних функцій.

```
# Кнопка для обчислення периметра
button_perimeter = Button(window, text="Обчислити периметр",
                          command=perimeter,
                          bg="lightblue", font=("Arial", 12))
button_perimeter.pack(pady=10)

# Кнопка для обчислення площі
button_area = Button(window, text="Обчислити площу",
                     command=area, bg="lightgreen",
                     font=("Arial", 12))
button_area.pack(pady=10)

# Кнопка для очищення полів
button_clear = Button(window, text="Очистити",
                      command=clear, bg="lightcoral",
                      font=("Arial", 12))
button_clear.pack(pady=10)
```

- Додайте текстовий елемент (**Label**) для відображення результатів обчислень.

```
# Текстовий елемент для відображення результатів
result_label = Label(window, text="", font=("Arial", 14),
fg="blue")
result_label.pack(pady=20)
```

- Опишіть функції для обчислення периметра, площі та очищення полів введення.

```
# Функція для обчислення периметра
def perimeter():
    length = float(entry_length.get())
    width = float(entry_width.get())
    perimeter = 2 * (length + width)
            result_label.config(text=f"Периметр:
{perimeter}")

# Функція для обчислення площі
def area():
    length = float(entry_length.get())
    width = float(entry_width.get())
    area = length * width
    result_label.config(text=f"Площа: {area}")

# Функція для очищення полів
def clear():
    entry_length.delete(0, END)
    entry_width.delete(0, END)
    result_label.config(text="")
```

Обговорити частини коду в групах.

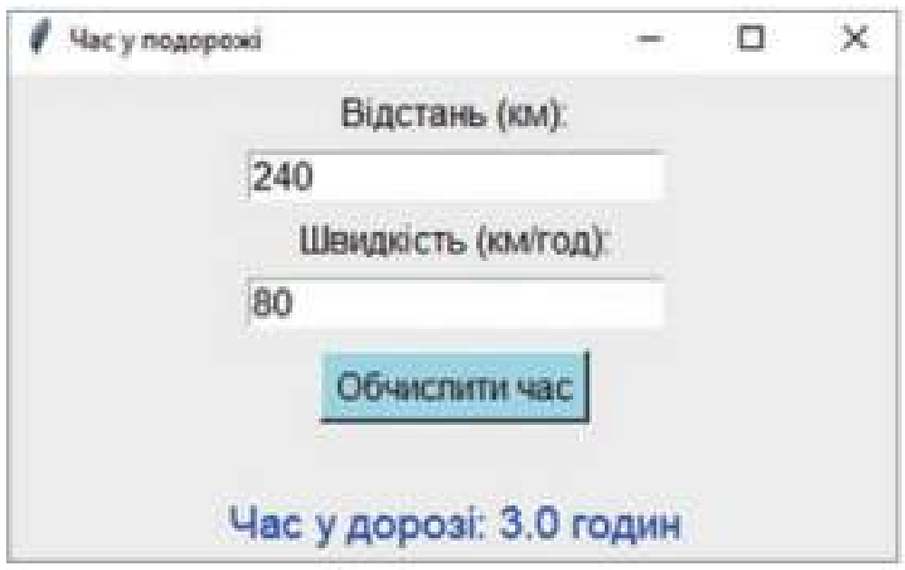
Завдання 3. Обчислення часу у подорожі

Створіть програму з графічним інтерфейсом, яка буде обчислювати час у дорозі на основі введених користувачем даних.

Програма повинна містити:

- Головне вікно розміром 400x250 пікселів із заголовком **Час у подорожі.**
- Дві мітки (**Label**) та два текстові поля (**Entry**) для введення відстані (в кілометрах) та швидкості (в км/год).
- Кнопку **Обчислити час** (**Button**), яка виводить час у дорозі в годинах.
- Текстовий елемент (**Label**) для відображення результатів обчислень.

Зразок вікна:



Завдання 4. Товар зі знижкою

Створіть програму з графічним інтерфейсом, яка буде знаходити, скільки коштуватиме певний товар в магазині, якщо діє знижка на нього. Значення ціни товару і відсоток знижки вводяться в окремих рядках користувачем з клавіатури.

Програма повинна містити:

- Головне вікно розміром 400x300 пікселів із заголовком **Акційний товар**.
- Три мітки (**Label**) та три текстові поля (**Entry**) для введення назви товару, ціни (в грн) та знижки (у %).
- Кнопку **Ціна зі знижкою**.
- Текстовий елемент (**Label**) для відображення результатів обчислень.

Завдання 5. Розрахунок вартості електроенергії

Створіть програму з графічним інтерфейсом для обчислення вартості спожитої електроенергії.

Програма повинна містити:

- Головне вікно розміром 450x300 пікселів із заголовком "Калькулятор електроенергії".
- Дві мітки (**Label**) та два текстові поля (**Entry**) для введення кількості спожитих кВт год та ціни за 1 кВт год (в гривнях).
- Кнопку **Обчислити вартість**, яка виводить загальну вартість електроенергії.
- Текстовий елемент (**Label**) для відображення результату.

Логічний тип даних. Операції відношень. Логічні оператори

Тема 31

Ключові поняття

Логічний тип даних
Операції відношень
Логічні оператори
Комбінування логічних операторів

Логічний тип даних



Логічний тип даних — один з основних типів у багатьох мовах програмування, включаючи **Python**.

Логічні дані часто використовуються для прийняття рішень в програмі, тобто для керування тим, які команди будуть виконуватися залежно від певних умов.

Логічний тип даних у **Python** представлений типом **bool** і має лише два можливі значення:

`False` — хиба

`True` — істина



Тобто логічний тип даних використовується для вираження істинності або хибності висловлювання.

- якщо змінна містить значення **True** (правдиве висловлювання) — то у консоль виведе **True**
- якщо змінна містить значення **False** (хибне висловлювання) — то у консоль виведе **False**

Приклад:

```
a = True
b = False
print(a)
print(b)
```

```
True
False
```

Перетворення типів у логічний

У **Python** можна перетворювати інші типи даних у логічний тип за допомогою функції **`bool()`**.

Будь-яке значення має свій логічний еквівалент:

- Порожні значення, такі як 0, **`None`**, **`[]`**, **`""`**, вважаються **`False`**.
- Всі інші значення вважаються **`True`**

За замовчуванням у числовому вигляді значення **`True`** відображається як 1, а **`False`** — як 0.

```
print(bool(0))          # Виведе: False
print(bool(42))         # Виведе: True
print(bool(""))         # Виведе: False
print(bool("Python"))   # Виведе: True
```

Операції відношень

Операції відношень використовуються для порівняння двох значень. Вони повертають логічне значення **`True`** або **`False`**, залежно від результату порівняння.

Основні операції відношень:

== — рівність (перевіряє, чи два значення рівні)
!= — нерівність (перевіряє, чи два значення нерівні)
> — більше
< — менше
>= — більше або дорівнює
<= — менше або дорівнює



Розглянемо кілька простих прикладів логічного порівняння у **Python**:

```
x = 10
y = 5
print(x == y) # False
print(x != y) # True
print(x > y) # True
print(x < y) # False
print(x >= y) # True
print(x <= y) # False
```

```
print(a + b == 15) # True
print(a * b == 40) # False
print(a % 3 == 0)  # False
print(a % b == 0)   # True
```

```
day = 'середа'

print(day == 'середа') # True
print(day == 'четвер') # False
```

Операції порівняння можна застосовувати не лише для чисел, а й, наприклад, для текстових значень.

Перше порівняння (**`day == 'середа'`**) повертає **`True`**, оскільки значення змінної **day** дійсно є `'середа'`.

Друге порівняння (**`day == 'четвер'`**) повертає **`False`**, оскільки значення **day** не збігається з `'четвер'`.

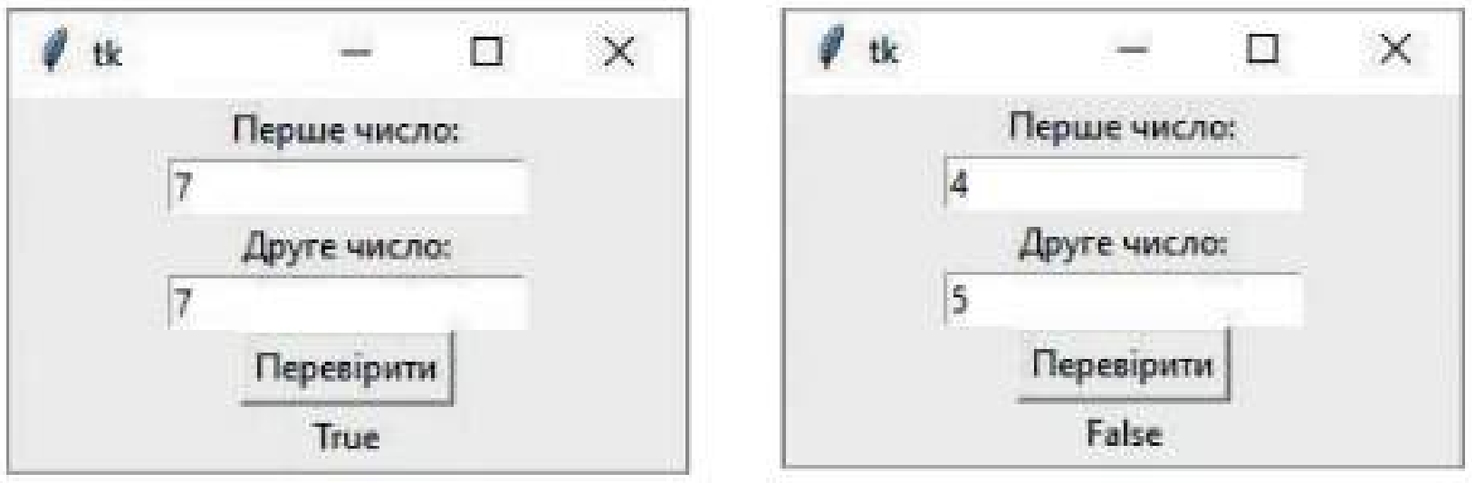
Із більш складнішими операціями над текстом та рядками ви зможете ознайомитися в курсі інформатики старших класів.

Функція для перевірки логічних операцій

Розглянемо приклад в **Tkinter**, в якому користувач вводить два числа, а програма повідомляє, рівні вони чи ні.

```
def equality():
    num1 = float(entry1.get())
    num2 = float(entry2.get())
    rezult = bool(num1 == num2)
    result_label.config(text= str(rezult))
```

Для реалізації цієї задачі потрібно створити у вікні **Tkinter** написи та поля для вводу користувачем двох значень. А також кнопку, яка викликатиме функцію **`equality()`**. Ця функція зчитує дані з полів, конвертує в числа, перевіряє на рівність та виводить результат у відповідний напис.



Логічні оператори

Логічні оператори дозволяють поєднати кілька умов і визначати їхню істинність або хибність.

Є три основні логічні оператори в **Python**:

`and` — логічне "І" — повертає **`True`**, якщо обидві умови істинні;

`or` — логічне "АБО" — повертає **`True`**, якщо хоча б одна умова істинна;

`not` — логічне "НЕ" — повертає протилежне значення: якщо вираз істинний, він стане хибним, і навпаки.



```
a = True
b - False

print(a and b)   # False (тільки обидва True дає True)
print(a or b)    # True (хоча б одна умова True)
print(not a)     # False (інвертація True)
```

```
a = 5
b = 10
c = 0

print(a > 0 and b > 0)  # Виведе: True (обидва числа додатні)
print(a > 0 and c > 0)  # Виведе: False (одне з чисел рівне нулю)
print(c > 0 and b > 0)  # Виведе: False (одне з чисел нуль)
```

```
x = -3
y = 7

print(x > 0 or y > 0)  # Виведе: True (одне число додатнє)
print(x > 0 or y < 0)  # Виведе: False (обидва числа не відповідають умові)
print(x < 0 or y > 0)  # Виведе: True (обидва числа відповідають умовам)
```

```
n = 15
m = 0

print(not n > 10)  # Виведе: False (n > 10 = True, тому not True
= False)
print(not m > 0)   # Виведе: True (m > 0 = False, тому not False
= True)
```

```
x = 100
y = 50
z = 0

print(x >= 100 and y <= 50)   # Виведе: True (обидві умови вико-
нуються)
print(z == 0 or y == 100)     # Виведе: True (z дорівнює 0, хоча
y не дорівнює 100)
print(not (x < 50))           # Виведе: True (оскільки x > 50,
not False = True)
```

Комбінування логічних операторів

Можна комбінувати кілька логічних операторів для побудови складних умов. При цьому **Python** виконує обчислення зліва направо і використовує пріоритет операторів:

- **not** має найвищий пріоритет
- потім йде **and**
- і останній — **or**



```
a = 8
b = -2
c = 12

print(a > 0 and b > 0 or c > 0)  # Виведе: True (оскільки c > 0)
print((a > 0 or b > 0) and c > 0)  # Виведе: True (оскільки a
> 0 і c > 0)
print(not (a < 0 or b > 0))  # Виведе: True (оскільки a < 0 або
b > 0 = False)
```

Логічні оператори в **Python** — це потужний інструмент для роботи з умовами і прийняття рішень. Вони дозволяють перевіряти кілька умов одночасно і будувати складні логічні вирази, що робить їх невід'ємною частиною програмування.

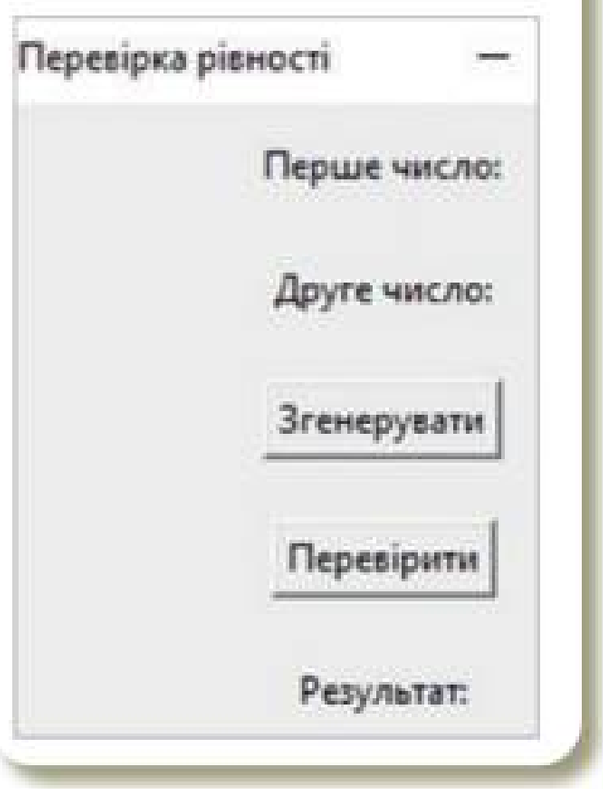
Практична робота №23.

Завдання 1. Перевірка на рівність двох випадкових чисел

Створіть програму, яка генерує два випадкові числа та перевіряє, чи вони рівні.

- імпортувати бібліотеки **tkinter** та **random**;
- створити вікно, яке буде містити дві мітки (**Label**) для інформації про два згенеровані числа, дві кнопки (**Button**), одна з яких генеруватиме два числа, а інша — перевірятиме їх на рівність, та мітку (**Label**), яка виводитиме логічну перевірку даних чисел (**True** або **False**);
- перша кнопка викликає функцію **generation()**, яка створює дві глобальні змінні, генерує в них два рандомні числа від 1 до 10 та виводить їх відповідно у дві перші мітки.

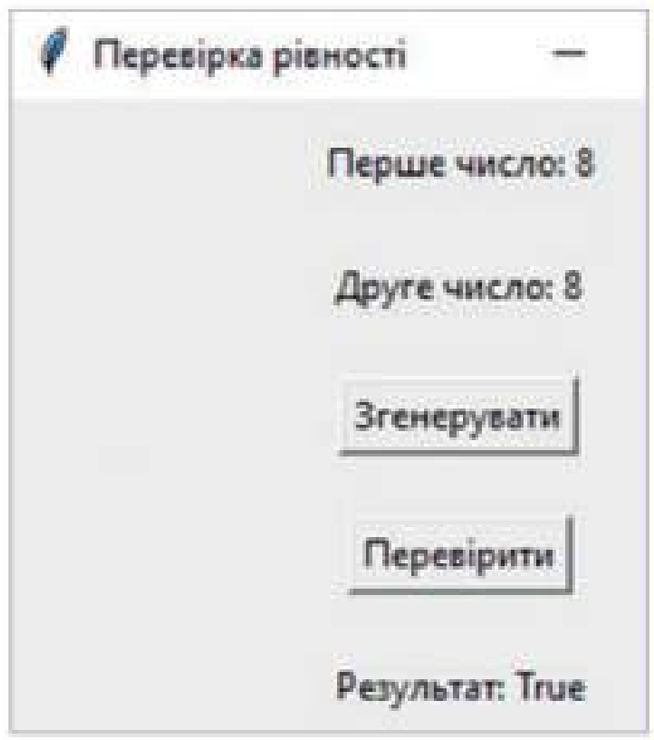
Зразок:




```
def generation():
    global num1
    global num2
    num1 = randint(1, 10)
    num2 = randint(1, 10)
    first_number.config(text=f"Перше число: {num1}")
    second_number.config(text=f"Друге число: {num2}")
```

- друга кнопка викликає функцію **check()**, яка перевіряє числа на рівність та виводить результат у третю мітку.

```
def check():
    result = bool(num1 == num2)
   result_label.config(text=f"Результат:
{result}")
```



Завдання 2. Перевірка, чи одне число більше за інше

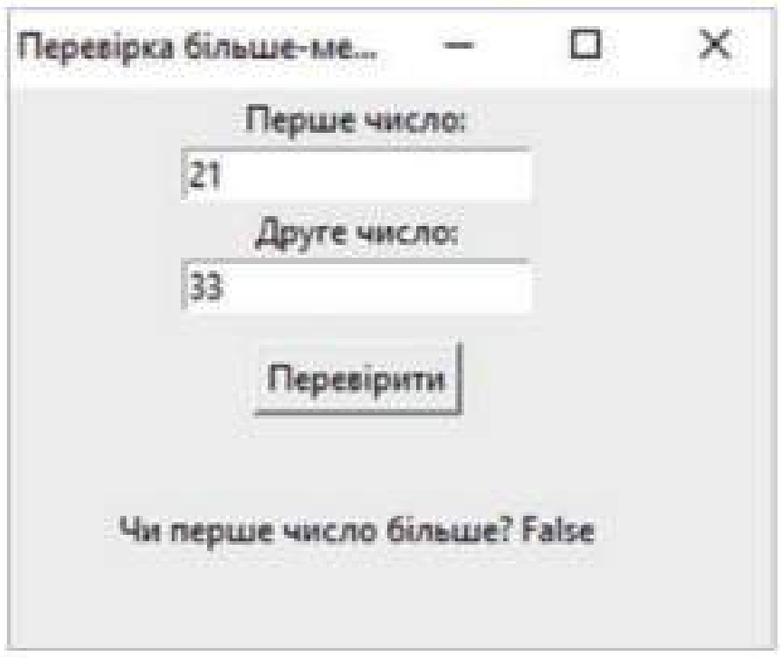
Створіть програму, яка запитує два числа і перевіряє, чи перше більше за друге.

- створити вікно, яке буде містити дві мітки (**Label**) та два поля (**Entry**) для введення двох чисел, кнопку (**Button**), яка перевірятиме умову задачі, та мітку (**Label**), яка виводитиме результат (**True** або **False**).

- кнопка викликає функцію `check()`, яка приймає два числа і перевіряє умову задачі

Частина коду функції:

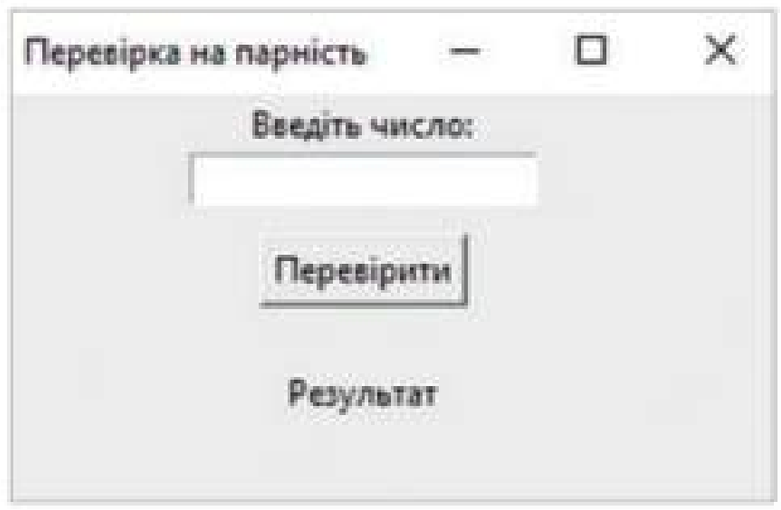
```
num1 = float(entry1.get())
num2 = float(entry2.get())
result = bool(num1 > num2)
```



Завдання 3. Перевірка на парність числа

Створіть програму, яка перевіряє, чи є введене число парним.

- створити вікно, яке буде містити мітку (**Label**) та поле (**Entry**), для введення цілого числа, кнопку (**Button**), яка перевірятиме умову задачі, та мітку (**Label**), яка виводитиме результат (**True** або **False**).
- кнопка викликає функцію **check()**



Перевірка на парність: `num % 2 == 0`

Завдання 4. Перевірка, чи є сума чисел більша за 100

Створіть програму, яка генерує два випадкові числа і перевіряє, чи їхня сума більша за 100.

Завдання 5. Перевірка на введення чисел одного знака (додатні чи від'ємні)

Створіть програму, яка перевіряє, чи введені два числа є одного знака (обидва додатні або обидва від'ємні).

Перевірка: `(num1 >= 0) == (num2 >= 0)`

(`правда = правда` — `True`, `хиба = хиба` — `True`)

Завдання 6. Перевірка існування трикутника

Створіть програму, яка дозволяє користувачеві ввести довжини трьох сторін і визначає, чи можуть ці сторони утворити трикутник (тобто чи виконується правило, що сума довжин будь-яких двох сторін має бути більшою за довжину третьої).

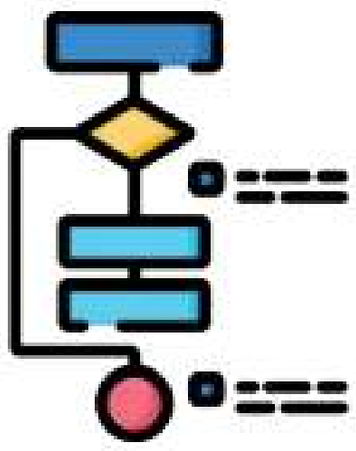
Перевірка: `(a + b > c) and (a + c > b) and (b + c > a)`, де `a`, `b`, `c` — довжини сторін трикутника.

Ключові поняття

Команда розгалуження
Оператор if
Оператор else
Конструкція elif
Віджет Checkbutton
Віджет Radiobutton

Команда розгалуження

У програмуванні часто буває потрібно, щоб програма виконувала різні дії залежно від того, чи виконується певна умова.



Для цього в **Python** використовується команда **розгалуження**. Вона дозволяє програмі приймати рішення на основі умов і виконувати різні блоки коду відповідно до результату перевірки.

Оператор if

Команда розгалуження — це інструкція, яка дозволяє програмі перевірити умову та залежно від її істинності або хибності виконати певний блок коду.

Синтаксис:

```
if умова:
    # код, який виконається, якщо умова істинна
```

Оператор **if** реалізує алгоритм неповного розгалуження. Він перевіряє, чи виконується певна умова. Якщо умова виконується, то виконується блок коду під ним. Якщо умова хибна, цей блок буде пропущено.

Приклад:

```
age = 18
if age >= 18:
    print("Ти дорослий")
```

У прикладі ми перевіряємо, чи вік **age** більший або дорівнює 18. Якщо це так, виведеться повідомлення "Ти дорослий". Якщо ні — нічого не відбудеться.

Оператор else

Команда **if-else** реалізує алгоритм повного розгалуження і використовується тоді, коли потрібно виконати одну дію, якщо умова істинна, або іншу дію, якщо вона хибна.

Синтаксис:

```
if умова:
    # код, який виконається, якщо умова істинна
else:
    # код, який виконається, якщо умова хибна
```

Приклад:

```
age = 15
if age >= 18:
    print("Ти дорослий")
else:
    print("Ти ще не дорослий")
```

Тобто оператор **else** відповідає за виконання коду, якщо умова в **if** не виконалась. Отже, програма виведе: "Ти ще не дорослий".

Операторів **if** в одній конструкції може бути кілька, проте оператор **else** — лише один раз.

Конструкція elif

Команда **elif** (скорочено від "else if" — інакше, якщо) дозволяє перевіряти додаткові умови, якщо попередня умова в **if** не виконалась.

Синтаксис:

```
if умова1:
    # код, який виконається, якщо умова1 істинна
elif умова2:
    # код, який виконається, якщо умова1 хибна, а умова2 істинна
else:
    # код, який виконається, якщо жодна з умов не є істинною
```

Приклад:

```
score = 6
if score >= 10:
    print("Відмінний результат")
elif grade >= 7:
    print("Добрий результат")
elif grade >= 4:
    print("Середній результат")
else:
    print("Низький результат")
```

Оператор **elif** можна використовувати кілька разів для додаткових умов (або не використовувати зовсім).

Вкладені розгалуження

Іноді потрібно перевірити нову умову всередині іншої умови. Для цього використовують вкладені розгалуження.

Вкладені розгалуження — це конструкції умов, де один оператор **if**, **elif** або **else** містить всередині себе інші оператори умов. Це дозволяє виконувати більш складну логіку, коли необхідно перевіряти додаткові умови всередині основної умови.

Синтаксис:

```
if умова1:
    # код, який виконується, якщо умова1 істинна
    if умова2:
        # код, який виконується, якщо умова2 істинна всередині умови1
    else:
      # код, який виконується, якщо умова2 хибна всередині умови1
else:
    # код, який виконується, якщо умова1 хибна
```

Приклад:

```
age = 20
has_id = True

if age >= 18:
    if has_id:
        print("Вам дозволено увійти.")
    else:
        print("Вам потрібно мати посвідчення.")
else:
    print("Вам ще не виповнилося 18.")
```

Зверніть увагу, як перевіряється друга умова — чи має користувач посвідчення (логічну перевірку **has_id == True** можна записати **has_id**).

Умови можуть бути складні

У **Python** можна поєднувати кілька умов за допомогою логічних операторів **and** (і), **or** (або), **not** (не).

Приклад:

```
temperature = 25
if temperature > 20 and temperature < 30:
    print("Приємна погода")
else:
    print("Погода не комфортна")
```

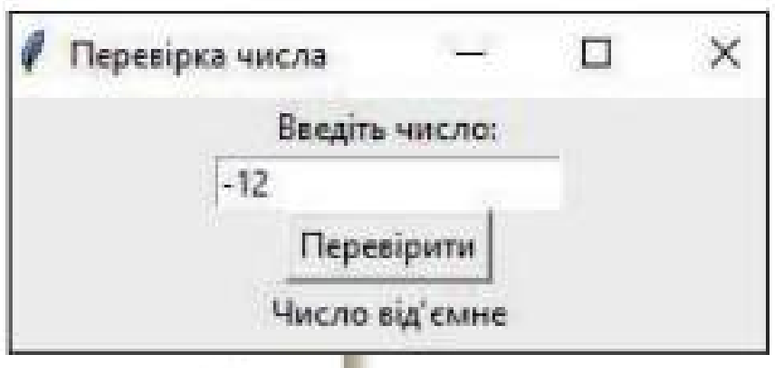
Програма перевіряє, чи температура перебуває в межах від 20 до 30 градусів. Якщо так, виводиться `"Приємна погода"`, в іншому випадку — `"Погода не комфортна"`.

Використовуючи подвійну нерівність, умову `temperature > 20 and temperature < 30` можна записати так: `20 < temperature < 30`.

Застосування розгалужень у Tkinter

Команди розгалуження в **Tkinter** можна використовувати для створення інтерактивних програм, де програма дозволить користувачу вводити числа, і на основі введених даних застосовувати умовні конструкції для різних сценаріїв.

Створимо інтерфейс, де користувач вводить число, а програма перевіряє, чи є це число додатним, від'ємним або нулем.




```
def check_number():
    num = float(entry.get())
    if num > 0:
        result_label.config(text="Число додатнє")
    elif num < 0:
        result_label.config(text="Число від'ємне")
    else:
        result_label.config(text="Число є нулем")
```

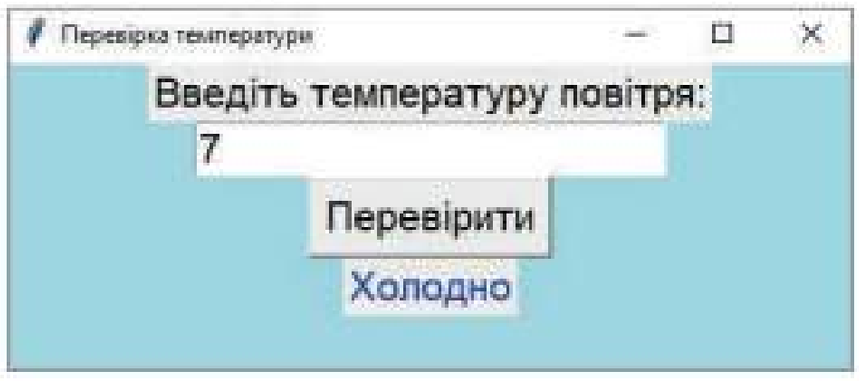
При натисканні на кнопку програма отримує введене користувачем значення з поля **Entry** і перетворює введене значення у число з плаваючою точкою - **float(entry.get())**.

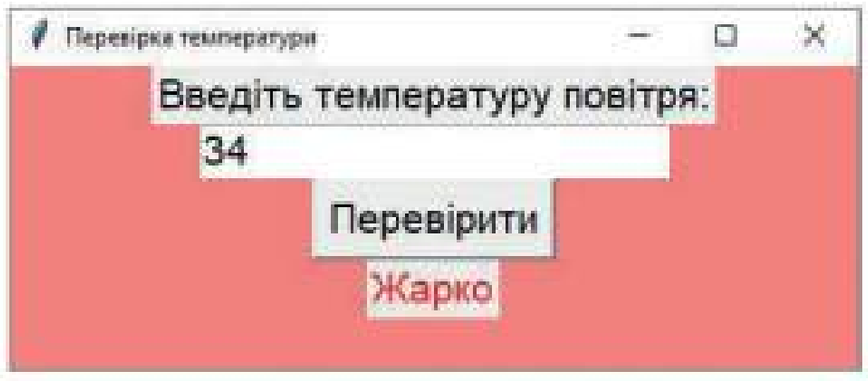
Після цього виконується конструкція **if-elif-else** для перевірки, чи є введене число додатнім, від'ємним або нулем. І насамкінець, виводить інформацію в мітку **result_label**.

Також, залежно від умови, можна змінювати зовнішній вигляд вікна та віджетів.

Розглянемо приклад, де користувач вводить температуру повітря, а програма повідомляє її стан та відповідно змінює стилі вікна та тексту.

```
def check_temperature():
  temperature = int(entry.get())  # Отримуємо значення з поля введення
  if temperature < 10:
    label_result.config(text="Холодно", fg="blue")
    window.config(bg="lightblue")  # Зміна кольору фону на "холодний"
  elif 10 <= temperature <= 25:
    label_result.config(text="Комфортно", fg="green")
   window.config(bg="lightgreen")  # Зміна кольору фону на "комфортний"
  else:
    label_result.config(text="Жарко", fg="red")
    window.config(bg="lightcoral")  # Зміна кольору фону на "гарячий"
```





Віджет Checkbutton

Віджет **Checkbutton** у бібліотеці **Tkinter** використовується для створення чекбоксів **(checkbox)**, які дозволяють користувачеві вибирати кілька опцій або перемикати стан між "вибрано" і "не вибрано" для ввімкнення або вимкнення певної опції.

Чекбокс може мати два стани:

Вибрано (**Checked**, стан **True**)

Не вибрано (**Unchecked**, стан **False**)

Основні особливості Checkbox

1. Може бути як одиночним, так і частиною групи.
2. Підтримує вибір кількох значень одночасно.
3. Його стан можна зв'язати зі змінною, щоб реагувати на зміни у виборі.

Синтаксис:

```
check_var = IntVar()
ch_btn = Checkbutton(parent, text="Текст",
                             variable=check_var,
                             onvalue=1, offvalue=0)
ch_btn.pack()
```

або




```
check_var = BooleanVar()
ch_btn = Checkbutton(parent, text="Текст",
                             variable=check_var,
                             onvalue=True, offvalue=False)
ch_btn.pack()
```

text — текст, який буде відображатися поруч із чекбоксом;

variable — це змінна, яка використовується для зв'язку між віджетом і значенням, яке цей віджет зберігає або контролює.

Ця змінна (**variable**) завжди має бути спеціальним типом змінних, таким як `IntVar()`, `DoubleVar()`, `STtrVar()` або `BooleanVar()`. Кожна зміна стану віджета (поставлена «галочка» або ні) автоматично змінює значення змінної.

За значення змінної **variable** відповідають такі властивості:

onvalue — значення змінної, коли чекбокс вибрано (для `IntVar` за замовчуванням 1).

offvalue — значення змінної, коли чекбокс не вибрано (для `IntVar` за замовчуванням 0).

Такі властивості не обов'язково прописувати, програма використовуватиме їхні значення за замовчуванням. Проте ці значення нам потрібні будуть для перевірки того, чи вибрано чекбокс, і при цьому виконувати різні функції.

Отримання стану Checkbox

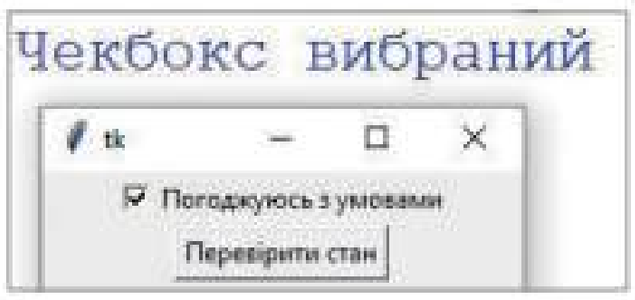
Розглянемо, як звернутися до змінної, що пов'язана з чекбоксом, і як за допомогою стану чекбокса виконувати різні дії.

```
is_checked = IntVar()

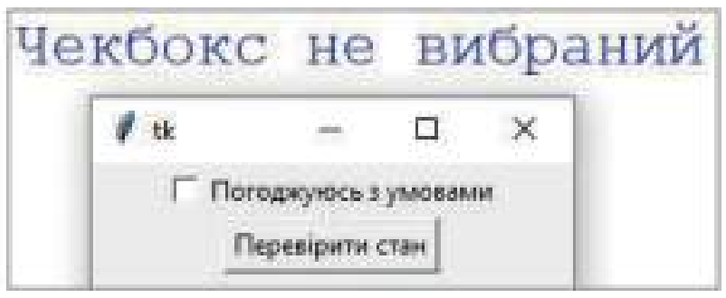
ch_btn = Checkbutton(window, text="Погоджуюсь з умовами",
                     variable=is_checked)
ch_btn.pack()
```

Спочатку створимо вікно з чекбоксом і кнопкою, яка викликатиме функцію **status()** для перевірки стану.

```
def status():
    if is_checked.get() == 1:
        print("Чекбокс вибраний")
    else:
        print("Чекбокс не вибраний")
```



Після цього опишемо цю функцію, а саме перевіримо, чи отримане значення змінної **is_checked.get()** дорівнює 1, тобто чи чекбокс вибрано, і в консолі виведемо повідомлення: `"Чекбокс вибрано"`. Якщо ж «галочка» не стоятиме — то виведемо повідомлення `"Чекбокс не вибрано"`.



Як ви уже зрозуміли, у функції можна описувати різні дії, які залежатимуть від того, поставлено «галочку» чекбоксу чи ні.

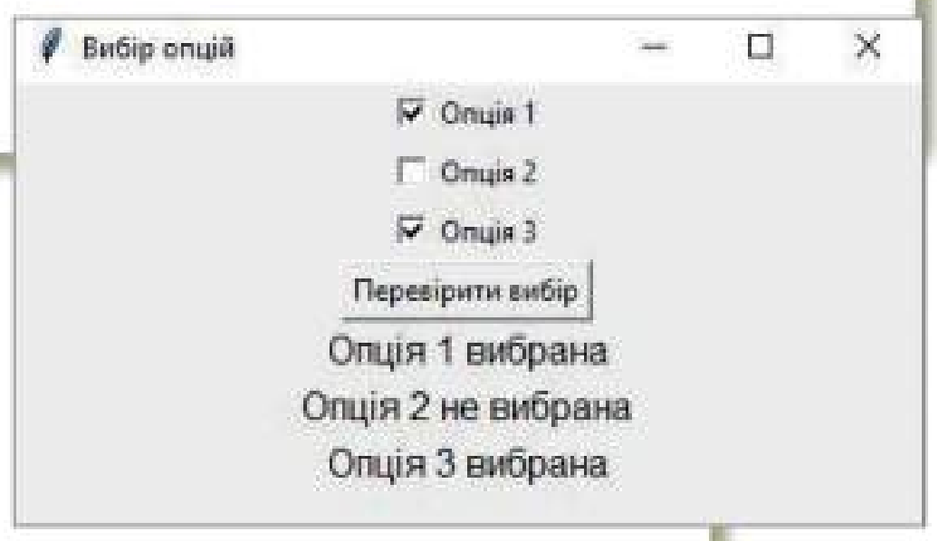
Використання декількох чекбоксів

Ви можете використовувати кілька чекбоксів (групу чекбоксів) у одному вікні. Кожен чекбокс матиме свою змінну для зберігання стану. Ось приклад, де створюються три чекбокси, і програма відслідковує їхній вибір.

```
# Створення змінних для кожного чекбокса
option1 = IntVar()
option2 = IntVar()
option3 = IntVar()

# Створення чекбоксів
check1 = Checkbutton(window, text="Опція 1", variable=option1)
check2 = Checkbutton(window, text="Опція 2", variable=option2)
check3 = Checkbutton(window, text="Опція 3", variable=option3)

check1.pack()
check2.pack()
check3.pack()
```



Кожен чекбокс повинен мати свою змінну, що дозволяє зберігати та відстежувати стан кожного з них.

```
def status():
    if option1.get() == 1:
        label_result1.config(text="Опція 1 вибрана")
    else:
        label_result1.config(text="Опція 1 не вибрана")

    if option2.get() == 1:
        label_result2.config(text="Опція 2 вибрана")
    else:
        label_result2.config(text="Опція 2 не вибрана")
```

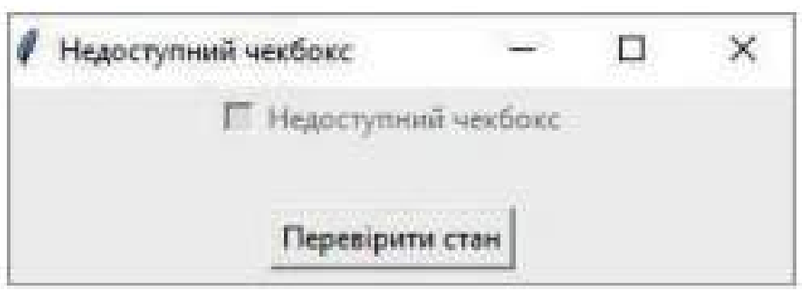
У функції, в свою чергу, повинна бути перевірка стану кожного з чекбоксів.

Стан недоступного чекбокса (disabled)

Іноді потрібно тимчасово зробити чекбокс недоступним для користувача (наприклад, коли слід заблокувати взаємодію). Це можна зробити за допомогою параметра **state**.

```
checkbutton.config(state=DISABLED)
```

Коли чекбокс перебуває в такому стані, він стає сірим, тому на нього не можна натискати.



Якщо вам потрібно зробити чекбокс знову доступним, ви можете змінити його стан за допомогою методу **`check.config(state='normal')`**.

Стилізація Checkbox

Аналогічно до раніше згаданих віджетів, в чекбоксах також можна змінювати колір фону, тексту, а також інші атрибути за допомогою певних параметрів:

`bg` — колір фону;
`fg` — колір тексту;
`font` — шрифт.




```
check = tk.Checkbutton(root, text="Опція",
                                    variable=check_var,
bg="lightblue", fg="red",
                           font=("Arial", 14))
check.pack()
```

Використання з іншими віджетами

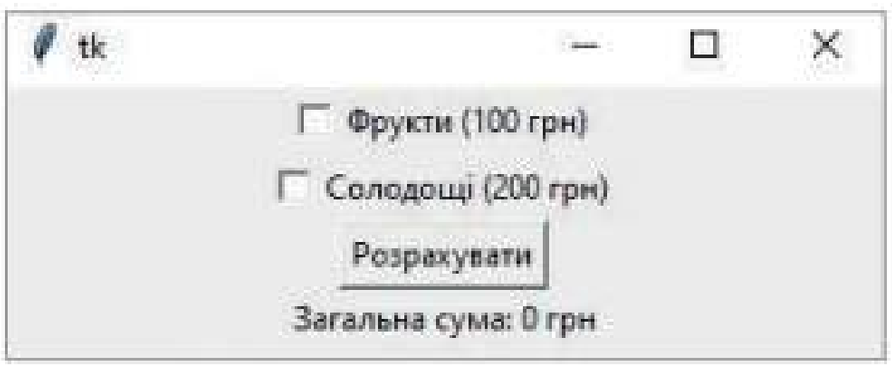
Чекбокси часто використовуються разом з іншими елементами інтерфейсу, такими як кнопки, текстові поля, списки, щоб реалізувати більш складні інтерфейси.

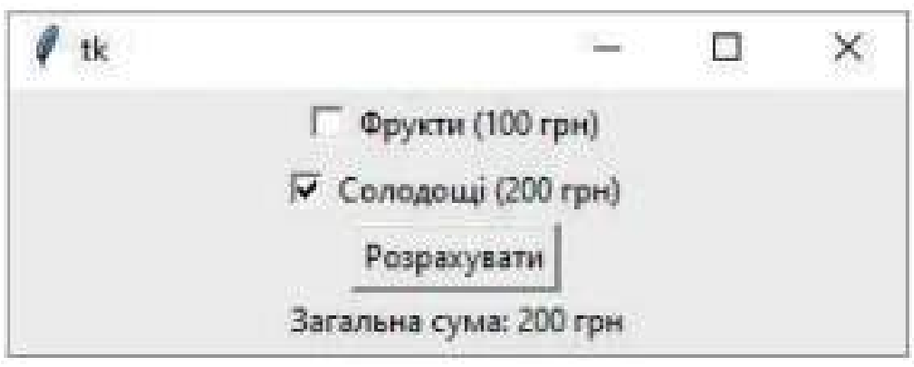
```
def calculate():
    suma = 0
    if check1_var.get() == 1:
        suma = suma + 100  # Додаємо до загальної суми
    if check2_var.get() == 1:
        suma = suma + 200
    result_label.config(text=f"Загальна сума: {suma} грн")
```

Цей код створює простий графічний інтерфейс, який дозволяє користувачеві вибирати певні опції (наприклад, покупку фруктів або солодощів), а потім підраховує загальну суму на основі вибору.

Функція **`calculate()`** працює таким чином, що перевіряє, чи вибрано чекбокси (значення змінних **`check1_var і check2_var`**). Якщо перший чекбокс вибрано, до суми додається 100 грн. Якщо другий — додається 200 грн.

Після цього сума відображається в мітці **`result_label`** за допомогою f-рядків.





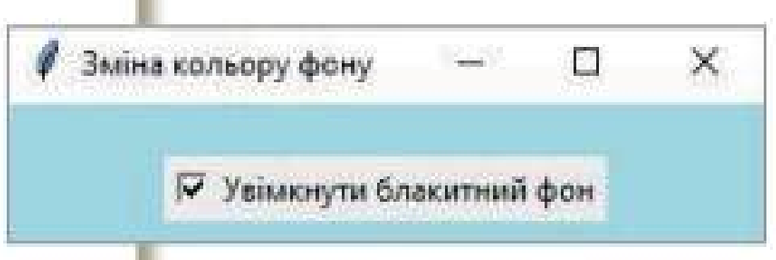
Використання функції command для обробки подій

Параметр **command** дозволяє вказати функцію, яка викликається при зміні стану чекбокса, аналогічно до кнопки. Це зручно для моментальної обробки вибору користувача без необхідності натискати окрему кнопку.

```
color_var = IntVar()

def change_background():
    if color_var.get() == 1:
        window.config(bg="lightblue")
    else:
        window.config(bg="white")

ch_btn = Checkbutton(window,
            text="Увімкнути блакитний фон",
            variable=color_var,
            command=change_background)
ch_btn.pack(pady=20)
```





У цьому прикладі створюється чекбокс **ch_btn** з текстом "Увімкнути блакитний фон", який:

- зв'язаний зі змінною **color_var**, що контролює його стан;
- при зміні стану викликає функцію **change_background()** для зміни кольору фону вікна;
- **pack(pady=20)** додає відступи по вертикалі для розміщення чекбокса.

Віджет Radiobutton

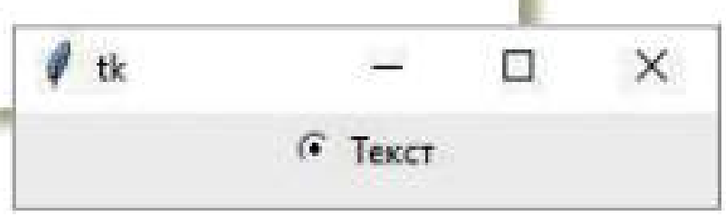
Віджет **Radiobutton** в **Tkinter** використовується для створення радіокнопок (**radio buttons**), які дозволяють користувачеві вибрати одну з кількох опцій. Кожна радіокнопка представляє певну опцію, і коли одну з них вибирають, інші стають недоступними для вибору (вибір одного виключає інші). Це відрізняє **Radiobutton** від **Checkbutton**, де можна вибрати кілька варіантів одночасно.

Синтаксис:

```
radio_var = IntVar()

radio_btn = Radiobutton(window, text="Текст",
            variable=radio_var, value=1)

radio_btn.pack()
```



variable — це змінна, яка використовується для зв'язку між віджетом і значенням, яке цей віджет зберігає або контролює.

value — значення, яке буде присвоєно змінній, коли цю радіокнопку вибрано.

Щоб зберігати вибір, зроблений за допомогою радіокнопок, використовується змінна типу **IntVar** або **StringVar**. Зазвичай для першої радіокнопки надають значення змінної 1, для другої — 2, і так далі.

Отримання стану Radiobutton

Аналогічно як із **CheckButton** можна звернутися до змінної, що пов'язана з радіокнопкою, і викликати функції.

```
radio_var = IntVar()

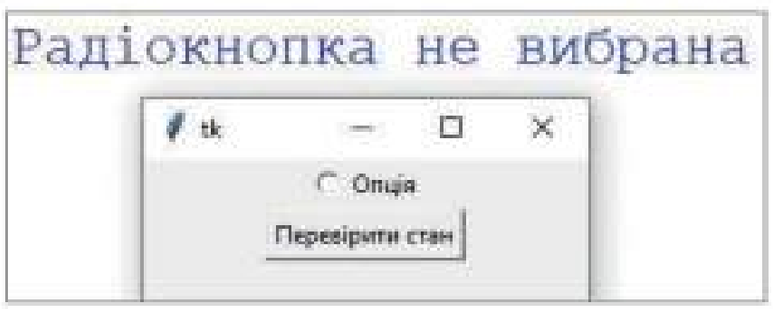
radio_btn = Radiobutton(window, text="Опція",
                        variable=radio_var, value=1)

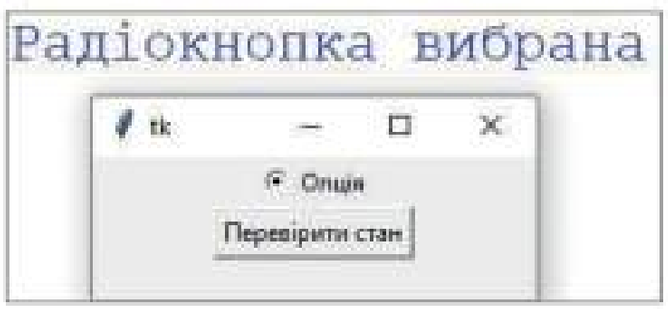
radio_btn.pack()
```

Спочатку створимо вікно з радіокнопкою і кнопкою, яка викликатиме функцію **status()** для перевірки стану.

```
def status():
    if radio_var.get() == 1:
        print("Радіокнопка вибрана")
    else:
        print("Радіокнопка не вибрана")
```

Після цього опишемо пропоновану функцію, а саме якщо оберемо цю радіокнопку, то значення змінної **radio_var** дорівнюватиме 1 і в консоль буде виведено відповідний текст `"Радіокнопка вибрана"`.





Проте, як ви вже помітили, коли в нас є лише одна радіокнопка, то після її вибору вже неможливо змінити її стан, аналогічно до чекбоксу. Тому доцільно працювати з кількома радіокнопками одночасно.

Організація групи Radiobutton

Для того щоб радіокнопки працювали як група (тобто дозволяли вибір лише однієї кнопки одночасно), їх потрібно прив'язати до однієї змінної **(variable)**.

```
Створення змінної для збереження вибору
choice = IntVar()

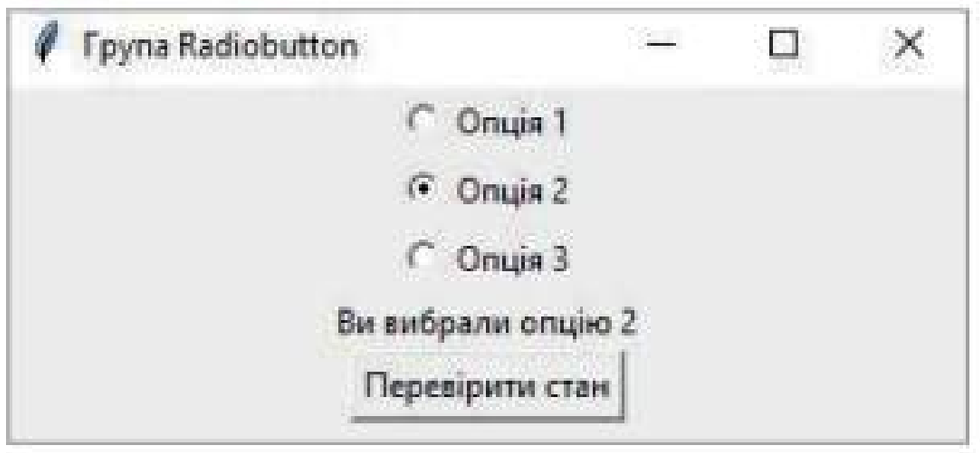
# Створення кількох Radiobutton, кожен із яких пов'язаний зі
змінною choice
radio1 = Radiobutton(window, text="Опція 1", variable=choice,
value=1)
radio2 = Radiobutton(window, text="Опція 2", variable=choice,
value=2)
radio3 = Radiobutton(window, text="Опція 3", variable=choice,
value=3)

radio1.pack()
radio2.pack()
radio3.pack()
```

Кожна кнопка **Radiobutton** має параметри:

- **variable=choice**: всі кнопки пов'язані зі змінною **choice**, типу **IntVar()**, яка буде зберігати вибране значення від однієї з кнопок. Значення змінної змінюватиметься на відповідне значення **value** тієї кнопки, яку обрав користувач;
- **value=1, 2, 3**: кожна кнопка має своє унікальне значення, яке присвоюється змінній `choice` при натисканні;
- **command=status**: кожного разу, коли користувач вибирає кнопку, викликається функція **status**, що показує вибране значення.

```
def status():
    # Отримуємо вибране значення зі змінної choice
    selected_option = choice.get()
   result_label.config(text=f"Ви вибрали опцію {selected_option}")
```



У функції, в свою чергу, спочатку отримаємо значення, яке зберігається у змінній **choice**.

```
selected_option = choice.get()
```

Ця змінна типу **IntVar** пов'язана з групою **Radiobutton** і містить значення тієї кнопки, яку обрав користувач. Наприклад, якщо користувач обрав кнопку з `value=2`, то `choice.get()` дорівнюватиме 2.

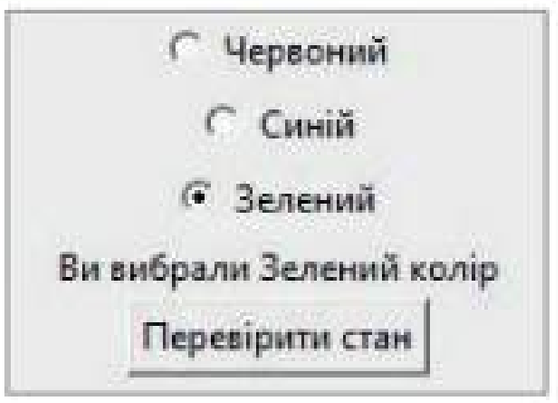
І виведемо інформацію в мітку за допомогою f-рядків.

Використання StringVar замість IntVar

Якщо ви хочете використовувати текстові значення для **Radiobutton**, можна обрати змінну типу **StringVar**.

```
choice = StringVar()

radio1 = Radiobutton(window, text="Червоний", variable=choice,
value="Червоний")
radio2 = Radiobutton(window, text="Синій", variable=choice,
value="Синій")
radio3 = Radiobutton(window, text="Зелений", variable=choice,
value="Зелений")
```



У цьому прикладі ми використовуємо **StringVar** для збереження текстового значення вибраного кольору. Текстове значення (наприклад, "Червоний", "Зелений", "Синій") зберігається в змінну **choice**. Кожного разу, коли користувач вибирає радіокнопку, текстовий рядок відображається у **Label**.

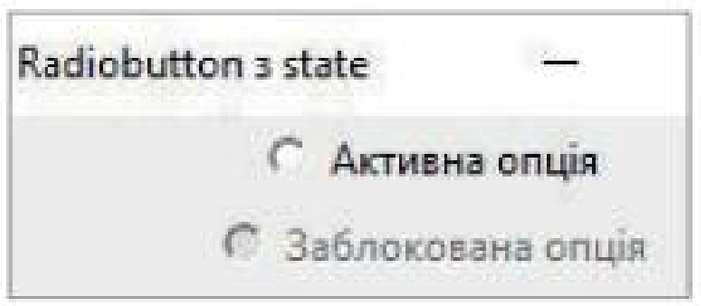
Вибір радіокнопки програмно: за допомогою методу **set()** можна програмно встановити значення змінної, що автоматично вибере відповідну радіокнопку.

choice.set(2) або **choice.set("Зелений")**

Стан недоступної кнопки (disabled)

Параметр **state** дозволяє контролювати, чи радіокнопка доступна для вибору. Якщо встановити **state="disabled"**, радіокнопка стане недоступною для взаємодії. Як і для чекбоксів, цей параметр можна як встановлювати в самому компоненті, так і змінювати її значення у функції.

Якщо вам потрібно зробити кнопку активною, ви можете змінити його стан за допомогою методу **check.config(state='normal')**.



Стилізація Radiobutton

Tkinter дозволяє налаштовувати зовнішній вигляд **Radiobutton**, змінюючи шрифт, колір тексту та фону.

```
radio_btn = Radiobutton(window, text="Варіант 1",
                        variable=selected_option,
                        value=1, font=("Arial", 12),
                        fg="blue", bg="lightblue")
radio_btn.pack()
```



Використання функції command для обробки подій

Параметр **command** дозволяє викликати функцію щоразу, коли вибирають певну радіокнопку. Це зручно, якщо потрібно виконати дію безпосередньо під час вибору.

```
radio1 = Radiobutton(window, text="Червоний", variable=choice,
                     value=1, command=change_background)
radio2 = Radiobutton(window, text="Зелений", variable=choice,
                     value=2, command=change_background)
radio3 = Radiobutton(window, text="Синій", variable=choice,
                     value=3, command=change_background)
```

У цьому прикладі ми присвоїмо кожному кольору певне числове значення і використовуватимемо **if-elif-else** для визначення кольору на основі вибраного числа.




```
def change_background():
    # Отримуємо числове значення, яке відповідає вибраному кольору
    selected_option = choice.get()

    # Змінюємо колір фону вікна залежно від значення
    if selected_option == 1:
        window.config(bg="red")
        result_label.config(text="Ви вибрали: Червоний")
    elif selected_option == 2:
        window.config(bg="green")
        result_label.config(text="Ви вибрали: Зелений")
    elif selected_option == 3:
        window.config(bg="blue")
        result_label.config(text="Ви вибрали: Синій")
```

Кілька груп Radiobutton

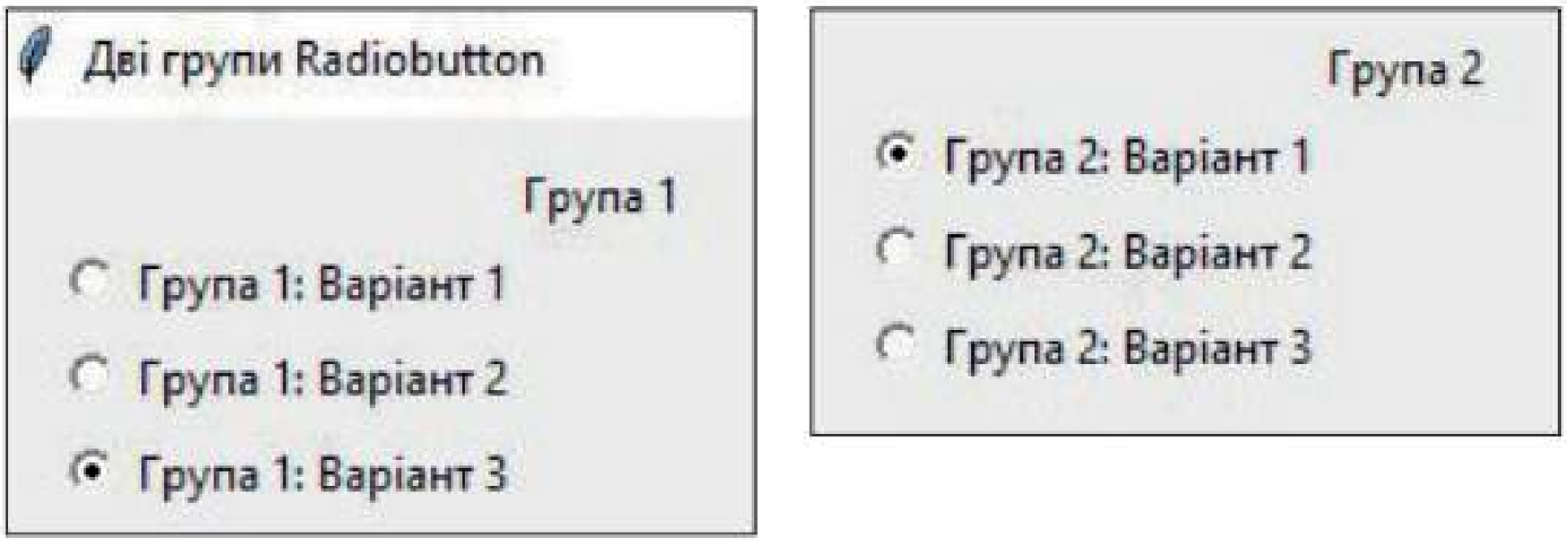
Якщо в програмі потрібно використовувати кілька груп радіокнопок, кожну групу потрібно прив'язати до окремої змінної. При цьому кожна група взаємодіятиме окремо одна від одної.

```
# Змінні для кожної групи
group1_var = IntVar()
group2_var = IntVar()
```

```
# Перша група радіокнопок
radiobutton1 = Radiobutton(window, text="Група 1: Варіант 1",
                           variable=group1_var, value=1)
radiobutton2 = Radiobutton(window, text="Група 1: Варіант 2",
                           variable=group1_var, value=2)
radiobutton3 = Radiobutton(window, text="Група 1: Варіант 3",
                           variable=group1_var, value=3)


# Друга група радіокнопок
radiobutton4 = Radiobutton(window, text="Група 2: Варіант 1",
                           variable=group2_var, value=1)
radiobutton5 = Radiobutton(window, text="Група 2: Варіант 2",
                           variable=group2_var, value=2)
radiobutton6 = Radiobutton(window, text="Група 2: Варіант 3",
                           variable=group2_var, value=3)
```

Зверніть увагу, що саме у параметрі **variable** обираємо, до якої групи належатиме та чи інша радіокнопка.



Зміна властивостей за допомогою груп радіокнопок

Розглянемо приклад, в якому буде створено вікно з двома групами **Radiobutton**. Перша група змінюватиме колір фону вікна, а друга — текст заголовка вікна.

```
# Змінні для кожної групи
bg_color_var = InVar()
title_var = InVar()
```

Створимо дві змінні, які відповідатимуть за кожну з груп радіокнопок: групу для зміни кольору фону та групу для зміни заголовка вікна.

```
# Заголовок для першої групи
label1 = Label(window, text="Виберіть колір фону")
label1.pack(pady=(10, 0))

# Перша група радіокнопок для вибору кольору фону
radiobutton1 = Radiobutton(window, text="Блакитний",
                           variable=bg_color_var, value=1,
command=change_bg_color)
radiobutton2 = Radiobutton(window, text="Зелений",
                           variable=bg_color_var, value=2,
command=change_bg_color)
radiobutton3 = Radiobutton(window, text="Рожевий",
                           variable=bg_color_var, value=3,
command=change_bg_color)

radiobutton1.pack(anchor="w", padx=20)
radiobutton2.pack(anchor="w", padx=20)
radiobutton3.pack(anchor="w", padx=20)
```



Після цього створимо першу групу кнопок, прив'язавши їх до першої змінної, та розташуємо у вікні.

Також додамо параметр **command**, який викликатиме функцію **change_bg_color()**.

Аналогічно створимо другу групу кнопок, яка викликатиме функцію **change_title()**, і прив'яжемо їх до другої змінної.

```
# Заголовок для другої групи
label2 = Label(window, text="Виберіть заголовок вікна")
label2.pack(pady=(20, 0))

# Друга група радіокнопок для зміни заголовка вікна
radiobutton4 = Radiobutton(window, text="Режим 1",
                           variable=title_var, value=1,
command=change_title)
radiobutton5 = Radiobutton(window, text="Режим 2",
                           variable=title_var, value=2,
command=change_title)
radiobutton6 = Radiobutton(window, text="Режим 3",
                           variable=title_var, value=3,
command=change_title)
```

І на завершення опишемо дві функції, які викликатимуть радіокнопки цих двох груп.

```
# Функція для зміни кольору фону
def change_bg_color():
    color = bg_color_var.get()
    if color == 1:
      window.config(bg="lightblue")
    elif color == 2:
     window.config(bg="lightgreen")
    elif color == 3:
      window.config(bg="lightpink")
```

```
# Функція для зміни заголовка вікна
def change_title():
    title = title_var.get()
    if title == 1:
       window.title("Tkinter: Режим 1")
    elif title == 2:
       window.title("Tkinter: Режим 2")
    elif title == 3:
       window.title("Tkinter: Режим 3")
```

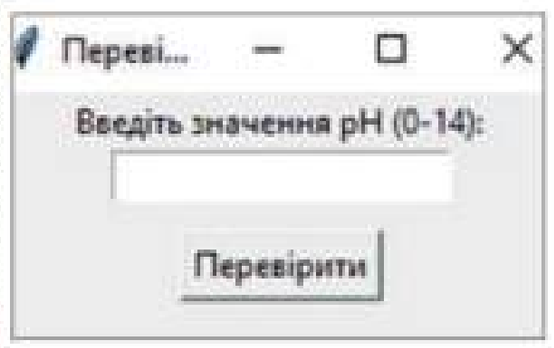
Практична робота №24.

Завдання 1. Перевірка кислотності

Створіть програму, яка дозволяє користувачеві ввести значення `pH` розчину, а на основі цього значення визначає, чи є розчин кислим `(pH < 7)`, нейтральним `(pH = 7)` або лужним `(pH > 7)`.

- створити вікно, яке буде містити мітку (**Label**) та поле (**Entry**) для введення кислотності, кнопку (**Button**) та мітку (**Label**), яка виводитиме результат;
- кнопка викликає функцію **`check_ph()`**, яка отримує дійсне число з поля **Entry**, перевіряє його на умову та виводить результат;
- додатково: залежно від результату, змінювати колір фону вікна.

Зразок:




```
def check_ph():
    ph = float(entry.get())
    if ph < 7:
        result_label.config(text="Кислий")
    if ph == 7:
        result_label.config(text="Нейтральний")
    if ph > 7:
        result_label.config(text="Лужний")
```

Завдання 2. Розрахунок покупки

Створіть програму, яка дозволяє покупцеві ввести назву товару, його ціну та суму грошей, які він має. Програма повинна розрахувати, чи вистачить покупцеві коштів для придбання товару. Якщо коштів достатньо, виводиться повідомлення про успішну покупку та залишок решти. Якщо грошей недостатньо, виводиться повідомлення про необхідну суму для доплати.

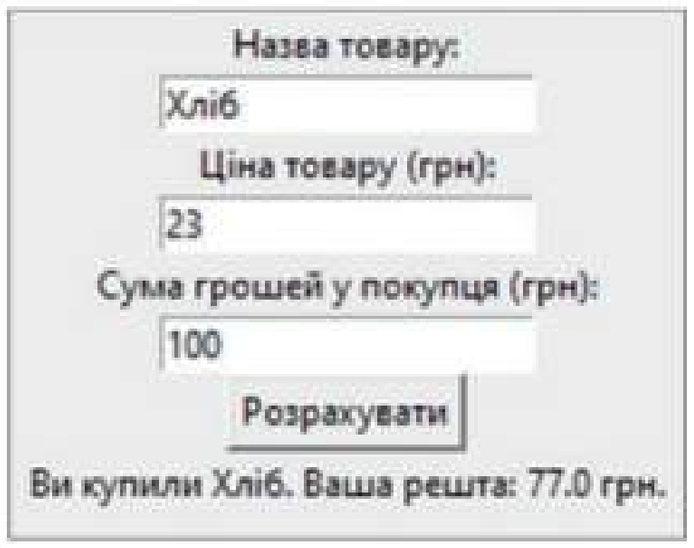
- створити вікно за зразком;
- кнопка **Розрахувати** викликає функцію **`calculate()`**, яка отримує назву товару та два дійсні числа, які відповідають за ціну товару та суму грошей. Якщо у покупця достатньо коштів, то виводити решту, якщо коштів недостатньо — повідомити, скільки ще потрібно покупцю додати грошей.

Зразок:



Частина коду умови:

```
if customer_money >= price:
        change = customer_money - price
            result_label.config(text=f"-
Ви купили {product_name}. Ваша решта:
{round(change, 2} грн.")
```



Завдання 3. Визначення пори року за місяцем

Користувач вводить номер місяця (від 1 до 12). Програма виводить назву пори року.

Програма повинна містити:

- Головне вікно із заголовком "Визначення пори року".
- Мітку (**Label**) та текстове поле (**Entry**) для введення номеру місяця.
- Кнопку **Визначити** (**Button**).
- Текстовий елемент (**Label**) для відображення результатів з такими властивостями `text="", font=("Arial", 14)`.
- Текстовий елемент (**Label**) для показу зображення (перед виконанням програми знайдіть та збережіть 4 зображення з форматом **png** або **gif** і назвіть їх відповідно **spring**, **summer**, **autumn**, **winter**.

```
image_label = Label(window)
image_label.pack(pady=20)
```

кнопка **Визначити** викликає функцію **season()**, яка отримує ціле число з поля **Entry**. Залежно від результату перевіряє, до якої пори року належить місяць під цим номером, виводить результат, змінює колір тексту пори року та додає зображення, яке відображатиме відповідну пору року.

Частина коду функції:

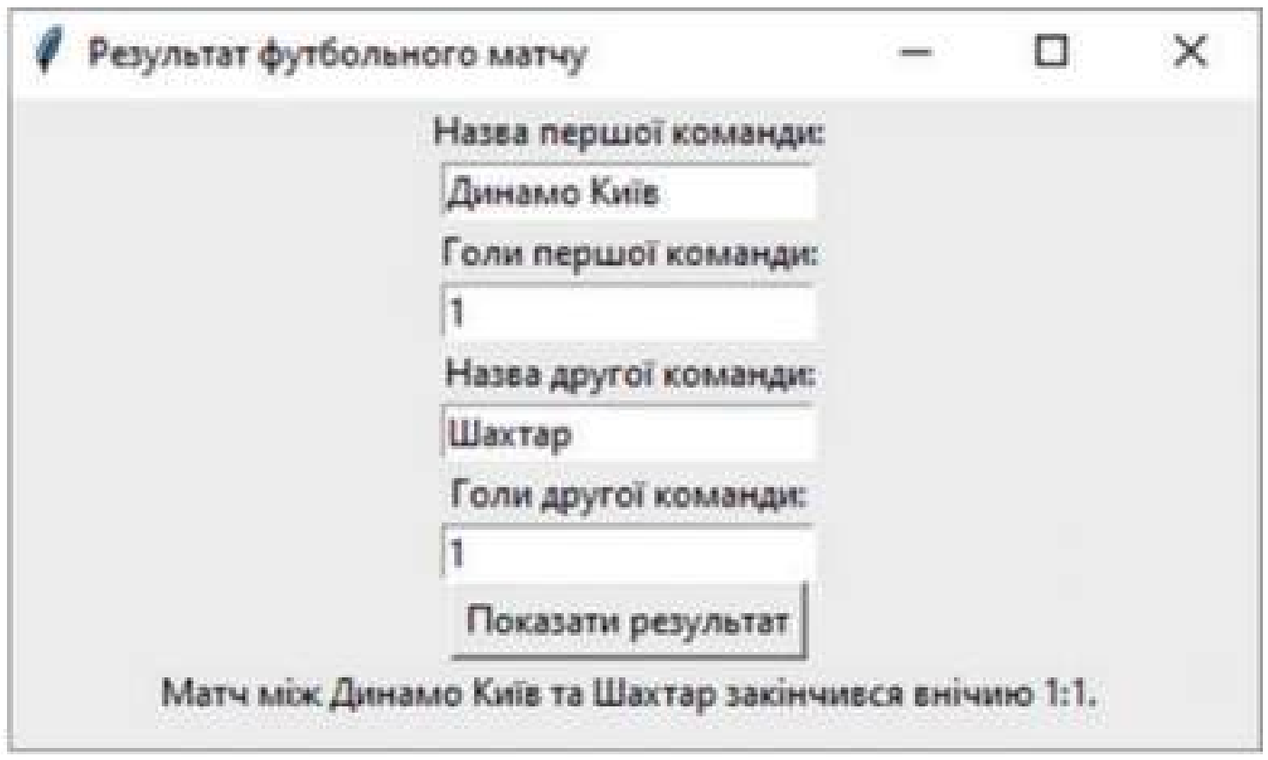
```
... (попередні умови)
elif 1 <= month <= 2 or month == 12:
        result_label.config(text=f"Пора
року: Зима", fg="blue")
        image = PhotoImage(file="winter.
gif")
      image = image.subsample(4, 4)
      image_label.config(image=image)
      image_label.image = image
 else:
      result_label.config(text="Некорек-
тно введений номер місяця")
```



Завдання 4. Визначення переможця футбольного матчу

Користувач вводить назви двох футбольних команд і кількість забитих ними голів. Програма визначає, яка команда виграла, або оголошує, що матч закінчився внічию.

Зразок:



Додатково: Стилізувати вікно та компоненти, змінюючи розмір тексту, колір тощо. (Для розміщення компонентів використовуйте фрейми для об'єднання окремих компонентів із змогою розміщати в рядок або використовуйте метод `place()`).

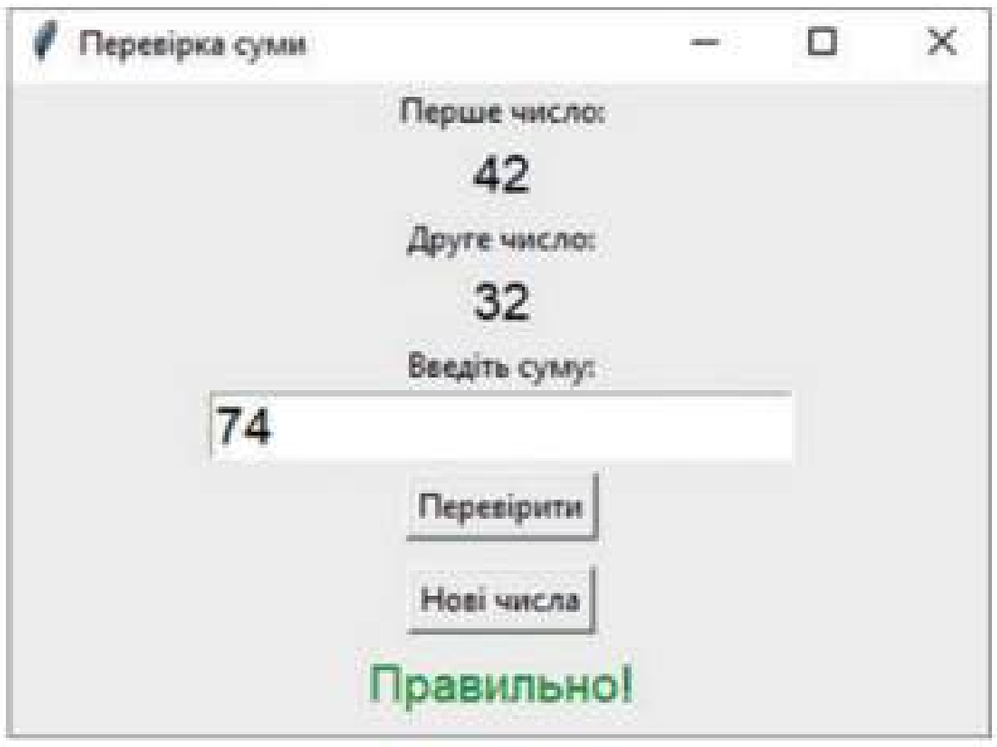
Завдання 5. Перевірка суми двох випадкових чисел

Створіть програму, яка:

1. Генерує два випадкові числа.
2. Виводить ці числа у вікно графічного інтерфейсу.
3. Дозволяє користувачеві ввести суму цих чисел у текстове поле.
4. Перевіряє правильність введеної суми та виводить результат:

- Якщо сума правильна, виводиться повідомлення **Правильно!** зеленим кольором.
- Якщо сума неправильна, виводиться повідомлення **Неправильно!** червоним кольором із правильною відповіддю.

Зразок вікна:



Кнопка **Нові числа** викликає функцію **generate_numbers()**, яка генерує числа, записує їх у відповідні мітки, очищає поле **Entry** та мітку результатів:

```
def generate_numbers():
# Генерація випадкових чисел
    num1 = randint(1, 100)
    ...рядок коду для другого числа

# Встановлення чисел у мітки
    num1_label.config(text=str(num1))
    ...рядок коду для другого числа
    # Очищення поля введення та результату
    answer_entry.delete(0, END)
    result_label.config(text="")
```

Кнопка **Перевірити** викликає функцію **check_sum()**, яка отримує два рандомні числа, відповідь користувача, перевіряє суму та виводить результат:

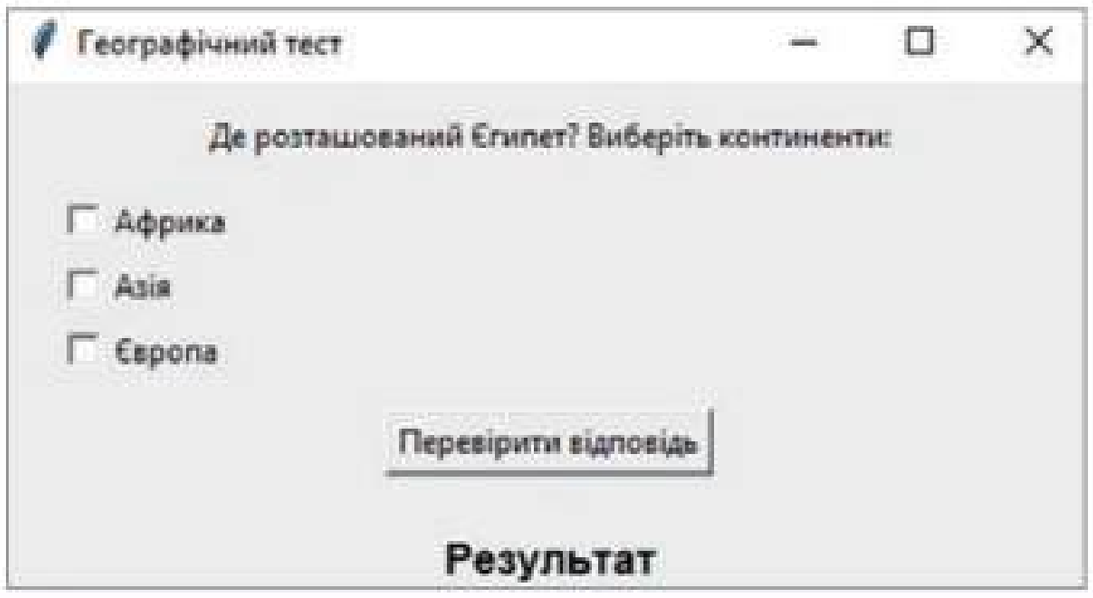
```
def check_sum():
    # Отримання чисел та відповіді
    num1 = int(num1_label.cget("text"))
    ...рядок коду для другого числа
    user_sum = int(answer_entry.get())

    # Перевірка суми
    if user_sum == num1 + num2:
        result_label.config(text="Правильно!", fg="green")
    else:
        result_label.config(text=f"Неправильно! Правильна
відповідь: {num1 + num2}", fg="red")
```

Завдання 6. Географічний тест – Вибір континентів

Користувач відмічає **Checkbox** для континентів, на яких розташована вибрана країна. Після натискання кнопки програма перевіряє, чи відповідь правильна.

створити вікно за зразком:



Частини коду: створення **Checkbutton** і його розміщення

```
africa_var = IntVar()

ch1 = Checkbutton(window, text="Африка",variable=africa_var)
ch1.pack(anchor="w", padx=20)
```

- кнопка **Перевірити відповідь** викликає функцію **check_answer()**, яка отримує значення з чекбоксів та перевіряє правильність відповідей.

```
def check_answer():

    africa = africa_var.get()
    asia = asia_var.get()
    europe = europe_var.get()

    if africa == 1 and asia == 1 and europe == 0:
        result_label.config(text="Правильно! Єгипет знаходиться
в Африці та Азії.", fg="green")
    else:
            result_label.config(text="Неправильно. Перевірте ще
раз.", fg="red")
```

Завдання 7. Середовище проживання тигра

Користувач відмічає **Checkbox** для середовищ проживання тигра. Після натискання кнопки програма перевіряє правильність відповіді.

- Вибір середовищ проживання:
  - Користувач обирає правильні варіанти *(Джунглі та Саванна)*.
  - Пустеля і Степи є неправильним варіантом.
- Виведення результату:
  - Якщо відповіді правильні, з'являється текст **Правильно!** зеленим кольором.
  - Якщо є помилки, текст **Неправильно** виводиться червоним.
- Змінні для чекбоксів:

`steppes_var`, `jungle_var`, `desert_var`, `savanna_var` зберігають стани вибору користувача.

Завдання 8. Математична вікторина

Користувач має обрати правильну відповідь на два математичні питання, використовуючи **Radiobutton**.

- створити вікно за зразком:



Частини коду: створення першого питання

```
question1 = Label(window, text="1. Скільки буде 5 * 3?")
question1.pack()

question1_var = StringVar(value="None")

rb1 = Radiobutton(window, text="12", variable=question1_var,
value="12")
rb1.pack()
rb2 = Radiobutton(window, text="15", variable=question1_var,
value="15")
rb2.pack()
rb3 = Radiobutton(window, text="18", variable=question1_var,
value="18")
rb3.pack()
```

- кнопка **Перевірити відповіді** викликає функцію **check_quiz()**, яка отримує значення з радіокнопок та перевіряє правильність відповідей.

```
def check_quiz():
    answer1 = question1_var.get()
    answer2 = question2_var.get()

    if answer1 == "15" and answer2 == "36":
        result_label.config(text="Ви правильно відповіли на
всі питання!")
    else:
        result_label.config(text="Є помилки у відповідях. Спро-
буйте ще раз.")
```

- додати ще одне питання і доповнити код функції для перевірки правильності.

Завдання 9. Налаштування стилів графічного інтерфейсу

Створіть програму, яка дозволяє користувачеві налаштовувати стиль вікна та його компонентів за допомогою:

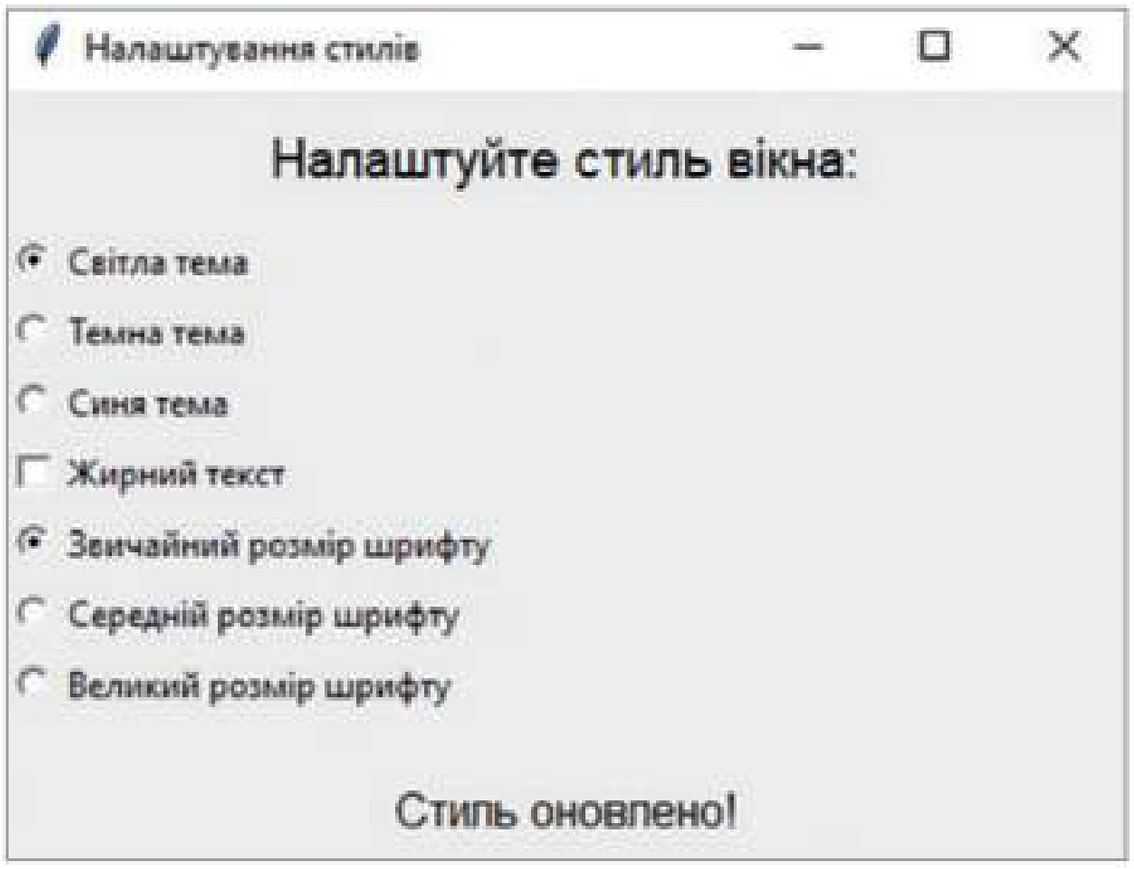
- **Radiobutton** для вибору кольорової схеми інтерфейсу (світла, темна, синя).
- **Checkbox** для налаштування текстових компонентів (зміна шрифту, розміру тексту тощо).

Вимоги до програми:

- Використовуйте **Radiobutton** для зміни кольорової схеми всього вікна (фон вікна і компонентів).
- Використовуйте **Checkbox** для увімкнення/вимкнення жирного тексту, збільшення або зменшення розміру шрифту.

- Після кожної зміни стиль компонентів повинен оновлюватися.

створити вікно за зразком



Частина коду для створення радіокнопок для вибору кольорової схеми:

```
color_var = StringVar(value="light")
color1    =    Radiobutton(window,    text="Світла    тема",
variable=color_var, value="light", command=apply_styles)
color1.pack(anchor="w")
```

Код для створення чекбоксу:

```
bold_var = IntVar()

check = Checkbutton(window, text="Жирний текст",
variable=bold_var, command=apply_styles)
check.pack(anchor="w")
```

Частина коду для створення радіокнопок для вибору розміру тексту:

```
size_var = IntVar(value=0)
size1 = Radiobutton(window, text="Звичайний розмір шрифту",
variable=size_var, value=0, command=apply_styles)
size1.pack(anchor="w")
```

```
# 0 — нормальний розмір, 1 — середній, 2 — великий
```

Функція **apply_styles()** перевіряє вибір користувача і застосовує зміни.

Частина коду:

```
def apply_styles():
    # Зміна кольорової схеми
    if color_var.get() == "light":
        window.config(bg="white")
```

```
            result_label.config(bg="white", fg="black")
  ...продовження умов

    # Налаштування шрифту
if bold_var.get() == 1:
font_weight = "bold"
else:
font_weight = "normal"
current_font = font.Font(size=12, weight=font_weight)

if size_var.get() == 1:
        current_font.config(size=16)
...продовження умов

   result_label.config(font=current_font)
```

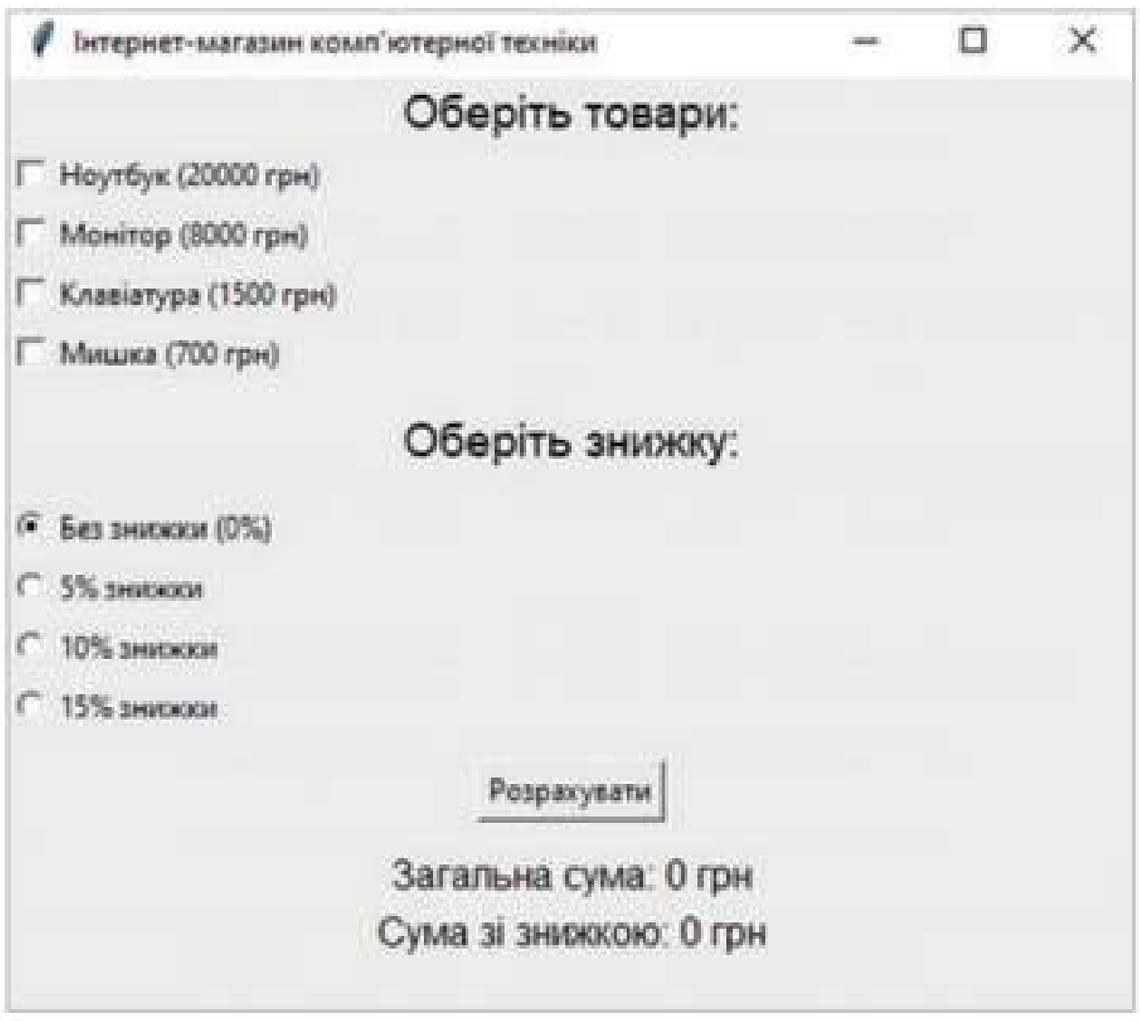


Додатково: змінювати колір фону інших компонентів.

Завдання 10. Інтернет-магазин комп'ютерної техніки

Створіть програму для інтернет-магазину комп'ютерної техніки, яка дозволяє покупцеві обрати товари з представленого списку (ноутбук, монітор, клавіатура, мишка тощо), дізнатися загальну суму покупки та застосувати знижку (за допомогою радіокнопок) та побачити підсумкову вартість товарів зі знижкою.

створити вікно за зразком:



кнопка **Розрахувати** викликає функцію **calculate()**, яка:

1) Шукає загальну суму покупки

Частина коду:

```
suma = 0
if laptop_var.get() == 1:
    suma = suma + 20000
    ... продовження умов
```

2) Застосування знижки

- отримати значення знижки з радіокнопки

```
discount = discount_var.get()
```

- знайти вартість покупки із знижкою

```
suma_discount = suma- (suma * discount / 100)
```

3) Виведення результатів

```
total_label.config(text=f"Загальна сума: {total} грн")
        discount_label.config(text=f"Сума зі знижкою:
{round(total_with_discount, 2)} грн")
```

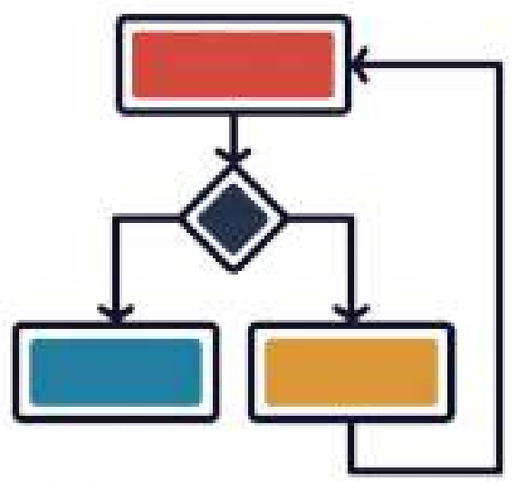
Ключові поняття

Команди повторення
Цикл for
Цикл while
Функція range()
Віджет Scale

Команди повторення

Як вам уже відомо, у програмуванні часто виникають ситуації, коли потрібно повторювати ту саму дію багато разів. Наприклад, відобразити всі числа від 1 до 100 або підсумувати кілька елементів тощо.

Для цього використовуються команди повторення, або цикли.



Команди повторення, або **цикли** — це одна із найважливіших конструкцій програмування, яка дозволяє виконувати певний блок коду багато разів.

Завдяки циклам можна автоматизувати повторювані завдання, що робить код коротшим та ефективнішим.

У Python існує два основні види циклів:

for while



Цикл for

Цикл **`for`** дозволяє пройтись по елементах списку, рядка або іншої колекції, а також виконати код певну кількість разів за допомогою функції **`range()`**.

Синтаксис:

```
for змінна in послідовність:
    # Команди, які виконуються для
    # кожного елемента
```

- **змінна**: кожен елемент послідовності послідовно присвоюється цій змінній;
- **послідовність**: будь-яка ітерована структура даних, наприклад, список, кортеж, рядок або результат виклику функції **`range()`**.

Приклади:

```
for i in range(5):
  print("Повторення:", i)
```

```
Повторення: 0
Повторення: 1
Повторення: 2
Повторення: 3
Повторення: 4
```

```
fruits = ["яблуко", "банан", "вишня"]
for fruit in fruits:
    print(fruit)
```

```
яблуко
банан
вишня
```

Зауважимо, що більш детально із списками та рядками ви ознайомитеся в старших класах.

Функція range()

Функція **range()** генерує послідовність чисел. Вона дуже зручна, коли потрібно повторити дію певну кількість разів.

Синтаксис **range()**:

range(кінцеве значення) — від 0 до вказаного значення (не включаючи його).

range(початок, кінець) — від початкового до кінцевого значення.

range(початок, кінець, крок) — крок визначає, наскільки кожне наступне число відрізняється від попереднього.

```
# Вивести числа від 2 до 8
for i in range(2, 9):
    print(i, end=" ")
```

```
2 3 4 5 6 7 8
```

```
# Вивести парні числа від 2 до 10
for i in range(2, 11, 2):
    print(i, end=" ")
```

```
2 4 6 8 10
```

Використання for з if

Цикл **for** можна комбінувати з умовними операторами, наприклад, для фільтрації елементів.

```
for num in range(1, 6):
    if num % 2 == 0:
        print(num, "- парне")
    else:
        print(num, "- непарне")
```

```
1 - непарне
2 - парне
3 - непарне
4 - парне
5 - непарне
```

Цикл while

Цикл **while** виконується доти, доки певна умова залишається істинною (**True**).

Зазвичай він використовується тоді, коли не знаємо точної кількості повторень заздалегідь, але знаємо умову, за якої потрібно зупинитися.

Синтаксис:

```
while умова:
    # блок коду, який виконується,
    # поки умова істинна
```

Умова — це логічний вираз. Коли він стає **False**, цикл припиняється.

Приклади:

```
i = 1
while i <= 5:
  print("Повторення:", i)
  i += 1
```

```
Повторення: 1
Повторення: 2
Повторення: 3
Повторення: 4
Повторення: 5
```

```
password = int(input("Введіть пароль: "))
while password != 1234:
  print("Спробуй ще")
  password = int(input("Введіть пароль: "))
print("Доступ дозволено!")
```

```
Введіть пароль: 125
Спробуй ще
Введіть пароль: 1234
Доступ дозволено!
```

Оператори break і continue в while

Іноді потрібно припинити цикл достроково або пропустити частину ітерації. Для цього в **Python** використовуються команди **break** та **continue**.

Команда break

```
i = 0
while i < 10:
    print(i, end=" ")
    if i == 5:
        break
    i += 1
```

```
0 1 2 3 4 5
```

```
for i in range(1, 10):
    print(i, end=" ")
    if i == 5:
        break
```

```
1 2 3 4 5
```

Команда **break** завершує виконання циклу достроково, якщо в певний момент треба припинити цикл незалежно від початкової умови.

Команда continue

```
i = 0
while i < 5:
    i += 1
    if i == 3:
        continue
    print(i, end=" ")
```

```
for i in range(1, 6):
    if i == 3:
        continue
    print(i, end=" ")
```

```
1 2 4 5
```

Команда **continue** пропускає поточну ітерацію і переходить до наступної. Це корисно, коли потрібно пропустити окремі значення.

Примітка! Ці оператори аналогічно використовуються у циклі **for**.

Нескінченний цикл while

```
while True:
    n = int(input("Введіть число (введіть 0 для завершення): "))
    if n == 0:
        break
    print("Подвоєне число:", 2 * n)
```

```
Введіть число (введіть 0 для завершення) : 7
Подвоєне число: 14
Введіть число (введіть 0 для завершення) : 3
Подвоєне число: 6
Введіть число (введіть 0 для завершення) : 0
```

Це приклад нескінченного циклу, який працює постійно, поки користувач не введе число 0. Тоді цикл зупиняється за допомогою **break**.

Використання else з циклами

```
for i in range(3):
    print(i)
else:
    print("Цикл завершено.")
```

```
0
1
2
Цикл завершено.
```

```
i = 0
while i < 3:
    print(i)
    i += 1
else:
    print("Цикл завершено.")
```

Цикли **for** та **while** можуть мати блок **else**, який виконується, коли цикл завершується "нормально", тобто без використання **break**.

Якщо б цикл перервався командою **break**, **else** не виконувався б.

Вкладені цикли

Цикли можуть бути вкладеними, тобто один цикл може міститися всередині іншого. Це корисно, коли потрібно працювати з двовимірними структурами, наприклад, для виведення таблиці або матриці.

Із вкладеними циклами ви вже ознайомлювалися в 7 класі, коли вивчали модуль **Turtle**.

```
from random import *

for i in range(1, 4):  # Зовнішній цикл
    for j in range(1, 5):  # Внутрішній цикл
        print(randint(1, 9), end=" ")
    print() # Перехід на новий рядок після
            # кожного внутрішнього циклу
```

```
5 2 9 5
6 7 1 5
4 9 6 8
```

Цей код генерує і виводить сітку випадкових чисел від 1 до 9, де кожен рядок має по 4 числа, а всього таких рядків є 3.

Опис коду:

Зовнішній цикл `for i in range(1, 4)`: – створює 3 ітерації, де змінна `i` приймає значення 1, 2 та 3. Кожна ітерація цього циклу відповідає новому рядку виводу.

Внутрішній цикл `for j in range(1, 5)`: – створює 4 ітерації, де змінна `j` приймає значення 1, 2, 3 та 4. Кожна ітерація цього циклу генерує одне випадкове число та виводить його у поточному рядку.

`print(randint(1, 9), end=" ")` – викликає функцію **`randint(1, 9)`** з модуля **`random`**, яка генерує випадкове ціле число від 1 до 9 включно, а параметр **`end=" "`** дозволяє вивести числа в один рядок з пробілами між ними.

`print()` (поза внутрішнім циклом) – завершує поточний рядок, переходячи до нового. Це виконується після завершення всіх ітерацій внутрішнього циклу, створюючи окремий рядок у виводі.

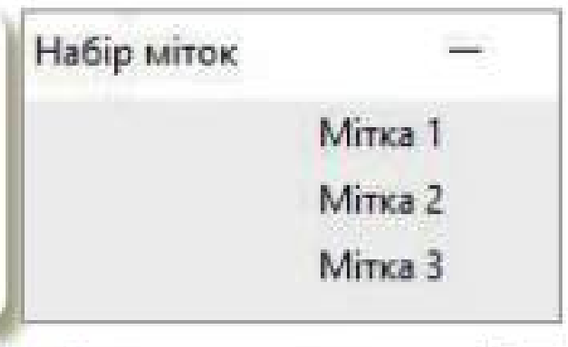
Використання циклів у Tkinter

Цикли дуже зручні для роботи в **`Tkinter`**, коли потрібно створювати або оновлювати кілька елементів на екрані. Нижче розглянемо, як за допомогою циклів **`for`** та **`while`** можна створювати інтерактивні інтерфейси, розміщувати елементи у вигляді таблиць, а також динамічно змінювати інтерфейс.

Створення набору віджетів за допомогою циклу `for`:

Цикл **`for`** допомагає зекономити час та код, коли потрібно створити набір подібних віджетів (наприклад, міток, чекбоксів або кнопок). Ось приклад, де створюються кілька міток з різним текстом.

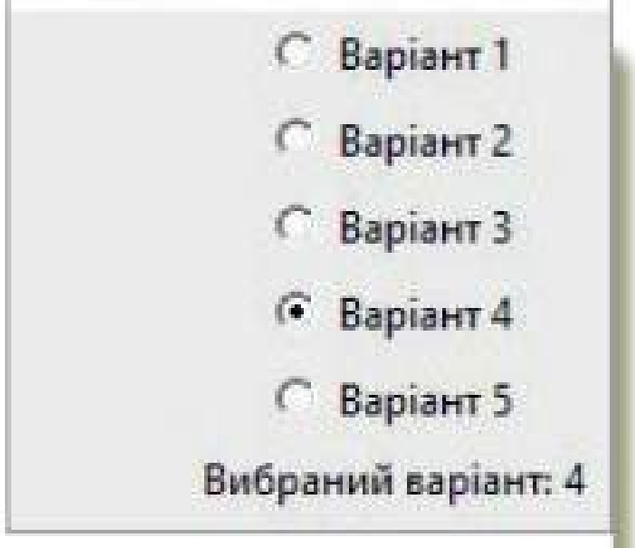
```
# Створення міток з різними текстами
for i in range(1, 4):
    label = Label(window, text=f"Мітка {i}")
    label.pack()
```




```
def status():
     label = Label(window, text=f"Вибраний
варіант: {selected_option.get()}")
    label.pack()

selected_option = IntVar(value=0)

for i in range(1, 6):
    radiobutton = Radiobutton(window, text=f"Варіант {i}",
                variable=selected_option, value=i,
                command=status)
    radiobutton.pack()
```



Розміщення віджетів у сітці за допомогою циклу `for`:

Вкладені цикли в **`Tkinter`** можна використовувати для створення таблиць або сіток елементів, використовуючи розміщення **`grid()`**.

```
# Створення числових кнопок у сітці 3x3
for i in range(3):                  # Рядки
    for j in range(3):              # Стовпці
        # Розрахунок номера кнопки
        button_number = i * 3 + j + 1
        # Створення кнопки з номером
      button = Button(window, text=str(button_number), width=5,
height=2)
        # Розміщення кнопки у відповідній позиції в сітці
        button.grid(row=i, column=j, padx=5, pady=5)
```

Цей код створить вікно з 9 кнопками, розташованими у сітці 3x3, з номерами від 1 до 9.



Опис коду:

Цикли для створення кнопок: Зовнішній цикл `for i in range(3)`: відповідає за рядки, а внутрішній `for j in range(3)`: за стовпці. Це дозволяє створити сітку 3x3.

Розрахунок номера кнопки: `button_number = i * 3 + j + 1` визначає значення кнопки, де кожне значення зростає від 1 до 9, формуючи сітку.

Створення кнопок: `tk.Button()` створює кнопку з текстом `button_number`, а `width=5` та `height=2` встановлюють її розміри.

Розміщення в сітці: Метод `grid(row=i, column=j, padx=5, pady=5)` розташовує кожну кнопку у відповідній позиції сітки з відступами для гарного вигляду.

Ця формула дозволяє обчислити номер кожної кнопки в сітці 3x3 так, що номери послідовно зростають від 1 до 9 зліва направо по рядках.

Всередині циклів також можна використовувати алгоритми з розгалуженням, створюючи цікаві та інтерактивні вікна.

```
for i in range(1, 6):
    for j in range(1, 6):
        label = Label(window, text=f"{i * j}",
        borderwidth=1, relief="solid",
        width=5, height=2)
        label.grid(row=i, column=j)
        if (i == 1) or (j == 1):
            label.config(bg="lightblue")
```

Цей код створює таблицю 5x5 у вікні **Tkinter**. Вміст кожної клітинки — це добуток номерів рядка та стовпця, а перший рядок і перший стовпець виділені світло-блакитним фоном для зручності.



Динамічне оновлення елементів за допомогою `while` та `after`:

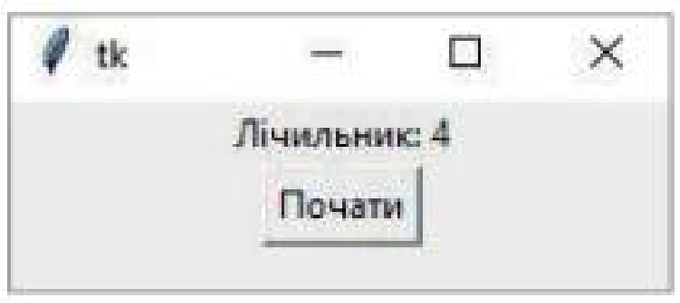
Цикл **while** в **Tkinter** може використовуватися для створення або оновлення інтерфейсу до виконання певної умови.

while зазвичай використовується обережно, оскільки він може блокувати основний цикл програми **(mainloop)** і зробити інтерфейс невідповідним (замороженим). Однак, щоб уникнути блокування інтерфейсу, краще використовувати функцію **after()** — вона дозволяє оновлювати елементи з певним інтервалом часу.

```
label = Label(window, text="Лічильник: 0")
label.pack()

start_button = Button(window, text="Почати", command=start)
start_button.pack()
```

Ось приклад використання циклу **while** в **Tkinter** для реалізації лічильника, який збільшується від 0 до 10 з інтервалом у 1 секунду, коли натискається кнопка **"Почати"**.



```
def start():
    i = 0
    while i <= 10:
        label.config(text=f"Лічильник: {i}")
        window.update()  # Оновлення інтерфейсу
        i += 1
        window.after(1000)  # Затримка на 1 секунду
```

i = 0: ініціалізується змінна **i**, яка виступає як лічильник.

Цикл **while i <= 10:** цикл виконується, доки значення **i** не досягне 10. У кожній ітерації лічильник **i** збільшується на 1.

- `label.config(text=f"Лічильник: {i}")`: оновлює текст віджета **label**, встановлюючи поточне значення **i**.
- `window.update()`: оновлює інтерфейс користувача, щоб зміни тексту на екрані стали видимими негайно.
- `window.after(1000)`: виконує затримку на 1 секунду (1000 мс) перед наступною ітерацією циклу, створюючи інтервал між оновленнями.

Віджет Scale

Віджет **Scale** використовується для створення повзунків, які дозволяють користувачеві вибрати числове значення з певного діапазону. Цей компонент особливо корисний для введення чисел або коли потрібне регулювання параметрів (наприклад, гучність, яскравість, швидкість, розмір тощо).

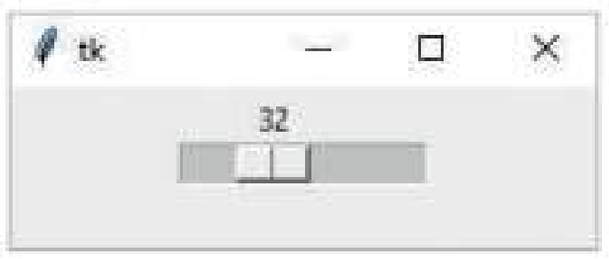
Основні властивості віджета **Scale**:

1. Горизонтальний або вертикальний напрямок: можна налаштувати, чи повзунок буде розташований горизонтально або вертикально.
2. Діапазон значень: задається мінімальне і максимальне значення, між якими переміщається повзунок.
3. Крок зміни: визначає, з яким кроком буде змінюватися значення.
4. Поточне значення: змінюється при переміщенні повзунка, відображаючи обране значення.

Синтаксис:

```
scale = Scale(parent, from_=мінімум, to=максимум,
              orient=напрямок)
scale.pack()
```

from — максимальне значення на шкалі
to — мінімальне значення на шкалі
orient — орієнтація повзунка: HORIZONTAL (горизонтально) або VERTICAL (вертикально).



Налаштування параметрів Scale

Крім вище згаданих, віджет **Scale** має багато інших параметрів, які дозволяють налаштовувати зовнішній вигляд, діапазон значень, крок і взаємодію. Розглянемо основні з них:

resolution — задає крок зміни значень. За замовчуванням дорівнює 1.

length — довжина шкали в пікселях.

tickinterval — інтервал, з яким на шкалі будуть відображатися мітки значень.

label — текст, що відображається поруч із повзунком.

```
scale = Scale(window, from_=0, to=50,
orient=HORIZONTAL, length=300,
          resolution=5, tickinterval=10,
label="Вибери значення",
              bg="lightblue", fg="blue",
font=('Arial', 14))
```

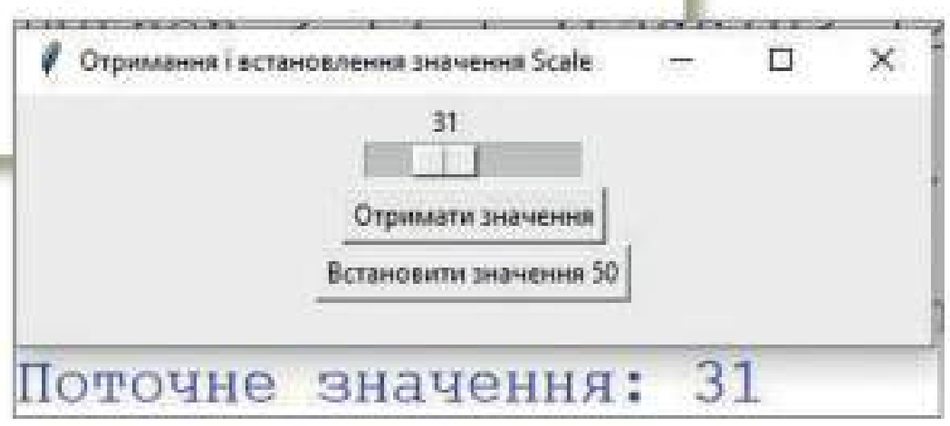


Отримання та встановлення значення Scale

Для роботи з повзунком можна використовувати методи **get()** та **set()**, щоб отримати поточне значення або змінити його програмно.

```
# Функція для отримання значення повзунка
def get_value():
    value = scale.get()
    print("Поточне значення:", value)

button_get = Button(window, text="Отримати значення",
             command=get_value)
button_get.pack()
```

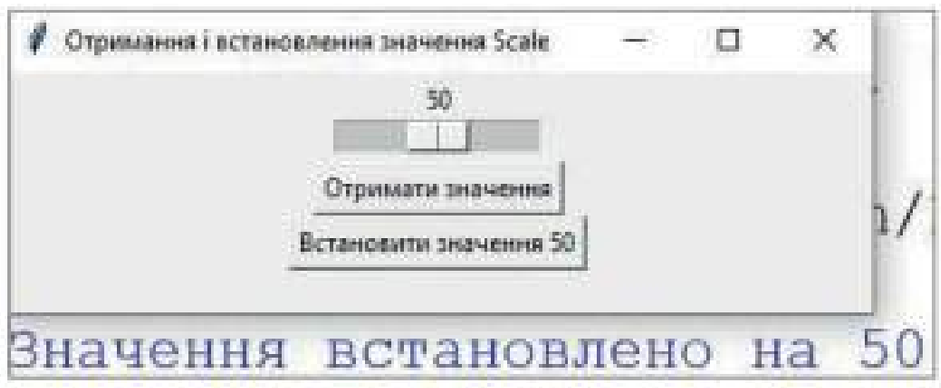



```
Функція для встановлення значення повзунка
def set_value():
    scale.set(50)  # встановлюємо значення 50
    print("Значення встановлено на 50")

button_set = Button(window, text="Встановити значення 50",
             command=set_value)
button_set.pack()
```

Опис:

- **`scale.get()`** — отримує поточне значення повзунка.
- **`scale.set(50)`** — встановлює значення повзунка на 50.



Використання command для обробки подій

Як і у раніше згаданих віджетах, параметр **`command`** дозволяє викликати функцію кожного разу, коли змінюється значення повзунка. Це зручно для оновлення інтерфейсу в реальному часі, наприклад, коли повзунок регулює гучність, яскравість або інші налаштування.

```
# Функція для обробки зміни значення повзунка
def on_scale(value):
    print("Поточне значення:", value)

# Створюємо повзунок з параметром command
scale = Scale(window, from_=0, to=100,
        orient = HORIZONTAL, command=on_scale)
scale.pack()
```



Опис:

- `command=on_scale`: функція **`on_scale()`** викликається кожного разу, коли користувач змінює значення повзунка.
- значення **`value`** передається у функцію як аргумент, що дозволяє виводити або використовувати поточне значення повзунка.

Зверніть увагу на запис **`on_scale(value)`**. Таким чином можна безпосередньо передавати значення у функцію, і в цій же функції використовувати дане значення.

У цьому прикладі параметр **`value`** отримує поточне значення повзунка кожного разу, коли користувач змінює його положення.

За допомогою **`Scale`** і **`command`** можна створити інтерактивний інтерфейс. Наприклад, можна змінювати колір фону вікна або розмір тексту на основі значення повзунка. Розглянемо приклад, коли користувач змінюватиме значення повзунка — і викликатиметься функція, наприклад, `change_font_size()`, яка змінюватиме розмір тексту на мітці.

```
# Мітка для відображення тексту
label = Label(window, text="Змінюваний текст",
              font=("Arial", 20))
label.pack()

# Функція для зміни розміру тексту
def change_font_size(value):
    label.config(font=("Arial", int(value)))

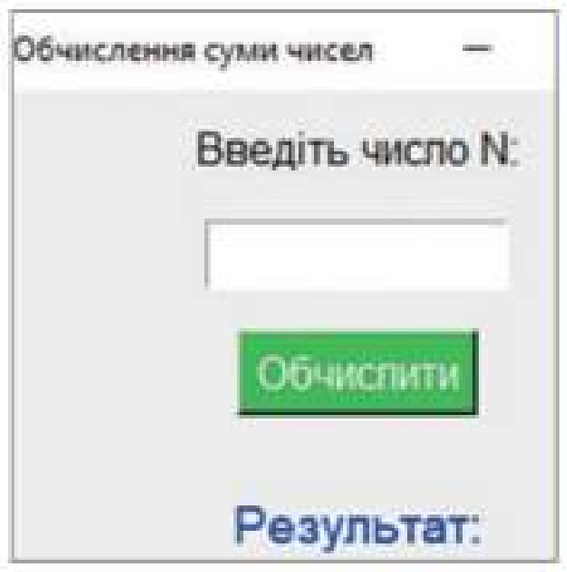
# Повзунок для вибору розміру тексту
scale = Scale(window, from_=10, to=50,
              orient=HORIZONTAL,
              command=change_font_size)
scale.pack()
```

Практична робота №25.

Завдання 1. Обчислення суми чисел від 1 до N.

Створіть програму, яка дозволяє користувачеві ввести значення N, а на основі цього значення обчислює суму всіх чисел від 1 до N. Результат виводиться в окремій мітці.

- створити вікно, яке буде містити мітку (**Label**) та поле (**Entry**) для введення числа N, кнопку (**Button**) та мітку (**Label**), яка виводитиме результат;
- кнопка викликає функцію `calculate_sum()`, яка отримує ціле число з поля **Entry** та за допомогою циклів знаходить суму чисел від 1 до N.




```
def calculate_sum():
n = int(entry.get())
s = 0
for i in range(1, n + 1):
            s = s + i
result_label.config(text=f"Сума чисел: {s}")
```

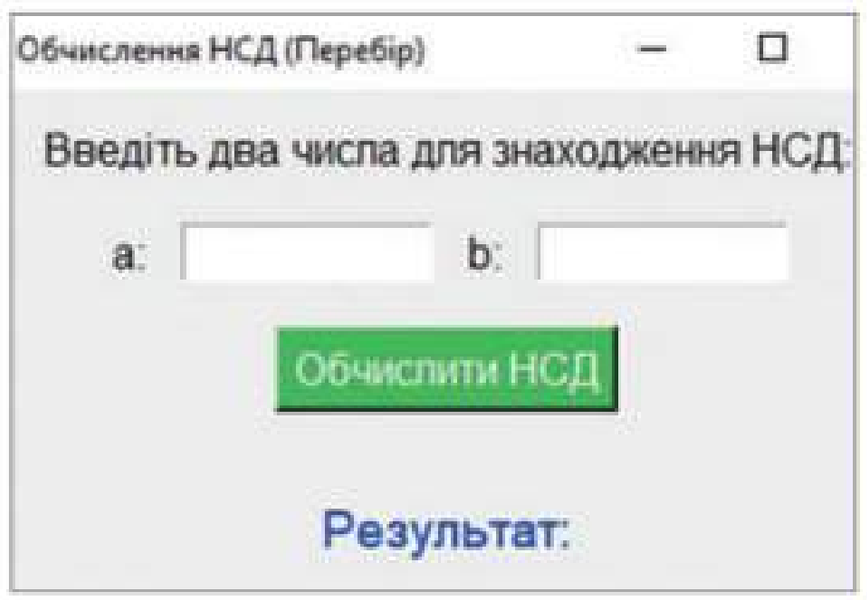
Завдання 2. Обчислення суми чисел від **A** до **B**.

Створіть програму, яка дозволяє користувачеві ввести значення **A** і **B**, а на основі цих значень обчислює суму всіх чисел від **A** до **B**. Результат виводиться в окремій мітці.

Завдання 3. Знаходження найбільшого спільного дільника (**НСД**) двох чисел

Створіть програму, яка знаходить найбільший спільний дільник (**НСД**) двох чисел **a** і **b** за допомогою перебору всіх можливих дільників.

- створити вікно за зразком:
- кнопка викликає функцію `calculate_nsd()`, яка отримує цілі числа **a** та **b** та знаходить **НСД** цих чисел.



Алгоритм знаходження НСД:

1. Визначте менше (`min`) з двох чисел **a** і **b**.
2. Перевіряйте всі числа від 1 до min, які одночасно ділять **a** і **b**.
3. Найбільше таке число — це **НСД**.

```
def calculate_nsd():
    a = int(entry_a.get())
    b = int(entry_b.get())
    if a < b:
        min = a
    else:
        min = b

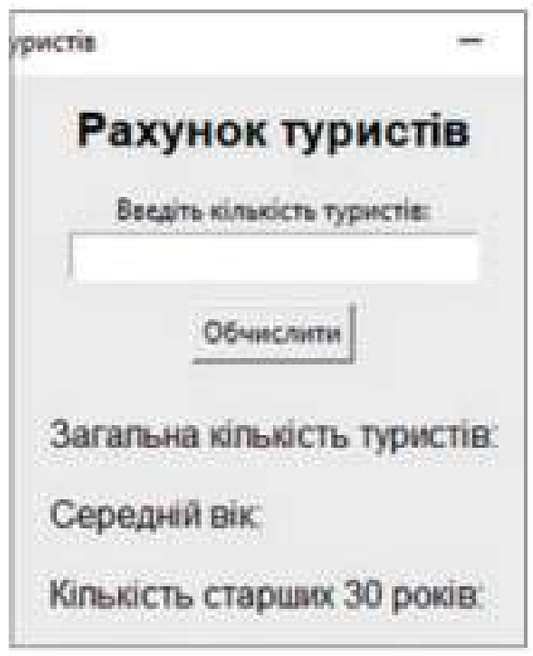
    nsd = 1
    for i in range(1, min + 1):
        if a % i == 0 and b % i == 0:
            nsd = i

    result_label.config(text=f"НСД({a},{b}): {nsd}")
```

Завдання 4. Рахунок туристів

Туристична група вирушає у подорож. Програма має запитувати у користувача кількість туристів та їхній вік. Вивести загальну кількість туристів, середній вік та кількість туристів, старших за 30 років.

- створити вікно за зразком:
- кнопка викликає функцію `tourists()`, яка отримує ціле число n, дозволяє користувачеві вводити в консоль вік туристів та знаходить середній вік туристів та кількість туристів, яким за 30 років.




```
def tourists():
    n = int(entry.get())
sum = 0
    count = 0

    for i in range(1, n + 1):
        age = int(input(f"Введіть вік туриста {i}: "))
    sum = sum + age
        if age > 30:
            count = count + 1

     average = sum / n
     result_1.config(text=f"Загальна кількість туристів: {n}"
result_2.config(text=f"Середній вік: {round(average, 2)}"
result_3.config(text=f"Кількість старших 30 років: {count}"
```

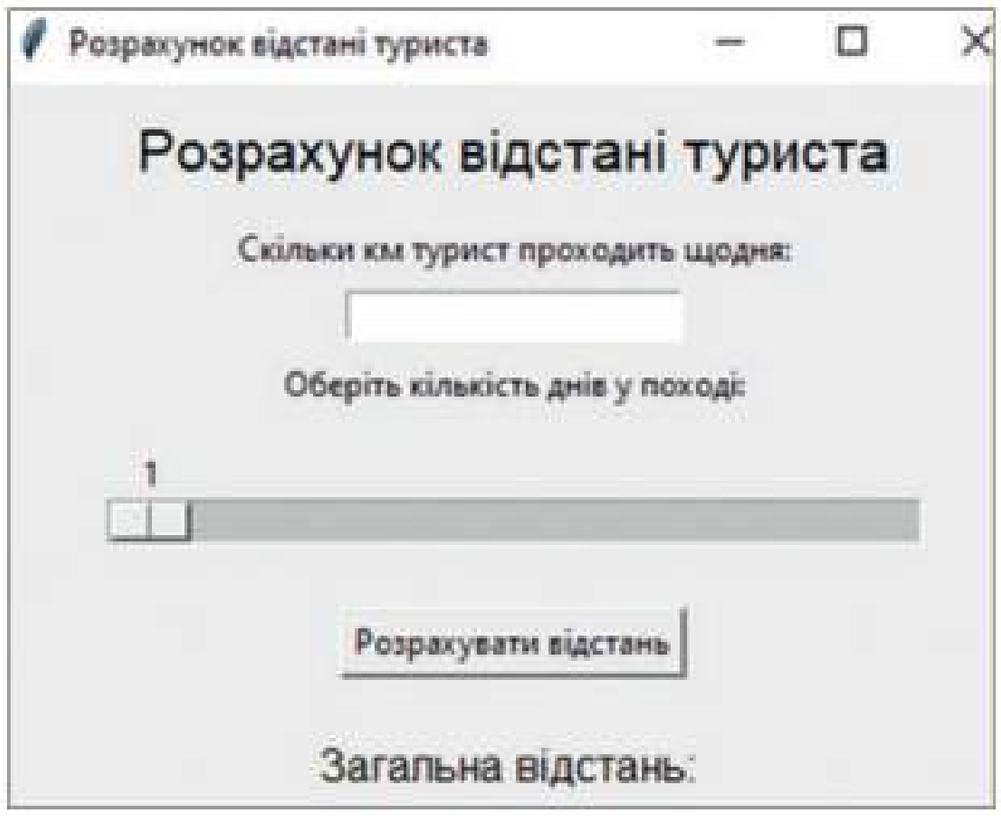
Завдання 5. Мандрівка автомобілем

Програма має запитувати кількість днів подорожі та відстань, яку автомобіль проїжджав кожного дня. Обчислити загальну відстань та середню швидкість за подорож.

Завдання 6. Розрахунок відстані, яку пройшов турист

Турист здійснює багатоденний похід. Кожного дня він проходить однакову кількість кілометрів. За допомогою компонента **Scale** користувач обирає кількість днів у поході (від **1** до **30**), а в поле для введення (**Entry**) вводить кількість кілометрів, які турист проходить щодня. Програма повинна розрахувати загальну відстань, яку подолав турист за обрану кількість днів, і вивести результат на екран.

- створити вікно за зразком:



Частини коду:

- створення **Entry**

```
# Введення кількості кілометрів на день
entry_label = Label(window, text="Скільки км турист проходить щодня:")
entry_label.pack()

entry = Entry(window)
entry.pack(pady=5)
```

- створення **Scale**

```
# Компонент Scale для вибору кількості днів
scale_label = Label(window, text="Оберіть кількість днів у поході:")
scale_label.pack()

scale = Scale(window, from_=1, to=30, orient="horizontal", length=300)
scale.pack(pady=10)
```

- кнопка **Розрахувати відстань** викликає функцію `calculate_distance()`, яка отримує значення із **entry** та **scale**, обчислює та виводить результат.

```
def calculate_distance():
    daily_distance = float(entry.get())  # Кілометри за день
    days = scale.get()  # Кількість днів

sum = 0  # Загальна відстань
    # Розрахунок загальної відстані
    for i in range(days):
        sum = sum + daily_distance

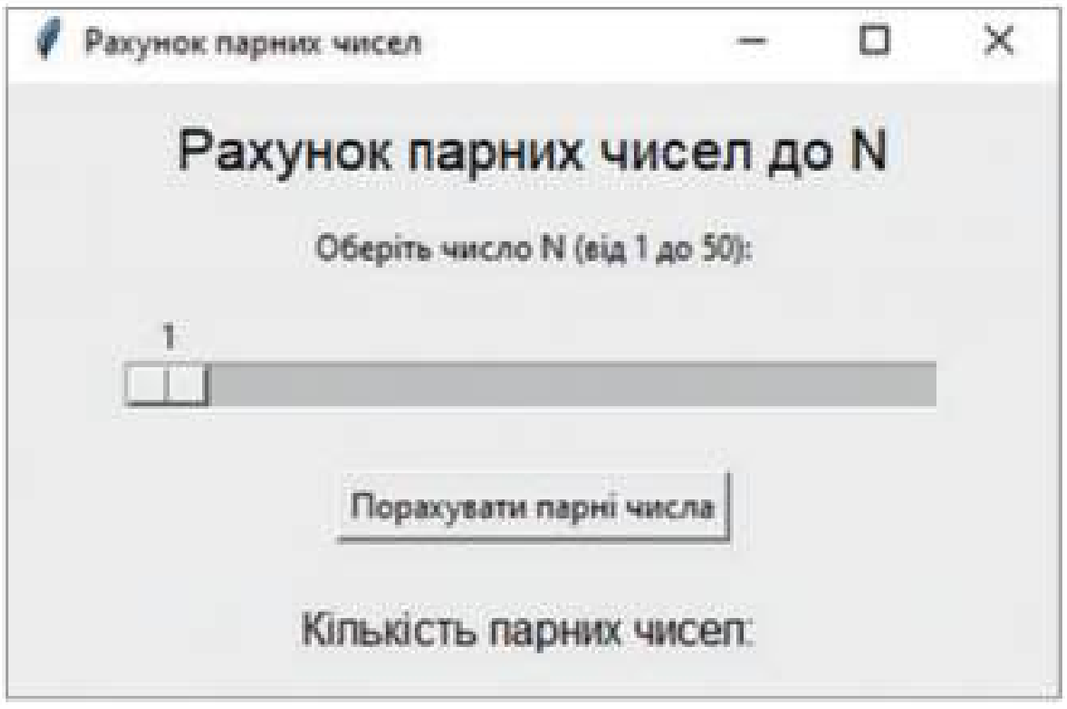
    # Вивід результату
    result_label.config(text=f"Загальна відстань: {sum} км")
```

- Додатково виконайте обчислення, не використовуючи циклу.

Завдання 7. Рахунок парних чисел до N

Програма дозволяє користувачу обрати число N за допомогою компонента **Scale** (від 1 до 50). Після цього програма рахує кількість парних чисел від 1 до обраного значення N і виводить результат у мітці.

- створити вікно за зразком:



- *Частини коду:* створення **Scale**

```
# Компонент Scale для вибору числа N
scale_label = Label(window, text="Оберіть число N (від 1 до 50):")
scale_label.pack()

scale = Scale(window, from_=1, to=50, orient="horizontal", length=300)
scale.pack(pady=10)
```

- кнопка **Порахувати парні числа** викликає функцію `count_even_numbers()`, яка отримує значення із `scale`, перевіряє за допомогою циклу на парність чисел від `1` до `n` та виводить результат.

```
def count_even_numbers():
    n = scale.get()                 # Отримуємо значення зі шкали
    count = 0                       # Лічильник парних чисел

    for i in range(1, n + 1):        # Цикл від 1 до N
        if i % 2 == 0:              # Перевірка на парність
            count = count + 1        # Збільшуємо лічильник


    result_label.config(text=f"Кількість парних чисел до {n}: {count}")
```

Ключові поняття

Коментарі
Перевірка коду
Типи помилок
Вікно повідомлення messagebox

Коментарі

Коментування програмного коду

Коментування коду — це процес додавання пояснень у програмний код. Коментарі допомагають зрозуміти логіку програми та полегшують роботу над кодом як для автора, так і для інших програмістів.

Коментарі — це рядки, які ігноруються інтерпретатором під час виконання програми.



У **Python** коментарі позначаються символом **#**, після якого пишуть текст коментаря. Коментарі особливо важливі під час створення складних програм.



Навіщо потрібні коментарі?

Коментарі не впливають на роботу коду, але дають розробникам додаткову інформацію:

- Описують логіку коду та пояснюють, як працює програма.
- Вказують на складні частини коду, щоб полегшити розуміння.
- Допомагають командній роботі: інші розробники зможуть швидше зрозуміти код.
- Полегшують підтримку коду через деякий час, коли автор може забути деталі реалізації.

Типи коментарів у Python

У **Python** є два основні типи коментарів: одно- та багаторядкові.

Однорядкові коментарі

Однорядкові коментарі використовуються для коротких пояснень і починаються із символа **#**. Коментар записується праворуч від коду або в окремому рядку.

```
# Це однорядковий коментар
name = "Анна"  # Ім'я користувача
```

Багаторядкові коментарі

Багаторядкові коментарі оформляються за допомогою потрійних лапок (''' або """). Однак вони більше схожі на рядкові значення і технічно є рядками, які інтерпретатор ігнорує, якщо не присвоєні змінній.

```
'''
Це багаторядковий коментар.
Його можна використовувати для довгих пояснень.
Зазвичай такий коментар описує призначення коду.
'''
```

Коментування в Tkinter

Коментарі в **Tkinter** використовуються для пояснення структури графічного інтерфейсу, налаштування елементів (віджетів) та опису функцій, що реагують на дії користувача.

Приклад коментування коду в **Tkinter**

```
from tkinter import *  # Імпортуємо функції модуля Tkinter

# Створення головного вікна програми
window = Tk()
window.title("Моя програма")  # Встановлення заголовка вікна

# Створення мітки з текстом і додавання її у вікно
label = Label(window, text="Вітаю у Tkinter!")  # Створення
мітки
label.pack()  # Відображення мітки у вікні

# Функція, яка виконується при натисканні кнопки
def on_button_click():
    print("Кнопку натиснуто!")  # Виведення повідомлення в
консоль

# Створення кнопки, яка викликає функцію on_button_click
button = Button(window, text="Натисни мене", command=on_
button_click)
button.pack()  # Відображення кнопки у вікні

# Запуск головного циклу програми
window.mainloop()  # Цей метод тримає вікно відкритим
```

Рекомендації щодо коментування коду



1 Короткі та інформативні коментарі

Коментарі повинні бути чіткими, без зайвих слів.

Наприклад, замість `#Створення змінної для збереження віку користувача` краще написати `#Вік користувача`.

2 Коментуйте складні ділянки коду

Якщо певна частина коду є незвичайною або складною, додайте коментар, що пояснює її логіку.



3 Не коментуйте очевидного

Якщо коментар дублює зміст коду, його можна уникнути.

Наприклад, `x = 10 #Присвоєння значення 10 змінній x` зайвий, бо код сам по собі очевидний.

4 Описуйте функції перед їх визначенням

Перед написанням функції додайте коментар, який описує її призначення, а також вхідні параметри та очікуваний результат.

Перевірка коду на помилки

Під час розробки програмного забезпечення в **Python**, особливо з використанням модуля **Tkinter** для створення графічного інтерфейсу, можуть виникати різні помилки.

Виявлення та виправлення помилок є важливим етапом, що дозволяє програмі працювати стабільно та без помилок.

Типи помилок у Python

Існує кілька типів помилок: синтаксичні, логічні та помилки виконання.

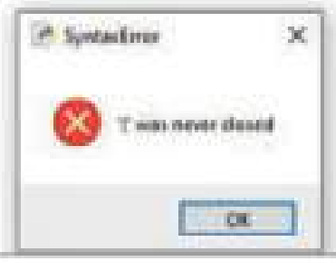
Синтаксичні помилки

Синтаксичні помилки виникають тоді, коли код не відповідає правилам мови **Python**. Такі помилки зазвичай виявляються при спробі запустити програму, і **Python** відразу ж повідомляє про помилку.

Приклади синтаксичних помилок:

- пропущено дужку;
- пропущено двокрапку після оголошення **`if`**, **`for`**, **`def`** тощо;
- неправильний відступ (що особливо важливо в **Python**).

```
# Неправильний синтаксис через пропущену дужку
print("Привіт, світ!"
```



Як виявити?

Python виведе повідомлення про помилку з текстом **SyntaxError**, вказавши на місце, де вона виникла.

❑ Помилки виконання (Run-time errors)

Ці помилки виникають під час виконання програми. Вони з'являються, коли **Python** не може виконати певний рядок коду, хоча синтаксис правильний.

Зазвичай виникають через:

- поділ на нуль;
- спробу доступу до хибного індексу в списку;
- використання невизначених змінних.

```
# Помилка виконання через поділ на нуль
number = 10 / 0
```

```
Traceback (most recent call last):
  File "C:/Users/Admin/Desktop/Підручник 8 клас/
Урок 36-37/3.py", line 5, in <module>
    number = 10 / 0
ZeroDivisionError: division by zero
```

Як виявити?

Python виведе повідомлення з текстом **ZeroDivisionError** або іншу помилку типу виконання, що вказує на проблему.

❑ Логічні помилки

Логічні помилки не викликають повідомлень про помилки і не припиняють роботу програми. Проте програма працює неправильно, тобто результат не відповідає очікуваному. Ці помилки є одними з найскладніших для виправлення, бо **Python** не повідомляє про них.

Приклади логічних помилок:

- написання помилкової формули обчислень;
- неправильне використання логічних операторів.

```
# Логічна помилка через неправильний оператор
age = 18
if age > 18:  # Очікувано, що 18 років також буде допускатися
    print("Повнолітній")
```

Як виявити?

Логічні помилки виявляються при тестуванні та перевірці результатів програми.

Методи перевірки коду на помилки в Python

Використання `try` та `except`

Конструкція **try-except** дозволяє обробляти помилки виконання. Якщо виникає помилка в блоці **try**, **Python** переходить до виконання блоку **except**, уникаючи аварійного завершення програми.

```
try:
    number = int(input("Введіть число: "))
    result = 10 / number
    print(result)
except ZeroDivisionError:
    print("Помилка: ділення на нуль!")
except ValueError:
    print("Помилка: введене значення не є числом.")
```

Пояснення:

- якщо користувач введе **0**, виникне помилка **ZeroDivisionError**, яка обробляється в першому блоці **except**;
- якщо користувач введе текст замість числа, виконається другий **except**, обробляючи помилку **ValueError**.

```
Введіть число: 5
2.0
```

```
Введіть число: 0
Помилка: ділення на нуль!
```

```
Введіть число: два
Помилка: введене значення не є числом.
```

Примітка! Поки що вам достатньо знати і використовувати два види помилок **ValueError** (некоректне значення) і **ZeroDivisionError** (ділення на нуль). З більшою кількістю помилок ви можете ознайомитися самостійно або досліджуватимете в шкільному курсі в старших класах.

`raise` — виклик помилок вручну

Команда **raise** дозволяє вручну викликати помилку, якщо потрібно зупинити виконання коду при певній умові.

```
number = -5
if number < 0:
    raise ValueError("Число не може бути від'ємним!")
```

`assert` — перевірка умов

assert дозволяє перевіряти певну умову, якщо умова не виконується, викликається помилка **AssertionError**.

```
age = 15
assert age >= 18, "Вік має бути 18 або більше"
```

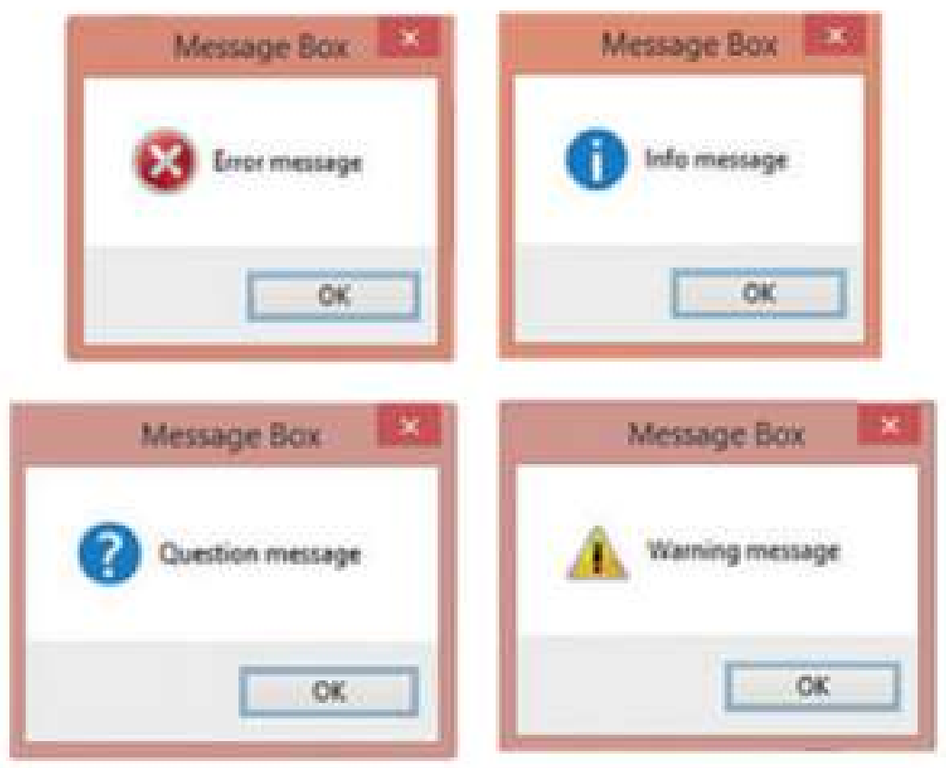
Вікно повідомлення messagebox

Одним із ключових інструментів для взаємодії з користувачем є вікно повідомлення, або **messagebox**.

messagebox — це спеціальний інструмент, який відкриває вікно повідомлення в **Tkinter**.

Його можна використовувати для відображення різних типів повідомлень, таких як:

- інформаційні повідомлення;
- попередження;
- повідомлення про помилки;
- підтвердження дії.



Messagebox спрощує взаємодію з користувачем і дозволяє йому приймати рішення чи отримувати важливу інформацію.

Імпорт модуля messagebox

messagebox є частиною модуля **Tkinter**, тому його можна імпортувати безпосередньо з **tkinter**:

```
from tkinter import *
from tkinter import messagebox
```

Примітка! Якщо вже імпортовано весь модуль **Tkinter** (**from tkinter import ***), вам все одно потрібно імпортувати **messagebox** окремо. Це пов'язано з тим, що **messagebox** є окремим підмодулем у **Tkinter**, і він не імпортується автоматично разом із *.

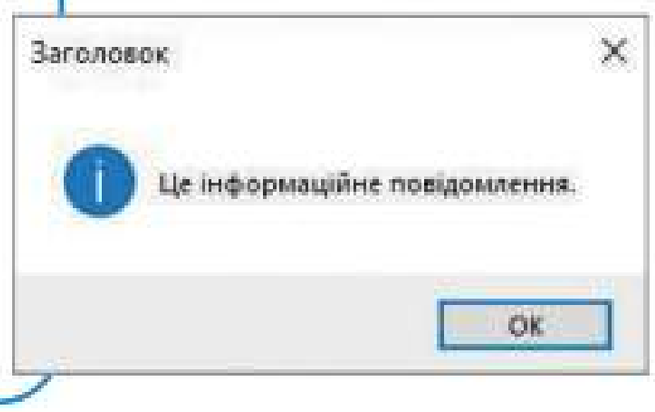
Основні типи повідомлень у messagebox

Tkinter має кілька типів повідомлень, кожен з яких використовується для конкретної ситуації.

showinfo

Відображає інформаційне повідомлення. Використовується, коли потрібно повідомити користувача про завершення певної операції чи важливу подію.

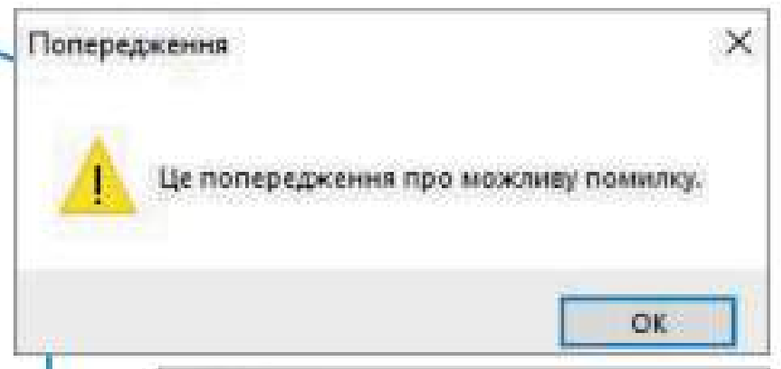
messagebox.showinfo(`"Заголовок", "Це інформаційне повідомлення."`)

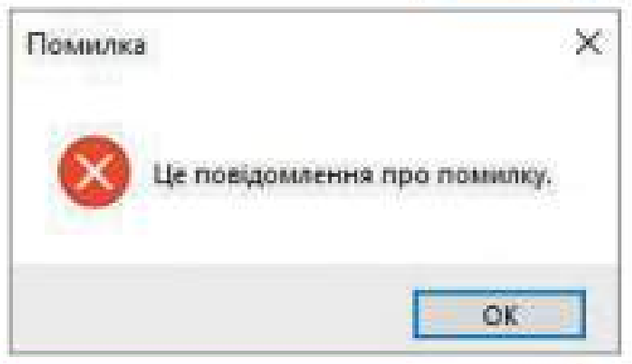


showwarning

Відображає попередження. Це корисно, коли потрібно привернути увагу користувача до можливих негативних наслідків певної дії.

messagebox.showwarning ("Попередження", "Це попередження про можливу помилку.")





showerror

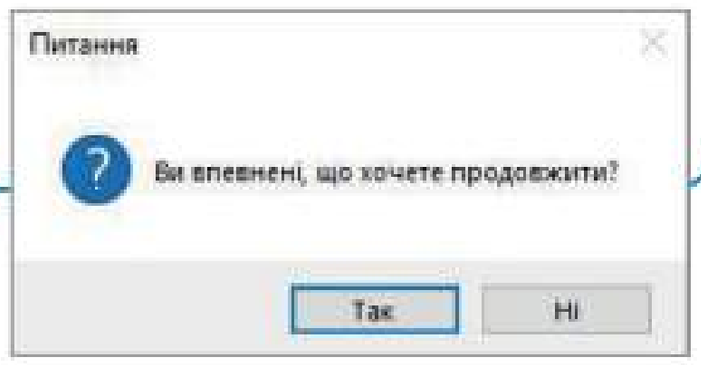
Відображає повідомлення про помилку. Це допомагає інформувати користувача про проблеми чи некоректні дії, які потрібно виправити.

messagebox.showerror ("Помилка", "Це повідомлення про помилку.")

askquestion

Відображає питання, яке очікує відповіді від користувача. Повертає **"yes"** або **"no"**, залежно від вибору.

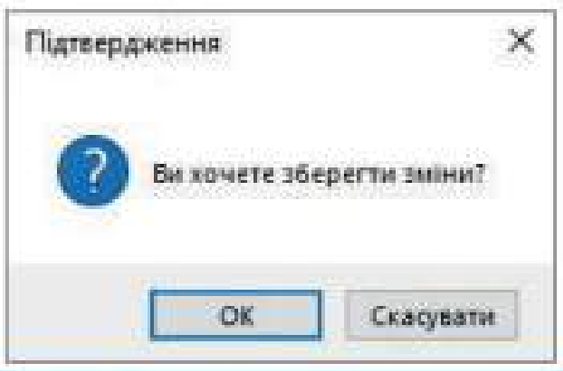
```
response = messagebox.askquestion("Питання", "Ви впевнені, що
хочете продовжити?")
   if response == "yes":
      print("Користувач обрав 'Так'")
   else:
      print("Користувач обрав 'Ні'")
```



askokcance

Показує підтвердження дії з кнопками **OK** та **Cancel**. Повертає **True**, якщо обрано **OK**, і **False**, якщо **Cancel**.

```
response = messagebox.askokcancel("Підтвердження", "Ви хочете
зберегти зміни?")
 if response:
     print("Користувач підтвердив дію")
 else:
     print("Користувач скасував дію")
```

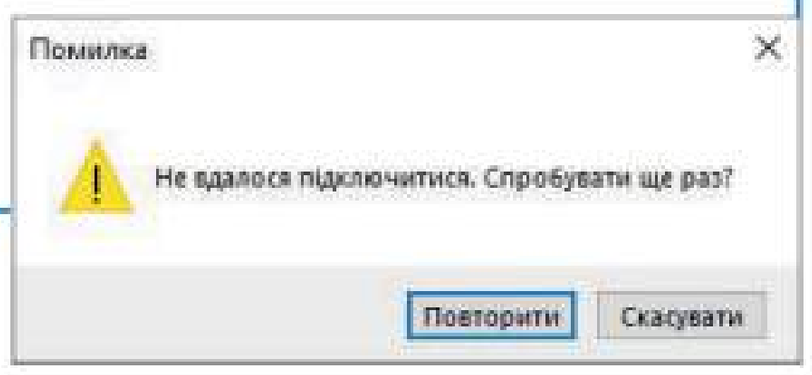


askretrycancel

Запитує у користувача, чи хоче він повторити дію або скасувати її. Корисно, коли потрібно дозволити користувачеві ще раз спробувати виконати дію.

```
response = messagebox.askretrycancel("Помилка", "Не вдалося підключитися. Спробувати ще раз?")

 if response:
    print("Користувач вибрав спробувати ще раз")
else:
    print("Користувач скасував дію")
```

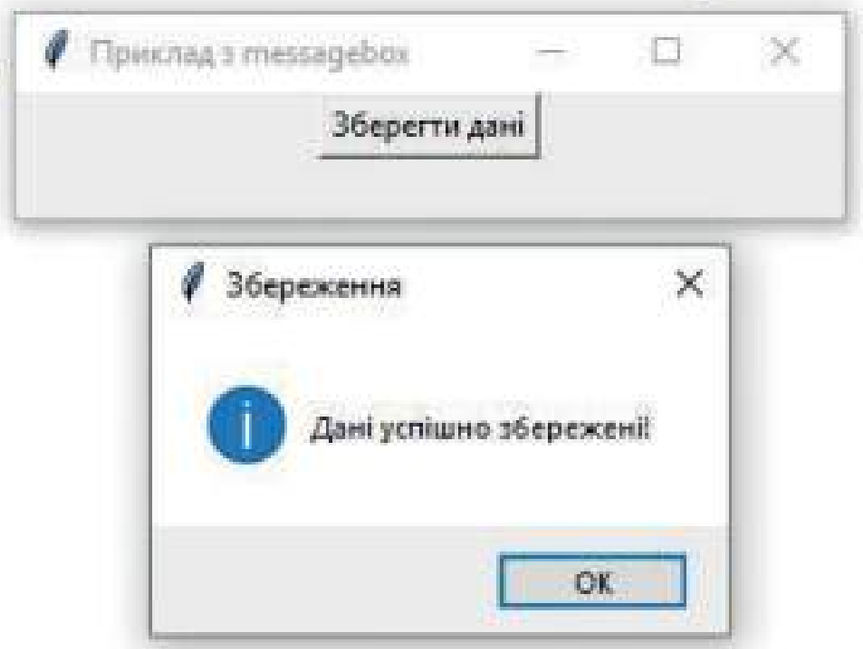


Практичне застосування messagebox у програмі Tkinter

Приклад 1: Інформаційне повідомлення після успішного збереження даних

```
def save_data():
    # Виконуються дії для збереження даних
    messagebox.showinfo("Збереження",
                        "Дані успішно збережені!")

save_button = Button(window, text="Зберегти дані",
                     command=save_data)
save_button.pack()
```



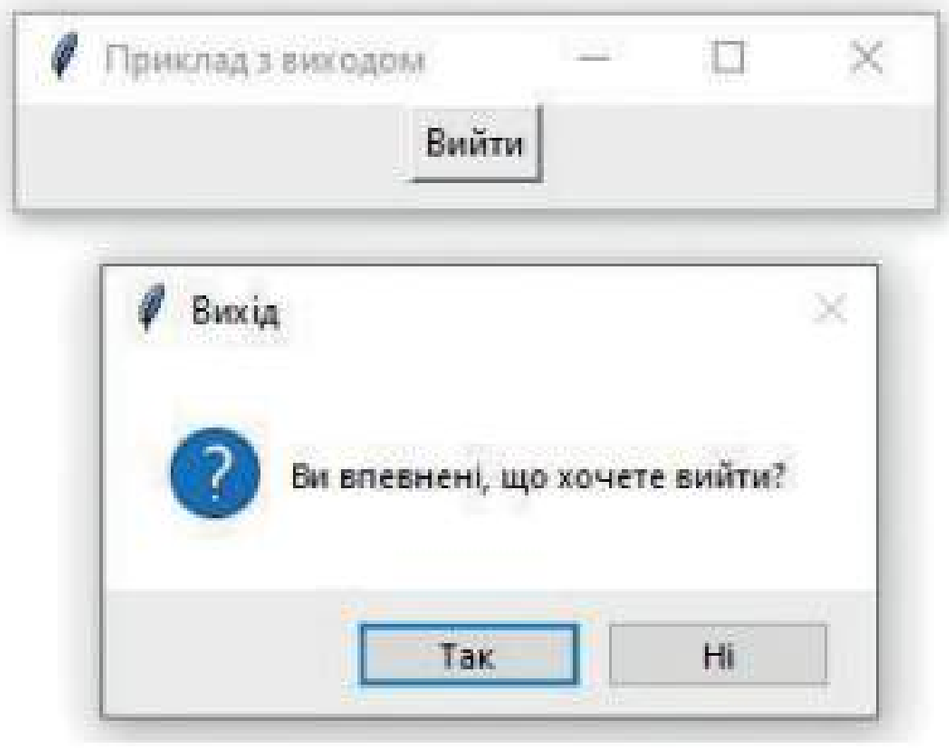
Пояснення: після натискання на кнопку "Зберегти дані" викликається функція **save_data**, яка показує вікно з інформацією про успішне збереження.

Приклад 2: Підтвердження виходу з програми

```
def on_exit():
     if messagebox.askyesno("Вихід", "Ви впевнені, що хочете вийти?"):
         window.destroy()

exit_button = Button(window, text="Вийти", command=on_exit)
exit_button.pack()
```

Пояснення: функція **on_exit** викликає **messagebox.askyesno**, запитуючи, чи хоче користувач вийти з програми. Якщо вибрано **Yes**, програма закривається. Функція **destroy()** відповідає за те, щоб вікно **Tkinter** закрилося.



Застосування messagebox для обробки помилок у Tkinter

Вікна **messagebox** корисні для обробки ситуацій, коли програма може отримати помилку під час введення даних або виконання операцій.

Розглянемо приклад ділення двох чисел:

```
window = Tk()
window.title("Калькулятор ділення")

entry1 = Entry(window)
entry1.pack()
entry2 = Entry(window)
entry2.pack()

button = Button(window, text="Поділити", command=calculate)
button.pack()
```

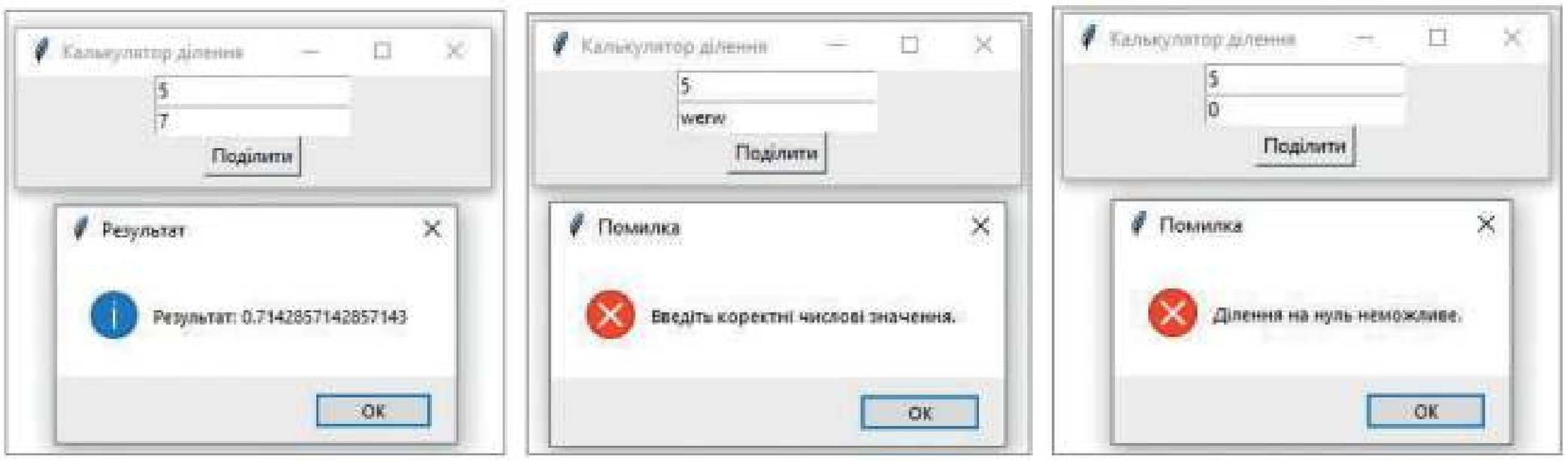
Створимо вікно з двома полями для вводу чисел та кнопки, яка викликатиме функцію ділення двох чисел.

```
def calculate():
    try:
        number1 = int(entry1.get())
        number2 = int(entry2.get())
        result = number1 / number2
      messagebox.showinfo("Результат", f"Результат: {result}")
    except ValueError:
      messagebox.showerror("Помилка", "Введіть коректні чис-
лові значення.")
    except ZeroDivisionError:
        messagebox.showerror("Помилка", "Ділення на нуль не-
можливе.")
```

Функція **calculate** намагається поділити два введені числа.

Якщо користувач вводить нечислове значення, з'являється **ValueError**, і відображається відповідне повідомлення.

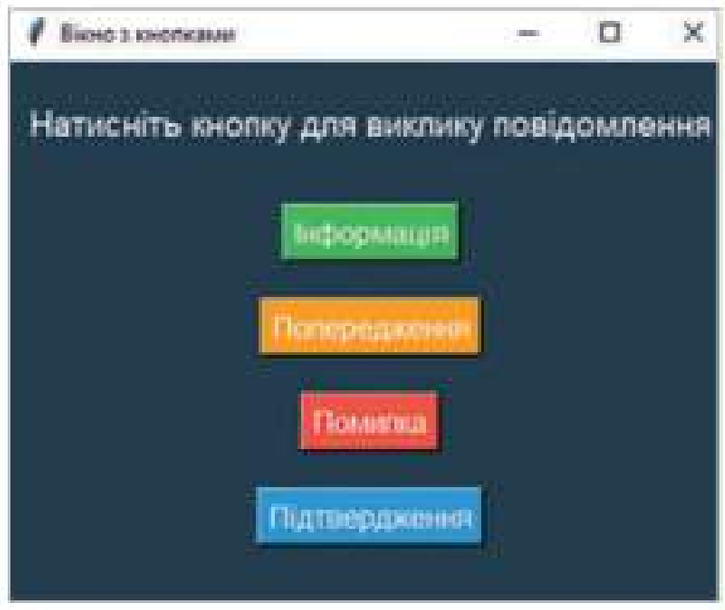
Якщо друге число дорівнює нулю, виникає **ZeroDivisionError**, і програма повідомляє про помилку ділення на нуль.



Оскільки в **Tkinter** вам доведеться часто працювати з віджетами вводу даних, доцільно завжди перевіряти код на введення коректних даних. Це підвищить надійність і зручність використання програми, особливо коли користувач може вводити некоректні дані.

Отже, завдяки **messagebox** вікнам програми стають більш дружніми до користувача, і взаємодія з додатком відбувається на зрозумілому та інтуїтивному рівні.

Практична робота №26.



Завдання 1. Види повідомлень

Створіть програму з чотирма кнопками, які викликають різні види **messagebox** повідомлень. Кожна кнопка змінює властивості головного вікна (колір фону, текст заголовка, розміри тощо) або властивості компонентів.

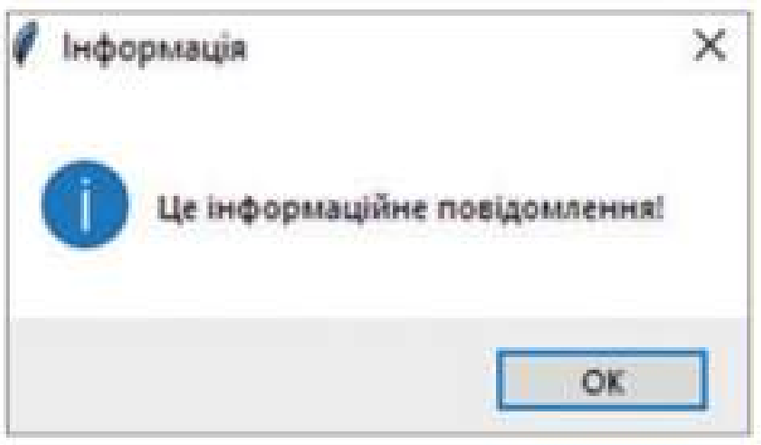
- створити вікно за зразком:

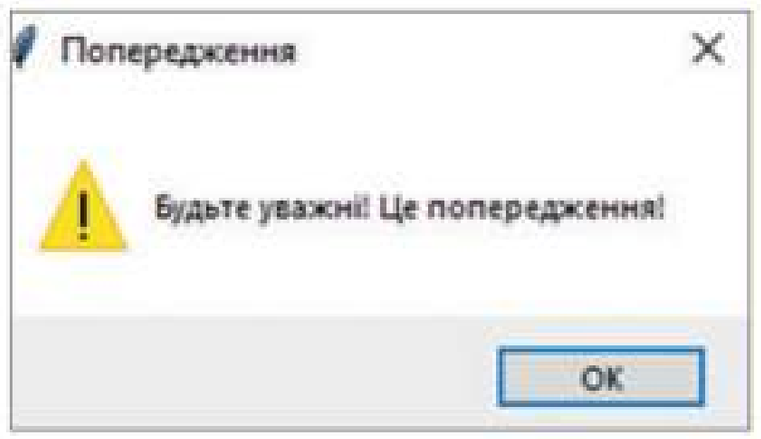
Рекомендовані властивості:

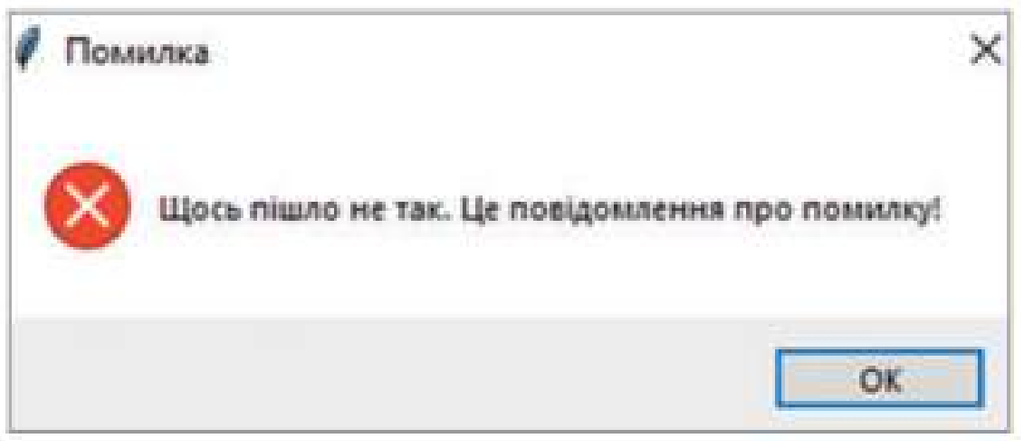
- вікно (розміри вікна `400x300`, колір фону `"#2c3e50"`)
- мітка **label** (шрифт `"Arial"`, `14`; колір фону `"#2c3e50"`, колір тексту `"white"`)
- 1 кнопка **button_info** (шрифт `"Arial"`, `12`; колір фону `"#2c3e50"`, колір тексту `"white"`)
- 2 кнопка **button_warning** (шрифт `"Arial"`, `12`; колір фону `"#f39c12"`, колір тексту `"white"`)
- 3 кнопка **button_error** (шрифт `"Arial"`, `12`; колір фону `"#e74c3c"`, колір тексту `"white"`)
- 4 кнопка **button_confirm** (шрифт `"Arial"`, 12; колір фону `"#3498db"`, колір тексту `"white"`)

Опис функцій:

- кнопка **button_info** викликає функцію `show_info()`, яка змінюватиме колір фону вікна і викликатиме інформативне вікно **showinfo**;
- кнопка **button_warning** викликає функцію `show_warning()`, яка викликатиме попереджувальне вікно **showwarning**;
- кнопка **button_error** викликає функцію `show_error()`, яка викликатиме повідомлення про помилку **showerror**;
- кнопка **button_confirm** викликає функцію `ask_confirmation()`, яка при підтвердженні змінює текст кнопки на `"Продовжено"`, а при скасуванні — змінює текст на `"Скасовано"`.







```
def ask_confirmation():
 response = messagebox.askyesno("Підтвердження", "Ви впевнені,
що хочете продовжити?")
 if response:
        button_confirm.config(text="Продовжено", bg="#4CAF50")
  else:
        button_confirm.config(text="Скасовано", bg="#f44336")
```

Завдання 2. Перевірка віку для доступу

Користувач вводить свій вік. Якщо йому ще не виповнилося 18 років, програма показує повідомлення про обмеження доступу (**messagebox.showerror**), а якщо більше — вітає користувача (**messagebox.showinfo**).

Завдання 3. Запит на вихід

Створіть кнопку «Вихід». Натискання на неї викликає вікно підтвердження дії (**messagebox.askokcancel**). Якщо користувач підтверджує, програма закривається, інакше продовжує працювати.

Завдання 4. Калькулятор квадрату числа

Користувач вводить число **N**. Після натискання кнопки обчислюється квадрат числа, і результат відображається через **messagebox.showinfo**.

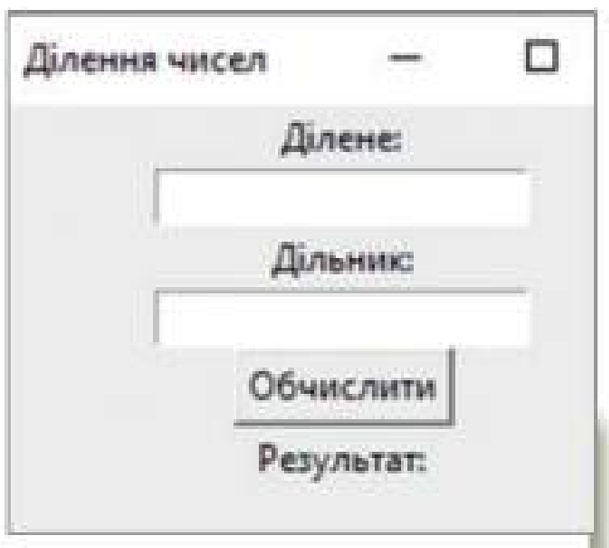
Завдання 5. Попередження про неправильний ввід

Користувач вводить число. Якщо поле залишилося порожнім або введено не число, програма показує **messagebox.showerror** із текстом «Введіть правильне значення».

Завдання 6. Перевірка ділення на нуль

Користувач вводить два числа: ділене та дільник. Потрібно виконати ділення, обробляючи випадок ділення на нуль.

- створити вікно за зразком:
- кнопка викликає функцію `calculate()`, яка отримує дійсні числа **a** та **b** з полів **Entry**, виконує ділення та обробляє випадок ділення на **0** та введення не числових даних в поля.

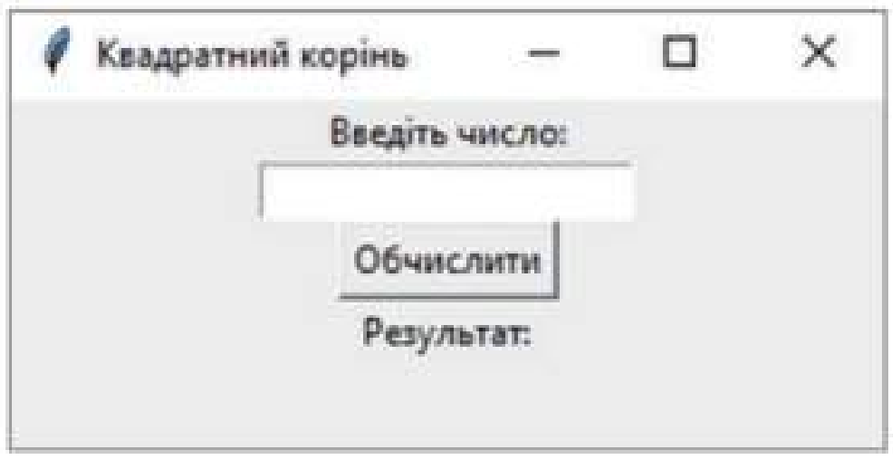



```
def calculate():
    try:
        a = float(a_entry.get())
        b = float(b_entry.get())
        result = a / b
      result_label.config(text=f"Результат: {round(result, 2)}")
    except ZeroDivisionError:
       messagebox.showerror("Помилка", "Ділення на нуль немож-
ливе!")
    except ValueError:
     messagebox.showerror("Помилка", "Введіть коректні числа!")
```

Завдання 7. Обчислення кореня числа

Користувач вводить число, для якого потрібно знайти квадратний корінь. Враховується випадок введення від'ємного числа.

- створити вікно за зразком:



- кнопка викликає функцію `calculate()`, яка отримує дійсне число `number`, знаходить корінь з цього числа (імпортуйте бібліотеку **math**) та обробляє випадок введення від'ємного числа.

```
def calculate():
    try:
        number = float(number_entry.get())
        if number < 0:
            raise ValueError("Від'ємне число!")
        result = sqrt(number)
        result_label.config(text=f"Корінь: {round(result, 2)}")
    except ValueError as e:
        messagebox.showerror("Помилка", f"Неможливо обчислити
корінь: {e}")
```

Завдання 8. Обчислення площі прямокутника

Користувач вводить довжину та ширину прямокутника. Програма знаходить площу прямокутника та обробляє випадок введення недодатних чисел.

Частина коду:

```
if length <= 0 or width <= 0:
    raise ValueError("Довжина та ширина мають бути додатними!")
```



Проєкт 1.
Розв'язування квадратних рівнянь

Теми 38–39

Етап 1. Організаційний

Завдання: Створити програму з графічним інтерфейсом, яка розв'язує квадратні рівняння виду:

$$ax^2 + b\,x + c = 0$$

Користувач буде вводити коефіцієнти a, b і c, а програма буде обчислювати корені рівняння та виводити результат у вікні. Якщо рівняння не має дійсних коренів, програма повинна повідомляти про це.

Етап 2. Підготовчий

Визначення цілі та функціоналу:

1. Створити простий інтерфейс для введення коефіцієнтів a, b, c.

2. Розрахувати корені квадратного рівняння за формулою:

$$D = b^2 - 4ac \qquad x_1 = \frac{-b + \sqrt{D}}{2a} \qquad x_2 = \frac{-b - \sqrt{D}}{2a}$$

3. Вивести результат у вікні.

4. Додати можливість очищення полів введення для нового обчислення.

Етап 3. Проєктний

Структура інтерфейсу:

- Головне вікно розміром 500x350 пікселів, із забороною зміни розмірів вікна та із заголовком **Розв'язування квадратних рівнянь.**
- Поля **Entry** для введення коефіцієнтів a, b і c.
- Кнопка **Розрахувати**, яка обчислює корені.
- Поля **Label** для інформаційного тексту та виведення результату.

Кольори, шрифти та інші властивості елементів вікна визначати самостійно.

Етап 4. Реалізація коду

Створення інтерфейсу

1. Підключення модулів:

- модуль **tkinter** — для створення вікна та змоги працювати з його об'єктами
- функція **messagebox** — для підключення інформативного вікна
- модуль **math** — для використання математичних функцій, а саме квадратний корінь

```
from tkinter import *
from tkinter import messagebox
from math import *
```

2. Створення вікна:

- Задати розміри вікна (“500x350”) та заборонити змінювати розміри
- Задати заголовок вікна **Розв’язування квадратних рівнянь**
- Задати колір фону вікна
- Запустити головний цикл **mainloop()**

```
window = Tk()
window.title(“Розв’язування квадратних рівнянь”)
window.geometry(“500x350”)
window.resizable(False, False)
window.configure(bg=”#f0f0f0”)
(майбутній код)
window.mainloop()
```

3. Створення віджетів

- Створити мітку для заголовку програми **label_title** з такими властивостями: текст: “Розв’язування квадратного рівняння”; шрифт: “Arial”, 16, “bold”; колір фону і колір тексту на особистий смак.

```
label_title = Label(window, text=”Розв’язування квадратного рівняння”,font=(“Arial”, 16, “bold”), bg=”#f0f0f0”, fg=”#333”)
label_title.pack(pady=10)
```

- Створити мітку для підзаголовку-умови програми **label_formula** з такими властивостями: текст: “Введіть коефіцієнти для рівняння:”; шрифт: “Arial”, 12, “bold”; колір фону і колір тексту на особистий смак.

```
label_formula.pack()
```

- Створити фрейм з мітками та полями для введення коефіцієнтів квадратного рівняння, які будуть розміщені в один рядок за допомогою **grid()**




```
frame_formula = Frame(window, bg=”#f0f0f0”)
frame_formula.pack(pady=20)
```

```
entry_a  =  Entry(frame_formula,  width=3,  font=(“Arial”,  12),
justify='center')
label_x2 = Label(frame_formula, text=”x2 +”, font=(“Arial”, 12),
bg=”#f0f0f0”)

entry_b  =  Entry(frame_formula,  width=3,  font=(“Arial”,  12),
justify='center')
label_x = Label(frame_formula, text=”x +”, font=(“Arial”, 12),
bg=”#f0f0f0”)

entry_c  =  Entry(frame_formula,  width=3,  font=(“Arial”,  12),
justify='center')
label_eq = Label(frame_formula, text=”= 0”, font=(“Arial”, 12),
bg=”#f0f0f0”)

entry_a.grid(row=0, column=0, padx=5)
label_x2.grid(row=0, column=1)
entry_b.grid(row=0, column=2, padx=5)
label_x.grid(row=0, column=3)
entry_c.grid(row=0, column=4, padx=5)
label_eq.grid(row=0, column=5)
```

- Створити кнопку для розрахунку **btn_solve** з такими властивостями: текст: "Розрахувати"; шрифт: "Arial", 12; колір фону і колір тексту на особистий смак, ім'я функції: solve.

```
btn_solve = Button(window, text=”Розрахувати”,
                   font=(“Arial”, 12), bg=”#4CAF50”, fg=”white”,
                    command=solve)
btn_solve.pack(pady=10)
```

- Створити кнопку для розрахунку **btn_clear** з такими властивостями: текст: "Очистити"; шрифт: "Arial", 12; колір фону і колір тексту на особистий смак, ім'я функції: **clear_fields**.

```
btn_clear.pack(pady=5)
```

- Створити мітку для результату **result_label** з такими властивостями: текст: " "; шрифт: "Arial", 14, "bold"; колір фону і колір тексту на особистий смак.

```
result_label.pack(pady=30)
```

4. Опис функцій.

- Створити та описати функцію **solve()**, яка матиме перевірки на помилки.

```
def solve():
    try:
(майбутній код)
    except ValueError:
        messagebox.showerror("Помилка", "Будь ласка, введіть
числові значення!")
    except ZeroDivisionError:
         messagebox.showerror("Помилка", "Коефіцієнт 'a' не
може бути нулем!")
```

Ця функція виконує такі дії:

- Зчитує коефіцієнти `a`, `b`, `c` з полів введення

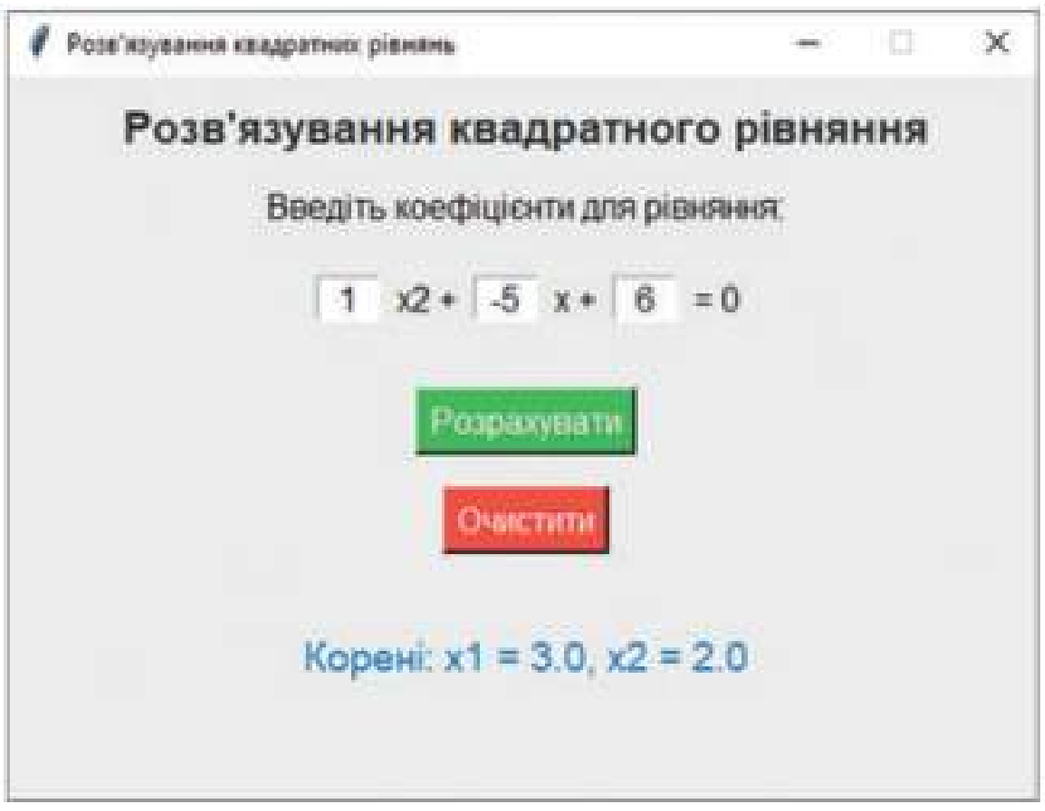
```
a = float(entry_a.get())
```

- Обчислює дискримінант `D`
- Перевіряє різні випадки дискримінанта та виводить результат (з двома знаками після коми)

```
if D > 0:
      x1 = (-b + sqrt(D)) / (2*a)
      x2 = (-b - sqrt(D)) / (2*a)
       result_label.config(text=f"Корені: x1 = {round(x1,2)},
x2 = {round(x2,2)}")
elif D == 0:
      (ваша частина коду)
else:
      (ваша частина коду)
```

- Створити та описати функцію `clear_fields()`, яка очищатиме поля для вводу та мітку результату

```
entry_a.delete(0, END)
```



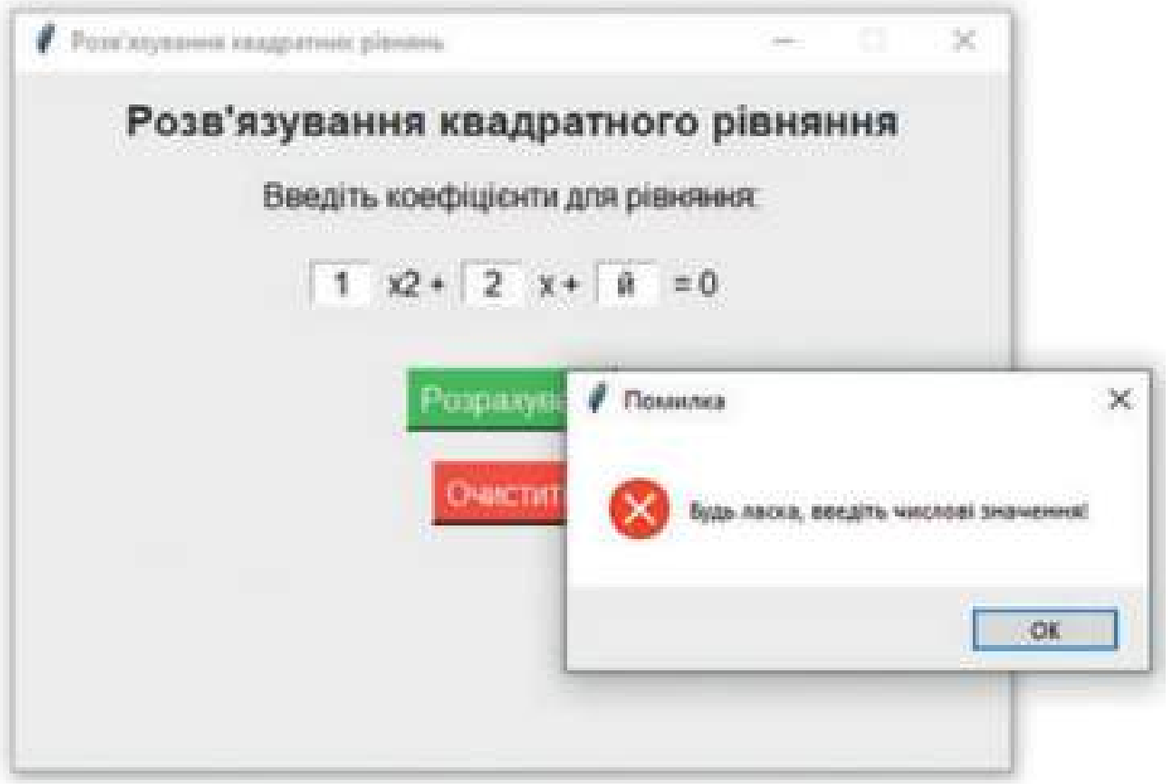
Етап 5. Тестувальний

Перевірте програму:

Введіть різні значення для a, b і c та перевірте результати.

Перевірте обробку помилок (наприклад, якщо $a = 0$ або введено нечислові значення).

Переконайтеся, що функція очищення працює коректно.



Додатково:

Етап 6. Оптимізація та Розширення

- Додайте кнопку для відображення інформації про програму або допомогу.
- Додайте кнопку, яка показуватиме розширене розв'язання і виводитиме значення Дискримінанта.

Етап 7. Документація

Створіть короткий опис функцій програми, коментарі до коду та інструкцію користувача для правильного використання програми.

Інструкція для користувача:

1. Введіть числові значення для коефіцієнтів a, b і c.
2. Натисніть **Розрахувати**, щоб отримати корені рівняння.
3. Натисніть **Очистити**, щоб ввести нові значення.



Проєкт 2.
Визначення індексу маси тіла (BMI)

Теми 40–41

Етап 1. Організаційний

Завдання. Створити програму з графічним інтерфейсом, яка розраховує індекс маси тіла (BMI) користувача на основі введених значень зросту та ваги. Програма має виводити результат у вигляді категорії стану ваги (норма, дефіцит маси, надлишкова вага тощо) та відповідного зображення.

Етап 2. Підготовчий

Цілі програми:

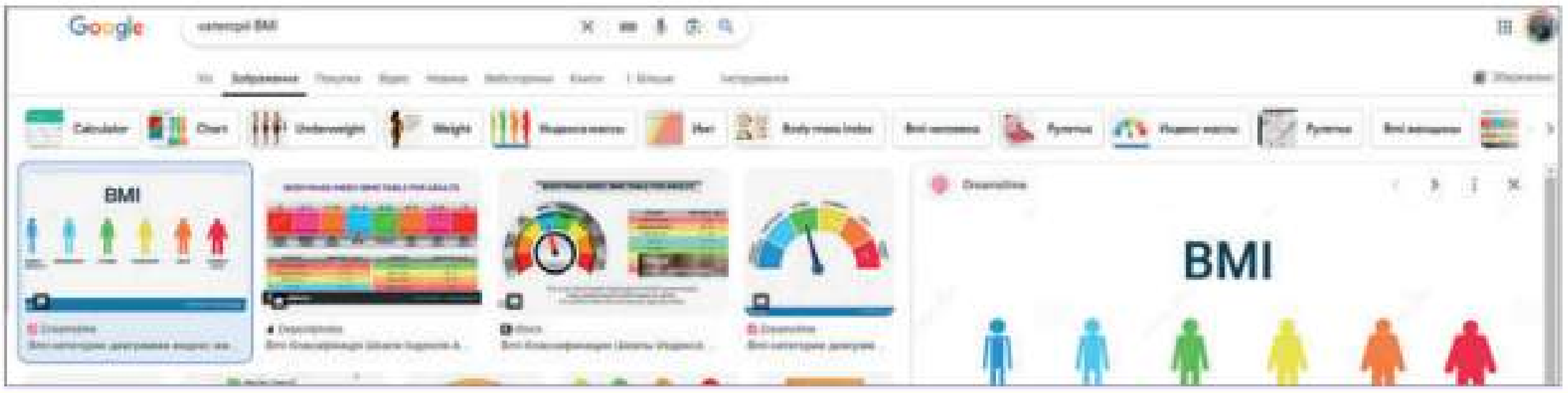
1. Створити простий інтерфейс для введення ваги (кг) і зросту (см).
2. Розрахувати BMI за формулою: $BMI = \frac{\text{вага (кг)}}{(\text{зріст (м)})^2}$ та визначити категорію:
   - Менш ніж 18.5 — дефіцит маси.
   - 18.5–24.9 — норма.
   - 25–29.9 — надлишок ваги.
   - 30–34.9 — ожиріння.
   - Понад 35 — ожиріння II ступеня.
3. Вивести результат із відповідною категорією стану ваги.
4. Показати тематичне зображення залежно від результату.

Функціонал:

- Поля введення зросту та ваги.
- Кнопка **Визначити** для запуску розрахунку.
- Виведення тексту та зображення з результатом.

Збереження зображень до проєкту:

в інтернеті знайти тематичне зображення до кожної категорії, для цього в пошуковому рядку ввести ключові слова на кшталт "`категорії ВМІ`", віднайти потрібне зображення або по окремій категорії чи групове



Якщо обрано групове зображення, то за допомогою скриншотів обрізати частини зображення. Саме таким способом ми отримаємо частинки зображення у форматі **png**. Одне із зображень має бути не із категорії, для початкового відображення у вікні програми.

збережіть дані зображення у папці з проєктом і переназвіть відповідно



Етап 3. Проєктний

Структура інтерфейсу:

1. Головне вікно:
   - Розмір: 500x600 пікселів.
   - Фіксований розмір вікна, щоб уникнути масштабування.
   - Заголовок: "BMI INDEX".
   - Фон: світло-рожевий (#FFCCCC).
2. Основні елементи:
   - Заголовок програми (на панелі).
   - Поля для введення зросту та ваги.
   - Кнопка для запуску розрахунку.
   - Поле для виводу результату.
   - Динамічне зображення, що змінюється залежно від результату.

Етап 4. Реалізація коду

Створення інтерфейсу

1. Підключення модулів.

Модуль **tkinter** — для створення вікна та змоги працювати з його об'єктами

2. Створення вікна:
   - Задати розміри вікна ("500x600") з відступами 100 та 200 зверху та зліва відповідно та заборонити змінювати розміри
   - Задати заголовок вікна "BMI INDEX"
   - Задати колір фону вікна
   - Запустити головний цикл mainloop()

3. Створення віджетів

Створити мітку для заголовку програми **label_title** з такими властивостями: текст: "ДІЗНАЙСЯ СВІЙ ІНДЕКС МАСИ ТІЛА"; шрифт: "Arial Black", 14, "bold"; колір фону "#584332" і колір тексту "#ffffff".

```
label_title.pack(pady=30)
```

- Створити мітку **label_height** з такими властивостями: текст: "Введи свій зріст:"; шрифт: 'Cambria', 18, 'bold'; колір фону "#FFCCCC" і колір тексту fg="#584332".

```
label_height.place(x=30, y=100)
```

- Створити поле **entry_height** для введення зросту з такими властивостями: width=10, шрифт: 'Cambria', 18, 'bold'.

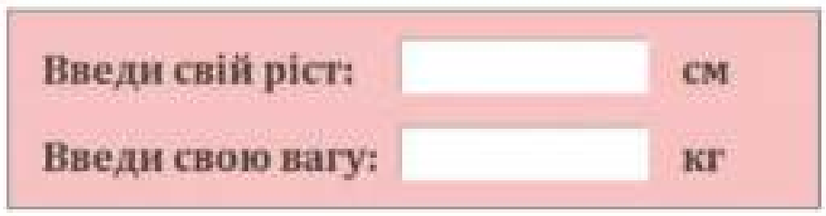
```
entry_height.place(x=240, y=100)
```

- Створити мітку **sm** .

```
sm.place(x=400, y=100)
```

- Аналогічно створити мітку **label_weight**, поле entry_height та мітку kg для введення ваги.

```
label_weight.place(x=30, y=150)
...
entry_weight.place(x=240, y=150)
...
kg.place(x=400, y=150)
```



- Імпортувати зображення та створити мітку для їх відображення у вікні програми.

```
img = PhotoImage(file="img.png")
img = img.subsample(2,2)
img1 = PhotoImage(file="img1.png")
img1 = img1.subsample(2,2)
img2 = PhotoImage(file="img2.png")
img2 = img2.subsample(2,2)
img3 = PhotoImage(file="img3.png")
img3 = img3.subsample(2,2)
img4 = PhotoImage(file="img4.png")
img4 = img4.subsample(2,2)
img5 = PhotoImage(file="img5.png")
img5 = img5.subsample(2,2)

label_img = Label(window, image = img)
label_img.place(x = 210, y = 230)
```

- Створити мітку **label_rezult** з такими властивостями: текст: " "; шрифт: 'Cambria', 24, 'bold', колір фону "#FFCCCC".

```
label_rezult.place(x = 250, y = 430, anchor = CENTER)
```

Створити кнопку **btn_rezult** з такими властивостями: текст: “ Визначити”; шрифт: ‘Arial black’, 14; колір фону “#323030” і колір тексту fg=”white”; command = calculate.

```
btn_rezult.place(x = 180, y = 480)
```



4. Опис функції

Створити та описати функцію **calculate()**, яка матиме перевірки на помилки

```
def calculate():
 try:
(майбутній код)
    except ValueError:
        label_rezult.config(text=”Помилка вводу!”)
```

Ця функція виконує такі дії:

- Зчитує дійсні числа — зріст та вагу з полів у змінні h і m відповідно
- переводить зріст із сантиметрів у метри, переприсвоюючи змінну h

```
h = h / 100
```

- обчислює BMI

```
BMI = m / (h ** 2)
```

- Перевіряє різні значення BMI та визначає категорії

```
if BMI < 18.5:
      label_rezult.config(text=”ДЕФІЦИТ МАСИ”, fg=”#08E1F0”)
           label_img.config(image = img1)
```

Етап 5. Тестувальний

Перевірте результат для кожної категорії BMI.
Введіть нечислові значення, щоб перевірити обробку помилок.
Додатково:

Етап 6. Оптимізація та Розширення

Додати кнопку для очищення полів введення.
Виводити значення самого BMI.

Етап 7. Документація

Створіть короткий опис функції програми та коментарі до коду.

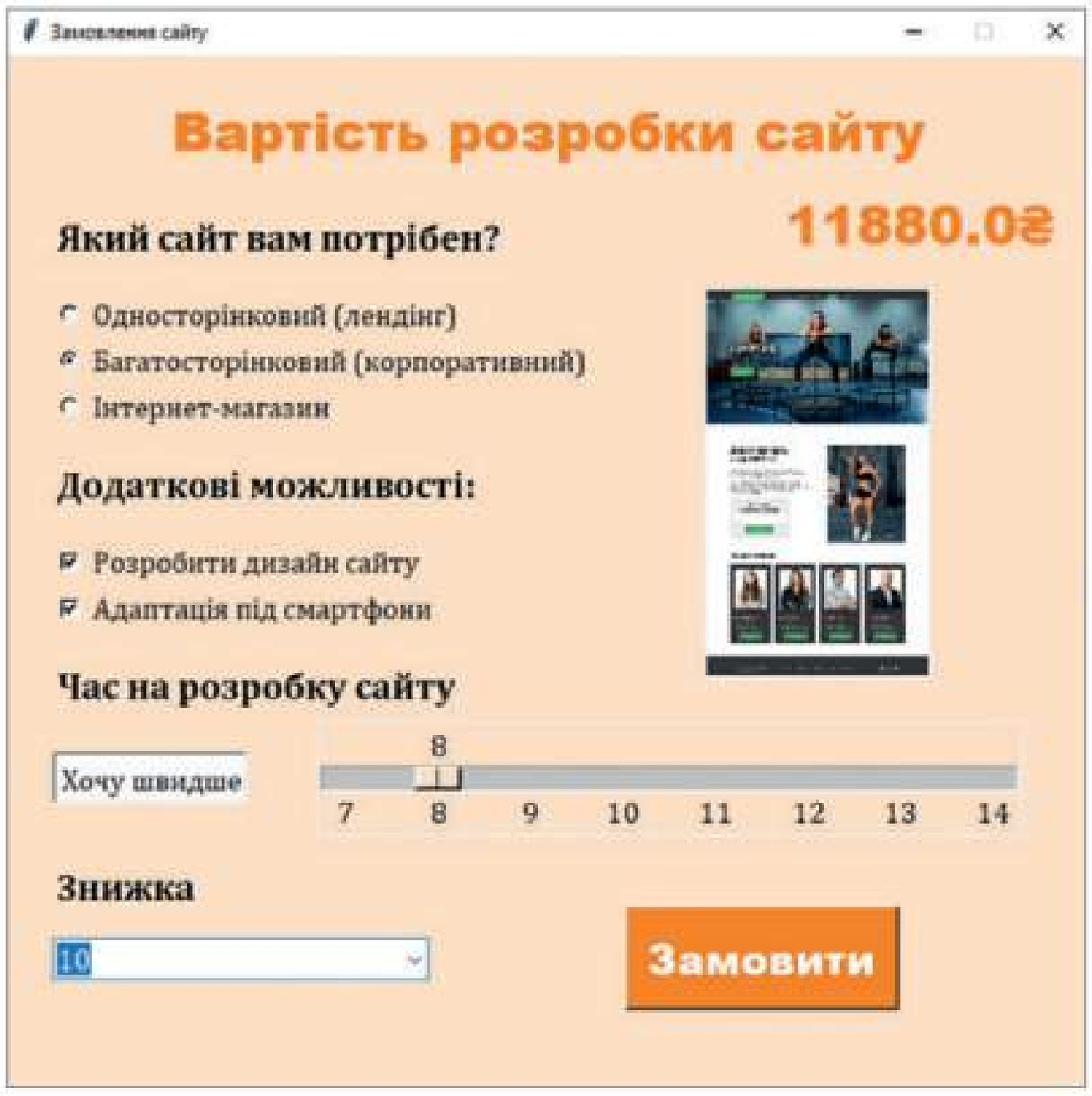


Проєкт 3.
Розрахунок вартості розробки сайту

Теми 42–43

Етап 1. Організаційний

Завдання: Розробити програму з графічним інтерфейсом для розрахунку вартості розробки сайту залежно від типу сайту, додаткових послуг, терміну виконання та знижки. Програма повинна також відображати відповідне зображення для вибраного типу сайту.



Мета програми:

1. Дати користувачеві можливість вибрати:
   - Тип сайту.
   - Додаткові можливості (розробка дизайну, адаптація під смартфони).
   - Термін виконання роботи.
   - Знижку.
2. Розрахувати вартість послуг залежно від вибору.
3. Вивести підсумкову вартість та відобразити відповідне зображення.

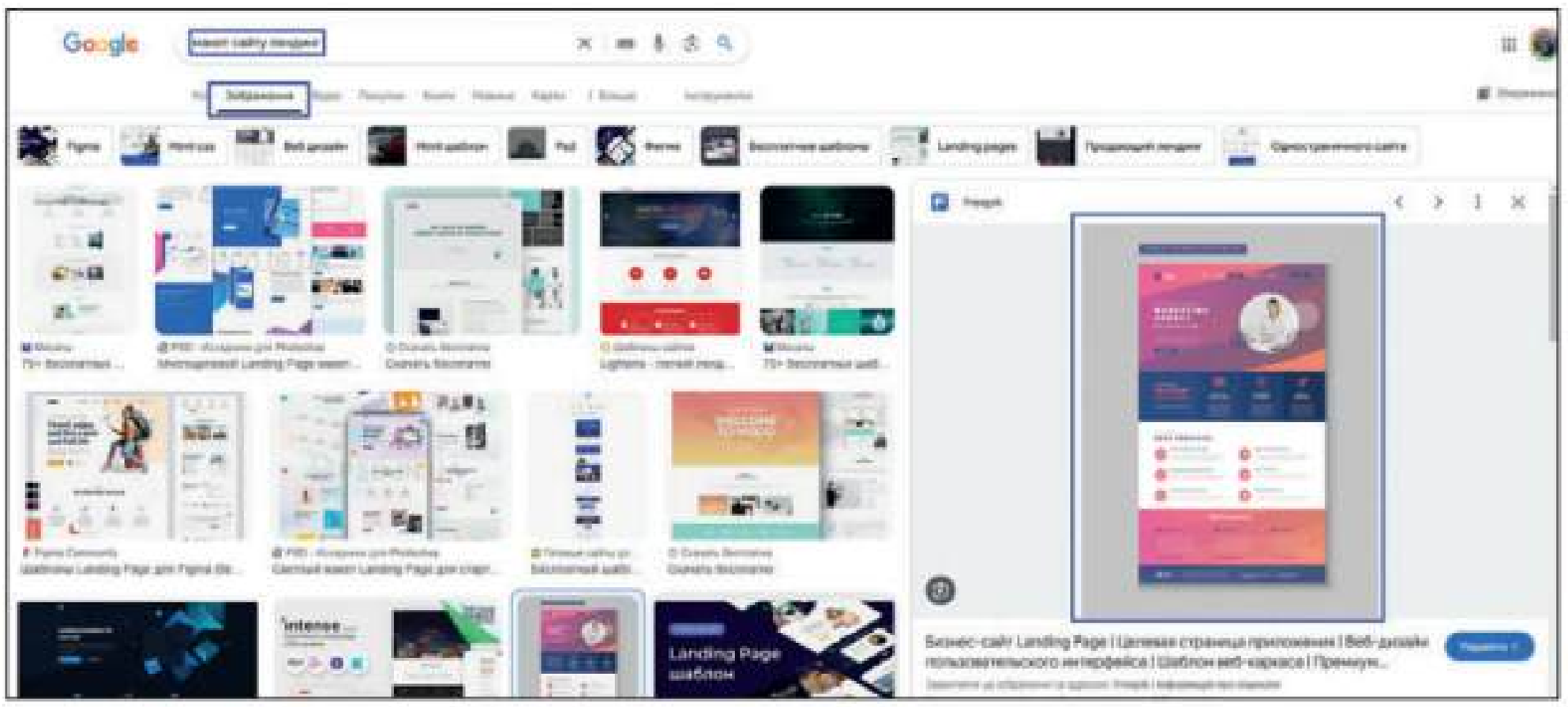
Етап 2. Підготовчий

Цілі програми:

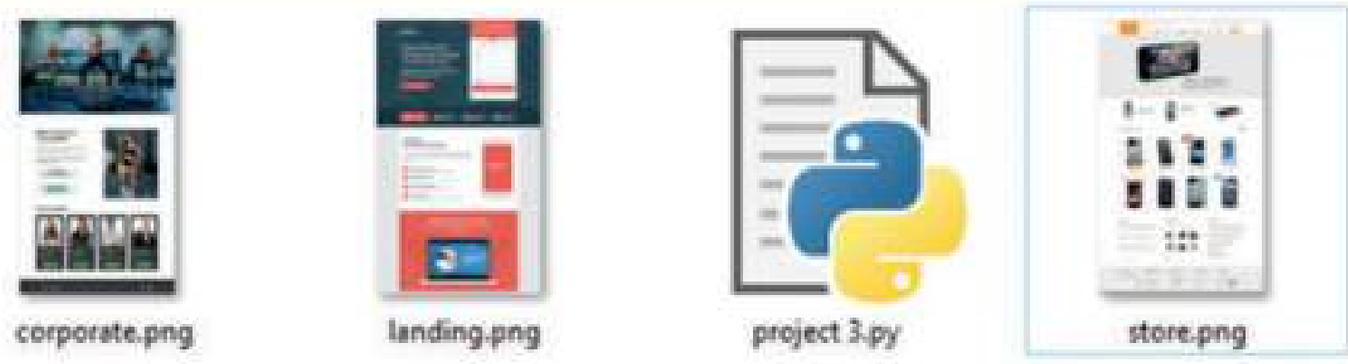
1. Тип сайту (обирається через **Radiobutton**):
   - Односторінковий (лендінг): 4000 грн.
   - Багатосторінковий (корпоративний сайт): 8000 грн.
   - Інтернет-магазин: 26000 грн.
2. Додаткові можливості (через **Checkbox**):
   - Розробка дизайну: додає 50% до вартості.
   - Адаптація під смартфони: додає 10% до вартості.
3. Термін розробки (через шкалу-слайдер). За бажанням користувача можна включити параметр **Хочу швидше**.
4. Знижка (через **Combobox**). Вибір значень: 0%, 10%, 20%, 30%.

Функціонал:

1. Розрахунок вартості: враховуються всі вибрані опції, знижка та термін виконання.
2. Відображення результату:
   - Загальна сума в гривнях.
   - Зображення відповідного типу сайту.
3. Динамічне оновлення: вартість і зображення змінюються одразу після вибору параметрів.
4. Збереження зображень до проєкту: в інтернеті знайти тематичне зображення до кожного типу сайту, для цього в пошуковому рядку ввести ключові слова на кшталт **макет односторінкового сайту**, **макет багатосторінкового сайту**, **макет сайту-магазину**, віднайти потрібні зображення та зберегти їх у папці з проєктом і перейменувати їх відповідно на **landing.png**, **corporate.png**, **store.png**.



Зауваження! Візьміть до уваги, що розширення має бути **png** або **gif**. Для зручності — зберігайте зображення, роблячи скриншоти потрібного зображення.



Етап 3. Проєктний

Структура інтерфейсу:

Головне вікно:

- Розмір: 700x670 пікселів.
- Фіксований розмір: заборонено змінювати розміри вікна.
- Фон: світло-помаранчевий (`#FFE5CC`).
- Заголовок вікна: `"Замовлення сайту"`.

Основні елементи:

1. Мітки (**Label**):
   - заголовок програми;
   - текст для кожного блоку (вибір типу сайту, додаткові можливості, термін розробки, знижка, результат).
2. **Radiobutton** (перемикачі): вибір типу сайту (лендінг, корпоративний сайт, інтернет-магазин).
3. **Checkbox** (прапорці):

Додаткові послуги:

- розробка дизайну;
- адаптація під смартфони.

4. **Scale** (шкала): вибір терміну виконання роботи (7-14 днів).
5. **Combobox** (спливаючий список): вибір знижки (0%, 10%, 20%, 30%).
6. Кнопки:
   - опція для вибору `"Хочу швидше"`;
   - обчислення підсумкової вартості.
7. Зображення (**Label** + **PhotoImage**): відображення картинки залежно від типу сайту.

Етап 4. Реалізація коду

Створення інтерфейсу

1. Підключення модулів:
   - модуль **tkinter** для створення графічного інтерфейсу;
   - модуль `ttk` для використання **Combobox**.

```
from tkinter import *
from tkinter.ttk import Combobox
```

2. Створення головного вікна:
1. Задання розмірів вікна (“700x670”)
2. Колір фону: світло-помаранчевий (#FFE5CC).
3. Фіксація розмірів (вимкнення масштабування).
4. Встановлення заголовка: “Замовлення сайту”.
5. Запустити головний цикл `mainloop()`

3. Створення віджетів.

- Створити мітку для заголовку програми **label_title** з такими властивостями: текст: “Вартість розробки сайту”; шрифт: “Arial Black”, 26; колір фону “#FFE5CC” і колір тексту “#FF8000”.

```
label_title.pack(pady=20)
```

- Створити мітку **label_rezult** з такими властивостями: текст: “0 ₴”; шрифт: “Arial Black”, 26; колір фону “#FFE5CC” і колір тексту “#FF8000”.

```
label_result.place(x=500, y=80)
```

- Створити мітку **label_type** з такими властивостями: текст: “Який сайт вам потрібен?”; шрифт: ‘Cambria’, 18, ‘bold’; колір фону “#FFE5CC” і колір тексту “#000000”.

```
label_type.place(x=30, y=100)
```

- Створити три радіокнопки для вибору типу сайту з такими властивостями: текст кожної з них: “Односторінковий (лендінг)”, “Багатосторінковий (корпоративний)”, “Інтернет-магазин”; шрифт: ‘Cambria’, 14; колір фону “#FFE5CC”; прив’язка до змінної `site_type`, і значення `value` відповідно 1, 2 та 3; виклик функції `command = check_type`.

```
site_type = IntVar(value=1)
site1 = Radiobutton(window, text="Односторінковий (лендінг)",
                    bg="#FFE5CC", variable=site_type, value=1,
                    font=('Cambria', 14), command=check_type)
site1.place(x=30, y=150)
...
site2.place(x=30, y=180)
...
site3.place(x=30, y=210)
```

- Створити мітку **label_plus** з такими властивостями: текст: “Додаткові можливості:”; шрифт: ‘Cambria’, 18, ‘bold’; колір фону “#FFE5CC” і колір тексту “#000000”.

```
label_plus.place(x=30, y=260)
```

- Створити два чекбокси для вибору додаткових можливостей з такими властивостями: текст кожного з них: "Розробити дизайн сайту", "Адаптація під смартфони"; **шрифт**: 'Cambria', 14; **колір фону** "#FFE5CC"; **прив'язка до змінних** design_var і mobile_var **відповідно.**

```
design_var = IntVar()
mobile_var = IntVar()

design = Checkbutton(window, text="Розробити дизайн сайту",
                             bg="#FFE5CC", variable=design_var,
font=('Cambria', 14))
design.place(x=30, y=310)

...

mobile.place(x=30, y=340)
```

- Створити мітку **label_time** з такими властивостями: текст: "Час на розробку сайту"; **шрифт**: 'Cambria', 18, 'bold'; **колір фону** "#FFE5CC" і **колір тексту** "#000000".

```
label_time.place(x=30, y=390)
```

- Створити чекбокс для активації шкали часу розробки з такими властивостями: текст: "Хочу швидше"; **шрифт**: 'Cambria', 14; **колір фону** "#FFE5CC"; **прив'язка до змінної** speed_var; onvalue = 1, offvalue = 0, indicatoron = 0 **(для зміни зовнішнього вигляду); виклик функції** command = check_scale.

```
speed_var = IntVar()
speed = Checkbutton(window, text = "Хочу швидше",
bg="#FFE5CC",variable=speed_var, onvalue = 1, offvalue = 0,
indicatoron = 0,font=('Cambria', 14), command = check_scale)
speed.place(x=30, y=450)
```

- Створити смугу прокручування **scale** з такими властивостями: орієнтація HORIZONTAL, довжина 450, показники з 14 до 7, інтервал шкали 1, **стан неактивний** state=DISABLED; **шрифт**: 'Cambria', 18, 'bold'; **колір фону** "#FFE5CC".

```
scale = Scale(window, orient=HORIZONTAL, length=450, from_=14,
to=7,tickinterval=1, bg="#FFE5CC", font=('Cambria', 14),
state=DISABLED)
scale.place(x=200, y=430)
```

- Створити мітку **label_discount** з такими властивостями: текст: "Знижка"; шрифт: **'Cambria'**, 18, 'bold'; колір фону "#FFE5CC" і колір тексту "#000000".

```
label_discount.place(x=30, y=520)
```

- Створити **combobox** для вибору знижки із спливаючого списку

```
combo = Combobox(window, values=[0, 10, 20, 30], font=('Cambria', 14))
combo.current(0)
combo.place(x=30, y=570)
```

- Створити кнопку **btn_rezult** з такими властивостями: текст: "Замовити"; шрифт: 'Arial black', 20; колір фону "#FF8000 і колір тексту fg="#FFFFFF"; виклик функції command = calculate.

```
btn_result.place(x=400, y=550)
```

- Імпортувати зображення та створити мітку для їх відображення у вікні програми.

```
landing = PhotoImage(file='landing.png')
landing = landing.subsample(3, 3)
corporate = PhotoImage(file='corporate.png')
corporate = corporate.subsample(3, 3)
store = PhotoImage(file='store.png')
store = store.subsample(3, 3)

label_img = Label(window, image=landing)
label_img.place(x=450, y=150)
```

4. Опис функцій

- Описати функцію **check_type()**, яка змінюватиме зображення при виборі тієї чи іншої радіокнопки

```
def check_type():
    if site_type.get() == 1:
        label_img.config(image=landing)
    elif site_type.get() == 2:
        label_img.config(image=corporate)
    elif site_type.get() == 3:
        label_img.config(image=store)
```

- Описати функцію **check_scale()**, яка вмикає або вимикає шкалу залежно від вибору

```
def check_scale():
    if speed_var.get() == 1:
        scale.config(state=NORMAL)
    else:
        scale.config(state=DISABLED)
```

- Описати функцію **calculate()**, яка
  - створює змінну **sum** і надає їй початкового значення **0**;
  - визначає базову вартість сайту залежно від вибору типу сайту.

```
def calculate():
    sum = 0


    if site_type.get() == 1:
        sum += 4000
...продовження умов
```

  - додає вартість додаткових послуг

```
if design_var.get() == 1:
    sum += sum * 0.5  # Дизайн додає 50%
if mobile_var.get() == 1:
   ...ваш рядок коду  # Адаптація додає 10%
```

  - застосовує знижку

```
discount = int(combo.get())
sum -= sum * (discount / 100)
```

  - оновлює текст з вартістю

```
label_result.config(text=f"{round(sum, 2)} ₴")
```

Етап 5. Тестувальний

- **Перевірка обчислення:**
  - Розрахунок для кожного типу сайту.
  - Додавання/вимкнення додаткових послуг.
  - Вибір різних значень знижки.
- Перевірка відображення зображення. Для кожного типу сайту відображається відповідна картинка.
- Перевірка функціоналу шкали. Шкала активується лише при виборі опції **Хочу швидше**.

Додатково:

Етап 6. Оптимізація та Розширення

Розширити можливості вибору: додати більше типів сайтів та додаткових послуг.

Етап 7. Документація

Створіть короткий опис функцій програми та коментарі до коду.



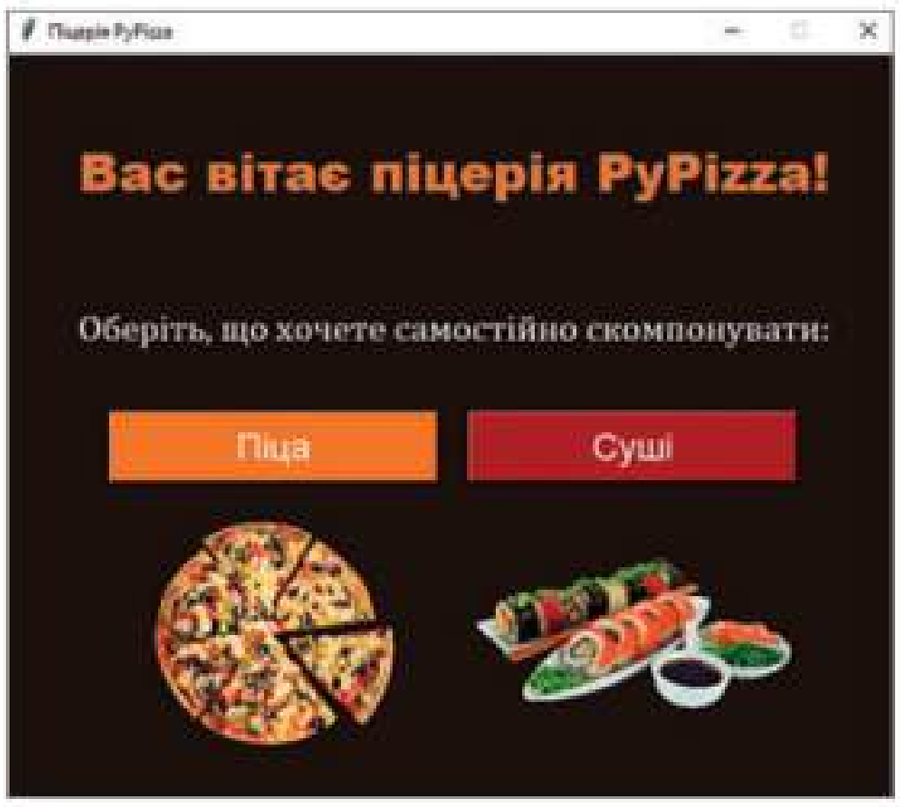
Проєкт 4.
Додаток для замовлення PyPizza

Теми 44–45

Етап 1. Організаційний

Завдання: Розробити програму з графічним інтерфейсом та кількома вікнами, яка дозволяє користувачеві:

1. Перейти зі стартового вікна до вікна створення піци або суші.
2. У вікні створення піци вибрати основу, компоненти та напій.
3. Розрахувати загальну вартість замовлення піци.
4. Оформити замовлення, відправивши свої контактні дані.



Мета програми:

1. Реалізувати три вікна:
   - Стартове вікно з вибором піци або суші.
   - Вікно для вибору складових піци (або суші).
   - Вікно оформлення замовлення з введенням контактних даних.
2. Забезпечити зручний інтерфейс для користувача.
3. Забезпечити динамічний розрахунок загальної вартості замовлення.
4. Реалізувати можливість вводу контактних даних користувача для завершення замовлення.

Етап 2. Підготовчий

Цілі програми:

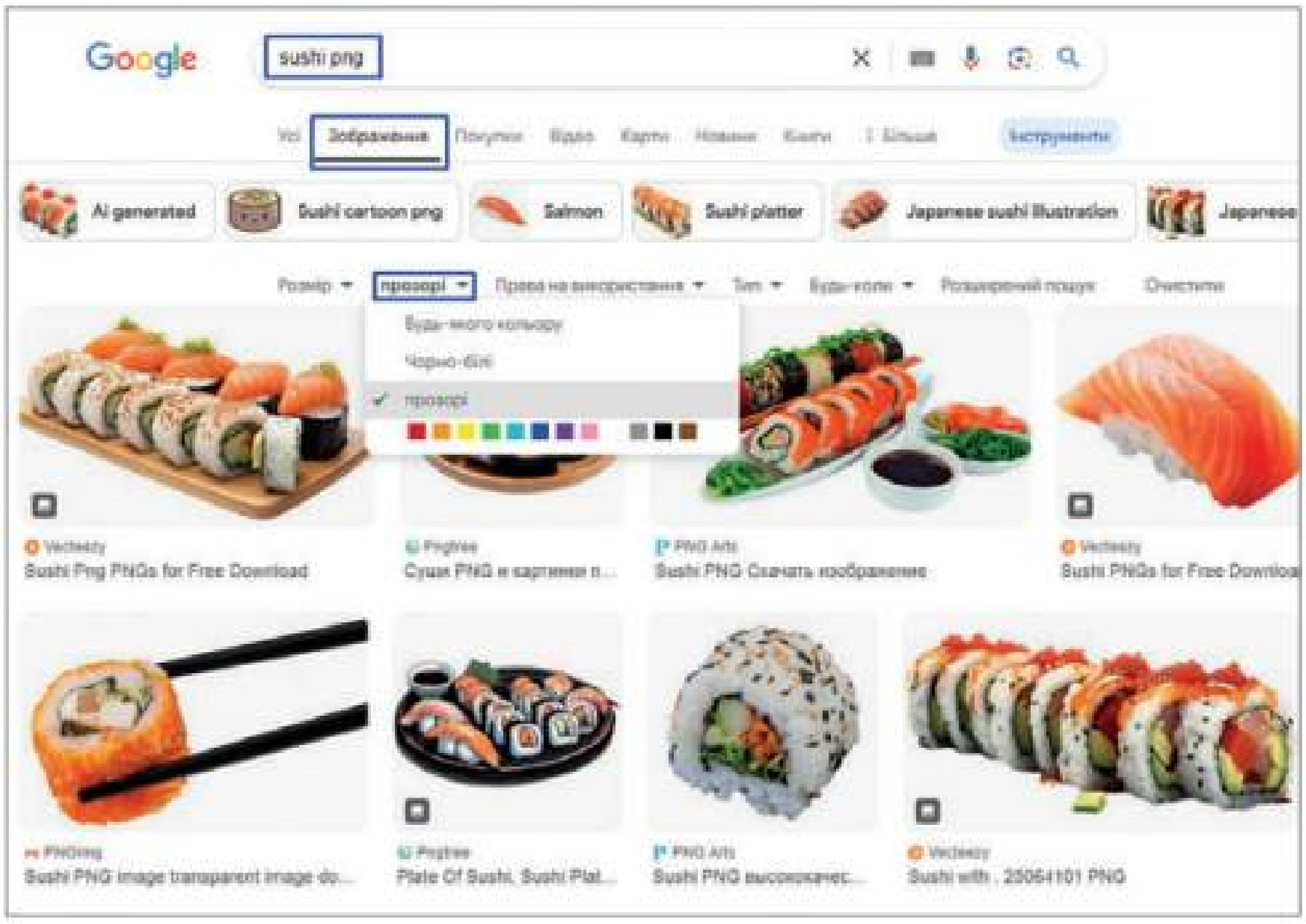
1. Стартове вікно:
   - Привітання.
   - Дві кнопки для переходу: **Піца** та **Суші**.

2. Вікно для створення піци:
- Основа для піци:
  - Тонка основа (100 грн).
  - Класична основа (120 грн).
  - Сирна основа (150 грн).
- Вміст піци:
  - Сир (+20 грн).
  - М'ясо (+30 грн).
  - Овочі (+15 грн).
- Напої:
  - Кока-кола (+15 грн).
  - Сік (+25 грн).
  - Мінеральна вода (+20 грн).
- Результат: виведення загальної вартості замовлення.

3. Вікно оформлення замовлення:
- Поля введення імені та телефону.
- Кнопка підтвердження з повідомленням про успішне замовлення.

Збереження зображень до проєкту

В інтернеті знайти тематичні зображення піци та суші, для цього в пошуковому рядку ввести ключові слова на кшталт "`pizza png`", "`sushi png`", віднайти потрібні зображення та зберегти їх у папці з проєктом і перейменувати зручним іменем.



Зауваження! Візьміть до уваги, що розширення має бути **png** або **gif.** Для зручності — зберігайте зображення, роблячи скриншоти потрібного зображення, а також з прозорим фоном.



Етап 3. Проєктний

Структура інтерфейсу:

1. Стартове вікно:
- Розмір: 600x500 пікселів.
- Колір фону: темний (#261b14).
- Елементи:
  - Мітка привітання та мітка інформування вибору.
  - Кнопки переходу до вікна піци або суші.

2. Вікно для створення піци:
- Розмір: 900x700 пікселів.
- Колір фону: темний.
- Елементи:
  - Зображення піци (ліворуч).
  - Елементи вибору основи, компонентів і напоїв (праворуч).
  - Кнопки:
    - **Розрахувати** (підсумок).
    - **Оформити замовлення** (перехід до вікна з даними).
    - **Назад** (повернення до стартового вікна).

3. Вікно оформлення замовлення:
- Розмір: 500x300 пікселів.
- Колір фону: темний.
- Елементи:
  - Поля введення імені та телефону.
  - Кнопка підтвердження.

Етап 4. Реалізація коду

Створення інтерфейсу

Підключення модулів:

- модуль **tkinter** для створення графічного інтерфейсу та роботи з основними віджетами;
- модуль **messagebox** для відображення інформативних повідомлень у діалогових вікнах.

```
from tkinter import *
from tkinter import messagebox
```

- Створення стартового вікна **start**:
  1. Задання розмірів вікна (“600x400”)
  2. Колір фону: темний (#261b14).
  3. Фіксація розмірів (вимкнення масштабування).
  4. Встановлення заголовка: “Піцерія PyPizza”.
  5. Запустити головний цикл mainloop()
- Створити мітку для привітання **label_greeting** з такими властивостями: текст: “Вас вітає піцерія PyPizza!”; шрифт: “Arial Black”, 26; колір фону “#261b14” і колір тексту “#de7439”.

```
label_greeting.pack(pady=50)
```

- Створити мітку для інформації вибору **label_choose** з такими властивостями: текст: “Оберіть, що хочете самостійно скомпонувати:”; шрифт: “Cambria”, 18; колір фону “#261b14” і колір тексту “#ffffff”.

```
label_choose.pack(pady=10)
```

- Створити фрейм для розміщення кнопок та зображення за допомогою методу **grid**

```
button_frame = Frame(start, bg=”#261b14”)
button_frame.pack(pady=20)
```

- Створити кнопку **btn_pizza** з такими властивостями: батьківський елемент button_frame, текст: “Піца”; шрифт: ‘Arial black’, 16; колір фону “#f76e08” і колір тексту fg=”#ffffff”; width=15; виклик функції command = open_pizza_window.

```
btn_pizza.grid(row=0, column=0, padx=10, pady=10)
```

- Створити кнопку **btn_sushi** з такими властивостями: батьківський елемент button_frame, текст: “Суші”; шрифт: ‘Arial black’, 16; колір фону “#af1618” і колір тексту fg=”#ffffff”; width=15; виклик функції command = open_sushi_window.

```
button_sushi.grid(row=0, column=1, padx=10, pady=10)
```

- Імпортувати зображення, створити їм відповідні мітки для їх відображення у вікні програми, масштабувати та розмістити під кнопками за допомогою **grid**.

```
pizza_img = PhotoImage(file=”pizza.png”)
pizza_img = pizza_img.subsample(3, 3)
label_pizza_img = Label(button_frame, image=pizza_img,
bg=”#261b14”)
label_pizza_img.grid(row=1, column=0, padx=10, pady=10)
```

```
sushi_img = PhotoImage(file="sushi.png")
sushi_img = sushi_img.subsample(5, 5)
label_sushi_img = Label(button_frame, image=sushi_img,
bg="#261b14")
label_sushi_img.grid(row=1, column=1, padx=10, pady=10)
```

Зауваження! Масштабування зображення встановлюєте самостійно, залежно від розмірів вашого зображення.

- Описати функцію **open_pizza_window()**, яка закриватиме стартове вікно та викликатиме наступну функцію **main_pizza_window**, яка створюватиме вікно із замовленням піци

```
def open_pizza_window():
        start.destroy() - закриває стартове вікно
        main_pizza_window() - виклик функції
```

- Описати функцію **main_pizza_window()**, яка створює вікно із замовленням піци.

Зауваження! Наступний код описуватиметься в цій функції. Функцію зручно розмістити не на початку вашого всього коду, як робили раніше, а під кодом, де описувався фрейм і його компоненти. Адже це функції нового вікна, і щоб не плутатися між кодом попередніх функцій.

- Створення вікна **pizza**:
  1. Задання розмірів вікна ("600x400")
  2. Колір фону: темний (#261b14).
  3. Фіксація розмірів (вимкнення масштабування).
  4. Встановлення заголовка: "Замовлення піци".
  5. Запустити головний цикл **mainloop()**

```
def main_pizza_window():
    pizza = Tk()
    pizza.title("Замовлення піци")
    ...інші властивості вікна
    ... майбутній код

pizza.mainloop()
```

- Створити мітку для заголовку **pizza_title** з такими властивостями: текст: "Скомпонуйте свою піцу!"; шрифт: "Arial Black", 24; колір фону "#261b14" і колір тексту "#ffa500".

```
pizza_title = Label(pizza, text="Скомпонуйте свою піцу!",
                    font=("Arial Black", 24), bg="#2b2b2b",
fg="#ffa500")
 pizza_title.pack(pady=20)
```

- Імпортувати зображення, масштабувати та розмістити зліва вікна.

```
pizza_img = PhotoImage(file="pizza1.png")
pizza_img = pizza_img.subsample(1, 1)
label_img = Label(pizza, image=pizza_img, bg="#261b14")
label_img.place(x=50, y=150)
```

Зауваження! Масштабування зображення `subsample(1, 1)` та розміщення `place(x=50, y=150)` встановлюєте самостійно, залежно від розмірів вашого зображення.

- Створити мітку для вибору основи **base_label** з такими властивостями: текст: `"Обирайте основу:"`; шрифт: `"Cambria"`, `16`, `"bold"`; колір фону `"#261b14"` і колір тексту `"#cf7374"`.

```
base_label.place(x=550, y=80)
```

- Створити три радіокнопки для вибору основи з такими властивостями: текст кожної з них: `"Тонка основа (100 грн)"`, `"Класична основа (120 грн)"`, `"Сирна основа (150 грн)"`; шрифт: `'Cambria'`, `14`; колір фону `"#261b14"`, колір тексту `fg="#ffffff"`, колір селекторів `"#4e4e4e"`; прив'язка до змінної **base_var**, і значення `value` відповідно `1`, `2` та `3`.

```
base_var = IntVar(value=1)

    base1 = Radiobutton(pizza, text="Тонка основа (100
грн)", variable=base_var, value=1, font=("Cambria",
14), bg="#261b14",fg="#ffffff", selectcolor="#4e4e4e")
    base1.place(x=500, y=120)
...
    base2.place(x=500, y=150)
...
    base3.place(x=500, y=180)
```

- Створити мітку для вибору вмісту **contents** з такими властивостями: текст: `"Виберіть вміст:"`; шрифт: `"Cambria"`, `16`, `"bold"`; колір фону `"#261b14"` і колір тексту `"#cf7374"`.

```
contents.place(x=550, y=230)
```

- Створити три чекбокси для вибору вмісту піци з такими властивостями: текст кожного з них: `"Сир (+20 грн)"`, `"М'ясо (+30 грн)"`, `"Овочі (+15 грн)"`; шрифт: `'Cambria'`, `14`; колір фону `"#261b14"`, колір тексту `fg="#ffffff"`, колір селекторів `"#4e4e4e"`; прив'язка до змінних `cheese_var`, `meat_var` і `veggies_var` відповідно.

```
cheese_var = IntVar()
meat_var = IntVar()
veggies_var = IntVar()

cheese = Checkbutton(pizza, text="Сир (+20 грн)",
variable=cheese_var, font=("Cambria", 14), bg="#261b14", fg="
#ffffff",selectcolor="#4e4e4e")
cheese.place(x=500, y=270)

...
meat.place(x=500, y=300)

...
veggies.place(x=500, y=330)
```

- **Створити мітку для вибору напою** `drinks` **з такими властивостями: текст:** `"Напої:"`**; шрифт:** `"Cambria"`, `16`, `"bold"`**; колір фону** `"#261b14"` **і колір тексту** `"#cf7374"`.

```
drinks.place(x=550, y=380)
```

- **Створити три радіокнопки для вибору напою з такими властивостями: текст кожної з них:** `"Кока-кола (+15 грн)"`, `"Сік (+25 грн)"`, `"Мінеральна вода (+20 грн)"`**; шрифт:** `'Cambria'`, `14`**; колір фону** `"#261b14"`**, колір тексту** `fg="#ffffff"`**, колір селекторів** `"#4e4e4e"`**; прив'язка до змінної** `drink_var`**, і значення** `value` **відповідно** `1`, `2` **та** `3`.

```
drink_var = IntVar()

...
drink1.place(x=500, y=420)
...
drink2.place(x=500, y=450)
...
drink3.place(x=500, y=480)
```

- **Створити мітку для виведення вартості замовлення** `price_label` **з такими властивостями: текст:** `"Загальна вартість: 0 грн"`**; шрифт:** `"Cambria"`, `18`, `"bold"`**; колір фону** `"#261b14"` **і колір тексту** `"#ffa500"`.

```
price_label.place(x=300, y=550)
```

- **Створити фрейм для розміщення кнопок в один рядок**

```
button_frame = Frame(pizza, bg="#261b14")
button_frame.place(relx=0.5, rely=0.95, anchor="s")
```

- Створити кнопку **btn_calculate** з такими властивостями: батьківський елемент `button_frame`, текст: `"Розрахувати"`; шрифт: `'Arial black', 14`; колір фону `"#af1618"` і колір тексту `fg="#ffffff"`; `width=15`; виклик функції `command = calculate_price`.

```
btn_calculate.pack(side=LEFT, padx=10)
```

- Створити кнопку **btn_order** з такими властивостями: батьківський елемент `button_frame`, текст: `"Оформити замовлення"`; шрифт: `'Arial black', 14`; колір фону `"#f76e08"` і колір тексту `fg="#ffffff"`; `width=15`; виклик функції `command = order`.

```
btn_order.pack(side=LEFT, padx=10)
```

- Створити кнопку **btn_back** з такими властивостями: батьківський елемент `button_frame`, текст: `"Назад"`; шрифт: `'Arial black', 14`; колір фону `"#af1618"` і колір тексту `fg="#ffffff"`; `width=15`; виклик функції `command = go_back`.

```
btn_back.pack(side=LEFT, padx=10)
```



- Описати функцію **calculate_price()**, яка знаходитиме ціну кожного блоку вибору, сумуватиме вартість та виводити інформацію користувачеві.

Зауваження! Ця функція описується всередині функції `pizza()`

```
def main_pizza_window():
    def calculate_price():
        base_price = 0
        if base_var.get() == 1:
            base_price += 100
        elif base_var.get() == 2:
            base_price += 120
        elif base_var.get() == 3:
            base_price += 150
```

Дії функції:

- знаходить вартість за основу піци: створити змінну **base_price** і надати їй початкового значення **0**; залежно від вибору, визначити вартість обраної основи

```
base_price = 0
if base_var.get() == 1:
    base_price = base_price + 100
...продовження умов
```

- знаходить вартість за вміст піци: створити змінну **content_price** і надати їй початкового значення **0**; залежно від вибору, визначити вартість замовленого вмісту

```
content_price = 0
if cheese_var.get() == 1:
   content_price = content_price + 20
    ...продовження умов
```

- знаходить вартість напою: створити змінну **drink_price** і надати їй початкового значення **0**; залежно від вибору, визначити вартість замовленого напою

```
drink_price = 0
if drink_var.get() == 1:
    drink_price = drink_price + 15
    ...продовження умов
```

- знаходить всю вартість замовлення та виводить результат

```
total_price = base_price + content_price + drink_price
price_label.config(text=f"Загальна вартість: {total_price} грн")
```

Описати функцію **order()**, яка створює вікно для введення контактних даних.

Зауваження! Наступний код описуватиметься в цій функції.
Створення вікна **order_window**:

1. Задання розмірів вікна ("500x300")
2. Колір фону: темний (#261b14).
3 Фіксація розмірів (вимкнення масштабування).
4. Встановлення заголовка: "Оформлення замовлення".
5. Запустити головний цикл mainloop()

Створити мітку для імені **name_label** з такими властивостями: текст: "Ваше ім'я:"; шрифт: "Arial Black", 16; колір фону "#261b14" і колір тексту "#ffffff".

```
name_label.pack(pady=10)
```

- Створити поле для введення імені **name_entry** з такими властивостями: шрифт: "Arial", 16; колір фону "#505050" і колір тексту "#ffffff".

```
name_entry.pack(pady=5)
```

- Створити мітку для телефону **phone_label** з такими властивостями: текст: "Ваш телефон:"; шрифт: "Arial Black", 16; колір фону "#261b14" і колір тексту "#ffffff".

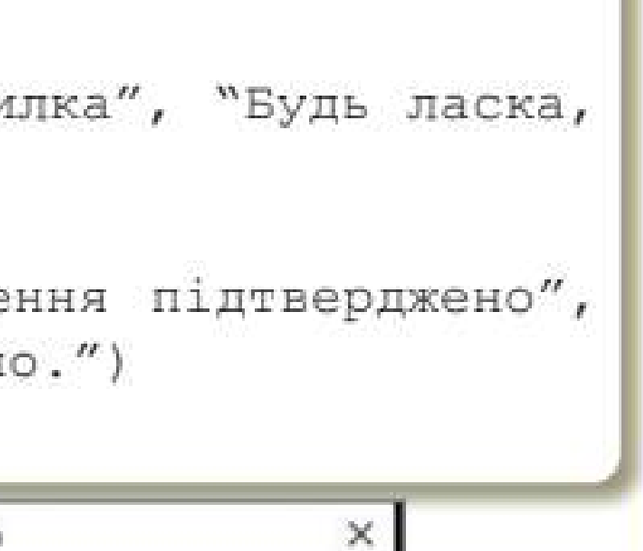
```
phone_label.pack(pady=10)
```

- Створити поле для введення номера телефону **phone_entry** з такими властивостями: шрифт: "Arial", 16; колір фону "#505050" і колір тексту "#ffffff".

```
phone_entry.pack(pady=5)
```

- Створити кнопку **btn_confirm** з такими властивостями: текст: "Підтвердити"; шрифт: 'Arial black', 16; колір фону "#f76e08" і колір тексту fg="#ffffff"; виклик функції command = confirm_order.

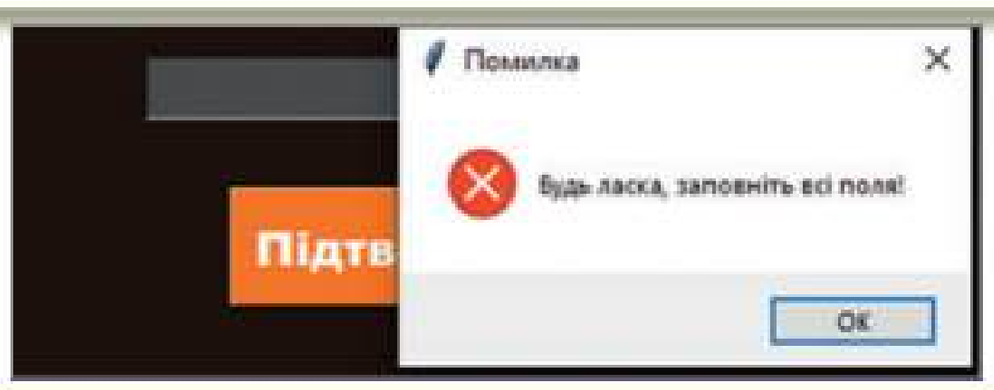
```
btn_confirm.pack(pady=25)
```



- Описати функцію **confirm_order**, яка приймає значення з полів. Якщо вони пусті — виводить діалогове вікно про те, що потрібно заповнити всі поля. Якщо поля введені, то вивести діалогове вікно про підтвердження замовлення та закриває вікно **order_window**.

Зауваження! Ця функція описується всередині функції order()

```
def confirm_order():
            name = name_entry.get()
            phone = phone_entry.get()
            if not name or not phone:
                messagebox.showerror("Помилка", "Будь ласка,
заповніть всі поля!")
                return
            messagebox.showinfo("Замовлення підтверджено",
f"Дякуємо, {name}! Ваше замовлення оформлено.")
            order_window.destroy()
```



Додатково:

Описати функцію **go_back()**, яка закриває вікно **pizza** та відкриває стартове вікно

```
def go_back():
        pizza.destroy()
        start_window()
```

Зауваження! Ця функція описується всередині функції **pizza()**

При запуску після опису цієї функції програма видасть помилку. Справді, в коді немає функції `start_window()`. Ця функція повинна запускати весь наш описаний код. Тому потрібно виконати такі дії:

- виділити та вирізати весь код програми, окрім імпортованих бібліотек
- створити функцію `start_window()`
- вставити вирізаний код в цю функцію
- підправити відступи, якщо це потрібно (поставити курсор на початок рядка і натиснути **Tab** або виділити ще раз вставлений код і натиснути **Tab**)
- в самому кінці вашого коду, без відступів викликати функцію `start_window()`

```
from tkinter import *
from tkinter import messagebox

def start_window():
    вирізаний код

start_window()
```

Етап 5. Тестувальний

1. Перевірка всіх вікон на відповідність функціоналу.
2. Перевірка обчислення та динамічного оновлення ціни: правильний підрахунок вартості для різних варіантів основи, компонентів та напоїв.
3. Перевірка вводу даних: виведення помилки, якщо поле імені або телефону залишається порожнім.
4. Перевірка кнопки **Назад**: повернення до стартового вікна.

Додатково:

Етап 6. Оптимізація та Розширення

1. Додавання нових опцій до піци (наприклад, розмір піци або соуси).
2. Різні розміри піци (мала, середня, велика).
3. При вибору напоїв динамічно змінюється зображення напою.
4. Створення аналогічного вікна для замовлення суші.
5. Зміна стилів та положення компонентів.

Етап 7. Документація

Створіть короткий опис функцій програми та коментарі до коду.

Розділ 6

СТИЛЬ ЦИФРОВОГО ОБРАЗУ

Теми 46-47 **Сутність бренду, його цінностей, місії і візії. Поняття фірмового (корпоративного) стилю. Роль зображень для формування цифрового образу. Елементи стилю (логотип, шрифти, фірмові кольори, гасло, стиль комунікацій тощо)**

Ключові поняття

Бренд
Фірмовий стиль
Логотип
Стиль комунікації
Персонаж бренду

Бренд

Бренд — це набір ідентифікуючих елементів, які відрізняють компанію, продукт, особу чи інший об'єкт.



Бренд є не тільки логотипом чи назвою компанії. Він відображає цінності, місію та візію, тобто те, що стоїть за продуктом або послугою, яку пропонує компанія.



Фірмови й стиль

Фірмовий стиль — це сукупність прийомів (графічних, колірних, пластичних, акустичних, відео), які забезпечують єдиний стиль та образ для всіх виробів фірми і заходів, покращують можливість запам'ятовувати, сприймати покупцями, партнерами, іншими незалежними спостерігачами не тільки товари фірми, а й усю її діяльність і відрізняти від конкурентних фірм.

Усі ці елементи дизайнери використовують для створення фірмового стилю, щоб зробити бренд упізнаваним, посилити позиціювання, виділитись серед конкурентів, підвищити цінність бренду.



Цінності бренду — це принципи та ідеї, на яких будується вся діяльність компанії. Вони визначають, що компанія вважає важливим і як вона хоче впливати на суспільство.

Місія бренду — це коротке формулювання, що описує основну мету компанії, її завдання та роль у житті клієнтів. Місія часто відповідає на запитання: «Чому ми існуємо?».

Візія (бачення) — це уявлення компанії про те, якою вона хоче бути в майбутньому і які зміни в суспільстві вона прагне зробити. Візія націлена на довгострокові цілі та плани розвитку.

Основні елементи фірмового стилю

Логотип	Це головний візуальний символ компанії. Він може містити назву компанії або бути графічним зображенням, яке асоціюється з брендом.
Фірмові шрифти	Спеціально обрані шрифти, що використовуються в усіх комунікаціях компанії. Вони повинні бути читабельними та відповідати стилю бренду.
Фірмові кольори	Це набір кольорів, що асоціюються з брендом і використовуються в усіх його візуальних матеріалах.
Гасло	Коротке й легке для запам'ятовування висловлювання, яке відображає цінності чи обіцянки бренду.
Стиль комунікацій	Манера, в якій компанія спілкується з клієнтами, зокрема в текстах, відео, рекламі.

Ідентичність бренду — це те, як компанія хоче, щоб її сприймали. Ідентичність визначає стиль комунікацій, візуальні елементи (логотип, кольори, шрифти), тон звернення, а також місію і цінності компанії.



Позиціонування бренду — це стратегія того, як бренд виділяється серед конкурентів. Позиціонування визначає унікальні риси, які роблять бренд привабливим для цільової аудиторії. Це відповідає на питання: «Чому споживачі повинні обрати саме цей бренд?»

Логотип

Логотип — це основний візуальний елемент, що ідентифікує бренд і дозволяє споживачам легко впізнавати його серед конкурентів. Це символ, який поєднує в собі стиль, фірмові кольори, шрифти та візуальні елементи, щоб створити унікальний образ компанії.

Логотип часто є першим, з чим взаємодіє потенційний клієнт, тому він має бути максимально простим, зрозумілим і запам'ятовуваним.

Основні функції логотипу

1. **Ідентифікація бренду**
Логотип — це «обличчя» бренду, яке дозволяє легко впізнавати його серед інших компаній на ринку. Успішний логотип швидко асоціюється зі специфічними товарами, послугами або цінностями компанії.

2. **Підвищення впізнаваності**
Чіткий і запам'ятовуваний логотип допомагає компанії виділитися серед конкурентів.

3. **Універсальність використання:**
Логотип використовують у різних маркетингових матеріалах, на продукції, вебсайтах, соціальних мережах, рекламних банерах та візитних картках. Важливо, щоб він був адаптивним і мав гарний вигляд у будь-якому форматі — як у цифровому, так і в друкованому.





Види логотипів

Текстові логотипи — Складаються тільки з тексту, часто це назва компанії, написана унікальним шрифтом.

Графічні символи — Містять прості графічні зображення або символи, які стають візуальною ідентифікацією бренду.

Абстрактні логотипи — Складаються з абстрактних форм або графічних елементів, які не мають прямого зв'язку з реальними об'єктами.

Комбіновані логотипи — Це поєднання тексту і графічного символу, що дозволяє компаніям використовувати їх разом або окремо.

Емблеми — Містять текст всередині символу або форми.



Основні принципи створення логотипу

Простота

Найуспішніші логотипи — це ті, які є простими і легко сприймаються. Простота допомагає зменшити кількість деталей, які можуть відволікати увагу, та робить логотип запам'ятовуваним.

Унікальність

Логотип має бути оригінальним та відрізнятися від інших на ринку. Унікальний дизайн допомагає бренду виділитися серед конкурентів.

Адаптивність

Логотип повинен мати гарний та чіткий вигляд як на великому рекламному банері, так і на невеликому значку чи візитній картці. Це означає, що дизайн повинен бути адаптований до різних форматів і розмірів.

Кольорова палітра

Як і в типографіці, кольори логотипу відіграють важливу роль у передачі настрою та емоцій. Кольори мають бути узгоджені з брендом та викликати правильні асоціації.

Шрифти

Шрифти — це важливий елемент фірмового стилю, який використовується для передачі повідомлень бренду у візуальній формі. Вони допомагають формувати настрій, стиль і характер компанії.

Шрифт — це не просто спосіб написати текст, а частина візуальної ідентичності бренду, яка впливає на сприйняття інформації споживачем.



1 Читабельність

Шрифт має бути легким для читання як у цифрових, так і в друкованих матеріалах. Він повинен добре сприйматися на різних носіях: на вебсайті, в мобільних додатках, на друкованій продукції.

2 Універсальність

Шрифт повинен бути гнучким для використання в різних форматах і розмірах. Наприклад, логотипи, рекламні оголошення та підписи повинні використовувати однакові шрифти, але адаптовані до різних розмірів та кольорів.

3 Відповідність стилю бренду

Важливо, щоб обраний шрифт відповідав стилю та цінностям компанії. Якщо бренд орієнтований на інновації та сучасність, підійде безсерифний шрифт. Якщо бренд прагне передати традиційність та надійність, краще вибрати серійний шрифт.

Шрифтові пари — це поєднання двох або більше шрифтів, що використовуються разом у дизайні, щоб створити гармонійний і зручний для читання текстовий контент.

Використання різних шрифтів для заголовків і основного тексту допомагає краще організувати інформацію, підвищуючи її сприйняття. Наприклад, жирний шрифт для заголовка привертає увагу, а більш простий шрифт для основного тексту робить його зручним для читання.

Поєднання шрифтів допомагає створити баланс у дизайні. Коли шрифти добре гармонують між собою, це підвищує загальне враження від матеріалу.



Як обирати шрифтові пари?

- **Контрастність.** Контраст може бути у розмірах, товщині ліній, формі або навіть кольорі.
- **Читабельність.** Перш ніж обирати шрифтові пари, переконайтеся, що обрані шрифти є читабельними у різних розмірах та форматах. Наприклад, декоративні шрифти часто краще підходять для заголовків, але можуть бути важкими для читання в основному тексті.
- **Тестування на різних платформах**. Шрифти можуть мати різний вигляд на екрані та у друкованому варіанті, тому важливо перевірити, як шрифтові пари сприйматимуться у різних форматах — від вебсайту до друкованої продукції.

Фірмові кольори — це набір кольорів, які використовуються компанією для ідентифікації свого бренду.

Ці кольори є важливим елементом фірмового стилю і використовуються на логотипах, у вебдизайні, рекламних матеріалах, на упаковці продукції, у соціальних мережах та інших візуальних комунікаціях бренду.

Візуальне сприйняття — це перше, що помічають споживачі при взаємодії з брендом. Чітко визначена кольорова палітра робить бренд миттєво впізнаваним.

Кольори можуть передавати емоції та асоціації. Кожен колір викликає певні психологічні реакції, тому правильно підібрані фірмові кольори можуть підкреслювати характер бренду, його цінності та посил.

Фірмові кольори не лише роблять бренд впізнаваним, але й впливають на емоційне сприйняття аудиторії, викликають певні асоціації та підсвідомі реакції.



Персонаж бренду

Персонаж бренду — це вигадана або реальна особа, яка уособлює цінності, стиль та дух бренду.

Це не обов'язково людина — це може бути тварина, робот або навіть вигаданий об'єкт, який стає символом компанії та допомагає формувати емоційний зв'язок із аудиторією. Персонаж бренду є частиною візуальної ідентичності бренду, і він часто використовується в рекламних кампаніях, маркетингових матеріалах або на пакуванні.



Гасло

Гасло (або слоган) — це коротка і запам'ятовувана фраза, яка передає головну ідею або цінність бренду.

Гасло повинно чітко і лаконічно висловлювати основну ідею або перевагу бренду. Воно дає зрозуміти споживачам, що бренд пропонує і в чому його унікальність.

Гасло часто викликає емоційний відгук у споживачів, оскільки воно повинно не просто описувати товар чи послугу, а й створювати позитивні емоції, що асоціюються з брендом.

Гасло має бути максимально стислим, адже короткі фрази краще запам'ятовуються. Ідеальне гасло складається з 3-6 слів.



Гасло є одним із ключових елементів фірмового стилю та маркетингової стратегії, оскільки допомагає створити сильне емоційне враження про бренд і передати його місію в кількох словах. Часто гасло використовується в рекламних кампаніях, на упаковці продукції, у соціальних мережах та на сайтах, щоб підкреслити унікальність компанії.

Стиль комунікації

Стиль комунікацій — це спосіб, яким бренд взаємодіє зі своєю аудиторією через різні канали комунікації: текст, візуальні матеріали.

Послідовний і чіткий стиль комунікацій допомагає побудувати довіру до бренду.

Стиль комунікацій допомагає зробити бренд впізнаваним. Якщо споживачі натрапляють на однаковий стиль у різних каналах — від соціальних мереж до друкованих матеріалів — це підвищує шанси, що бренд запам'ятається.

Це не тільки мова, якою спілкується бренд, а й тон, емоційна складова, манера подачі інформації, що впливають на те, як бренд сприймається споживачами.

Стиль комунікацій — це важливий аспект фірмового стилю, який допомагає створити цілісний та впізнаваний образ бренду.

Основні елементи стилю комунікацій

1 **Тон і манера спілкування**

Це спосіб, яким бренд звертається до своєї аудиторії. Тон може бути формальним, неформальним, серйозним, жартівливим, дружнім або професійним.

2 **Мова**

Бренд може обрати різні стилі мови — просту, формальну, технічну чи навіть сленг. Це залежить від цільової аудиторії та індустрії, в якій працює бренд.

3 **Тональність**

Тональність визначає емоційний вплив на аудиторію. Вона може бути позитивною, заохочувальною, нейтральною або навіть провокаційною, залежно від того, як бренд хоче бути сприйнятим.

4 **Візуальна комунікація**

Стиль комунікацій передбачає не лише текст, а й візуальні елементи: кольори, графіку, зображення, що підтримують загальну ідею бренду.

5 **Структура і форма комунікацій**

Це спосіб організації інформації. Вона може бути чіткою та лаконічною або більш деталізованою та описовою, залежно від аудиторії та контексту.

Легенда бренду (brand story) – це захоплива історія про продукт або компанії.

В основі історії бренду можуть лежати:

- історичні факти – вікові традиції, історичні події, легендарні місця, історія заснування компанії, суспільно значущі події, пов'язані з брендом;
- реальні події – достовірні події, що вплинули на становлення і розвиток компанії.



Легенда бренду може бути сформульована як у вигляді вербальних, так і графічних елементів.

Ребрендинг — це процес зміни або оновлення фірмового стилю компанії, її візуальної ідентичності або стратегії комунікації.

Якщо бренд існує тривалий час і його візуальний стиль чи комунікаційний підхід більше не відповідають сучасним стандартам або смакам аудиторії, ребрендинг допоможе освіжити фірмовий стиль і зробити його більш актуальним.

Це може передбачати зміни в логотипі, фірмових кольорах, шрифтах, гаслі, продуктах, а також у стилі комунікацій і навіть місії та цінностях компанії.

Зображення допомагають створити впізнаваний і унікальний образ бренду. Вони використовуються для підтримки фірмового стилю: кольорової палітри, стилю фотографій, ілюстрацій або графічних елементів.

Зображення повинні бути узгоджені з загальним стилем бренду. Використання одних і тих самих фірмових кольорів, стилю фотографій та графіки допомагає створити впізнаваний і послідовний образ бренду на всіх платформах.

Фотографії та ілюстрації можуть стати частиною фірмового стилю бренду. Наприклад, деякі бренди обирають певний стиль фотографії — мінімалістичний, яскравий, кольоровий або чорно-білий — щоб підкреслити свої унікальні риси та встановити певний настрій.

Практична робота №27.

Індивідуальна робота

Створення фірмового стилю для бізнес-проєкту на основі хобі

Ваше завдання — розробити фірмовий стиль для бізнес-проєкту на основі свого хобі. Виконуючи практичну роботу, ви створите всі елементи бренду: логотип, шрифти, кольори, гасло і стиль комунікацій, а потім зберете їх у брендбук (на наступних уроках)

Працюємо в **Google Документах**

Надаємо доступ педагогу

1. Придумайте бізнес-проєкт на основі хобі

Виберіть своє хобі та перетворіть його на бізнес-проєкт. Наприклад, це може бути бізнес з виготовлення виробів ручної роботи, кулінарний блог, спортивні тренування тощо.

Завдання: опишіть свій проєкт у 2-3 реченнях. Що ви пропонуєте? Хто ваша аудиторія?

2. Створіть логотип

Логотип — це головний символ вашого бренду. Він має бути простим, але водночас відображати суть вашого бізнесу.

Завдання: використовуйте **Canva**

Покликання: **Canva** для створення логотипів.

3. Підберіть шрифтові пари

Шрифти допомагають створити стиль бренду. Один шрифт можна використовувати для заголовків, а інший — для основного тексту. Вони мають гармоніювати між собою.

Завдання: оберіть шрифтові пари для заголовків і тексту.

Покликання: **Google Fonts** для підбору шрифтів або **Font Pair** для підбору шрифтових пар.

4. Оберіть фірмові кольори

Кольори мають передавати настрій і стиль вашого бренду. Наприклад, зелений часто асоціюється з природою, синій — з довірою, червоний — з енергією.

Завдання: виберіть 2-4 основні кольори для вашого бренду.

Покликання: **Coolors** для підбору кольорових палітр або **Adobe Color** для створення палітри.

5. Створіть гасло для бренду

Гасло — це коротка фраза, що передає основну ідею вашого бренду. Воно повинно бути простим, запам'ятовуватися і підкреслювати вашу унікальність.

Завдання: придумайте гасло, яке відображає ваш бізнес і його головні цінності.

6. Визначте стиль комунікацій

Як ваш бренд спілкуватиметься з аудиторією? Стиль комунікацій може бути дружнім, офіційним, креативним або мотиваційним. Подумайте, як це допоможе залучити клієнтів.

Завдання: напишіть приклад повідомлення для соціальних мереж або рекламного оголошення у стилі, який ви обрали для свого бренду.

7. Створіть макет упаковки або рекламного постера

Використовуйте всі елементи фірмового стилю — логотип, шрифти, кольори та гасло — щоб створити макет упаковки для вашого продукту або рекламний постер.

Завдання: створіть макет у **Canva https://www.canva.com/**

Колективна робота

Створення презентації в **Canva**

Провести колективне дослідження на задану тему, зібрати та організувати інформацію в презентацію за допомогою **Canva**.

В групі 3-4 учні

Обрати свою тему

Організувати всі матеріали в **Canva Презентації**

1. 10 найвідоміших брендів світу

- Дослідження брендів, що стали дуже відомими.
- *Що врахувати:* Логотип, кольори, гасло, фірмовий стиль, вплив бренду на споживачів.

2. Принципи успішного логотипу

- Дослідження основних характеристик, що роблять логотипи успішними: простота, унікальність, впізнаваність, адаптивність.
- *Що врахувати:* Приклади успішних і невдалих логотипів, які елементи важливі для створення впізнаваного логотипу.

3. Як створити влучне гасло для бренду

- Дослідження процесу створення гасла: чому воно повинно бути коротким, емоційним і легко запам'ятовуватися.
- *Що врахувати:* Приклади відомих гасел, їхній емоційний вплив і значення для бренду.

4. Психологія кольорів у брендингу

- Дослідження, як різні кольори викликають емоції та формують сприйняття бренду.
- *Що врахувати:* Значення кольорів у різних культурах, приклади брендів, що вдало використовують кольорову палітру.

5. Еволюція логотипів відомих брендів

- Дослідження, як логотипи змінювалися з часом і чому компанії вирішили провести ребрендинг.

- *Що врахувати:* Причини змін логотипів, аналіз старих і нових версій логотипів

6. Роль типографіки у брендингу
- Дослідження впливу шрифтів на створення образу бренду та сприйняття його стилю.
- *Що врахувати:* Приклади брендів з унікальними шрифтами, які підсилюють їхній образ.

7. Історія створення найвідоміших гасел
- Аналіз того, як створювалися відомі гасла, які процеси ідуть за їх розробкою.
- *Що врахувати:* Приклади брендів, чиї гасла стали легендарними.

8. Фірмовий стиль як ключовий елемент успіху бренду
- Дослідження важливості єдиного стилю у всіх візуальних і текстових матеріалах бренду.
- *Що врахувати:* Приклади брендів, що послідовно дотримуються фірмового стилю.

9. Як соціальні мережі формують образ бренду
- Дослідження, як бренди використовують **Instagram**, **Facebook**, **TikTok** для просування і формування свого візуального стилю.
- *Що врахувати:* Стратегії просування, використання візуальних матеріалів, відгуки клієнтів.

Вимоги до колективної роботи:

- Використання надійних джерел для збору інформації (офіційні сайти брендів, статті, дослідження, відеоінтерв'ю).
- Створення візуально привабливої презентації у **Canva**. Всі слайди мають бути оформлені у єдиному стилі, який підкреслюватиме обрану тему.
- Кожен слайд має містити коротку, чітку та інформативну інформацію. Не перевантажуйте слайди текстом, використовуйте заголовки та короткі пункти.
- Обов'язкові приклади та ілюстрації для кожної теми.
- Презентувати результати дослідження перед класом. Кожна група має представити свою тему, пояснити ключові ідеї та відповісти на запитання аудиторії.

Ключові поняття

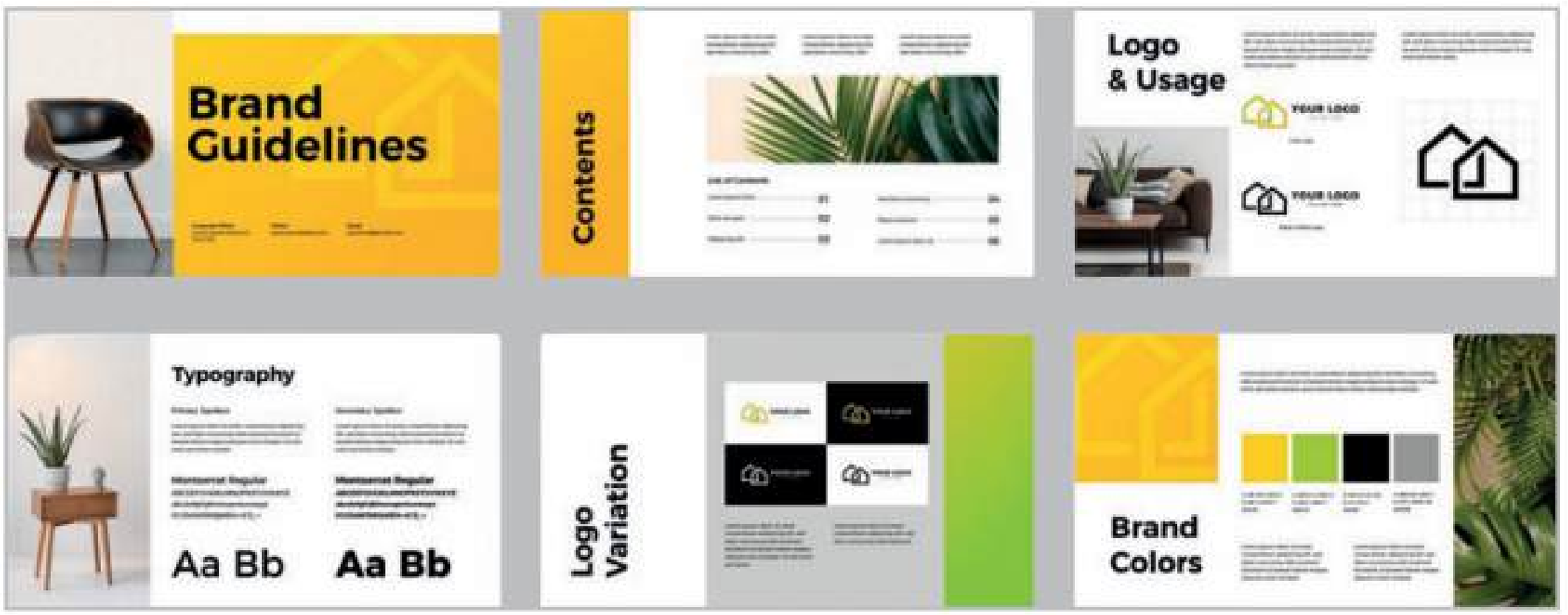
Брендбук
Шрифт та шрифтові пари
Слоган
Стиль комунікації

Брендбук

Брендбук — це документ, в якому зібрані всі правила і елементи, що стосуються фірмового стилю бренду.

Уявіть, що брендбук — це наче інструкція, як зробити так, щоб усе, що стосується бренду, мало б однаково красивий та впізнаваний вигляд на всіх платформах: від соціальних мереж до реклами на вулицях.

Брендбук допомагає будь-якому дизайнеру, маркетологу чи співробітнику компанії правильно використовувати логотип, шрифти, кольори та інші візуальні елементи бренду, щоб все мало узгоджений і гармонійний вигляд.



Що описати в брендбуці?

Логотип

- Як правильно використовувати логотип, яких розмірів він повинен бути, на яких фонах можна його розміщувати (світлих або темних).
- Додайте приклади правильного та неправильного використання логотипу, показуйте варіанти логотипу в різних кольорах та форматах.



Кольорова палітра

- Точні коди кольорів (RGB, HEX) і правила їх використання в різних матеріалах.
- Створіть зразки кольорів у вигляді блоків, які легко показують, як кольори поєднуються між собою. Можете додати приклади використання кольорів на фонах або в текстах.



Шрифти та шрифтові пари

- Які шрифти використовувати для різних матеріалів, розміри шрифтів і правила їх комбінування.
- Покажіть приклади заголовків і тексту в різних шрифтах, щоб продемонструвати, який вони мають вигляд разом. Додайте рекомендації щодо того, де і як застосовувати ці шрифти.

Гасло (слоган)

- Як правильно використовувати гасло, де його краще розміщувати, який стиль комунікацій використовувати поруч із гаслом.
- Покажіть гасло у різних стилях: на плакатах, у соціальних мережах, у відео. Додайте приклади його використання поруч із логотипом.

Стиль комунікацій

- Яким тоном і стилем треба писати тексти для постів, оголошень, реклам і на сайті.
- Додайте приклади текстів: один у неправильному стилі (наприклад, занадто офіційний), інший — у правильному стилі, який відповідає бренду.



Як візуально красиво оформити брендбук?

- Всі елементи в брендбуці мають бути узгоджені між собою: використовуйте однакові шрифти, кольори та візуальні стилі для заголовків, текстів та ілюстрацій.
- Не перенасичуйте сторінки текстом. Використовуйте короткі абзаци, списки і багато простору між елементами, щоб усе було чисте і легко читалося.
- Кожен елемент, що ви описуєте (логотип, кольори, шрифти), повинен мати реальні приклади використання. Це допоможе краще зрозуміти, який має вигляд бренд у реальних ситуаціях.
- Ілюстрації, фотографії або макети (mockups) допоможуть краще показати, як працює бренд у реальному житті. Наприклад, який має вигляд логотип на футболках або на сайті.
- Кожен розділ брендбуку має починатися з яскравого заголовка, щоб одразу було зрозуміло, про що йдеться (Логотип, Кольори, Шрифти тощо).

Canva — чудовий інструмент для створення красивих презентацій. Там ви можете знайти багато готових шаблонів для оформлення брендбуку, що допоможуть вам зробити його професійним і гармонійним.

Практична робота №28.

Індивідуальна практична робота

Створення брендбуку

Об'єднати всі створені раніше елементи фірмового стилю (логотип, шрифти, кольори, гасло, стиль комунікацій) у брендбук. Оформити його красиво та гармонійно у вигляді презентації в **Canva**, щоб показати, як ці елементи працюють разом і допомагають створити цілісний образ вашого бренду.

Етапи виконання практичної роботи

1. Зібрати всі елементи фірмового стилю
   - Перегляньте всі елементи, які ви створювали на попередніх уроках:
   - Логотип.
   - Шрифти (для заголовків та основного тексту).
   - Фірмова кольорова палітра.
   - Гасло.
   - Приклади комунікацій (пости для соціальних мереж або реклами).
   - Підготуйте ці елементи для використання в оформленні брендбуку.
2. Структуруйте свій брендбук
   - *Сторінка 1:* Обкладинка
   - Оформте заголовок: «Брендбук [Назва вашого бренду]».
   - Додайте логотип та гасло.
   - *Сторінка 2:* Вступ
   - Напишіть коротке пояснення, що це за бренд і яка його основна ідея. Кому призначений цей бренд і які його цінності.
   - *Сторінка 3:* Логотип
   - Додайте ваш логотип.
   - Опишіть, як правильно його використовувати (наприклад, розміри, на яких фонах можна використовувати, які кольори для нього підходять).
   - Додайте приклади правильного та неправильного використання логотипу.
   - *Сторінка 4:* Фірмова кольорова палітра
   - Додайте свої основні фірмові кольори.
   - Підпишіть точні коди кольорів (RGB або HEX).
   - Поясніть, як ці кольори можуть використовуватися (наприклад, для фону, заголовків, основного тексту).

- *Сторінка 5:* Шрифти
  - Покажіть вибрані шрифти для заголовків і основного тексту.
  - Напишіть, для яких матеріалів використовуватимуться ці шрифти (наприклад, заголовки на сайті, текст у рекламних буклетах).
  - Додайте приклади текстів у різних шрифтах.
- *Сторінка 6:* Гасло
  - Додайте своє гасло.
  - Покажіть, як воно виглядає разом із логотипом, кольорами та шрифтами.
  - Поясніть, як і де гасло можна використовувати (у рекламі, на сайті, на товарах).
- *Сторінка 7:* Стиль комунікацій
  - Опишіть, яким тоном і стилем ваш бренд буде спілкуватися з аудиторією (дружнім, офіційним, креативним тощо).
  - Додайте приклади текстів (наприклад, пост для соціальної мережі або рекламне оголошення).

3. Оформте брендбук у **Canva**
   - Використовуйте **Canva** для створення вашого брендбуку. Підберіть стильний шаблон, що підходить для оформлення презентації.
   - Всі елементи (логотип, кольори, шрифти) повинні гармонійно поєднуватися один з одним.
   - Додайте багато простору між текстом і зображеннями, щоб усе було чисто і професійно.
   - Не забувайте, що презентація має бути легкою для читання і розуміння.
4. Збережіть брендбук і підготуйте його до презентації
   - Збережіть свій брендбук у форматі PDF або презентації, щоб показати його під час презентації перед класом.
   - Будьте готові коротко пояснити кожен розділ і те, чому ви обрали саме такі елементи для свого бренду.

Теми 50–51 **Вплив інформації на адресата. Сильні та слабкі аргументи. Розробка дизайну шаблонів інформаційних продуктів бренду відповідно до погодженого стилю**

Ключові поняття

Шаблон інформаційних продуктів
Стиль бренду
Аргументи

Шаблон інформаційних продуктів

Шаблони інформаційних продуктів — це стандартні дизайни для створення візуальних і текстових матеріалів, які можна використовувати з різною метою: для постів у соціальних мережах, презентацій, листівок, банерів, інфографік тощо.

Види шаблонів інформаційних продуктів:

- шаблони для постів у соціальних мережах — презентаційні шаблони
- шаблони для інфографік
- шаблони рекламних банерів
- шаблони для електронних листів (email-маркетинг)



Стиль бренду

Погоджений стиль бренду — це сукупність візуальних елементів, які створюють єдиний образ бренду.



Використання цього стилю у всіх шаблонах допомагає:

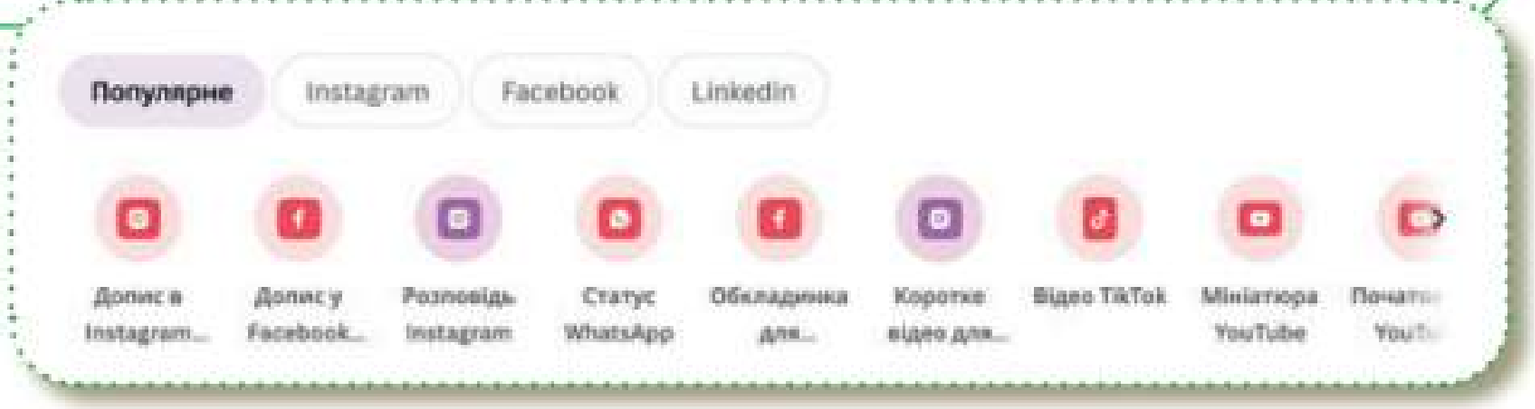
- Підвищити впізнаваність бренду.
- Створити професійний та естетичний вигляд продуктів.

Створення шаблонів для інформаційних продуктів

Шаблон для постів у соціальних мережах

- Використовуйте логотип у верхньому або нижньому куті поста.
- Виберіть фоновий колір, що відповідає фірмовому стилю.
- Використовуйте погоджений шрифт для заголовка та основного тексту.
- Додайте гасло або коротке повідомлення у нижній частині поста.





Шаблон для презентацій

- Використовуйте однаковий стиль для всіх слайдів (один тип заголовка, тексту і кольорових блоків).
- Логотип може бути на титульному слайді і на заключному слайді.
- Дотримуйтеся фірмової кольорової палітри у фонах і текстах.
- Додайте візуальні елементи (ілюстрації, іконки), які відповідають стилю бренду.

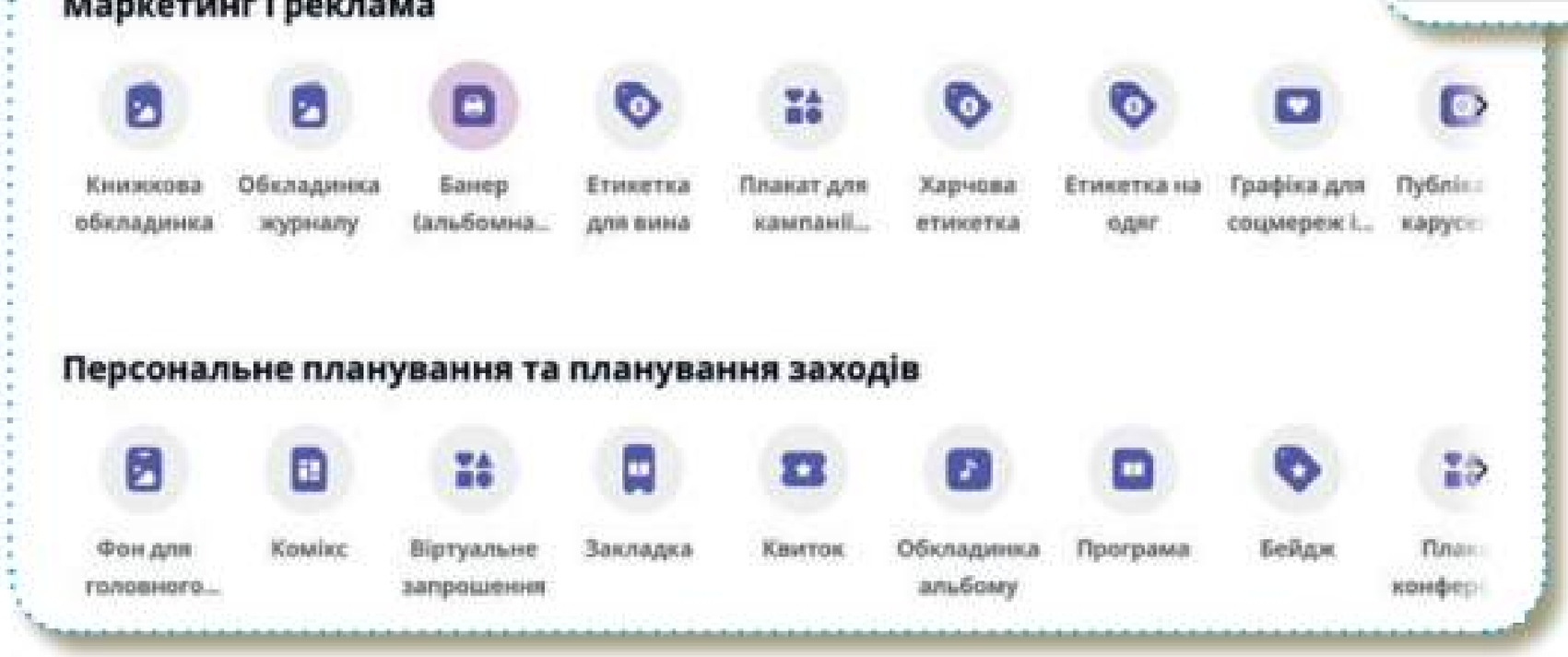




Шаблон для інфографік

- Використовуйте прості та зрозумілі іконки та графіки, що відповідають стилю бренду.
- Використовуйте погоджену палітру для всіх кольорових елементів (графіків, діаграм).
- Текст та заголовки мають бути виконані у погоджених шрифтах.





Шаблон для листівки або флаєра

- Розмістіть логотип у верхній частині листівки (наприклад).
- Використовуйте погоджену кольорову палітру для фону та елементів дизайну.
- Заголовок листівки має бути великим і контрастним, щоб привертати увагу.
- Додайте короткий текст з основною інформацією про подію, продукт або послугу.
- У нижній частині флаєра додайте контактну інформацію та посилання на вебсайт або соціальні мережі.
- Використовуйте прості візуальні елементи або іконки для привернення уваги.





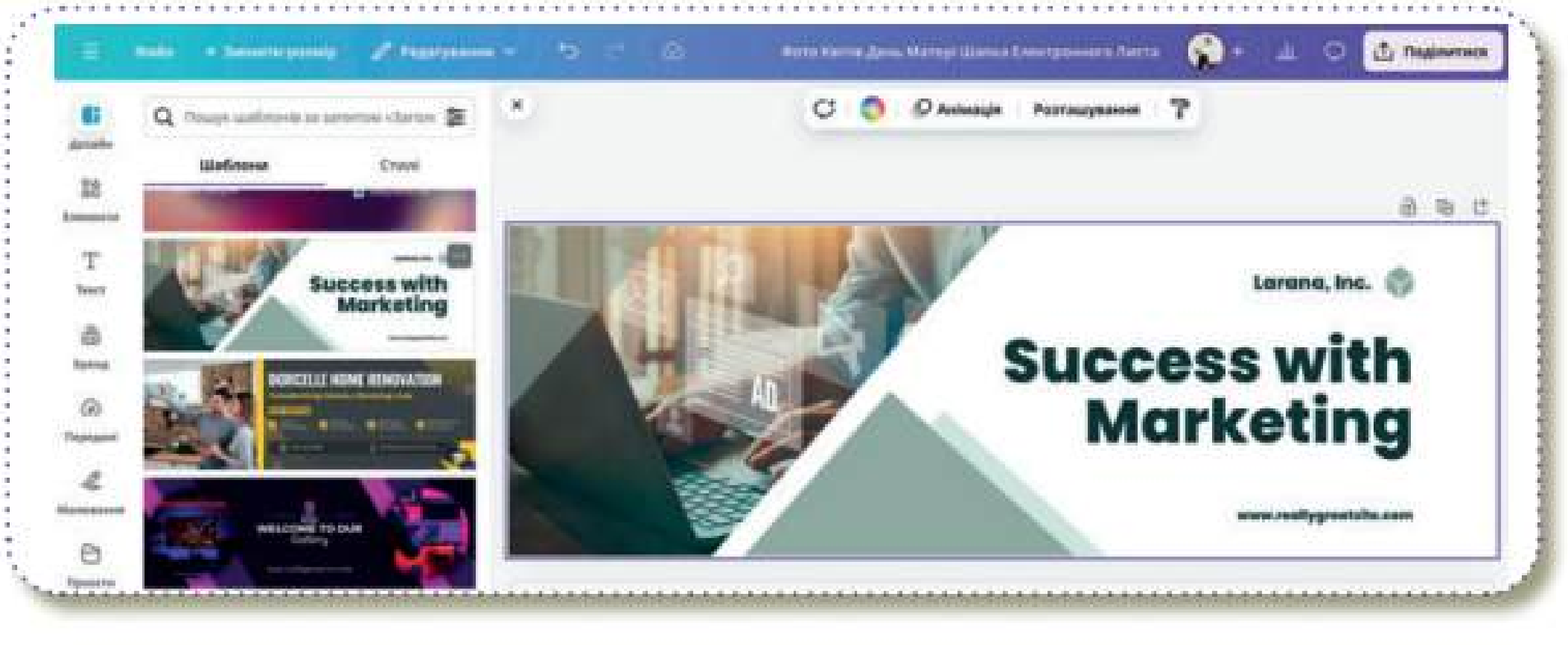
Шаблон для візитівки

- Розмістіть логотип бренду в одному з кутів або по центру візитівки.
- Використовуйте мінімалістичний дизайн із фірмовими кольорами.
- Оберіть чіткий і простий шрифт для імені, посади та контактної інформації (номер телефону, email, сайт).
- Додайте слоган або коротке повідомлення бренду на зворотному боці.
- Використовуйте візуальні елементи або іконки для соціальних мереж чи контактних даних.



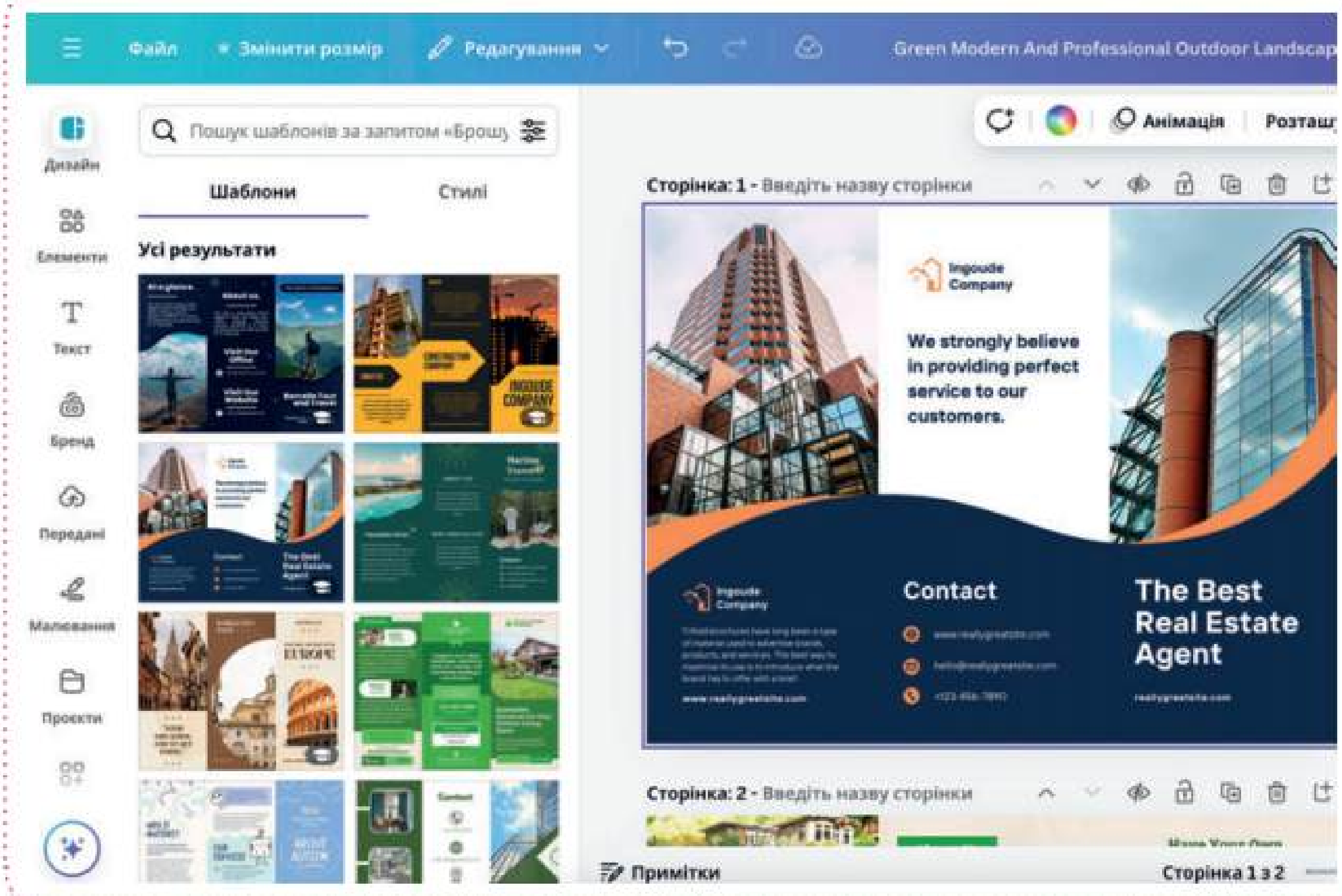
Шаблон для електронного листа (email-маркетинг)

- Розмістіть логотип у верхній частині листа або в «шапці».
- Використовуйте заголовок у погодженому шрифті для короткого повідомлення (наприклад, «Спеціальна пропозиція» або «Новини бренду»).
- Використовуйте основний текст у фірмовому стилі (шрифти, кольори), дотримуючись простоти й читабельності.
- Додайте візуальні елементи або фотографії, що підкреслюють основний зміст листа.
- У нижній частині листа розмістіть заклик до дії (наприклад, «Дізнатися більше», «Замовити зараз») і контакти бренду.



Шаблон для брошури

- Логотип має бути розташований на титульній сторінці брошури, зверху або по центру.
- Використовуйте фірмові кольори у всіх елементах брошури (обкладинка, заголовки, блоки тексту).
- Додайте погоджені шрифти для заголовків та основного тексту, зберігаючи читабельність і гармонію.
- У кожному розділі брошури додайте візуальні елементи (фото, графіки, іконки), що відповідають темі.
- На останній сторінці брошури додайте контактні дані та слоган бренду.



Вплив інформації на адресата



Інформація, яку ми отримуємо, здатна викликати різні реакції: від байдужості до бажання діяти.

На адресата можуть впливати:

- *Емоції:* інформація, що викликає сильні емоції (радість, страх, співчуття), зазвичай має сильний вплив.
- *Авторитет джерела:* якщо джерело інформації є надійним і авторитетним, це підвищує рівень довіри до повідомлення.

- *Зрозумілість та ясність:* що простіше і зрозуміліше подано інформацію, то легше її сприймає адресат.
- *Контекст:* обставини, у яких подана інформація, також можуть підсилити або послабити її вплив.

Аргументи

Аргумент — це твердження, яке використовується для переконання адресата в чомусь або для пояснення певної точки зору.



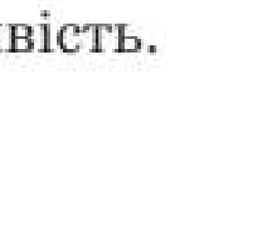
Аргументи можна поділити на сильні та слабкі залежно від того, наскільки переконливо вони підтверджують основну ідею або позицію.

Сильні аргументи — це твердження, які базуються на фактах, доказах та логіці.

Вони зазвичай:

- *Підтримуються надійними джерелами:* коли аргумент підтверджений науковими дослідженнями, статистикою або даними з авторитетних джерел.
- *Логічно побудовані:* сильні аргументи слідують за логікою та не містять суперечностей.
- *Зрозумілі та конкретні:* чіткі, однозначні факти або приклади підсилюють аргументацію.
- *Звертаються до потреб та інтересів аудиторії:* аргумент стає сильнішим, коли він відповідає на питання або виклики, з якими стикається адресат.

Слабкі аргументи — це твердження, які не підтримуються переконливими доказами або не мають логічної основи.



Вони зазвичай:

- *Базуються на особистих думках або припущеннях:* якщо аргумент не підкріплений фактами або доказами, його вплив буде слабким.
- *Мають загальний або невизначений характер:* неясні або неточні аргументи, які не містять конкретних прикладів або доказів.
- *Можуть використовувати логічні помилки:* такі як хибні аналогії або «підміна понять».
- *Надто емоційні:* коли аргументи базуються лише на емоціях без підтвердження фактами, це послаблює їхню переконливість.

Практична робота №29.

Індивідуальна практична робота

Створення шаблонів інформаційних продуктів у **Canva**

Розробити серію шаблонів для інформаційних продуктів бренду, дотримуючись погодженого фірмового стилю.

Шаблони, які потрібно створити:

- Шаблон для постів у соціальних мережах
- Шаблон для візитівки
- Шаблон для листівки або флаєра
- Шаблон для брошури
- Шаблон для електронного листа
- Шаблон для обкладинки презентації
- Шаблон для запрошення на подію
- За бажанням можете створити інші шаблони

Загальні вимоги до виконання практичної роботи

1. **Дотримання фірмового стилю**: усі шаблони мають бути виконані у погоджених кольорах, із використанням логотипу, гасла, шрифтів та інших елементів бренду.
2. **Зручність і читабельність**: шрифти мають бути чіткими і читабельними. Фон і текст повинні контрастувати один з одним для зручності сприйняття.
3. **Креативність та унікальність**: використовуйте креативні ідеї для розміщення елементів на шаблонах, але при цьому зберігайте загальну гармонію дизайну.
4. **Консистентність**: усі шаблони мають бути узгоджені між собою, зберігаючи єдиний стиль на всіх матеріалах.
5. **Візуальна привабливість**: шаблони мають бути естетично привабливими, з чіткими візуальними акцентами і продуманим розміщенням елементів.

Результат

До всіх шаблонів надати доступ педагогу.

Створіть презентацію в **Canva**, де будуть представлені всі розроблені шаблони, а також короткий опис кожного з них.

Технології генерування ідей. Принципи, етапи та методи дизайн-мислення. Складання контент-плану щодо публікацій бренду

Тема 52

Ключові поняття

Дизайн-мислення
Контент-план
Мозковий штурм
Мозкова карта
Метод генерації нових ідей

Генерування ідей є важливим етапом у процесі розв'язання будь-якої задачі.

Мозковий штурм (Brainstorming)

Це одна з найбільш популярних технік для генерації ідей, яка дозволяє учасникам висловлювати будь-які думки без критики. Основна мета — запропонувати якомога більше ідей за короткий проміжок часу.

Метод шести капелюхів мислення (Six Thinking Hats)

Цей метод допомагає розглядати проблему з різних точок зору. Шість різних капелюхів представляють шість різних способів мислення: логічний, креативний, критичний, емоційний, об'єктивний і керівний.

Мозкова карта (Mind Map)

Візуальний спосіб організації думок. Основна ідея розташовується у центрі, а навколо неї розширюються гілки з новими ідеями та підтемами, що дозволяє структурувати думки.

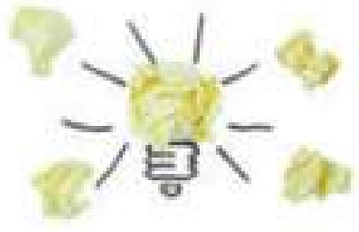


Метод генерації нових ідей (Scamper)

Поставити завдання: визначити проблему, яка потребує розв'язання, або взяти існуючу ідею, на основі якої необхідно згенерувати нову.

Задати питання за схемою «SCAMPER» для генерації нових ідей.

S – *Substitute* (замінити)
C – *Combine* (комбінувати)
A – *Adapt* (адаптувати)
M – *Modify / Magnify* (модифікувати)
P – *Put to other uses* (застосувати для чогось іншого, в іншій галузі)
E – *Eliminate* (усунути або звести до мінімуму)
R – *Rearrange* (переставити, перевернути, змінити порядок)



Дизайн-мислення

Дизайн-мислення — це методологія, яка використовується для розв'язання складних проблем з фокусом на користувачів та їхні потреби. Цей підхід передбачає емпатію, інновації та експерименти.

1 Орієнтація на користувача

Потреби користувача є головним пріоритетом у процесі створення продукту або послуги.



2 Креативність

Необхідно шукати нові ідеї та рішення, навіть якщо вони спочатку здаються незвичайними.

3 Експерименти

Частина процесу полягає у створенні прототипів і тестуванні ідей на реальних користувачах для отримання зворотного зв'язку.

4 Співпраця

Робота у команді дозволяє бачити проблему з різних сторін, що сприяє створенню кращих рішень.



Дизайн-мислення (Design Thinking)

Як це працює? Етапи дизайн-мислення

Що лежить в основі дизайн-мислення?

Human-centered design є однією з ключових складових дизайн-мислення. Основою дизайн-мислення є творчий процес, в якому дизайнери, розробники та інші спеціалісти працюють разом, щоб створити інноваційні рішення для різних проблем та викликів. Цей підхід виник з потреби розв'язання складних проблем та непередбачених ситуацій, які виникають у різних галузях діяльності.



1 **Емпатія**
Збір інформації про користувача, розуміння його потреб через спостереження, опитування або дослідження.

2 **Формулювання проблем**
На основі зібраної інформації чітко формулюється проблема, яку потрібно розв'язати.

3 **Генерація ідей**
Використовуючи різні методи, такі як мозковий штурм, команда шукає рішення для проблеми.

4 **Прототипування**
Створення робочих моделей або простих версій продукту для тестування.

5 **Тестування**
Прототип тестується на реальних користувачах для збору зворотного зв'язку, який використовується для вдосконалення продукту.

Контент-план

Контент-план — це графік або система, яка визначає, що, коли і де публікуватиметься від імені бренду.



Контент-план допомагає впорядкувати процес публікацій, що дозволяє бренду залишатися послідовним у своєму комунікаційному підході та ефективно взаємодіяти зі своєю аудиторією.

Основні етапи створення контент-плану

1 Аналіз цільової аудиторії

- Визначення основних характеристик аудиторії (вік, інтереси, стиль життя).
- Вивчення потреб та запитів, щоб зрозуміти, який контент буде цікавий та корисний.

2 Постановка цілей для контенту

Визначення чітких цілей, яких бренд прагне досягти за допомогою контенту (наприклад, підвищення впізнаваності, просування нових продуктів, залучення нових підписників).

3 Вибір тем для публікацій

Розробка ключових тем, що відповідають інтересам аудиторії і пов'язані з діяльністю бренду. Наприклад, для IT-компанії це можуть бути огляди технологій, поради з налаштування пристроїв або інформація про новинки в індустрії.

4 Вибір форматів контенту

Пости, статті, відео, інфографіка, прямі ефіри тощо. Різні формати допомагають підтримувати інтерес аудиторії та дозволяють більш гнучко передавати інформацію.

5 Створення календаря публікацій

- Розподіл публікацій за днями та годинами для досягнення максимального залучення.
- Вказання конкретних дат, тем та форматів, щоб забезпечити регулярність і послідовність контенту.

6 Аналіз ефективності

- Відстеження реакцій на публікації, показників залучення, кількості лайків, коментарів та репостів.
- Оцінка результатів та внесення коректив у контент-план для подальшого вдосконалення.

Типи контенту для бренду

- **Освітній контент.** Додає цінність для аудиторії, наприклад, поради, інструкції, рекомендації, інфографіка.
- **Розважальний контент.** Допомагає залучити аудиторію та підвищити її активність, наприклад, меми, конкурси, опитування, інтерактиви.
- **Рекламний контент**. Спрямований на просування продуктів і послуг, наприклад, акції, презентації, спеціальні пропозиції.
- **Новини та оновлення бренду.** Інформація про важливі події, новини, зміни в бренді або його продуктах.
- **Контент, створений користувачами.** Відгуки, фото клієнтів, приклади використання продукту (створює довіру та підвищує лояльність).

Принципи ефективного контент-плану

- *Різноманітність:* поєднання різних форматів і типів контенту для утримання уваги аудиторії.
- *Актуальність:* публікація контенту, який відповідає поточним інтересам і потребам аудиторії.
- *Частота і регулярність:* забезпечення стабільного розкладу публікацій для підтримання постійного зв'язку з аудиторією.
- *Візуальна складова:* якісні зображення, графіка та оформлення, які підсилюють сприйняття інформації та привертають увагу.



Створення шаблону контент-плану в Canva

У **Canva** є багато шаблонів для контент-плану, які легко адаптувати для власних потреб. Для початку можна обрати шаблон календаря або таблиці.

1. **Дата публікації:** визначте конкретну дату для кожної теми.
2. **Тема публікації:** запишіть основну ідею або тему посту.
3. **Формат:** наприклад, пост, інфографіка, відео, історія.
4. **Мета:** вкажіть, який це контент: розважальний, інформаційний або освітній.

Плануйте теми так, щоб вони не повторювалися надто часто. Наприклад, якщо блог про в'язання (плетіння), то варто чергувати публікації з оглядами інструментів та інструкціями.

Частота публікацій залежить від вашого розкладу і доступного часу, але для стабільного розвитку блогу рекомендується публікувати хоча б один пост на тиждень.



Практична робота №30.

Сервіс — **Canva**
Шаблон — календар

1. Оберіть тему хобі для вашого блогу (наприклад, фотографія, в'язання (плетіння), малювання).
2. Створіть контент-план на два тижні у **Canva**, включаючи дати, теми та формати.
3. Додайте візуальне оформлення для одного з постів, використовуючи **Canva**, і підготуйте його для публікації.
4. Результат. Надайте доступ для вашого педагога з можливістю редагування.

Проєктування, розроблення, публікація та представлення інформаційного продукту з комбінацією тексту, зображень, звуків, анімацій, відео тощо

Ключові поняття

Інформаційні продукти
Сторіс
Допис
GIF-файли
Чек-листи
Планувальники

Інформаційні продукти

Інформаційні продукти — це різні форми контенту, що створюються для передачі важливої інформації аудиторії та забезпечують ефективну комунікацію.

У сучасному цифровому середовищі такі продукти, як сторіс, дописи для соціальних мереж, презентації та інші візуальні матеріали, можуть значно підвищити впізнаваність бренду та залученість користувачів.

1 Сторіс для соціальних мереж

Сторіс — короткі відео чи зображення, які зникають через 24 години, ідеально підходять для швидкого інформування аудиторії.

Використовуючи шаблони в **Canva**, можна створювати сторіс для анонсів, промо-кампаній, опитувань або ділитися закулісними кадрами.

2 Допис для соціальних мереж

Пости можуть містити текст, зображення, іконки та елементи для створення інфографіки або наочних оглядів продуктів.

Canva має різноманітні шаблони для **Instagram**, **Facebook**, **Twitter** тощо. Ви можете вибрати шаблон відповідно до мети посту — наприклад, для продажу продукту, залучення до обговорення або повідомлення важливої новини.

3 Анімовані GIF-файли та відео

Canva дозволяє створювати короткі відео чи анімовані GIF-файли, які можуть бути використані для сторіс, постів або рекламних кампаній.

Інтерактивний формат допомагає привертати увагу та підтримувати залучення аудиторії.

4 **Чек-листи**

Чек-лист — це простий документ, який допомагає користувачам перевірити виконання певних завдань або кроків у процесі.

Це зручно для надання покрокових інструкцій або списку справ, що супроводжують продукт чи послугу.

Приклад використання: «Чек-лист для планування подорожі».

Переваги:

- Забезпечує чітку структуру і допомагає користувачам слідувати певним крокам.
- Легкий у створенні, доступний і корисний як в електронній, так і в друкованій версії.

5 **Планувальники**

Планувальники можуть бути різного типу: щоденні, тижневі, місячні або тематичні. Вони використовуються для організації часу, завдань і мети.

Приклад використання: «Тижневий планувальник завдань для продуктивності».

Переваги:

- Допомагає користувачам структурувати свої задачі та досягати мети.
- Легко налаштовується під конкретні потреби та теми.

6 **Календарі**

Календарі можна використовувати для планування особистих або робочих завдань. **Canva** дозволяє створювати календарі на місяць або рік, додаючи власні дати чи примітки.

Приклад використання:

- «Календар подій на рік для планування навчання».
- «Місячний календар зі святковими датами та завданнями».

Переваги:

- Може бути як цифровим, так і друкованим.
- Використовується для довготривалого планування і нагадування важливих подій.

Практична робота №31.

Розробити інформаційні продукти в **Canva** на основі вашого контент-плану, створеного на попередньому уроці. Ви створите як мінімум 8 різних видів контенту, які підходять для вашого бренду або особистого блогу.

Необхідні види інформаційних продуктів:

1. Сторіс
2. Допис для соціальних мереж
3. Чек-лист
4. Інфографіка
5. Календар або планувальник
6. Рекламний банер або обкладинка для соцмереж
7. Картка з порадою або цікавим фактом
8. Листівка або запрошення

Інструкція для виконання практичної роботи

1. Підготовка:

- Відкрийте свій контент-план, створений на попередньому уроці. Проаналізуйте теми, мету та типи контенту, який ви планували опублікувати.
- Визначте, який вид інформаційного продукту (сторіс, пост, інфографіка тощо) найкраще підходить для кожної з тем вашого плану.

2. Створення інформаційних продуктів: виконайте певні дії для кожного виду контенту:

- **Сторіс:**
  - У **Canva** виберіть формат сторіс (**Instagram Story**).
  - Додайте зображення або текст, що відповідає одній з тем вашого контент-плану.
  - Використовуйте елементи дизайну, що підходять для стилю вашого бренду.
- **Допис для соціальних мереж:**
  - Оберіть шаблон для допису (наприклад, для **Instagram**, **Facebook**).
  - Створіть візуально привабливий пост з текстом, зображенням або графікою, які підтримують вашу ідею.
- **Чек-лист:**
  - У **Canva** знайдіть шаблон для чек-листів (наприклад, А4 або **PDF** формат).
  - Створіть перелік важливих кроків або завдань, що відповідають вашій темі (наприклад, «Чек-лист для початку роботи над проєктом»).
- **Інфографіка:**
  - Виберіть шаблон інфографіки.

- Додайте графіки, діаграми або текстові блоки для ілюстрації інформації з вашого контент-плану.

Календар або планувальник:

- Виберіть шаблон календаря або планувальника (наприклад, тижневий чи місячний).
- Додайте дати та теми, що допоможуть організувати ваші публікації або завдання.

Рекламний банер або обкладинка для соцмереж:

- Виберіть формат банера або обкладинки (наприклад, обкладинка для **Facebook**).
- Створіть дизайн з інформацією про подію, акцію або анонс нового продукту.

Картка з порадою або цікавим фактом:

- Виберіть формат картки для соцмереж.
- Додайте коротку пораду або цікавий факт на тему вашого блогу.

Листівка або запрошення:

- Виберіть шаблон листівки.
- Додайте текст і графіку, які підходять для привітання або запрошення на подію.

3. Збереження та організація інформаційних продуктів:

- Збережіть кожен файл у відповідному форматі (наприклад, **PNG** для соцмереж, **PDF** для чек-листів).
- Створіть окрему папку для збереження всіх інфопродуктів, щоб їх було легко знайти та використовувати.

4. Презентація результатів:

- Після завершення всіх дизайнів підготуйте коротку презентацію, де опишіть кожен створений інформаційний продукт, його мету та чому ви обрали саме цей вид контенту.
- Продемонструйте свої дизайни в **Canva** або завантажте файли у презентацію для надання зворотного зв'язку.

Результат практичної роботи: кожен учень повинен мати мінімум 8 готових інформаційних продуктів, розроблених у **Canva** на основі свого контент-плану.

ЗМІСТ

РОЗДІЛ 1. ПРОДУКТИВНІСТЬ ТА БЕЗПЕКА ЦИФРОВОГО СЕРЕДОВИЩА

Тема 1. Цифрове робоче місце. Види та можливості сучасних комп'ютерів. Архітектура та елементна база цифрових пристроїв. Процесор. Пам'ять. Кодування даних на цифрових носіях. 4
Практична робота №1 ..... *19*

Тема 2. Пристрої введення та виведення даних. Драйвери 21
Практична робота №2 ..... *36*

Тема 3. Апгрейд та проблеми утилізації електронного сміття. Поняття про файлову системи та її види. Підготовка дисків для роботи та їхне обслуговування. Поняття про логічні та фізичні ушкодження дисків..... 38
Практична робота №3 ..... *44*

Тема 4. Операційна система та її складові. Спеціальні функції операційної системи для користувачів з особливими потребами..... 45
Практична робота №4 ..... *50*

Тема 5. Авторське право на програмне забезпечення. Ліцензії та їхні види. Поняття ліцензійного ключа 52
Практична робота №5 ..... *58*

Тема 6. Сумісність програмного забезпечення. Автозавантаження. Оновлення програмного забезпечення. 59
Практична робота №6 ..... *65*

Тема 7. Проблеми зберігання даних та як їм запобігти. Втрата даних: причини, способи запобігання. Резервне копіювання даних на цифрових пристроях та у хмарних сервісах. Архівування даних. Поняття про методи стиснення даних..... 67
Практична робота №7 ..... *72*

Тема 8. Віруси та антивірусні програми. Брандмауер. Захист власного цифрового середовища від зловмисних дій 74
Практична робота №8 ..... *87*

РОЗДІЛ 2. ЗБІР І ОПРАЦЮВАННЯ ДАНИХ

Тема 9. Збір даних. Персональні дані. Опитування, анкета, тест. Сервіси для збору даних..... 89
Практична робота №9..... *93*

Теми 10–11. Візуалізація даних (презентація, інфографіка, карти, діаграми тощо)..... 94
Практична робота №10 ..... *97*

Теми 12–13. Текстові функції. Математичні та логічні функції. Статистичні функції. Фінансові функції. Процес дослідження. Проблема та мета дослідження. Гіпотеза дослідження. 103
Практична робота №11..... *111*

РОЗДІЛ 3. САЙТ ТА ЙОГО ВЕРСТКА

Тема 14. Поняття мови гіпертекстової розмітки. Тег та його атрибути. Валідація сторінок. Теги створення елементів вебсторінки: зображень, списків, таблиць 117
Практична робота №12 ..... *126*

Тема 15. Покликання та форми. Мультимедіа на вебсторінках. Поняття таблиці каскадних стилів. Інструменти розробника. Стильове оформлення сторінки. Підключення шрифтів 129
Практична робота №13 ..... *145*

Тема 16. Модуль гнучких блоків. Макет сайту. Цільова аудиторія та технічне завдання з розробки вебсайту. Аналіз відвідування сайту..... 152
Практична робота №14 ..... *161*

Тема 17. Поняття хостингу. Інструменти координації роботи команди (Scrum, Kanban тощо). Система контролю версій. 165
Практична робота №15 ..... *174*

Теми 18–19. Проєкт № 1. Створення односторінкового вебсайту (лендинг) «Книжковий Рай». 177

Теми 20–21. Проєкт № 2. Створення багатосторінкового вебсайту «Подорожі». 183

РОЗДІЛ 4. ПРОЄКТУВАННЯ ГРАФІЧНОГО ІНТЕРФЕЙСУ

Тема 22. Графічний інтерфейс продукту. Дизайн UI/UX. Інтуїтивність та зручність використання, інклюзивність та доступність інформаційних продуктів. 191
Практична робота №16 ..... *198*

Теми 23–24. Векторний графічний редактор. Комбінування та налаштування графічних примітивів. Робота з векторними контурами. Текст та типографіка у векторному графічному редакторі. Ефекти та фільтри. 200
Практична робота №17 ..... *206*

Теми 25–26. Композиція та макет у дизайні. Створення прототипу. Документація інформаційного продукту 208
Практична робота №18 ..... *212*

РОЗДІЛ 5. АЛГОРИТМИ ТА ЇХНІ КОДИ

Тема 27. Середовища для написання коду. Транслятори. Програмний проєкт 213
Практична робота №19 ..... *218*

Тема 28. Мова програмування. Поняття змінної. Типи даних. 220
Практична робота №20 ..... *226*

Тема 29. Введення та виведення даних. Присвоєння. 228
Практична робота №21 ..... *237*

Тема 30. Арифметичні операції. Математичні функції. 241
Практична робота №22 ..... *246*

Тема 31. Логічний тип даних. Операції відношень. Логічні оператори 250
Практична робота №23 ..... *254*

Теми 32–33. Команда розгалуження 256
Практична робота №24 ..... *272*

Теми 34–35. Команди повторення 281
Практична робота №25 ..... *292*

Теми 36–37. Коментування програмного коду.. 296
Практична робота №26 ..... *306*

Теми 38–39. Проєкт № 1. Розв'язування квадратних рівнянь 309

Теми 40–41. Проєкт № 2. Визначення індексу маси тіла (BMI) 314

Теми 42–43. Проєкт № 3. Розрахунок вартості розробки сайту 318
Теми 44–45. Проєкт № 4. Додаток для замовлення PyPizza 325

РОЗДІЛ 6. СТИЛЬ ЦИФРОВОГО ОБРАЗУ

Теми 46–47. Сутність бренду, його цінностей, місії і візії.
Поняття фірмового (корпоративного) стилю.
Роль зображень для формування цифрового образу.
Елементи стилю (логотип, шрифти, фірмові кольори,
гасло, стиль комунікацій тощо).. 336
Практична робота №27 ..... *345*

Теми 48–49. Брендбук 348
Практична робота №28 ..... *351*

Теми 50–51. Вплив інформації на адресата. Сильні та слабкі
аргументи. Розробка дизайну шаблонів інформаційних
продуктів бренду відповідно до погодженого стилю..... 353
Практична робота №29 ..... *359*

Тема 52. Технології генерування ідей. Принципи, етапи та методи
дизайн-мислення. Складання контент-плану щодо
публікацій бренду. 360
Практична робота №30 ..... *365*

Теми 53–54. Проектування, розроблення, публікація
та представлення інформаційного продукту
з комбінацією тексту, зображень, звуків, анімацій,
відео тощо. 366
Практична робота №31 ..... *368*

ДЛЯ НОТАТОК

Відомості про стан підручника

№	Прізвище та ім'я учня/учениці	Навчальний рік	Стан підручника	
			на початку року	в кінці року
1				
2				
3				
4				
5				

Навчальне видання

ТРІЩУК Інна Володимирівна
ЛАЗАРЕЦЬ Олександр Юрійович

ІНФОРМАТИКА

Підручник для 8 класу
закладів загальної середньої освіти
Рекомендовано Міністерством освіти і науки України

Підручник відповідає Державним санітарним нормам і правилам
«Гігієнічні вимоги до друкованої продукції для дітей»

У підручнику використано ілюстрації та фотографії з таких інтернет-джерел:
www.freepik.com www.wikipedia.org

Головний редактор *Богдан Будний*
Редактор *Володимир Дячун*
Обкладинка *Ростислава Крамара*
Комп'ютерна верстка *Зоряни Сидор*
Художній редактор *Ростислав Крамар*
Технічна редакторка *Неля Домарецька*

Підписано до друку ____. Формат 70×100/16. Папір офсетний.
Гарнітура CentSchbook Win95BT. Друк офсетний. Умовн. друк арк. 23,40.
Умовн. фарбо-відб. 93,60. Обл.-вид. арк. 13,80.

Видавництво «Навчальна книга – Богдан»
Свідоцтво про внесення суб'єкта видавничої справи до Державного
реєстру видавців, виготівників і розповсюджувачів видавничої продукції
ДК № 4221 від 07.12.2011 р.

Навчальна книга – Богдан, просп. С. Бандери, 34а, м. Тернопіль, 46002
Навчальна книга – Богдан, а/с 529, м. Тернопіль, 46008
тел./факс (0352)52-06-07
office@bohdan-books.com www.bohdan-books.com

ОСНОВНІ ТЕГИ HTML

Основні теги – теги для відкриття/закриття документів і їхніх частин

`<html></html>`	Початок і кінець документа; вказує програмі перегляду сторінок, що це HTML-документ
`<head></head>`	Метаінформація сторінки
`<body></body>`	Вміст сторінки

Теги заголовків та форматування тексту

`<title></title>`	Заголовок сторінки (відображається на вкладці браузера)
`<h1>..<h6>`	Створює заголовки (1 — найбільший, 2, 3, 4, 5 — проміжних розмірів, 6 — найменший заголовок)

`<p></p>`
`<p align="left">`
`<p align="right"`
`<p align="center"`
`<p align="justify">`
	Створює новий абзац
	Текст зліва
	Текст справа
	Текст по центру
	Текст по ширині

`<ul></ul>`	Маркований список
`<ol></ol>`	Нумерований список
`<b></b>`	Виділення жирним шрифтом (болд)
`<i></i>`	Виділення нахиленим шрифтом (курсивом)
`<u></u>`	Виділення підкресленням
`<strike></strike>`	Виділення перекреслюванням

Теги для створення таблиць

`</table>... </table>`	Визначає початок і кінець таблиці
`</tr>...</tr>`	Визначає рядок у таблиці
`<td>...</td>`	Визначає комірку таблиці
`<th>...</th>`	Визначає заголовок комірки (використовується для першого рядка або першого стовпця таблиці)

`<img>`	Вставлення зображення
`<audio>`	Вставлення аудіофайлів
`<video>`	Вставлення відеофайлів
`<a>`	Вставлення посилання

ОСОБЛИВОСТІ ДИЗАЙНУ САЙТІВ

Зрозуміла навігація

- Меню повинно бути інтуїтивним і легким у використанні.
- Використовуйте чіткі назви розділів.

Читабельність тексту

- Використовуйте зручні шрифти.
- Оптимальний розмір тексту для читання.

Оптимізація для швидкості

Сайт має швидко завантажуватись, навіть із повільним інтернетом.

Адаптивність

Більшість користувачів заходять на сайт зі смартфонів, тому адаптивність є обов'язковою.

Кнопки закліку до дії (CTA)

Кнопки мають бути видимими й закликати до дії (**Зареєструватися**, **Замовити**).

ОСНОВНІ ЕЛЕМЕНТИ ФІРМОВОГО СТИЛЮ

Логотип	Це головний візуальний символ компанії. Він може містити назву компанії або бути графічним зображенням, яке асоціюється з брендом.
Фірмові шрифти	Спеціально обрані шрифти, що використовуються в усіх комунікаціях компанії. Вони повинні бути читабельними та відповідати стилю бренду.
Фірмові кольори	Це набір кольорів, що асоціюються з брендом і використовуються в усіх його візуальних матеріалах.
Гасло	Коротке й легке для запам'ятовування висловлювання, яке відображає цінності чи обіцянки бренду.
Стиль комунікацій	Манера, в якій компанія спілкується з клієнтами, зокрема в текстах, відео, рекламі.