Інна Тріщук
Олександр Лазарець

ІНФОРМАТИКА

ПІДРУЧНИК ДЛЯ 9 КЛАСУ
закладів загальної середньої освіти

Рекомендовано Міністерством освіти і науки України

ТЕРНОПІЛЬ
НАВЧАЛЬНА КНИГА — БОГДАН
2026

УДК 004(075.2)
Т 67

Рекомендовано Міністерством освіти і науки України
(наказ Міністерства освіти і науки України від 27.01.2026, № 110)

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 9 кл. закладів загальн. середн. освіти / І. В. Тріщук, О. Ю. Лазарець. — Тернопіль : Навчальна книга — Богдан, 2026. — _ с.

ISBN 978-966-10-9224-1

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 9 класу.

УДК 004(075.2)

Завантажуйте безкоштовний інтерактивний електронний додаток, використовуючи детальну інструкцію, за покликанням:
https://edodatok.com/9224-1/

ISBN 978-966-10-9224-1

Шановні дев'ятикласниці та дев'ятикласники!

Перед вами — підручник з інформатики, який допоможе вам не лише закріпити вже здобуті знання, а й зануритися у захопливий світ цифрових технологій ще глибше! У 9 класі на вас чекає практична робота з тривимірним моделюванням, базами даних, створенням цифрових продуктів та розробкою власних мініпроєктів.

Ви навчитеся:

- створювати 3D-моделі будинків, меблів та інтер'єрів;
- працювати з базами даних і створювати запити;
- програмувати у середовищі Python;
- безпечно користуватись цифровими сервісами та штучним інтелектом;
- реалізовувати власні ідеї за допомогою сучасних цифрових інструментів.

Підручник поділено на модулі, кожен з яких містить:

- структуровану та чітку теорію;
- практичні завдання на комп'ютері;
- творче та додаткове завдання;
- QR-коди для перегляду презентацій.

Розділи поділено на теми, кожна з яких містить теоретичний матеріал, практичну роботу за персональним комп'ютером та домашнє завдання. Також, скориставшись покликанням 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

ЖИТТЯ З РОЗУМНИМИ ПРИСТРОЯМИ

Що таке інтернет речей (IoT)? Архітектура інтернету речей

Тема 1

Ключові поняття

Інтернет речей
Архітектура інтернету речей
Рівень пристроїв (сенсори й виконавчі механізми)
Мережевий рівень
Рівень обробки даних (сервер або хмара)
Рівень застосування (інтерфейс користувача)

Інтернет речей (Internet of Things, IoT) — це мережа фізичних пристроїв, які мають сенсори, програмне забезпечення та можливість підключення до інтернету, щоб обмінюватися даними без участі людини.

Ключові ознаки IoT-пристроїв:
- Збір даних
- Передача даних через мережу
- Обробка та реагування на дані

Термін «Інтернет речей» (Internet of Things, IoT) був запропонований в 1999 році Кевіном Ештоном, одним з трьох засновників Центру автоматичної ідентифікації Массачусетського університету (Auto-ID Center).

Приклади IoT у повсякденному житті

Розумний термостат

Автоматично регулює температуру

Смарт-браслет

Вимірює пульс, сон, активність

GPS-трекер

Визначає місцезнаходження в реальному часі

Інтерактивна дошка

Вона фіксує активність на уроці (що і коли відображалось), збирає аналітику взаємодії з контентом, а також може передавати дані в хмару для збереження або спільного доступу. Це дозволяє створити безперервний цифровий освітній процес.

відеокамера з детектором руху

Повідомляє про рух, надсилає відео у додаток

Розумні ваги

Аналізують масу тіла, індекс жиру, відправляють дані в додаток

Як працює IoT? Загальний принцип

Сенсор/пристрій збирає дані (температура, рух, звук, серцебиття тощо).

Передача даних через інтернет або локальну мережу.

Обробка даних на сервері або в хмарі.

Реакція або дія: команда пристрою (увімкнути, повідомити, змінити стан).

Архітектура інтернету речей

Рівень пристроїв (сенсори й виконавчі механізми)

Це «очі» та «вуха» системи. Пристрої на цьому рівні фіксують зміни у навколишньому середовищі — температуру, світло, рух, вологість, пульс тощо.

Приклад: сенсор температури в термостаті виявляє, що в кімнаті стало холодно.

Мережевий рівень

Цей рівень відповідає за передачу зібраних даних від пристрою до обробника або сервера. Для цього використовуються **Wi-Fi**, **Bluetooth**, мобільні мережі (3G/4G/5G) чи спеціальні IoT-протоколи (наприклад, **LoRaWAN**).

Приклад: фітнес-браслет надсилає дані про кількість кроків через **Bluetooth** на смартфон.

Рівень обробки даних (сервер або хмара)

Тут дані зберігаються, аналізуються і перетворюються на корисну інформацію. Обробка може відбуватись на хмарному сервері або локальному комп'ютері.

Приклад: хмарна система аналізує показники з датчиків і визначає, коли вмикати полив рослин у теплиці.

Рівень застосування (інтерфейс користувача)

Це все, з чим взаємодіє людина: мобільний додаток, вебінтерфейс, панель керування. Саме тут користувач бачить результати, змінює налаштування або отримує сповіщення.

Приклад: у додатку на смартфоні користувач бачить температуру в будинку й натискає кнопку, щоб увімкнути обігрівач.

Чим IoT корисний і небезпечний водночас?

Інтернет речей значно поліпшує якість життя: автоматизує побут, забезпечує контроль і підвищує комфорт.

Проте водночас IoT-пристрої збирають велику кількість персональних даних, що створює ризики для приватності та безпеки.

Переваги

- Автоматизація повсякденних дій
- Комфорт і зручність
- Економія ресурсів (електроенергії, води тощо)
- Швидкий доступ до даних у реальному часі
- Можливість задовольнити потреби людей з інвалідністю чи обмеженнями (зір, слух, мовлення, мобільність), особливостями розвитку або необхідністю контролювати прийом ліків

Ризики

- Збір і витік персональних даних
- Потенційне стеження за користувачем
- Залежність від стабільного інтернет-з'єднання
- Низький рівень захисту багатьох IoT-пристроїв

Практична робота №1.

Мій цифровий профіль: який вигляд має моя цифрова тінь?

Середовище виконання ПР: таблиця в **Canva** та презентація в **Canva**

Хід роботи

КРОК 1. Збір інформації про свої цифрові сліди (15–20 хв)

Випишіть всі гаджети та пристрої, якими користуєтеся щодня: смартфон, ноутбук, планшет, фітнес-браслет, смарт-годинник, колонки тощо.

Визначте всі типи даних, які збираються з цих пристроїв:

- геолокація (де ти перебуваєш)
- історія переглядів (**YouTube**, браузер)
- фото та відео
- біометричні дані (сон, пульс)
- списки контактів
- історія покупок
- голосові команди (**Google Assistant**, **Siri**)
- листування (месенджери)
- дії в соцмережах (лайки, коментарі)

КРОК 2. Побудова таблиці «Моя цифрова тінь»

Пристрій/ сервіс	Які дані збирає	Для чого це використовується	Потенційні ризики

Приклад

Google акаунт

Google фіксує історію наших пошуків, переглядів на **YouTube** та дій у браузері **Chrome**. Ці дані допомагають зробити пошук ефективнішим і краще підбирати контент. Проте виникає ризик профілювання — коли створюється детальний цифровий портрет особи, який може бути використаний з рекламною або навіть політичною метою.

Instagram

Instagram зберігає фотографії, які ми публікуємо, наші вподобання (лайки), коментарі та місце перебування. Ця інформація використовується для створення рекомендацій і формування стрічки новин. Проте надмірна присутність у соцмережі може призводити до залежності від уваги, а також до прихованого трекінгу нашої активності.

КРОК 3. Створення цифрового профілю
Презентація (5–7 слайдів)
(за бажанням можете змінити порядок слайдів):

- Слайд 1: Мій цифровий профіль
- Слайд 2: Мої гаджети
- Слайд 3: Які дані я залишаю
- Слайд 4: Як ці дані використовуються
- Слайд 5: Мої цифрові ризики
- Слайд 6: Як я можу себе захистити
- Слайд 7: Висновок / Рефлексія (написати свої думки щодо цифрового сліду)

КРОК 4. Надіслати свої роботи (презентацію та таблицю) на визначену адресу / завантажити за покликанням для ознайомлення.

Тема 2 **Сфери застосування інтернету речей. Безпека, конфіденційність та етика в IoT**

Ключові поняття

Безпека IoT
Витік персональних даних

Інтернет речей впроваджується в різні сфери, де потрібен контроль, автоматизація, безпека або моніторинг даних.

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.

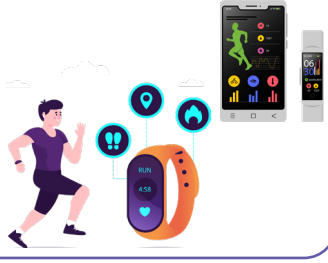
Основні сфери застосування IoT

Побут

У розумному будинку IoT-пристрої автоматизують освітлення, клімат-контроль, сигналізацію та побутові прилади. Наприклад, смарт-розетка може вмикати кавоварку за графіком, а сигналізація — надсилати сповіщення на телефон. Це створює комфорт і економить електроенергію.

Медицина

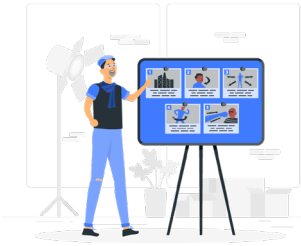
У медицині фітнес-браслети та медичні монітори постійно збирають дані про стан здоров'я людини: пульс, тиск, сон. Лікар може дистанційно отримати ці дані та вчасно зреагувати на загрозу. Це особливо важливо для людей з хронічними захворюваннями.



Транспорт

У розумному місті IoT керує світлофорами, парковками та громадським транспортом. Наприклад, датчики руху можуть автоматично перемикати світлофор, а паркомати — повідомляти про вільні місця. Це зменшує затори та підвищує безпеку на дорогах.

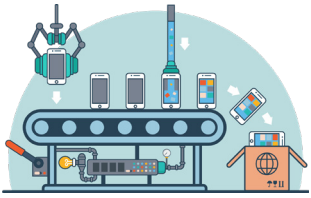
Освіта



В освітньому процесі використовуються смарт-дошки, цифрові щоденники, системи відстеження присутності. Учні можуть працювати з матеріалами в реальному часі, а вчитель бачить аналітику активності. Це робить навчання сучасним і персоналізованим.

Виробництво

На заводах IoT-сенсори контролюють температуру, вологість і стан обладнання. Система може попередити про можливу поломку ще до того, як вона станеться. Це економить кошти і запобігає аваріям.



Сільське господарство

Фермери використовують сенсори для моніторингу вологості ґрунту, температури та освітлення в теплицях. Система автоматично вмикає полив лише тоді, коли це справді потрібно. У результаті — економія води, кращі врожаї та менше ручної праці.

Конфіденційність: що про нас «знають» пристрої?

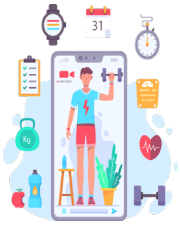
- Геолокація: де ми були й куди їдемо
- Стан здоров'я: пульс, сон, активність
- Поведінка: коли прокидаємось, як довго не рухаємось
- Смаки й інтереси: на основі переглядів, покупок

Безпека IoT означає захист даних, які пристрої збирають і передають.

Потенційні загрози

1. **Хакерські атаки**
злам інтернет-камер, дверних замків, сигналізацій, фітнес-трекерів. Наприклад, хакер може переглядати, що бачить камера, або віддалено відчиняти двері.

2. **Витік персональних даних**
ваші дані про здоров'я, місцезнаходження чи щоденні звички можуть потрапити до рук сторонніх осіб або бути продані без вашої згоди.



3. **Використання пристроїв як «ботів»**
зловмисники можуть використати IoT-пристрій у великій кібератаці на інші сервери (ботнет).

Етичні питання використання IoT

- Чи має право пристрій збирати дані без згоди людини?
- Як розумні пристрої впливають на автономію особистості?
- Чи мають розумні іграшки право спостерігати за дітьми?

Практична робота №2.

«Інтернет речей у сучасному світі: як він працює в різних галузях» — дослідницько-презентаційна робота
Колективна групова робота (5 команд)
Кожна команда обирає свою галузь:
1. Медицина
2. Транспорт
3. Промисловість
4. Побут (розумний дім)
5. Освіта

Завдання. Дослідіть, як у цій галузі використовуються пристрої інтернету речей (IoT). Знайдіть:

- що саме ці пристрої роблять,
- які дані вони збирають,
- яку користь це дає людям або підприємствам,
- чи є якісь ризики (безпека, приватність).

Організуйте результати пошуку в презентацію (можна це зробити в **Canva**)

1. Як у цій галузі використовується IoT?
Про що:
Коротке пояснення ролі IoT у вибраній сфері.
Як працює? Які процеси автоматизує?
Наприклад: У медицині пристрої в режимі реального часу передають фахівцям показники життєдіяльності (стану) пацієнта/пацієнтки.

2. Приклади пристроїв
Про що:
Наведіть 2–3 конкретні IoT-пристрої: назва + короткий опис, що вони роблять.
Наприклад: фітнес-браслет, сенсор температури, розумна лампа.

3. Які дані збираються цими пристроями?
Про що:
Опис типів даних, які пристрої фіксують і передають.
Наприклад: пульс, геолокація, температура, кількість кроків, вологість.

4. Економічна користь
Про що:
Пояснення, яку вигоду або економію дає впровадження IoT у цій галузі.
Наприклад: економія часу персоналу, менше витрат на енергію, підвищення ефективності.

5. Користь для людей з особливими потребами фізичного або ментального здоров'я.

6. Потенційні ризики

Про що:

Перелік можливих небезпек або проблем, пов'язаних із використанням пристроїв.

Наприклад: витік особистих даних, кібератаки, порушення конфіденційності.

Оформіть результати у вигляді:

- презентації (7-9 слайдів)
- інфографіки/постера (1 аркуш або слайд)
- усна розповідь до презентації

Підготуйтеся до виступу:

- Тривалість презентації: до 3 хвилин
- Можна виступати індивідуально (від свого імені) або розподілити порядок презентації частин усередині команди
- Розповідайте просто, зрозуміло, чітко.

Джерела для пошуку інформації:

- **Google** (ключові слова: IoT + назва галузі)
- Сайти: **http://iotforall.com** та інші
- Вікіпедія (українська/англійська)

Ключові поняття

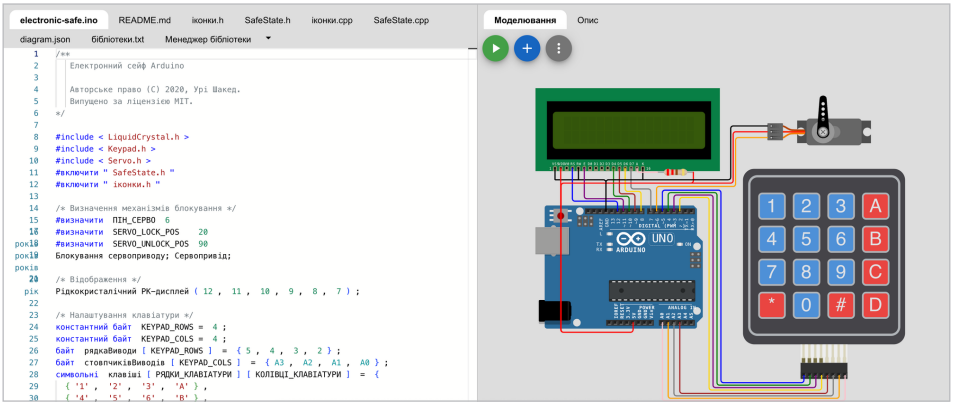
Інформаційна система IoT
Як працює система IoT

Що таке інформаційна система IoT?

Інформаційна система IoT — це набір пристроїв, які:

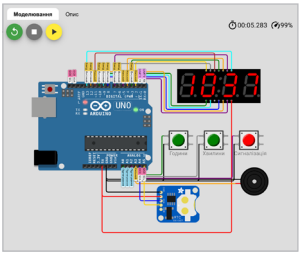
- збирають дані (датчики);
- обробляють їх (контролер або мікропроцесор);
- реагують на них (виконавчі пристрої);
- часто передають інформацію через інтернет.

Датчик руху ➡ контролер ➡ повідомлення на смартфон ➡ вмикання лампи.



Складові IoT-системи

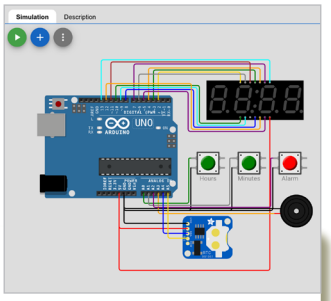
- Датчики (input): світла, температури, руху, вологості.
- Контролер (обробка): Arduino, ESP32, micro:bit тощо.
- Виконавчі пристрої (output): LED, двигуни, реле.
- Зв'язок: Wi-Fi, Bluetooth, іноді GSM.
- Програмне забезпечення: логіка «якщо — то», умови, обробка сигналів.



Як працює система IoT (приклад сценарію)

Умова:

Якщо поточний час збігається з встановленим часом будильника → увімкнути сигнал будильника.



Алгоритм:

- RTC-модуль (Real Time Clock) постійно передає поточний час у систему.
- Користувач встановлює години та хвилини будильника за допомогою кнопок Години і Хвилини.
- Контролер **Arduino** порівнює поточний час із часом будильника.
- Якщо час збігається → подається сигнал на дзвінок (buzzer) або динамік.
- Кнопка **Сигналізація** дозволяє увімкнути/вимкнути спрацювання будильника.

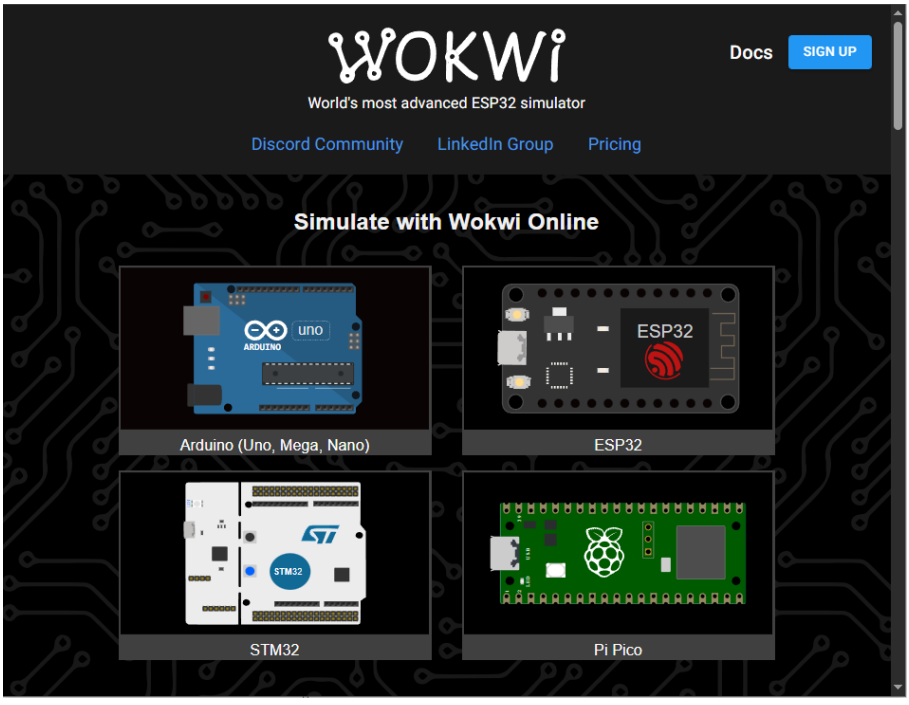
Практична робота №3.

Перегляд симуляцій IoT-систем
Сайт **https://wokwi.com/**

Wokwi — це безкоштовний онлайн-симулятор **Arduino, ESP32, Raspberry Pi** та інших мікроконтролерів. Він дозволяє створювати схеми, писати код, запускати симуляцію безпосередньо в браузері — без потреби мати фізичне обладнання.

Дослідження готових симуляцій

1. У розділі **Explore** перегляньте приклади:
   - Smart Thermometer
   - Motion Sensor Alarm
   - IoT Home Automation
2. Виберіть один приклад, відкрийте та проаналізуйте:
   - Що робить ця схема?
   - Які компоненти використано?
   - Яка логіка у коді?



Проєкт 1
Технічний/дизайнерський проєкт «Розумний клас майбутнього»

Теми 4–5

Мета проєкту: розробити макет класної кімнати, яка використовує пристрої інтернету речей (IoT) для підвищення безпеки, комфорту, ефективності навчання та здоров'я учнів.

QR-код / NFC вхід до класу – ідентифікація осіб, які заходять до класної кімнати.

Під час проєктування макету класу важливо врахувати принципи інклюзивності: забезпечити безбар'єрний доступ, адаптовані рішення для учнів з порушенням зору, слуху, мовлення, мобільності чи іншими особливими потребами. Це сприятиме створенню комфортного та безпечного середовища для всіх учасників навчального процесу.

Ідеї IoT-пристроїв для класу майбутнього:

- Система розпізнавання облич – дозволяє автоматично відмічати присутність.
- Кнопка тривоги або «SOS» – її натискають ідентифіковані особи в небезпечній ситуації.
- Автоматичне провітрювання – відчиняє вікна, якщо рівень CO_2 високий.
- Електронний розклад на стіні – автоматичне оновлення занять.
- Сонячні жалюзі з керуванням – реагують на рівень освітлення.
- Регулювання освітлення – лампи змінюють яскравість залежно від часу доби.
- Для кожного пристрою вкажіть, чи є він доступним для використання всіма учнями, включно з тими, хто має особливі освітні потреби.

Середовище виконання проєкту

Для створення макету або 3D-схеми класу:

Tinkercad
онлайн-сервіс для 3D-моделювання.

SketchUp
для більш просунутих користувачів, які хочуть деталізовану 3D-модель.

Для створення графічної схеми та презентації:

Canva – інтуїтивно зрозумілий сервіс для створення презентацій, макетів, схем.

Google Slides – прості у використанні інструменти для графічної схеми або презентації.

Google Документи – для текстів, сценаріїв, описів

Створення текстового документу для:
- формулювання проблем і завдань проєкту;
- опису кожного IoT-пристрою;
- підготовки тексту презентації (хто що говорить).

Trello – для планування кроків проєкту

Візуальна дошка задач із картками:
- **Зробити** – що заплановано.
- **У процесі** – над чим зараз працюють.
- **Готово** – виконані завдання.

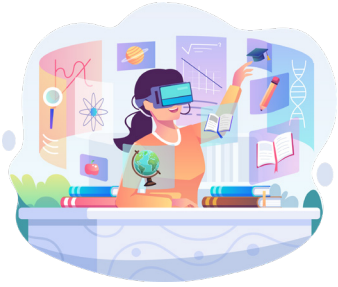
До кожної задачі можна прикріпити відповідальну особу.

Google Диск — для збереження всіх матеріалів команди

Спільна папка команди, де зберігаються:
- ескізи, чернетки, скріншоти;
- **Canva**-посилання;
- макети з **Tinkercad**.

Результат вашої командної роботи:

- Макет або графічна схема класу з розміщенням 5–7 IoT-пристроїв (**Tinkercad**, **SketchUp**, **Canva**).
- Опис кожного пристрою (назва, функція, проблема, яку усуває, розташування).
- Презентація команди (3–5 хвилин) із поясненням логіки рішення. Презентація здійснюється недискримінаційною мовою.
- Коротка рефлексія: який пристрій найкорисніший, які ризики існують, як зробити рішення безпечним.



Ідея і план

- Обговоріть проблеми звичайного класу (шум, повітря, освітлення, контроль часу, безпека, інклюзивність, рівень доступності та пристосованості до навчання для людей із особливими потребами).
- Оберіть 5–7 IoT-пристроїв, які розв'язують ці проблеми.

Створення макету

- Оберіть зручний інструмент:
  - **Tinkercad** / **SketchUP** – для **3D**
  - **Canva** / **Google Slides** — презентація **IoT**
- Додайте підписи:
  - Назва
  - Що робить?
  - Яку проблему розв'язує?
  - Чи збирає дані?
  - Чи безпечний для користувача?

Додатково визначте, як ваше рішення враховує потреби учнів і учениць з інвалідністю або інших вразливих груп, та які функції пристроїв забезпечують інклюзивність.

Матеріали для презентації проєкту

- Підготовки презентації проєкту
  - Назва проєкту та склад команди
  - Проблеми звичайного класу
  - Рішення через IoT
  - Опис пристроїв (5–7)
  - Схема або макет класу
  - Висновки та рефлексія
  - Підсумок і відповіді на запитання
- Ролі у виступі:
  - Вступ – коротко про назву і мету
  - Основна частина – по черзі розповідають про пристрої
  - Висновок і рефлексія – про користь, безпеку, інноваційність

Презентація проєкту

- Представте свій проєкт за 3–5 хвилин.
- Поясніть, чому саме ці пристрої вибрані.

Пам'ятайте про застосування недискримінаційної мови.

Індивідуальна дослідницька робота

- Дослідити, як інтернет речей уже впливає на ваше життя, проаналізувати приклади застосування, переваги, ризики та особливості конфіденційного використання.
- Скласти план 5-7 речей IoT, які ви хочете використовувати у своєму житті.

Приклади IoT-пристроїв:

- Розумна колонка
- Розумний термостат
- Браслет здоров'я
- Розумне освітлення
- QR-коди на транспорті
- GPS-трекер для домашніх тварин

Середовище виконання проєкту

Canva	інтуїтивно зрозумілий сервіс для створення презентацій, макетів, схем
Google Slides	прості у використанні інструменти для графічної схеми або презентації
Google Документи	для текстів виступу

Результат індивідуального дослідницького проєкту

Презентація в Canva

- Опис 3–5 IoT-пристроїв, які використовуєте або плануєте використовувати (що робить, яку проблему усуває, які дані збирає).
- План: 5–7 IoT-речей, які ви хочете мати в майбутньому, з поясненням вибору.
- Аналіз впливу на ваш цифровий слід: які дані зберігаються, які ризики можуть бути.
- Рекомендації щодо конфіденційності: як захищати свої дані, що варто врахувати.

Структура презентації дослідницької роботи

- Аналіз 3–5 прикладів IoT-пристроїв, які використовуєте або можете використовувати:
  - Що це за пристрій?
  - Як він працює?
  - Які дані збирає?
  - Яку користь приносить?
  - Які ризики має?
- 5–7 IoT-речей, які хочете мати у своєму житті
  - Складіть перелік з коротким описом, чому саме ці пристрої (переваги, зручність, безпека, ефективність тощо).

Наприклад: смарт-лампа, датчик повітря, фітнес-браслет, відеодзвінок, система поливу, розумний холодильник.

- Вплив на цифровий слід:
  - Які дані ці пристрої можуть зберігати про вас (місцезнаходження, режим сну, звички, емоції тощо)?
  - Чи можливо ці дані використовувати не за призначенням?
  - Яку частину цифрового сліду ви залишаєте щодня з такими пристроями?

Презентація проєкту

- Представте свій проєкт за 3–5 хвилин.
- Поясніть, чому саме ці пристрої вибрані.

Колективна робота

Робота у парах або трійках.

Мета проєкту: розробити ідею інноваційного IoT-пристрою (опис, яку проблему розв'язує, як працює).

Результат вашої командної роботи:

- Ідея інноваційного IoT-пристрою: опис, яку проблему розв'язує, як працює.
- 3D-модель пристрою (**ThinkerCad** або **SketchUp**).
- Назва, логотип і слоган пристрою (візуальний бренд).
- Рекламний продукт:
  або відеореклама (30–60 с.)
  або рекламний постер / банер.
- Презентація команди (до 5 хв).

Середовище виконання проєкту

Tinkercad | SketchUp | Canva | Google Slides

Google Документи | Trello | Google Диск

Генерація ідеї + проєктування

- Визначення проблеми
  – Вправа: «Що мене щодня турбує? Що я хочу поліпшити?»
- Мозковий штурм рішень
  – Які функції має пристрій? Які сенсори? Зв'язок зі смартфоном? Збір даних?
- Розробка бренду
  – Назва, слоган, логотип.

Пам'ятайте про врахування потреб вразливих груп, уникнення їх дискримінації та забезпечення інклюзії

Створення прототипу пристрою

- У середовищі **Tinkercad** або **SketchUp**.
- Простий 3D-макет або модель, що демонструє зовнішній вигляд та елементи пристрою.

Рекламний продукт

- Відеореклама (30-60 секунд).
- Рекламний постер / банер.

Все робимо в **Canva**

Презентація проєкту

- Представте свій проєкт за 3–5 хвилин.
- Поясніть, чому саме цей пристрій вибраний.

Поняття бази даних. Основні терміни (поле, запис, типи даних, ключі)

Тема 11

Ключові поняття

- База даних
- Поле, записи
- Основні типи даних
- Ключове поле

Що таке база даних?

Уявіть собі шкільну бібліотеку. У ній є сотні книг. Щоб знайти потрібну, бібліотекарка користується спеціальним каталогом. У ньому зберігається інформація про всі книги: назва, автор, рік видання, жанр тощо. Таким чином бібліотекарка швидко знайде цю книгу.

Такий каталог — приклад бази даних, тобто впорядкованої структури для зберігання та обробки інформації.

База даних (БД) — це впорядкований набір інформації, яку можна зберігати, шукати, змінювати або аналізувати.

У комп'ютері база даних працює схожим чином, але замість книг там зберігаються, наприклад, списки учнів, розклади уроків чи навіть інформація про ваші вподобання (улюблений спортивний клуб, кіно, книги, музику тощо) або додаткові потреби (безпечні продукти харчування, медична інформація, перелік та режим приймання ліків, рухової активності тощо).

Приклади баз даних:

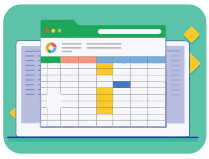
- у магазині — накладна, де вказано назви товарів, ціну, кількість;
- у смартфоні — список контактів;
- у класному журналі — база оцінок і відомостей про учнів та учениць

Основні терміни

Щоб добре зрозуміти, як працює база даних, потрібно знати кілька основних понять.

Поле — це окремий елемент інформації в базі даних, як колонка в таблиці.

Уявіть таблицю про своїх друзів. У ній є колонки: «Ім'я», «Вік», «Улюблений колір». Кожна колонка — це поле. Наприклад, поле «Ім'я» може містити значення «Оля», «Максим» або «Софія». У кожному полі зберігається певний тип даних (наприклад, текст або число).



Запис — це рядок у таблиці, який містить повну інформацію про один об'єкт. Кожен запис складається з кількох полів.

У тій же таблиці про друзів один рядок (запис) може бути такий: «Оля, 12, синій». Цей запис містить усю інформацію про одну людину — Олю.

Або ж на прикладі учня запис можна подати в такому вигляді:

Ім'я: Олександр. Прізвище: Лазарець. Дата народження: 12.02.2010. Клас: 9-А. Середній бал: 10.5.

Потрібно зауважити: одна особа у таблиці або інформація про одного друга чи подругу — це один запис.

Типи даних

Кожне поле зберігає інформацію певного типу. Тип даних — це вказівка, якого виду буде інформація.

Основні типи даних

Текстовий	У тій же таблиці про друзів один рядок (запис) може бути такий: «Оля, 12, синій». Цей запис містить усю інформацію про одну людину — Олю.	
Числовий	Для віку, оцінок, цін, наприклад, 14 або 95.50.	
Дата/час	Для дат народження, розкладу, наприклад, 15.03.2010.	
Логічний	Для відповідей «так» чи «ні», наприклад: «Чи любить піцу? — Так».	

Уявіть, що записуєте дані про домашніх тварин. Поле «Кличка» — це текст (наприклад, «Мурка»), поле «Вік» — це число (наприклад, 3), а поле «Дата вакцинації» — це дата (наприклад, 10.07.2025).

Ключ (ключове поле) — це поле, яке унікально ідентифікує кожен запис.

Приклад:

У класному журналі двоє учнів можуть мати однакові імена. Але якщо кожному учневі присвоїти **унікальний номер (ID)** — це і буде ключ.

Наприклад:

```
ID: 101, Ім'я: Марія, Прізвище: Іваненко
ID: 102, Ім'я: Марія, Прізвище: Петрів
```

Ключ допомагає уникнути плутанини та дозволяє точно знайти потрібний запис.

Як це працює на практиці?

Уявіть, що вам потрібно створити базу даних для шкільної бібліотеки. У ній є таблиця «Книги», яка має такий вигляд:

ID	Назва книги	Автор	Рік видання	Наявність
001	Гаррі Поттер	Дж. Ролінг	2017	Так
002	Енеїда	І. Котляревський	2021	Так
003	Гоббіт	Дж. Толкін	2021	Ні

Поля: ID, Назва книги, Автор, Рік видання, Наявність.

Записи: кожен рядок — це інформація про одну книгу.

Типи даних: ID — числовий, Назва книги та Автор — текстові, Рік видання — числовий, Наявність — логічний.

Ключове поле: ID, бо він унікальний для кожної книги.

Чому бази даних важливі?

Бази даних використовуються всюди: у магазинах (для обліку товарів), у лікарнях (для медичних карток), у соцмережах (для зберігання постів і профілів). Вони допомагають швидко обробляти великі обсяги інформації.

Уявіть, що ви шукаєте улюблену пісню в музичному додатку. База даних цього додатка швидко знаходить пісню за назвою чи виконавцем/виконавицею, бо вся інформація чітко організована.

Практична робота №11

Індивідуальна робота

Завдання 1. Складіть та заповніть таблицю з 4-5 учнями свого класу. Поля: ID, Ім'я, Прізвище, Клас, Середній бал.

- Укажіть, які типи даних підходять для кожного поля у вашій таблиці.
- У таблиці оберіть поле, яке можна зробити ключовим. Чому?

Завдання 2. Створіть таблицю для зберігання інформації про улюблені фільми своїх друзів. Визначте:

- Які поля будуть у таблиці (наприклад, «Назва фільму», «Рік», «Жанр»).
- Який тип даних матиме кожне поле.
- Яке поле буде ключовим.
- Заповніть таблицю мінімум трьома записами.

Завдання 3. Створіть базу даних для парку розваг, щоб вести облік атракціонів. Виконайте:

- Визначте мінімум 5 полів (наприклад, «Назва атракціону», «Тип», «Мінімальний вік», «Ціна квитка», «Час роботи»).
- Вкажіть тип даних для кожного поля.
- Виберіть ключове поле.
- Заповніть таблицю 3–4 записами.
- Напишіть два запитання, які можна розв'язати за допомогою цієї бази даних (наприклад, «Які атракціони підходять для дітей до 10 років?»).

Реляційна база даних. Визначення об'єктів предметної області

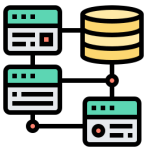
Тема 12

Ключові поняття

Реляційна база даних
Предметна область
Зв'язки між таблицями

Що таке реляційна база даних?

Реляційна база даних — це спосіб організації інформації, де дані зберігаються у вигляді таблиць, які пов'язані між собою.



Кожна таблиця містить інформацію про певний тип об'єктів, наприклад, про учнів/учениць, книги чи товари. Таблиці з'єднуються за допомогою спеціальних полів, які називаються ключами, щоб можна було легко знаходити потрібну інформацію.

Приклад із життя: уявіть шкільну їдальню. У них є таблиця з меню (страви, ціни) і таблиця з учнями (імена, класи). Якщо хочете знати, хто замовив борщ, ці таблиці можна пов'язати через унікальний номер замовлення. Це і є реляційна база даних — інформація розбита на окремі таблиці, але їх можна об'єднати.

ID	Назва страви	Ціна (грн)
1	Борщ	30
2	Вареники	25
3	Каша гречана	20

Номер замовлення	Ім'я учня/учениці	Клас	ID страви
2001	Марія	7–В	1
2002	Іван	9–А	3

Із запису Z001 відомо, що Марія замовила борщ, бо ID страви — 1. Це і є зв'язок між таблицями.

Що таке предметна область?

Предметна область — це частина реального світу, про яку ми хочемо зібрати та зберігати інформацію в базі даних.

Наприклад:

Якщо ми створюємо базу даних шкільної бібліотеки, предметною областю буде саме бібліотека: книги, учні, автори, видачі книг.

Якщо створюємо базу даних кафе, предметна область — це кафе: страви, персонал, замовлення, гості.

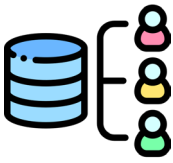
Тут потрібно вирішити, які об'єкти важливі для бази даних.

Об'єкти предметної області

Об'єкти предметної області — це основні елементи, про які ми хочемо зберігати інформацію.

Кожен об'єкт стає окремою таблицею в реляційній базі даних. У кожній таблиці є поля (колонки), які описують властивості об'єкта, і записи (рядки), які містять конкретні дані.

Як визначити об'єкти?



1. **Визначте, про що база даних**
   Наприклад, якщо це база даних для бібліотеки, то вона про книги, читачів(-ок) і видачу книг.

2. **Назвіть основні об'єкти**
   У бібліотеці це можуть бути «Книги», «Читачі(-чки)», «Видача книг».

3. **Опишіть властивості об'єктів**
   Для «Книги» це можуть бути поля: назва, автор, рік видання. Для «Читача(-чки)» — ім'я, прізвище, номер читацького квитка.

4. **Визначте зв'язки між об'єктами**
   Наприклад, таблиця «Видача книг» пов'язує «Книги» і «Читачі(-чки)», бо показує, хто взяв яку книгу.

Як пов'язуються таблиці?

У реляційній базі даних таблиці з'єднуються за допомогою ключів.

Первинний ключ — унікальний ідентифікатор для кожного запису в таблиці (наприклад, ID учня).

Зовнішній ключ — поле в одній таблиці, яке посилається на первинний ключ іншої таблиці.

Наприклад, маємо базу даних для улюбленого футбольного клубу. Є таблиці:

Гравці/гравчині: ID, Прізвище, Позиція.

Матчі: ID, Дата, Суперник.

Голи: ID, ID гравця, ID матчу, Час голу.

Таблиця «Голи» пов'язує «Гравці/гравчині» і «Матчі» через зовнішні ключі (ID гравця та ID матчу). Це дозволяє дізнатися, хто і коли забив гол.

Типи зв'язків між таблицями

Тип зв'язку	Що означає	Приклад
Один до одного (1:1)	Один об'єкт пов'язаний лише з одним іншим	У кожного учня є один учнівський квиток
Один до багатьох (1:N)	Один об'єкт пов'язаний з кількома іншими	Один автор написав кілька книг
Багато до багатьох (M:N)	Кілька об'єктів пов'язані з кількома іншими	Учні записані на кілька гуртків

Розклад транспорту

Об'єкти: Автобуси, Маршрути, Водії

Зв'язки:

Один автобус → один маршрут

Водій/водійка → декілька автобусів

Шкільна бібліотека

Об'єкти: Книги, Учні(-ці), Видачі

Зв'язки:

Учень/учениця → (може взяти) → багато книг

Книга → (може бути видана) → багатьом учням/ученицям

Чому це важливо?

Реляційні бази даних допомагають ефективно організувати інформацію. Завдяки зв'язкам між таблицями можна швидко знаходити потрібні дані, наприклад:

- Які книги взяла в бібліотеці певна особа?
- Які товари людина купила в магазині?
- Хто грає головну роль у спектаклі такого-то дня?

Якщо ж ви шукаєте улюблену гру в онлайн-магазині, то база даних магазину швидко знаходить гру, показує її ціну, відгуки та чи є вона в наявності, бо всі ці дані пов'язані між собою.

Практична робота №12

Завдання 1. Оберіть одну з тем для бази даних:
Шкільна їдальня
Онлайн-магазин
Домашні тварини

Опишіть предметну область.
Визначте 3–4 об'єкти.
Для кожного об'єкта — придумайте 3–5 полів.
Укажіть зв'язки між об'єктами.

Завдання 2. На основі попереднього завдання:
Складіть таблиці для кожного об'єкта.
Укажіть ключові поля.
Придумайте 2–3 записи для кожної таблиці.

Завдання 3.

БД для кабінету (курсу, уроків тощо) біології

Створіть базу даних «Рослини шкільного двору».

Таблиці:
- Рослини (назва, тип, висота, цвітіння).
- Місце посадки (ділянка, сторона школи, тип ґрунту).
- Чергові (відповідальні особи тощо), які доглядають.

Опиши зв'язки:
- Одна рослина — одне місце посадки.
- Одна особа може доглядати за кількома рослинами.



Ключові поняття

Інтерфейс бази даних
Можливості інтерфейсу
Обмеження інтерфейсу

Що таке інтерфейс бази даних?

Інтерфейс бази даних — це спосіб, за допомогою якого люди взаємодіють із базою даних, щоб додавати, отримувати, переглядати, змінювати чи видаляти інформацію.

Іншими словами, це «вітрина» («ґанок», «вхідні ворота» тощо) бази даних, тобто зручний спосіб, за допомогою якого користувач може працювати з даними, навіть не знаючи програмування чи внутрішньої будови БД.

Уявімо приклад із життя:

Шкільна їдальня має комп'ютер, де збережено базу даних про страви, ціни, учнів/учениць та замовлення.

Пані Оксана, яка не є програмісткою, працює з базою через готовий інтерфейс: вона відкриває форми, натискає кнопки — і отримує інформацію, хто що замовив.

Приклади інтерфейсів

Програми на комп'ютері

Наприклад, **Microsoft Access**, **LibreOffice Base**, **MySQL**, **MongoDB** тощо.

Вебсайти

Мобільні додатки

Прості форми

Наприклад, **Microsoft Excel**, **Google Sheets**.

Тип інтерфейсу	Приклад	Опис
Графічний інтерфейс	Microsoft Access, Google Таблиці	Таблиці, кнопки, форми, звіти
Веб-інтерфейс	Онлайн-магазин, шкільний щоденник	Користувач працює з базою через браузер
Мобільний інтерфейс	Програма доставки їжі	Дані вводяться/отримуються через телефон

Основні можливості інтерфейсу бази даних

Інтерфейс бази даних допомагає виконувати різні дії з інформацією. Ось основні можливості:

Зрозуміла навігація

Зрозуміла навігація передбачає врахування потреб стосовно способів отримання інформації у людей (наприклад, із порушеннями зору чи слуху). Завдяки зрозумілій навігації можна ознайомитися з інформацією у зручний для користувача/користувачки спосіб.

Пошук даних

Можна швидко знайти потрібну інформацію, ввівши запит. Наприклад, у базі даних шкільного журналу можна знайти всіх, хто отримали оцінку «10» з математики. Або в інтернет-магазині шукаєте навушники. Вводите у пошуковий рядок «бездротові навушники до 1000 грн», і на сайті знаходите список товарів. Це інтерфейс бази даних магазину шукає товари за вашими критеріями.

Додавання даних

Інтерфейс дозволяє додавати нові записи до бази даних. Наприклад, у зоомагазині з'явилася нова тварина. Працівник/працівниця відкриває програму, заповнює форму з інформацією (кличка, вид, ціна), і дані зберігаються в базі.

Редагування даних

Можна змінювати вже наявну інформацію. Наприклад, якщо в учня/учениці інша адреса, її можна оновити в базі через інтерфейс.

Видалення даних

Інтерфейс дозволяє видаляти непотрібні записи. Наприклад, якщо книга в бібліотеці загубилася, її можна видалити з бази.

Фільтрація та сортування

Інтерфейс допомагає відбирати дані за певними умовами (фільтрація) або впорядковувати їх (сортування). Наприклад, у музичному додатку можна відфільтрувати пісні за жанром (тільки попмузику) або відсортувати їх за датою додавання.

Звіти та аналітика

Інтерфейс може створювати звіти, які показують узагальнену інформацію. Наприклад, у шкільній їдальні менеджер/менеджерка може через інтерфейс отримати звіт, які страви є найпопулярнішими.

Обмеження інтерфейсу бази даних

Залежність від дизайну інтерфейсу

Якщо інтерфейс погано спроєктований (наприклад, незрозумілі кнопки, чи інструкції щодо введення або складні форми), з ним буде незручо і складно працювати.

Уявіть, що в бібліотеці комп'ютерна програма для пошуку книг має маленькі кнопки і заплутане меню: тоді вам буде важко знайти потрібну книгу, навіть якщо вона є в базі.

Обмежений доступ до даних

Інтерфейс показує тільки те, що запрограмовано. Якщо розробники не додали можливість шукати за певним критерієм, ви не зможете цього зробити.

У базі даних магазину є інформація про колір товару, але якщо в пошуковій формі немає фільтра за кольором, ви не зможете, наприклад, знайти всі червоні рюкзаки.

Помилки користувача

Інтерфейс не завжди може перевірити, чи правильно ви вводите дані. Наприклад, у шкільній базі даних відповідальна особа випадково ввела неправильну дату народження учня, і тепер система вважає, що йому 99 років.

Обмеження прав доступу

Інтерфейс може обмежувати доступ до певних даних залежно від ролі користувачів. Наприклад, у шкільному журналі всі бачать свої оцінки через додаток, але змінювати їх може тільки вчитель/вчителька.

Технічні обмеження

Інтерфейс залежить від програми чи пристрою. Якщо програма застаріла або не працює на вашому комп'ютері, ви не зможете отримати доступ до бази даних.

Уявіть, що в бібліотеці програма для пошуку книг працює тільки на старому комп'ютері, який постійно зависає. Це ускладнює роботу.

Який вигляд має інтерфейс у реальному житті?

Уявімо, що в шкільній бібліотеці є база даних.

Через інтерфейс бібліотеки можна:

- шукати книгу за автором/авторкою або назвою;
- переглядати, у кого зараз книга;
- видати нову книгу на запит.

Але:

- не можна змінити назву таблиці;
- не можна додати поле «Жанр», якщо його спочатку не було;
- не можна змінити логіку пошуку (наприклад, за двома критеріями одночасно — автор/авторка і рік).

Практична робота №13

Завдання 1. Наведіть три приклади з життя, де ви взаємодієте з базою даних через інтерфейс. Наприклад:

- Сайт запису до медичного закладу
- Замовлення піци через застосунок
- Онлайн-кіноплатформа (**Netflix**, **Megogo**)

Для кожного прикладу напишіть:
— які дії ви можете робити через інтерфейс?
— що вам не дозволяють?
— які обмеження найчастіше трапляються?

Завдання 2. Спроєктуйте інтерфейс бази даних магазину спортивних товарів, яка містить інформацію про товари (назва, тип, ціна, кількість на складі)

- Опишіть, які функції матиме інтерфейс (наприклад, пошук товарів за типом, додавання нового товару).
- Намалюйте макет або створіть у **PowerPoint/Canva/Google Form**.

Приклад полів форми:

- Назва товару: Футбольний м'яч
- Тип: М'яч
- Ціна: 600
- Кількість на складі: 10
- (Натиснути «Додати товар»)

Ключові поняття

Таблиці в базах даних
Режими створення та перегляду таблиць
Поле OLE
Створення зв'язків між таблицями

Створення таблиць у Microsoft Access

Таблиця — це основний об'єкт бази даних, у якому зберігаються всі дані.

Microsoft Access — це система управління базами даних (СУБД), яка входить до пакету **Microsoft Office**. Вона використовує реляційну модель даних, де інформація зберігається у таблицях, пов'язаних між собою за допомогою ключів.

ID	Назва книги	Автор	Жанр	Наявність
1	Гаррі Поттер	Дж. К. Ролінґ	Фентезі	☑
2	Маленький принц	Антуан де Сент-Екзюпері	Казка	☐
3	Пригоди Тома Сойєра	Марк Твен	Пригоди	☑
4	1984	Джордж Орвелл	Антиутопія	☑

Кожна таблиця складається з рядків (записи) і стовпців (поля).

Основні принципи створення таблиць

Планування структури бази даних

- Визначте мету бази даних.
- Визначте, які таблиці потрібні та які дані вони міститимуть.
- Встановіть зв'язки між таблицями (наприклад, один до багатьох, багато до багатьох).
- Уникайте надлишковості даних (нормалізація).

Угода про іменування

- Використовуйте зрозумілі назви для таблиць і полів (наприклад, «Клієнти/Клієнтки», «Замовлення»).
- Уникайте пробілів, спеціальних символів і зарезервованих слів у назвах. Наприклад: замість «Список клієнтів» використовуйте «Список_Клієнтів» або «Клієнти».

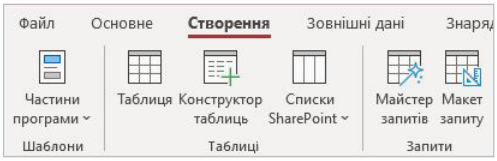
Вибір правильних типів даних

Кожне поле таблиці має тип даних, який визначає, які значення можна вводити (текст, число, дата тощо).

Створення таблиць у Microsoft Access

У **Microsoft Access** таблиці можна створювати кількома способами.

У режимі таблиці
(введення даних із автоматичним створенням структури)



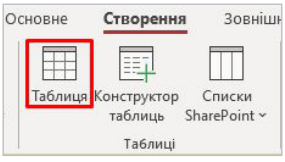
У режимі конструктора
(ручне визначення структури таблиці)

За допомогою майстра таблиць
(автоматизоване створення на основі шаблонів)

Імпортом
або зв'язком із зовнішніми джерелами даних

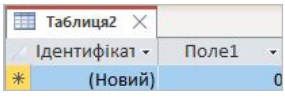
Створення таблиці в режимі таблиці

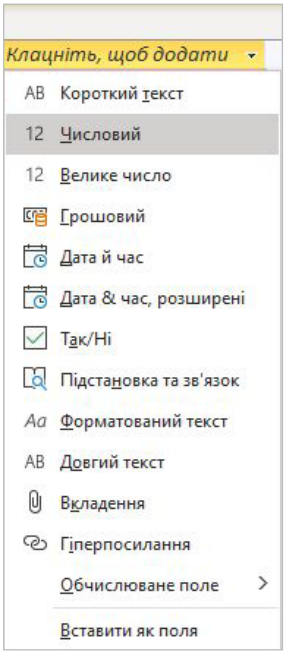
Цей спосіб підходить для швидкого створення таблиці з автоматичним визначенням структури.



Кроки

1. Відкрити **Microsoft Access** і створити нову базу даних.
2. На вкладці **Створити (Create)** у групі **Таблиці (Tables)** натиснути **Таблиця (Table)**.
3. З'явиться порожня таблиця з полем **Ідентифікатор** (автонумерація).
4. Натиснути кнопку **Клацніть, щоб додати**, щоб додати інші поля, обравши тип даних цього поля.
5. Подвійним натисканням змінити назву поля.
6. Ввести дані безпосередньо в таблицю, змінюючи назви полів і типи даних за потреби.
7. Після введення даних зберегти таблицю, присвоївши їй ім'я.



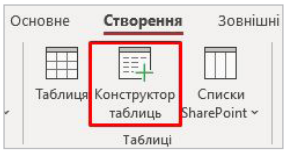


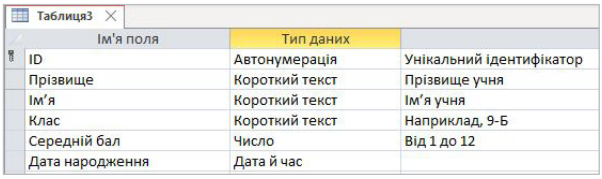
Особливості

- Поле **Ідентифікатор** автоматично створюється як первинний ключ із типом **Автонумерація**.
- Для зміни типів даних або структури потрібно перейти до режиму конструктора.
- Цей спосіб зручний для початківців, але менш гнучкий, ніж режим конструктора.

Створення таблиці в режимі конструктора

Режим конструктора дозволяє точно визначити структуру таблиці, включаючи імена полів, типи даних і властивості.





Кроки

1. На вкладці **Створити (Create)** у групі **Таблиці (Tables)** вибрати **Конструктор таблиць (Table Design)**.
2. У верхній частині вікна ввести назви полів, вибрати типи даних і, за потреби, додати описи.
3. У нижній частині (властивості полів) за потреби налаштувати додаткові параметри (розмір поля, формат, значення за замовчуванням тощо).
4. Визначити ключове поле, виділивши поле та натиснувши кнопку **Ключове поле (Primary Key)** на вкладці **Конструктор (Design)** або натиснувши ПКМ на поле.
5. Збережіть таблицю, присвоївши ім'я.

Типи даних

- **Короткий текст (Short Text)**: для текстових даних до 255 символів (наприклад, імена, адреси).
- **Довгий текст (Long Text)**: для великих текстових даних (наприклад, описи, примітки).
- **Число (Number)**: для числових даних (цілі числа, дробові числа тощо).
- **Дата/Час (Date/Time)**: для зберігання дат і часу.
- **Грошова одиниця (Currency)**: для фінансових даних із фіксованою точністю.
- **Автонумерація (AutoNumber)**: автоматично генерує унікальні номери (часто використовується для первинного ключа).
- **Логічний (Yes/No)**: для значень «Так/Ні» (**True/False**).
- **Гіперпосилання (Hyperlink)**: для зберігання URL-адрес.
- **Вкладення (Attachment)**: для зберігання файлів (зображень, документів).
- **Об'єкт OLE**: для додавання об'єктів, як-от зображення тощо.
- **Обчислювальний (Calculated)**: для автоматичних обчислень на основі інших полів.
- **Майстер підстановок (Lookup Wizard)**: для створення списків вибору або зв'язків з іншими таблицями.

Нюанси

- Для наповнення таблиці записами — потрібно перейти в режим таблиці.
- Якщо первинний ключ не задано, **Access** запропонує створити поле **Ідентифікатор** із типом **Автонумерація** під час збереження таблиці.
- Використовуйте вкладку **Властивості полів** для додаткових налаштувань, таких як:
  - Маска вводу: для форматованого введення (наприклад, телефонів або кодів).
  - Умова на значення: для перевірки даних (наприклад, >0 для позитивних чисел).
  - Підпис: для відображення у формах замість назви поля.

Створення таблиці за допомогою майстра таблиць

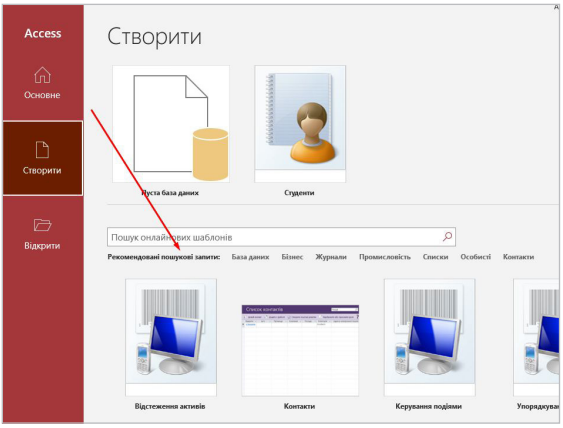
Майстер таблиць пропонує шаблони для швидкого створення таблиць.

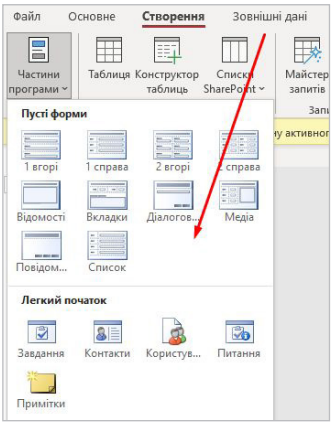
Кроки

1. На вкладці **Створити** виберіть **Майстер таблиць (Table Wizard)**.
2. Виберіть шаблон (наприклад, **Контакти**, **Замовлення**).
3. Додайте потрібні поля з шаблону.
4. Вкажіть ім'я таблиці та первинний ключ.
5. Виберіть, чи відкривати таблицю для введення даних, чи переходити до конструктора.

Нюанси

- Майстер зручний для початківців, але обмежує гнучкість порівняно з конструктором.
- Після створення таблиці її можна доопрацювати в режимі конструктора.
- Шаблон створює не лише таблиці, а й звіти, форми, запити.





Імпорт або зв'язок із зовнішніми даними

Таблиці можна створювати шляхом імпорту даних із зовнішніх джерел (**Excel**, **SharePoint,** текстових файлів) або зв'язком із зовнішніми базами даних.

Імпорт даних

1. На вкладці **Зовнішні дані (External Data)** виберіть **Нове джерело даних (New Data Source) > З файлу (From File)** або **З бази даних (From Database)**.
2. Виберіть джерело (наприклад, файл **Excel**) і дотримуйтесь інструкцій майстра.
3. Виберіть опцію **Імпортувати дані до нової таблиці** або **Створити пов'язану таблицю**.

Нюанси

- Пов'язані таблиці доступні лише для читання, якщо джерело даних (наприклад, вебслужба) не підтримує редагування.
- Переконайтеся, що формат даних у джерелі відповідає типам даних **Access**.

Наповнення таблиць даними

Після створення таблиці її потрібно наповнити даними.

Як наповнити таблицю

1. Відкрити таблицю в режимі таблиці (двічі клацнувши на ній в області переходів).
2. Вводьте дані в кожне поле, для зручності переміщуючись між полями клавішею **Tab.**
3. Після введення запису натисніть **Enter**, щоб перейти до наступного запису. **Access** автоматично зберігає кожен новий запис.

Нюанси

- Якщо поле має умову на значення (наприклад, >0), **Access** видасть помилку при введенні некоректних даних.
- Поле **Автонумерація** заповнюється автоматично і не редагується.
- Можна використовувати форму або запити для зручнішого введення.

Поле OLE в базах даних Access

Поле OLE — це спеціальний тип поля в **Microsoft Access**, який дозволяє зберігати об'єкти, вбудовані з інших програм.

До такого поля можна вставити:

- зображення (**BMP**, **JPG**, **PNG** тощо),
- документи **Word** або **Excel**,
- інші об'єкти, що підтримують **OLE**.

Як вставити зображення у поле OLE в Access:

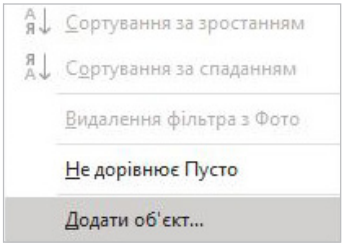
1 Створення таблиці з полем OLE

- Відкрийте **Microsoft Access**.
- Створіть нову базу даних або відкрийте існуючу.
- У режимі конструктора таблиці додайте поле:
  - Ім'я поля: **Фото**
  - Тип даних: Об'єкт **OLE**

Ім'я поля	Тип даних
ID	Автонумерація
Назва	Короткий текст
Тип	Короткий текст
Дата посадки	Дата й час
Кількість	Число
Фото	Об'єкт OLE

2 Відкриття таблиці для введення даних

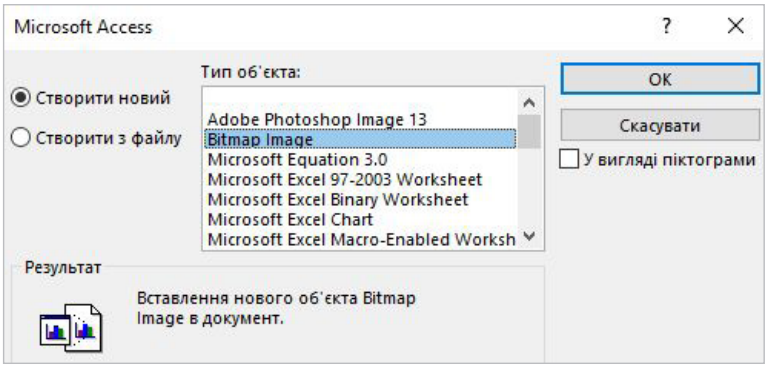
- Перейдіть у **Режим таблиці.**
- У полі **Фото** клацніть правою кнопкою миші.
- У меню виберіть **Додати об'єкт…**



3 Вибір джерела зображення

У діалоговому вікні з'явиться два варіанти:

- Створити новий – відкриється **Paint** або інший редактор (рекомендовано обрати **Bitmap Image**), можна намалювати малюнок або вставити скопійоване зображення із інтернету або скриншот.
- Створити з файлу – вставити готовий файл (**JPG**, **PNG**, **BMP**).



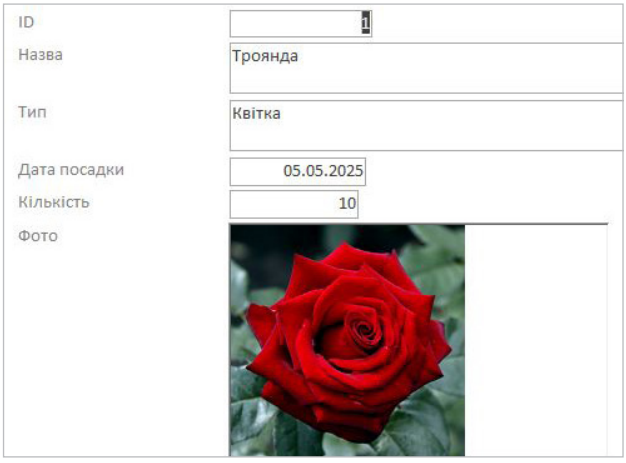
4 Збереження зображення в таблиці

- Натисніть **ОК**.
- У таблиці у полі **Фото** з'явиться позначка типу **Bitmap Image**.
- Це означає, що картинка успішно додана.

ID	Назва	Тип	Дата посади	Кількість	Фото	Кл
1	Троянда	Квітка	05.05.2025	10	Bitmap Image	
2	Дуб	Дерево	10.04.2025	5	Bitmap Image	
3	Ромашка	Квітка	12.06.2025	15	Bitmap Image	
4	Бузок	Кущ	15.05.2025	8	Bitmap Image	
5	Сакура	Дерево	14.04.2025	3	Bitmap Image	

5 Перегляд зображення у формі

Дане зображення можна буде переглядати у спеціально створеній формі



Зв'язки між таблицями

У реляційній базі даних інформація зберігається в різних таблицях, але ці таблиці можна зв'язати між собою через спільні поля для об'єднання даних.

Мета зв'язків:

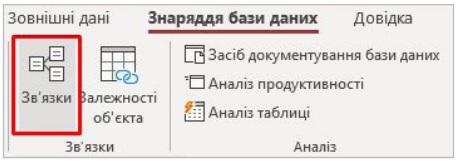
- щоб не дублювати інформації;
- щоб дані були структурованими;
- щоб при зміні однієї таблиці пов'язані дані оновлювались автоматично.

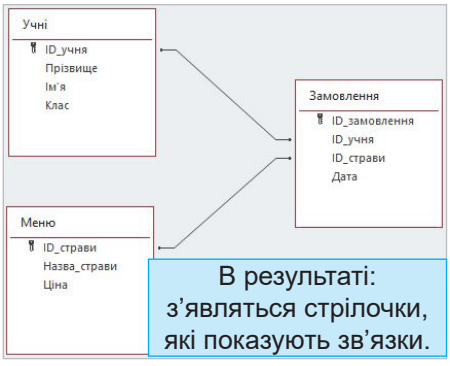
Види зв'язків між таблицями

- **Один до одного (1:1):** кожен запис в одній таблиці відповідає одному запису в іншій.
- **Один до багатьох (1:N):** один запис в одній таблиці відповідає кільком записам в іншій (наприклад, один клієнт — багато замовлень).
- **Багато до багатьох (N:M):** багато записів з однієї таблиці можуть відповідати багатьом записам з іншої таблиці. Реалізується через проміжну таблицю. Наприклад, учні можуть відвідувати багато гуртків, гуртки — мати багато учнів.

Створення зв'язків

1. Імпортуйте або створіть таблиці. Наприклад, **Учні**, **Меню** і **Замовлення.**
2. На вкладці **Знаряддя бази даних (Database Tools)** виберіть **Зв'язки (Relationships).**





3. Додайте потрібні таблиці.
4. Перетягніть мишкою поле **ID_учня** з таблиці **Учні** на поле **ID_учня** в таблиці **Замовлення.**
5. У вікні, що відкриється:
   - Поставте галочку «**Забезпечити цілісність даних**».
   - Оберіть тип зв'язку: **Один до багатьох.**
6. Повторіть для зв'язку **ID_страви ↔ ID_страви.**

Що таке "забезпечення цілісності даних"?

Це налаштування не дозволяє:
- Створити замовлення на неіснуючого учня/учениці.
- Видалити страву, яка вже використовується в замовленнях.

Це захищає дані від помилок і втрати зв'язків.

Поради

- Кожна таблиця повинна мати первинний ключ.
- Зовнішній ключ — це поле, яке посилається на іншу таблицю.
- Імена полів краще задавати однаково для зв'язаних елементів (ID_учня/учениці ↔ ID_учня/учениці).
- Після створення зв'язків можна використовувати запити, які автоматично «знають», як поєднувати дані з кількох таблиць.

Практична робота №14

Завдання 1. Шкільний буфет.

Уявіть, що ви створюєте БД для шкільного буфету. Створіть таблицю, яка має містити:

- Назву товару
- Категорію (наприклад: солодощі, напої)
- Ціну
- Кількість у наявності

Додайте щонайменше 5 товарів

Завдання 2. Кінотеатр мрії.

Уявіть, що створюєте базу даних для кінотеатру. Вона повинна зберігати інформацію про фільми, які показуються. Створіть таблицю **Фільми** зі щонайменше 10 полями різних типів даних та введіть щонайменше 5 записів.

Рекомендована структура таблиці:

Поле	Опис	Тип даних
ID_фільму	Унікальний номер фільму	Автонумерація (Primary Key)
Назва	Назва фільму	Короткий текст
Жанр	Наприклад, комедія, драма, фантастика	Короткий текст
Тривалість (хв)	Тривалість у хвилинах	Числовий
Дата прем'єри	Дата першого показу	Дата/час
Вікове обмеження	Наприклад, 12+, 16+, 18+	Короткий текст
У прокаті (так/ні)	Чи фільм ще показують?	Логічний (Yes/No)
Глядацький рейтинг (1-10)	Середня оцінка	Числовий з плаваючою цяткою
Мова озвучення	Наприклад, українська, англійська	Короткий текст
Постер (шлях до зображення)	Файл-зображення фільму	Гіперпосилання / Вкладений об'єкт/Об'єкт OLE

Завдання 3. «Плануємо шкільну екскурсію».

Учні/учениці планують поїздку до іншого міста. Уся інформація про учасників/учасниць, подорожі та транспорт зберігається в єдиній базі даних.

Створіть три таблиці з вказаними полями.

Таблиця **Учні**

Поле	Тип даних
ID_Учня (Primary Key)	Автонумерація
Прізвище	Короткий текст
Ім'я	Короткий текст
Клас	Короткий текст
Телефон	Короткий текст

Таблиця **Поїздки**

Поле	Тип даних
ID_Поїздки (Primary Key)	Автонумерація
Місце	Короткий текст
Дата	Дата/час
Ціна за 1 особу	Числовий

Таблиця **Участь** *(зв'язкова таблиця)*

Поле	Тип даних	Примітка
ID_Участі (Primary Key)	Автонумерація	
ID_Учня	Числовий	Зовнішній ключ → Учні.ID_Учня
ID_Поїздки	Числовий	Зовнішній ключ → Поїздки.ID_Поїздки
Сплачено (так/ні)	Логічний	

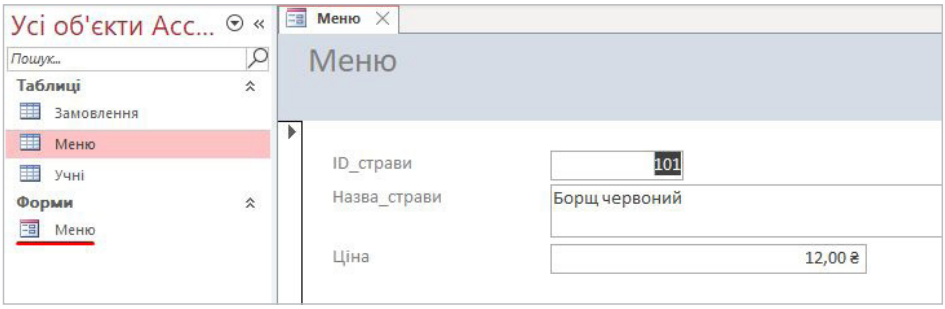
- Встановіть ключові поля (**Primary Key**) у кожній таблиці.
- Перейдіть до **Схеми даних** (**Access**: **Знаряддя бази даних** → **Схема даних**).
- Встановіть зв'язки один-до-багатьох між таблицями **Учні**, **Поїздки** та **Участь**.
- Уведіть по кілька записів у кожну таблицю:
  - 3–5 учнів
  - 2–4 поїздки
  - участь учня / учениці в одній або кількох поїздках

Ключові поняття

Форми в базах даних
Технології No Code
Об'єкти форми: текстове поле, напис, кнопка

Що таке форми в базах даних?

Форми в Microsoft Access — це спосіб, за допомогою якого люди взаємодіють із базою даних, щоб додавати, переглядати, змінювати чи видаляти інформацію.



У таблиці ми бачимо всю інформацію одразу. А у формі — лише один запис, але красиво оформлений і без зайвих деталей.

Форми в **Access** можна налаштувати так, щоб вони мали гарний вигляд й були зручними для користувача. Наприклад, можна додати кнопки, випадаючі списки чи навіть картинки.

Можливості форм

Форми в **Access** мають багато можливостей, які роблять роботу з базою даних зручною.

1 Введення даних

Форми дозволяють швидко додавати нові записи до таблиці. Наприклад, у зоомагазині персонал використовує форму, щоб додати нову тварину до бази даних, вказавши кличку, вид, вік і ціну.

2 Перегляд даних

Форми дозволяють переглядати записи по одному, а не всі одразу, як у таблиці. Це зручно, якщо ти хочеш перевірити інформацію про конкретну особу чи книгу.

3 Редагування даних

Можна змінювати дані через форму, наприклад, оновити ціну страви в їдальні чи змінити адресу людини в базі.

4 Фільтрація та пошук

Форми можуть мати кнопки чи поля для пошуку даних. Наприклад, можна знайти всі книги певного жанру, ввівши «фентезі» у поле пошуку.

5 Зручний дизайн

Форми можна налаштувати, щоб вони мали привабливий вигляд: додати кольори, змінити розмір полів чи додати логотип школи.

Технології No Code

Технології No Code — це інструменти, які дозволяють створювати програми, бази даних чи навіть вебсайти без програмування (написання коду).

Microsoft Access є прикладом такої технології, бо ви можете створювати форми, запити та звіти, просто використовуючи мишку та готові інструменти програми. Це як конструктор LEGO: складаєте потрібні деталі (поля, кнопки), щоб отримати бажаний результат.

No Code інструменти, такі як Access, дозволяють:

- Створювати бази даних і форми за допомогою графічного інтерфейсу.
- Налаштовувати вигляд форм (наприклад, додавати кольори чи підписи).
- Автоматизувати завдання, такі як пошук чи сортування даних.

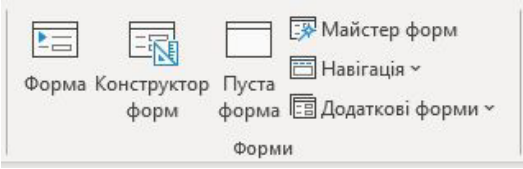
Основні елементи форми

Текстові поля	
Випадаючі списки (Combo Box)	Для вибору значення зі списку (наприклад, жанр книги: фентезі, пригоди).
Прапорець (Check Box)	Для позначки «Так/Ні» (наприклад, чи є книга в наявності).
Кнопки	Для збереження, видалення чи пошуку даних.
Написи (Labels)	Для пояснення, що означає кожне поле (наприклад, «Введіть ім'я»).

Створення форм

У **Microsoft Access** є кілька способів створення форм:

- Майстер форм
- Конструктор форм
- Автоматична форма



Майстер форм — це інструмент в **Access**, який автоматизує створення форм на основі таблиць або запитів, дозволяючи швидко налаштувати інтерфейс без глибоких знань дизайну. Допомагає створити форму покроково, вибираючи поля з таблиці.

Конструктор форм — дозволяє налаштувати форму вручну, додаючи поля, кнопки чи зображення.

Автоматична форма — **Access** може створити просту форму за одне натискання на основі таблиці.

Створення форм за допомогою майстра

Крок 1. Відкриття майстра форм

- Відкрийте базу даних у **Microsoft Access**.
- Перейдіть на вкладку **Створити (Create)** у верхній частині стрічки.
- У групі **Форми (Forms)** натисніть **Майстер форм (Form Wizard)**.

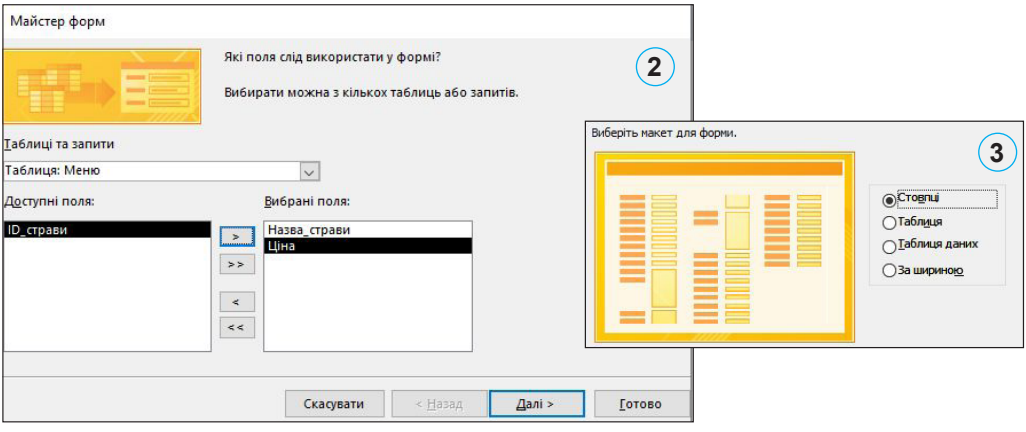
Крок 2. Вибір джерела даних

- У вікні майстра виберіть таблицю або запит, на основі яких створюватиметься форма. Наприклад, таблиця «Меню».
- У списку **Доступні поля (Available Fields)** виберіть поля, які потрібно включити до форми, і перенесіть їх до списку **Вибрані поля (Selected Fields)** за допомогою кнопок зі стрілками (> для одного поля, >> для всіх полів). *Наприклад:* для таблиці Меню виберіть поля Назва_страви та Ціна.
- Натисніть **Далі**.

Крок 3. Вибір макета форми

Майстер пропонує кілька макетів для відображення даних.

- **Стовпці (Columnar)**: Поля розташовані вертикально, один запис на сторінці. Найпоширеніший для введення даних.
- **Таблиця (Tabular)**: Поля розташовані в рядках, як у таблиці, для перегляду кількох записів одночасно.
- **Таблиця даних (стрічка)**: Форма має вигляд таблиці, подібна до режиму таблиці.
- **За шириною (Justified)**: Поля розташовані компактно, заповнюючи весь простір форми. Виберіть макет, наприклад, **Стовпчиковий**, і натисніть **Далі**.



Крок 4. Вибір стилю

- Виберіть стиль оформлення форми (наприклад, Стандартний, Офісний, Сучасний). Стиль впливає на кольори, шрифти та фон форми.
- Натисніть **Далі**.

Крок 5. Назва форми та завершення

- Задайте назву форми.
- Виберіть одну з опцій:
  - Відкрити форму для перегляду або введення даних: Форма відкриється в режимі введення.
  - Змінити макет форми: Форма відкриється в режимі конструктора для подальшого редагування.
- Натисніть **Готово**.

Після створення форми за допомогою майстра її можна доопрацювати в режимі конструктора (**Design View**).

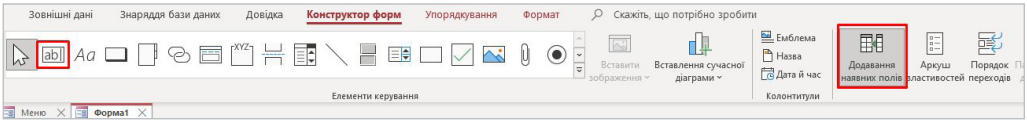
Створення форм за допомогою конструктора

Режим конструктора — це ручний режим, у якому користувач самостійно створює зовнішній вигляд і **структуру форми** — розміщує поля, змінює їхній розмір, додає надписи, кнопки тощо.

Підходить для досвідченіших користувачів, які хочуть повний контроль над виглядом форми.

Крок 1. Створення нової форми

- Відкрийте базу даних у **Microsoft Access**.
- На вкладці **Створити (Create)** у групі **Форми (Forms)** натисніть **Конструктор форм (Form Design)**. З'явиться порожнє полотно форми в режимі конструктора.
- Установіть джерело даних для форми:
  - Клацніть правою кнопкою миші на порожній області форми та виберіть **Властивості форми (Form Properties)**.
  - У вкладці **Дані (Data)** у полі **Джерело записів (Record Source)** виберіть таблицю або запит.



Крок 2. Додавання полів у форму

- Перейдіть до вкладки **Конструктор (Design)**.
- Натисніть **Додати поля існуючої таблиці** (іноді зліва — **Додати наявні поля → Показати всі таблиці**).
- У панелі, що з'явиться:
  - Знайдіть потрібні поля таблиці.
  - Перетягніть їх мишею на форму.

Крок 3. Додавання елементів керування

- Відкрийте панель **Елементи керування** (**Controls**) на вкладці **Конструктор** (**Design**).

- Додайте потрібні елементи: **Напис**, **Текстове поле**, **Список**, **Кнопка**, **Поле зі списком**, **Прапорці**, **Перемикачі**, **Підформа** тощо.
- Перетягніть елемент керування на область форми (наприклад, в область даних).
- Для текстових полів або списків зв'яжіть елемент із полем таблиці.
- У властивостях елемента в полі **Джерело елемента керування** (**Control Source**) виберіть поле з джерела записів.

Крок 4. Налаштування елементів керування

- Клацніть правою кнопкою миші на елементі керування та виберіть **Властивості** (**Properties**) або натисніть **F4**.
- Налаштуйте ключові властивості:
  - **Формат** (**Format**): Наприклад, Грошова одиниця для поля Ціна.
  - Джерело елемента керування (яке поле показувати).
  - Назву елемента.
  - Колір, шрифт, вирівнювання, відступи.
  - **Підпис** (**Caption**): Текст, який відображається поруч із елементом (наприклад, Назва книги).
  - **Умова на значення** (**Validation Rule**): Наприклад, >0 для поля Ціна.
  - **Текст підказки** (**ControlTip Text**): Спливаюча підказка для користувача.
  - **Події** (**Events**): Наприклад, додайте макрос або VBA-код для події **On Click** кнопки.

Крок 5. Налаштування макета форми

- Розташуйте елементи керування логічно, використовуючи сітку для вирівнювання.
- Змініть розміри елементів, перетягуючи їхні межі.
- Додайте заголовок у розділ **Заголовок форми**:
  - Перетягніть елемент **Напис** і введіть текст, наприклад, Введення даних про клієнтів.
  - Налаштуйте шрифт, розмір і колір у вкладці **Формат** (**Format**).
- Згрупуйте елементи за допомогою **Макета керування** (**Control Layout**):
  - Виберіть елементи, клацніть **Упорядкувати** (**Arrange**) > **Табличний** або **Стовпчиковий**.

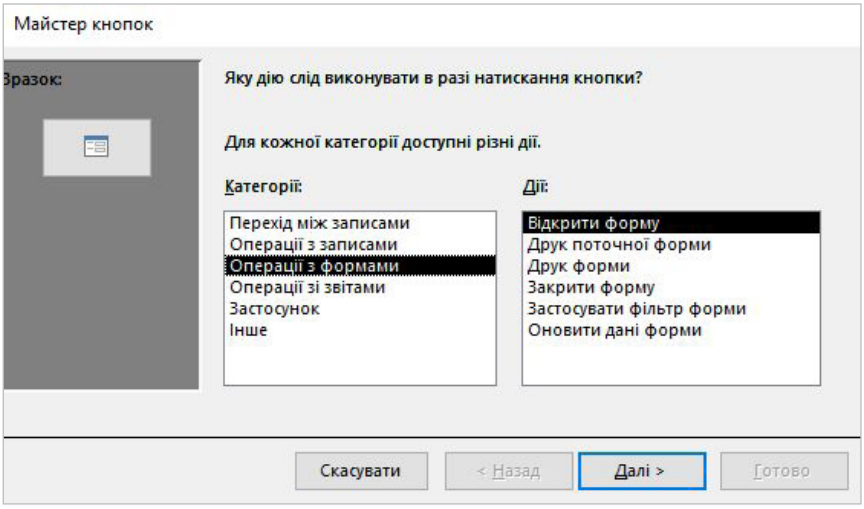
Крок 6. Збереження та тестування

- Збережіть форму, натиснувши **Файл** > **Зберегти** (**File** > **Save**) і задавши ім'я.
- Перейдіть до **Режиму форми** (**Form View**), щоб перевірити її функціональність.
- Перевірте введення даних, навігацію та відображення зв'язаних даних.

Додавання кнопок

Додавання кнопок до форм у **Microsoft Access** у режимі конструктора дозволяє створювати інтерактивні та зручні інтерфейси для роботи з даними.

Кнопки можна налаштувати за допомогою майстра, макросів або VBA, що забезпечує гнучкість для реалізації як простих, так і складних дій.



Як додати кнопку в Access

1. У режимі конструктора форми виберіть вкладку **Конструктор**
2. Клацніть **Кнопка**
3. Натисніть мишею на формі — запуститься **майстер кнопок**
4. Виберіть:
   - Категорію дій (наприклад: Навігація між записами)
   - Конкретну дію (наприклад: Перейти до наступного запису)
5. Назвіть кнопку → натисніть **Готово**

Найбільш поширені дії:

- **Відкрити форму**
- **Перехід до наступного/попереднього запису**
- **Додати новий запис**
- **Закрити форму**

Практична робота №15

Завдання 1. Форма «Бібліотека»

Створіть таблицю Книги з такими полями або відкрийте готову базу даних:

- Назва книги (текст)
- Автор/авторка (текст)
- Рік видання (число)
- Жанр (текст)
- Дата надходження (дата)

Далі:

- Заповніть таблицю 5-ма записами.
- Перейдіть у вкладку **Створити** → **Майстер форм**.
- Створіть форму на основі таблиці Книги.
- Оберіть зовнішній вигляд **Стрічка** або **Стовпчик**.
- Відкрийте форму в режимі **Форми**, перегляньте записи, додайте нову книгу.

Завдання 2. Форма «Класні заходи»

Створіть таблицю Події:

- Назва події
- Дата
- Клас
- Кількість учасників/учасниць
- Місце проведення

1. Створіть форму за допомогою **Майстра форм**.
2. Використайте усі поля, але змініть порядок: спершу Назва, потім Дата, Клас і т.д.
3. Оберіть стиль форми, що вам найбільше до вподоби.
4. Увімкніть вікно переходу по записах внизу форми.
5. Перевірте: чи можете через форму змінити клас або місце?

Завдання 3. Створення форми «Каталог зоотоварів»

Створіть таблицю Зоотовари з такими полями або відкрийте готову базу даних:

Поле	Тип даних	Примітка
ID_товару	Автонумерація	Первинний ключ
Назва товару	Короткий текст	Напр.: Корм для котів
Категорія	Короткий текст	Напр.: Корм, Іграшка, Аксесуар

Тварина	Короткий текст	Напр.: Собака, Кішка, Птах
Ціна (грн)	Числовий	
В наявності	Логічний (Yes/No)	Прапорець
Виробник	Короткий текст	Напр.: Royal Canin
Дата надходження	Дата/час	
Опис	Довгий текст	Додаткова інформація

- Перейдіть до вкладки **Створити** → **Конструктор форм**.
- Створіть порожню форму, пов'язану з таблицею Зоотовари.
- Розмістіть поля у зручному та логічному порядку, наприклад:
  - Блок 1: Назва товару, Категорія, Тварина.
  - Блок 2: Ціна, В наявності, Виробник.
  - Блок 3: Дата надходження, Опис.
- Додайте заголовок форми — Каталог зоотоварів.
- Додайте кольорове тло (наприклад, м'яко-зелений чи блакитний).
- Для логічного поля В наявності використовуйте прапорець.
- Для поля Опис задайте довге текстове поле з вертикальною прокруткою.
- Додайте кнопки:
  - **Додати новий запис**
  - **Видалити запис**
  - **Зберегти**
  - **Закрити форму**

Ключові поняття

- Сортування в базах даних
- Фільтрація в базах даних
- Типи фільтрів
- Звіти в базах даних

Що таке сортування та фільтрація даних?

Сортування та фільтрація — це інструменти в базах даних, які допомагають упорядковувати та знаходити потрібну інформацію.

Сортування — це організація даних у певному порядку, наприклад, за алфавітом чи від найменшого до найбільшого значення.

Фільтрація — це відбір даних, які відповідають певним умовам, наприклад, тільки ті записи, де ціна менша за 50 гривень.

У **Microsoft Access** ці функції дозволяють швидко працювати з великими обсягами інформації, роблячи її зрозумілою та зручною для аналізу. Уявіть, що ви шукаєте улюблену книгу в бібліотеці: сортування допоможе розташувати книги за назвою чи автором, а фільтрація покаже тільки книги в жанрі «фентезі».

Чому сортування та фільтрація важливі?

У базах даних часто зберігається багато інформації, і без сортування чи фільтрації знайти потрібне було б складно. Ці інструменти допомагають:

- **Організовувати дані**

Наприклад, розташувати список учнів / учениць за прізвищем в алфавітному порядку.

- **Знайти потрібне**

Наприклад, відібрати всіх учнів/учениць, які отримали оцінку «10» з математики.

- **Аналізувати інформацію**

Наприклад, подивитися, які товари в магазині найдешевші.

Продавець/продавчиня хоче показати вам всі машинки, які коштують до 200 гривень. Він/вона використовує базу даних, щоб відфільтрувати тільки машинки і відсортувати їх за ціною від найнижчої до найвищої.

Сортування в Microsoft Access

Сортування в **Access** дозволяє впорядкувати записи в таблиці чи запиті за одним або кількома полями. Можна сортувати:

За зростанням

(A–Z для тексту, від меншого до більшого для чисел)

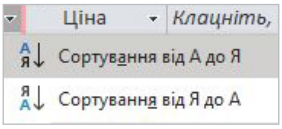
За спаданням

(Z–A для тексту, від більшого до меншого для чисел)

Як виконати сортування?

У таблиці

1. Відкрити таблицю в **Access**.
2. Клацнути правою кнопкою миші на заголовок стовпця (наприклад, Прізвище).
3. Вибрати **Сортувати за зростанням** або **Сортувати за спаданням**.



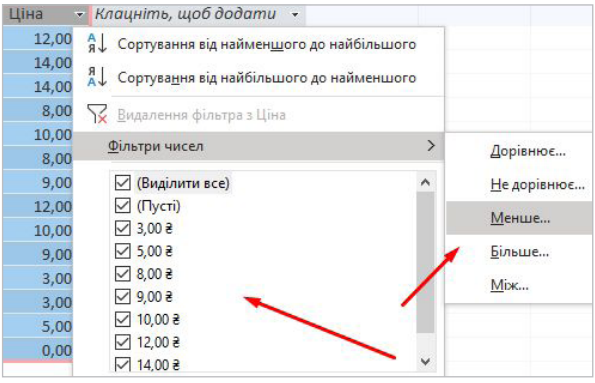
У запиті

1. Створити запит у конструкторі запитів.
2. У рядку **Сортування** вибрати **За зростанням** або **За спаданням** для потрібного поля.
3. Запустити запит.

У формі

Додати кнопку сортування до форми, щоб користувач міг упорядкувати дані.

Фільтрація в Microsoft Access



У таблиці

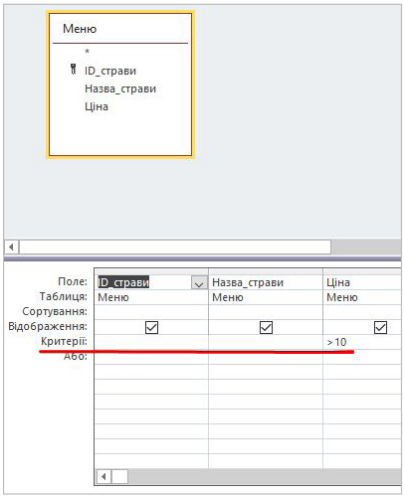
- Відкрийте таблицю в **Access**.
- Клацніть на стрілку в заголовку стовпця (наприклад, Жанр).
- Виберіть потрібне значення (наприклад, Фентезі) або задайте умову (наприклад, Ціна < 50).

У запиті

- Створіть запит у конструкторі запитів.
- У рядку **Критерії** вкажіть умову, наприклад, >10 для поля Вік або Фентезі для поля Жанр.

У формі

- Додайте до форми поле для фільтрації, наприклад, випадаючий список із жанрами.
- Користувач вибирає значення, і форма показує тільки відповідні записи.



Типи фільтрів у Microsoft Access

- **Фільтр за значенням**: вибирає записи з конкретним значенням. Наприклад, показати всі книги, написані — «Дж. К. Ролінґ».
- **Фільтр за кількома умовами**. Наприклад, показати всі книги, де жанр — «Фентезі» та ціна < 200 грн.
- **Фільтр за умовою:** використовує логічні вирази, наприклад, «Ціна < 100» або «Дата > 01.01.2025».
- **Фільтр у формах:** дозволяє користувачу вибирати значення зі списку чи вводити умову.

Переваги й недоліки сортування та фільтрації

Переваги

- **Швидкість**. Можна знайти потрібні дані за секунди.
- **Простота**. **Access** дозволяє сортувати та фільтрувати кількома натисками.
- **Гнучкість**. Можна комбінувати кілька умов фільтрації чи сортувати за кількома полями.

Недоліки

- **Обмеження інтерфейсу.** У таблицях чи формах можна фільтрувати тільки за тими полями, які є в них. Якщо потрібного поля немає, доведеться створювати запит.
- **Складність із кількома умовами.** *Приклад:* У бібліотеці важко відфільтрувати книги, які одночасно є в наявності, належать до жанру «Фентезі» і видані після 2010 року, якщо форма не підтримує такі комбінації.
- **Помилки в даних.** Якщо дані введено неправильно, фільтр може не знайти потрібні записи.

Звіти в базах даних

Звіт — це спеціальний документ, створений на основі даних таблиць або запитів.

Звіт зручно:

- переглядати
- друкувати
- зберігати у PDF
- передавати іншому користувачу



Звіти часто створюються для того, щоб підсумувати інформацію, зробити її читабельною та гарно оформленою.

Уявімо, що вчитель хоче надрукувати список учнів, які беруть участь у олімпіадах.

Дані зберігаються в базі — але друкувати таблицю незручно, тому вчитель натискає **Створити звіт**, і **Access**:

- відображає дані красиво у вигляді списку,
- додає заголовки, дати, рамки,
- автоматично нумерує сторінки.

Чому звіти важливі?

Звіти допомагають швидко зрозуміти великі обсяги даних, не переглядаючи всі таблиці. Вони корисні, коли потрібно:

Узагальнити інформацію

Наприклад, підрахувати, скільки книг узяли в бібліотеці за місяць.

Презентувати дані

Наприклад, підрахувати, скільки книг узяли в бібліотеці за місяць.

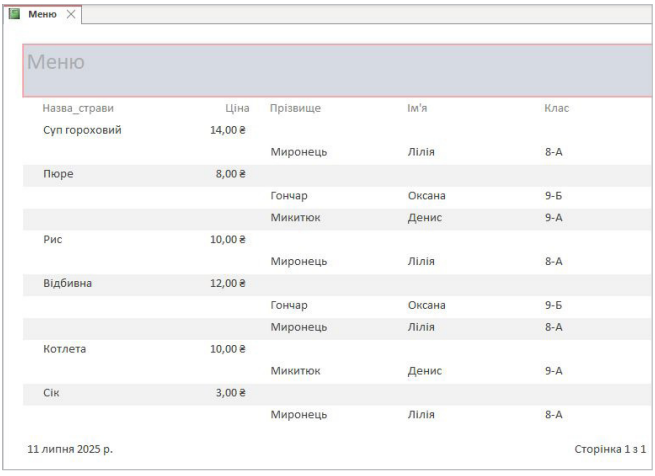
Приймати рішення

Наприклад, визначити, які товари в магазині закінчуються, щоб замовити нові.

Як виглядає звіт?

Звіт в **Access** — це документ, що має:

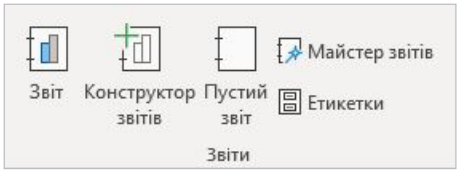
- Заголовок (наприклад, **Меню**)
- Табличну частину (дані)
- Примітки (наприклад, **Згенеровано: 11.07.2025**)
- Підписи, нумерацію сторінок



Звіт не редагується напряму — це вивідна форма, а не таблиця.

Як створювати звіти в Microsoft Access?

У **Microsoft Access** звіти створюються за допомогою **Майстра звітів** або **Конструктора звітів**. Ось як це працює:



Майстер звітів (Report Wizard)

1. **Створити → Майстер звітів**
2. Обери джерело (таблицю або запит)
3. Обери поля, які мають відображатися
4. Визнач порядок сортування
5. Обери макет: **табличний / блоковий**
6. Дай ім'я звіту — готово!

Конструктор звітів (Report Design)

1. Виділи потрібну таблицю або запит
2. Вкладка **Створити → Звіт**
3. **Access** створить простий звіт автоматично

Автоматичний звіт

Для досвідченіших користувачів:

- Можна вручну додати колонтитули, підписи, логотип
- Додавати розрахункові поля (наприклад, **Сума**, **Середнє**, тощо)

Практична робота №16

Завдання 1. Сортування даних у таблиці

Уявіть, що створюєте базу даних для шкільної бібліотеки з таблицею Книги (поля: ID, Назва, Автор, Жанр, Рік видання).

Виконайте:

- Створіть або відкрийте готову базу даних Бібліотека і таблицю Книги.
- Відсортуйте таблицю за полем Назва в алфавітному порядку.
- Відсортуйте таблицю за полем Рік видання від найновішого до найстарішого.

Завдання 2. Фільтрація даних у таблиці

Уявіть, що працюєте з базою даних зоомагазину з таблицею Тварини (поля: ID, Кличка, Вид, Вік, Ціна).

Виконайте:

- Створіть або відкрийте готову базу даних Зоомагазин і таблицю Тварини.
- Відфільтруйте таблицю, щоб показати тільки котів.
- Відфільтруйте таблицю, щоб показати тварин із ціною до 1000 грн.

Завдання 3. Звіт для шкільного клубу

Вам потрібно створити базу даних у **Microsoft Access** для шкільного клубу любителів природи з таблицею Рослини (поля: ID, Назва, Тип, Дата посадки, Кількість, Фото).

Виконайте:

- Створіть або відкрийте таблицю Рослини і заповніть її мінімум 5 записами.
- Створіть звіт про всі рослини, групуючи їх за типом (квітка, дерево, кущ).
- Відформатуйте звіт.

Порядок дій

Створення звіту:

- Відкрийте **Майстер звітів** у **Access**.
- Виберіть таблицю **Рослини** і поля: Назва, Тип, Кількість.
- Групуйте за полем **Тип**.

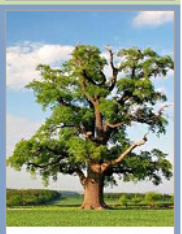
Оформлення:

- Додайте заголовок Звіт про рослини клубу 2025.
- Використайте зелений колір для оформлення, щоб підкреслити екологічну тему.

Зразок:

Рослини

Звіт про рослини клубу 2025

Назва	Тип	Дата посадки	Кількість	Фото
Бузок	Кущ	15.05.2025	8	
Дуб	Дерево	10.04.2025	5	
Ромашка	Квітка	12.06.2025	15	
Сакура	Дерево	14.04.2025	3	

Ключові поняття

Запити в базах даних

Типи запитів: з вибіркою, з параметром, перехресний запит, на обчислення

Основи SQL

Що таке запит?

Запити в базах даних — це інструменти, які дозволяють тобі знаходити, вибирати, сортувати або об'єднувати потрібну інформацію з однієї чи кількох таблиць.

У **Microsoft Access** запити діють як «розумний помічник», який швидко шукає дані за твоїми умовами.

Приклад з життя

Уяви, що в бібліотеці є велика база книг. Але тобі потрібно дізнатися:

- які книги авторства Тараса Шевченка,
- видані після 2000 року,
- або вартістю до 200 грн.

Замість гортати таблицю, ти ставиш запит — і програма сама виводить потрібне.

Чому запити важливі?

Запити допомагають працювати з великими обсягами даних ефективно. Вони дозволяють:

Знайти потрібне

Наприклад, показати всіх учнів 9-Б класу

Узагальнити дані

Наприклад, підрахувати, скільки книг узяли в бібліотеці за місяць

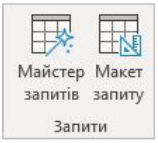
Об'єднати інформацію

Наприклад, поєднати дані про учнів і їхні оцінки аьо навіть об'єднати таблиці

Автоматизувати роботу

Запити можна зберегти й використовувати повторно

Як створити простий запит у Microsoft Access?



За допомогою Майстра запитів

1. Перейди на вкладку **Створити → Майстер запитів**
2. Обери таблицю або запит, з яким хочеш працювати
3. Додай потрібні поля (Прізвище, Клас, Середній бал тощо)
4. **Заверши** — **Access** покаже результати

   Цей метод — найпростіший для новачків!

У режимі конструктора

1. Перейди на вкладку **Створити → Макет запиту**
2. Обери таблиці (або запити), які потрібно використати
3. У нижній частині конструктора:
   - укажи поля, які хочеш вивести
   - додай умови фільтрації
   - задай сортування
4. Запусти запит кнопкою **Виконати** (червоний знак !)

Основні типи запитів

1 Вибірковий запит (Select Query)

Вибирає дані за певними умовами.

Приклад: Знайти всі книги жанру «Фентезі» в бібліотеці.

2 Запит із параметрами

Запитує в користувача умову під час виконання.

Приклад: Запит, який питає «Введіть клас» і показує учнів цього класу.

3 Запит на обчислення

Виконує підрахунки, наприклад, суму чи середнє значення

Приклад: Підрахувати загальну кількість рослин у шкільному саду.

4 Перехресний запит (Crosstab Query)

Показує дані в зведеній таблиці.

Приклад: Показати кількість замовлень кожної страви за днями тижня.

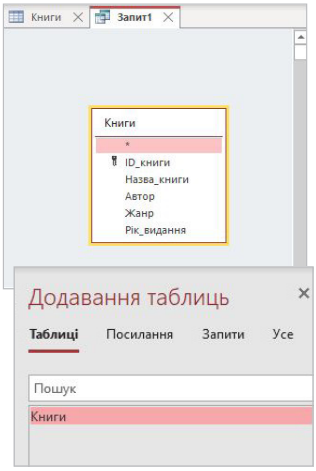
Алгоритм створення вибіркового запиту

КРОК 1: Відкрити конструктор запитів

- Відкрий свою базу даних в **Microsoft Access**
- Перейди на вкладку **Створити** (**Create**)
- Натисни **Макет запиту** (**Query Design**)

КРОК 2: Додати таблицю **Книги**

- У вікні **Додавання таблиць** вибери **Книги**
- Натисни **Додати**→ **Закрити**



КРОК 3: Вибрати потрібні поля

У нижній частині вікна (сітка запиту), подвійним клацанням додай поля:

- Назва_книги
- Автор
- Жанр
- (за потреби — інші)

КРОК 4: Задати умову фільтрації

У рядку **Умова** під полем Жанр введи: Фентезі (дві лапки, якщо це текстовий тип поля)

Access шукатиме лише ті записи, де жанр точно дорівнює Фентезі (без помилок у написанні)

Поле:	Назва_книги	Автор	Жанр
Таблиця:	Книги	Книги	Книги
Сортування:			
Відображення:	☑	☑	☑
Критерії:			"Фентезі"
Або:			

Можна задавати різні умови для фільтрації даних:

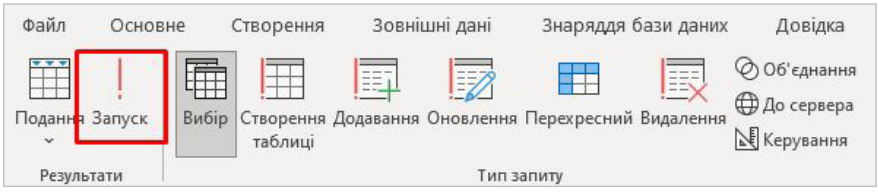
- **Точне значення**
- **Числові порівняння**
- **Діапазони**
- **Частковий збіг**

Що потрібно	Критерій
Учні 9-А класу	«9-А»
Учні з балом більше 10	> 10
Учні з балом від 8 до 11	>=9 And <=11
Прізвище починається на «П»	«П*»
В наявності	Так або True

Використовуй **AND** і **OR** для складних умов, наприклад, Жанр = «Фентезі» AND Рік > 2000

КРОК 5: Запустити запит

- Натисни кнопку **Виконати** (червоний знак оклику) на вкладці **Конструктор**
- З'явиться таблиця з усіма книгами жанру Фентезі



КРОК 6: Зберегти запит

- Натисни **Ctrl + S**
- Назви запит, наприклад: Запит_Фентезі

ID_книги	Назва_книги	Автор	Жанр	Рік_виданн
1	Шерлок Холмс: Собака Баскервілів	Артур Конан Дойл	Детектив	1902
2	Колгосп тварин	Джордж Орвелл	Антиутопія	1945
3	1984	Джордж Орвелл	Антиутопія	1949
4	Хроніки Нарнії: Лев, чаклунка і шафа	Клайв Льюїс	Фентезі	1950
5	Володар перснів: Дві вежі	Дж. Р. Р. Толкін	Фентезі	1954
6	Володар перснів: Братство персня	Дж. Р. Р. Толкін	Фентезі	1954
7	Володар перснів: Повернення корол	Дж. Р. Р. Толкін	Фентезі	1955
8	**Убити пересмішника**	Гарпер Лі	Соціальна драма	1960
9	Гра престолів	Джордж Мартін	Фентезі	1996
10	Пісня льоду й полум'я	Джордж Мартін	Фентезі	1996
11	Гаррі Поттер і філософський камінь	Дж. Роулінг	Фентезі	1997
12	Гаррі Поттер і таємна кімната	Дж. Роулінг	Фентезі	1998
13	Ерагон	Крістофер Паоліні	Фентезі	2002
14	Код да Вінчі	Ден Браун	Детектив	2003
15	Інферно	Ден Браун	Детектив	2013

Назва_книги	Автор	Жанр
Гаррі Поттер і філософський камінь	Дж. Роулінг	Фентезі
Гаррі Поттер і таємна кімната	Дж. Роулінг	Фентезі
Володар перснів: Братство персня	Дж. Р. Р. Толкін	Фентезі
Володар перснів: Дві вежі	Дж. Р. Р. Толкін	Фентезі
Володар перснів: Повернення корол	Дж. Р. Р. Толкін	Фентезі
Хроніки Нарнії: Лев, чаклунка і шафа	Клайв Льюїс	Фентезі
Ерагон	Крістофер Паоліні	Фентезі
Пісня льоду й полум'я	Джордж Мартін	Фентезі
Гра престолів	Джордж Мартін	Фентезі

Запит із параметрами

КРОК 1: Відкрити конструктор запитів
- Відкрий свою базу даних в **Microsoft Access**
- Перейди на вкладку **Створити** (**Create**)
- Натисни **Макет запиту** (**Query Design**)

КРОК 2: Додати таблицю **Учні**
- У вікні «**Додавання таблиці**» вибери таблицю **Учні**
- Натисни **Додати** → **Закрити**

КРОК 3: Додати поля
У сітку запиту внизу додай поля: Прізвище, Ім'я, Клас.

КРОК 4: Додати параметр у полі **Клас**
У рядку **Критерії** під полем Клас введи таке: [Введіть клас:]
Квадратні дужки — це ознака параметра.
Access покаже це як діалогове вікно, коли ви запустите запит.

КРОК 5: Запустити запит
- Натисни **Виконати** (червоний знак оклику) у вкладці **Конструктор**
- З'явиться вікно з питанням: Введіть клас:
- Напиши, наприклад: 9-А
- Після цього **Access** покаже всіх учнів класу 9-А

КРОК 6: Зберегти запит
Натисни **Ctrl + S**
Назви запит, наприклад: Запит_по_класу

Ідентифікат	Прізвище	Ім'я	Клас
1	Іваненко	Олена	9-А
2	Петренко	Ігор	9-Б
3	Сидорова	Марія	9-А
4	Коваленко	Андрій	8-А
5	Мельник	Наталія	9-А
6	Ткачук	Богдан	8-Б
7	Шевченко	Катерина	9-Б
8	Бондар	Юрій	9-А
9	Козак	Оксана	8-А
10	Литвин	Сергій	9-Б
11	Мороз	Анна	8-Б
12	Романенко	Микита	9-А
13	Гнатюк	Лілія	9-Б
14	Остапчук	Олександр	8-А
15	Заболотна	Ірина	8-Б

Прізвище	Ім'я	Клас
Іваненко	Олена	9-А
Сидорова	Марія	9-А
Мельник	Наталія	9-А
Бондар	Юрій	9-А
Романенко	Микита	9-А

Порада:
Можна використовувати часткові збіги з **Like**, наприклад:
Like [Введіть початок прізвища:] & «*»
Показує всіх учнів, прізвища яких починаються на введене.

Алгоритм створення запиту на обчислення

КРОК 1: Відкрити конструктор запитів

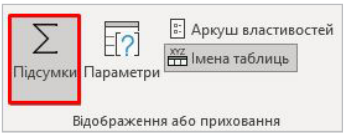
- Відкрий свою базу даних в **Microsoft Access**
- Перейди на вкладку **Створити** (**Create**)
- Натисни **Макет запиту** (**Query Design**)

КРОК 2: Додати таблицю **Рослини**

- У вікні **Додавання таблиці** вибери таблицю **Рослини**
- Натисни **Додати** → **Закрити**

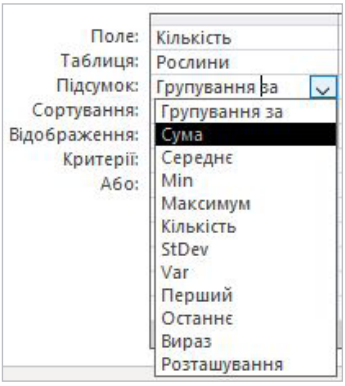
КРОК 3. Додати поле **Кількість**

Подвійним клацанням додай поле **Кількість** до сітки запиту (внизу)



КРОК 4. Увімкнути підсумкові обчислення

- На вкладці **Конструктор** натисни кнопку **Підсумки** (Σ)
- У сітці з'явиться новий рядок **Підсумок**



КРОК 5. Вибрати функцію обчислення

У рядку «**Підсумок**» під полем **Кількість** розкрий список і обери **Sum** (**Сума**)

КРОК 6. Запустити запит

Натисни кнопку **Виконати** (червоний знак оклику)

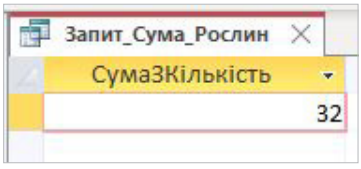
Access покаже таблицю з одним полем і одним значенням — загальна кількість рослин

КРОК 7. Зберегти запит

Натисни **Ctrl + S**

Назви запит, наприклад: Запит_Сума_Рослин

Прізвище	Ім'я	Клас
Іваненко	Олена	9-А
Сидорова	Марія	9-А
Мельник	Наталія	9-А
Бондар	Юрій	9-А
Романенко	Микита	9-А



Можливі варіанти замість Sum:

Функція	Пояснення
Sum(Сума)	Підраховує загальну суму
Avg(Середнє)	Обчислює середнє значення
Min(Мінімум)	Мінімальне значення
Max(Максимум)	Максимальне значення
Count(Кількість)	Кількість записів (рядків), а не значень

Алгоритм створення перехресного запиту

КРОК 1. Відкрити майстер перехресного запиту

- Відкрий **Access**
- Перейди на вкладку **Створити** (**Create**)
- Обери **Майстер запитів** (**Query Wizard**)
- Вибери **Перехресний запит** (**Crosstab Query Wizard**) і натисни **ОК**

КРОК 2: Вибрати таблицю або запит-джерело

Вибери таблицю **Замовлення** → натисни **Далі**

КРОК 3. Вибрати поле для рядків (стовпчиків у першому стовпці)

Вибери поле **Страва** → натисни **Далі**

КРОК 4. Вибрати поле для стовпців

Вибери поле **День_тижня** → натисни **Далі**

КРОК 5. Вибрати поле для обчислення і функцію

- Вибери поле ID_замовлення
- Обери функцію **Count** (**Підрахунок**)

Access підрахує кількість замовлень кожної страви по кожному дню тижня

КРОК 6. Назвати запит і зберегти

- Назви, наприклад: Замовлення_по_днях
- Обери: **Переглянути запит** → **Готово**

ID_замовле	Страва	День_тижн
1	Борщ	Понеділок
2	Вареники	Вівторок
3	Борщ	Вівторок
4	Котлета	Понеділок
5	Борщ	Середа
6	Каша	Середа
7	Котлета	Середа
8	Вареники	Четвер
9	Каша	П'ятниця
10	Борщ	П'ятниця
11	Каша	Четвер
12	Каша	П'ятниця
13	Котлета	Понеділок
14	Вареники	Вівторок
15	Борщ	Четвер

Страва	Усього ID_замовлення	Вівторок	П'ятниця	Понеділок	Середа	Четвер
Борщ	5	1	1	1	1	1
Вареники	3	2				1
Каша	4		2		1	1
Котлета	3			2	1	

Основи SQL

SQL (Structured Query Language, або мова структурованих запитів) — це спеціальна мова, яку використовують для роботи з базами даних.

Вона дозволяє "розмовляти" з базою даних, щоб знаходити, додавати, змінювати чи видаляти дані. У **Microsoft Access** ти можеш створювати запити за допомогою графічного інтерфейсу (Конструктора запитів), але за лаштунками **Access** перетворює твої дії в SQL-код. Це як замовляти їжу в ресторані: ти можеш вибрати страву з меню (Конструктор), але кухар готує її за рецептом (SQL).

Чому **SQL** важливий?

SQL — це універсальна мова, яка працює не лише в **Microsoft Access**, а й в інших базах даних, таких як **MySQL** чи **Oracle**.

Навіть прості знання **SQL** допоможуть тобі:

- **Швидко знаходити дані:** наприклад, відібрати всіх учнів 9-А класу.
- **Аналізувати інформацію:** наприклад, підрахувати, скільки разів замовляли борщ у їдальні.
- **Автоматизувати роботу:** збережені SQL-запити можна використовувати повторно.
- **Працювати з різними програмами:** SQL — це стандарт, який використовується в багатьох базах даних.

Основні команди SQL

SQL складається з команд, які дозволяють працювати з даними. Ось основні команди для створення простих запитів:

- **SELECT**: Вибирає поля, які ти хочеш бачити в результаті.
  Приклад: SELECT Назва, Автор — вибирає назви та авторів книг.
- **FROM:** Указує таблицю, із якої беруться дані.
  Приклад: FROM Книги — дані беруться з таблиці «Книги».
- **WHERE**: Задає умови для відбору даних.
  Приклад: WHERE Жанр = 'Фентезі' — вибирає тільки книги жанру «Фентезі».
- **ORDER BY:** Сортує результати.
  Приклад: ORDER BY Назва ASC — сортує книги за назвою в алфавітному порядку.
- **GROUP BY:** Групує дані для підрахунків.
  Приклад: GROUP BY Тип — групує рослини за типом (квітка, дерево).
- **SUM, COUNT, AVG**: Виконують обчислення (сума, кількість, середнє).
  Приклад: SUM(Кількість) — підраховує загальну кількість рослин.

Як використовувати SQL у Microsoft Access?

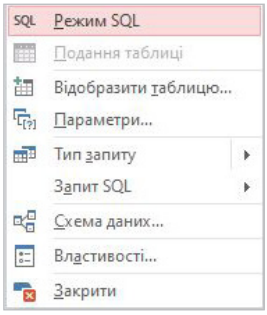
У **Microsoft Access** ти можеш створювати запити двома способами:

Конструктор запитів
Графічний інтерфейс, де ти вибираєш таблиці, поля та умови мишкою.

Режим SQL
Написання SQL-коду вручну для точнішого контролю

Як побачити SQL у Access?

1. Створи запит у Конструкторі запитів
2. Клацни правою кнопкою миші на запиті та вибери **Режим SQL** (**SQL View**).
3. Ти побачиш поле для SQL-коду, де потрібно описати запит
4. Натисни **Виконати** (кнопка з червоним знаком оклику)




```
SELECT Страва, День_тижня
FROM Замовлення
WHERE Страва = 'Борщ';
```

Страва	День_тижн
Борщ	Понеділок
Борщ	Вівторок
Борщ	Середа
Борщ	П'ятниця
Борщ	Четвер

Типи SQL-запитів у Microsoft Access

Вибірковий запит (SELECT). Вибирає дані за умовами.

Приклад: Знайти всі книги, видані після 2010 року

```
SELECT Назва, Автор
FROM Книги
WHERE Рік_видання > 2010;
```

Запит із параметрами. Дозволяє користувачу вводити умову під час виконання

Приклад: Запит, який питає «Введіть вид тварини» і показує всіх тварин цього виду.

```
SELECT Кличка, Ціна
FROM Тварини
WHERE Вид = [Введіть вид тварини:];
```

Запит на обчислення. Виконує підрахунки, наприклад, суму чи середнє значення

Приклад: Підрахувати загальну кількість рослин.

```
SELECT SUM(Кількість)
FROM Рослини;
```

Перехресний запит (Crosstab Query). Показує дані у вигляді зведеної таблиці

Приклад: Показати кількість замовлень кожної страви за днями тижня

```
TRANSFORM SUM(Кількість)
SELECT Назва
FROM Страви INNER JOIN Замовлення
ON Страви.ID = Замовлення.ID_страви
GROUP BY Назва
PIVOT Format(Дата, 'dddd');
```

Практична робота №17

Завдання 1. Вибірка учнів з високим балом

1. Створи таблицю **Учні** з полями: Прізвище, Ім'я, Клас, Середній_бал
2. Додай щонайменше 8 записів
3. Створи простий запит, який:
   - відображає учнів із середнім балом більше 10
   - сортує їх за спаданням бала

Завдання 2. Магазин товарів

1. Створи таблицю Товари: Назва, Категорія, Ціна, В_наявності
2. Створи запит, який:
   - показує тільки товари з категорії Іграшки
   - є в наявності
   - сортує за ціною (від найдешевшого)

Завдання 3: Запит із параметрами

Створи або відкрий базу даних зоомагазину з таблицею Тварини (поля: ID, Кличка, Вид, Вік, Ціна).

Порядок виконання:

- Створення запиту:
  - Відкрий Конструктор запитів.
  - Додай таблицю Тварини.
  - Вибери поля: Кличка, Вік, Ціна.
  - У критеріях для поля Вид укажи [Введіть вид тварини:].
  - Збережи запит як Пошук_тварин_за_видом.
- Додаткова умова: Додати критерій для поля «Ціна» <1000, щоб показати тільки доступні тварини певного виду.

Завдання 4. SQL у зоомагазині

1. Створи або відкрий базу даних зоомагазину з таблицею Тварини (поля: ID, Кличка, Вид, Вік, Ціна).
2. Напиши SQL-запит:

```
SELECT Назва, Ціна
FROM Зоотовари
WHERE Категорія = «Корм» AND В_наявності = True
ORDER BY Ціна;
```

Поясни, що саме виконує цей запит.

Мета проєкту: створити базу даних для автоматизованого введення інформації про наявні автомобілі, їх моделі та виробників. Навчитися створювати таблиці, будувати зв'язки, використовувати запити, формувати звіти й працювати з формами.

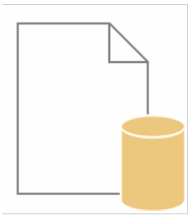
Етап 1. Організаційний

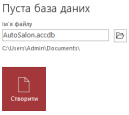
Завдання: Створити нову базу даних **AutoSalon.accdb** у **Microsoft Access**.

Етап 2. Підготовчий

План дій:

1. Спроєктувати структуру таблиць
2. Встановити зв'язки між таблицями
3. Продумати можливі запити та звіти
4. Створити зручні форми для введення та перегляду даних





Етап 3. Проєктний:

1. Структура таблиць:

Поле	Тип даних
Код_Виробника	Автонумерація
Назва_Виробника	Короткий текст (ключове поле)
Країна	Короткий текст
Логотип	Поле OLE

Таблиця: Виробники

Поле	Тип даних
Код_Моделі	Автонумерація
Назва_Моделі	Короткий текст (ключове поле)
Тип_Кузова	Короткий текст
Об'єм_Двигуна	Числовий
Потужність	Числовий
Виробник	Короткий текст

Таблиця: Моделі

Поле	Тип даних
VIN_Номер	Текстовий (ключове поле)
Модель	Короткий текст
Колір	Короткий текст
Рік випуску	Короткий текст
Ціна	Грошова одиниця
Наявність	Так/Ні (логічний)
Фото_Автот	Поле OLE

Таблиця: Автомобілі

Заповнити таблиці мінімум:

- Виробники – 5 записів (наприклад: Mercedes, Toyota, BMW, Ford, Tesla…)
- Моделі (до кожного виробника підшукати інформацію в інтернеті про їх моделі) – 10 записів
- Автомобілі – 15 записів

Всі фото до таблиць знаходити в інтернеті. Якщо в учня немає змоги віднайти зображення, замість вставки можна додати текст «Фото відсутнє». Для цін, потужностей та років – реалістичні дані

Зразки таблиць:

Код_Виробника	Назва_Виробника	Країна	Логотип
1	BMW	Німеччина	Bitmap Image
2	Mercedes	Німеччина	Bitmap Image
3	Ford	США	Bitmap Image
4	Renault	Франція	Bitmap Image
5	Opel	Німеччина	Bitmap Image
6	Tesla	США	Bitmap Image
7	Nissan	Японія	Bitmap Image
8	Skoda	Чехія	Bitmap Image

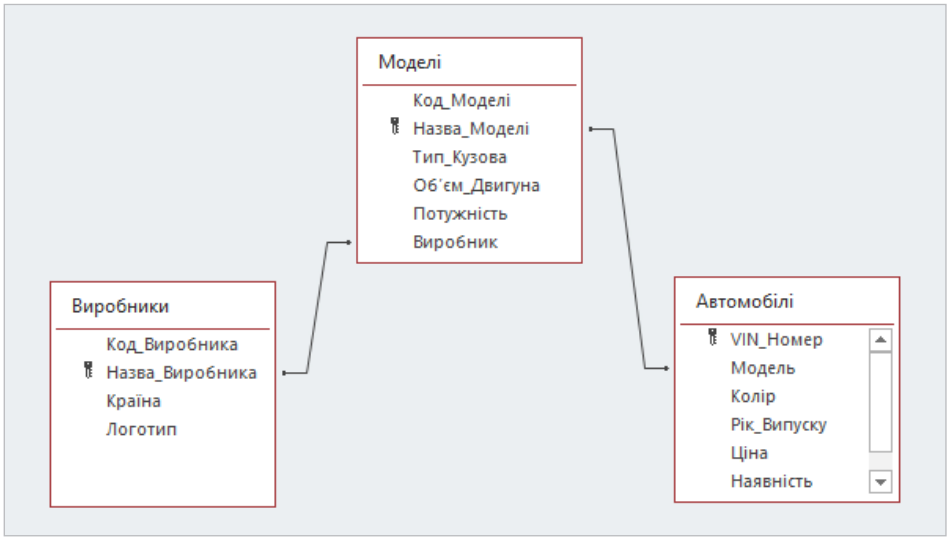
Код_Моделі	Назва_Моделі	Тип_Кузова	Об'єм_Двигуна	Потужність	Виробник
1	Scenic	універсал	2	140	Renault
2	Clio	хетчбек	1,6	109	Renault
3	Focus 2	універсал	1,6	115	Ford
4	Edge	хетчбек	2	156	Ford
5	Fabia	седан	1,4	94	Scoda
6	Clio u	універсал	1,8	137	Renault
7	Model X	седан		670	Tesla
8	Focus 3	універсал	2	132	Ford

VIN_Номер	Модель	Колір	Рік_Випуску	Ціна	Наявність	Фото_Авто
2FMTK4J92FBB74534	Edge	Сірий металік	2015	14 000,00 ₴	☑	Bitmap Image
2FMPK4K98PBA14748	Edge	Сірий металік	2024	24 900,00 ₴	☑	Bitmap Image
VF1BR2HDH46494545	Clio	Чорний	2008	5 500,00 ₴	☐	Bitmap Image
WF0SXXGCDSAB27477	Focus 2	Білий	2011	6 400,00 ₴	☑	Bitmap Image

2. Схема зв'язків:

Виробники (1) → (∞) Моделі – за полем Назва_Виробника

Моделі (1) → (∞) Автомобілі – за полем Назва_Моделі



Звіритись чи зв'язки працюють належним чином, переглянувши свої таблиці

Код_Виробника	Назва_Виробника	Країна	Логотип	Клацніть, щоб додат
1	BMW	Німеччина	Bitmap Image	
2	Mercedes	Німеччина	Bitmap Image	
3	Ford	США	Bitmap Image	

Код_Моделі	Назва_Моделі	Тип_Кузова	Об'єм_Двигуна	Потужність	Клацніть
3	Focus 2	універсал	1,6	115	
4	Edge	хетчбек	2	156	

VIN_Номер	Колір	Рік_Випуску	Ціна	Наявність	Фото_Авто	Клацніть, ц
2FMTK4J92FBB	Сірий металік	2015	14 000,00 ₴	☑	Bitmap Image	
2FMPK4K98PB	Сірий металік	2024	24 900,00 ₴	☑	Bitmap Image	
		0	0,00 ₴	☐		

8	Focus 3	універсал	2	132
(Новий)			0	0

4	Renault	Франція	Bitmap Image
5	Opel	Німеччина	Bitmap Image
6	Tesla	США	Bitmap Image
7	Nissan	Японія	Bitmap Image
8	Skoda	Чехія	Bitmap Image

3. Запити:

Вибірка автомобілів з потужністю більше 120 к.с.

Вивести виробника, модель, рік, потужність, ціну

Поле:	Назва_Виробника	Модель	Рік_Випуску	Потужність	Ціна
Таблиця:	Виробники	Автомобілі	Автомобілі	Моделі	Автомобілі
Сортування:					
Відображення:	☑	☑	☑	☑	☑
Критерії:					
Або:					

Назва_Виробника	Модель	Рік_Випуску	Потужність	Ціна
Ford	Edge	2015	156	14 000,00 ₴
Ford	Edge	2024	156	24 900,00 ₴
Ford	Focus 2	2011	115	6 400,00 ₴
Renault	Scenic	2013	140	5 800,00 ₴

Авто певного класу (з параметром)

Користувач вводить клас авто (наприклад, універсал) – виводяться усі відповідні

Поле:	Назва_Виробника	Назва_Моделі	Тип_Кузова
Таблиця:	Виробники	Моделі	Моделі
Сортування:			
Відображення:	☑	☑	☑
Критерії:			
Або:			

Назва_Виробника	Назва_Моделі	Тип_Кузова
Ford	Focus 2	універсал
Renault	Scenic	універсал

Додаткові запити:

Авто наявні в салоні, відсортовані за ціною

Тільки ті, в яких Наявність = «Так», сортування за зростанням

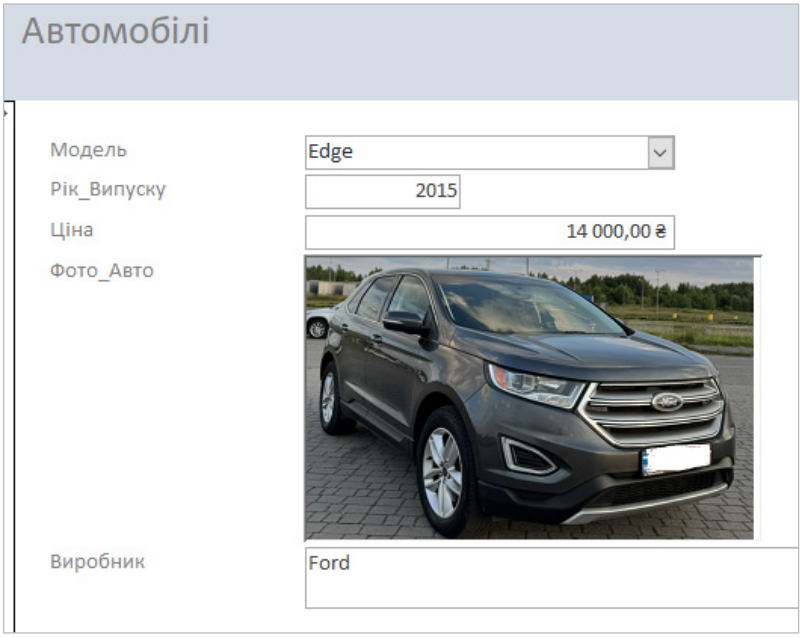
Авто за роком випуску

Параметр: рік. Виводить усі авто, випущені не раніше заданого року

4. Форми:

Головна форма «Каталог автомобілів»

Включає інформацію з трьох таблиць: Виробник → Модель → Автомобіль. Наприклад:



Додатково: у режимі конструктора відформатувати форму, змінивши кольори, шрифти та положення об'єктів.

5. Звіти:

Візитка моделі

На кожній сторінці — окрема модель (з фото, технічними характеристиками)

Додатково:

Прайс-лист наявних авто

Табличний звіт із назвою, кольором, роком, ціною

Відсортований за виробником

4 етап. Тестувальний

Внесення тестових даних

Перевірка правильності зв'язків, запитів, форм і звітів

Виправлення виявлених помилок

5 етап. Презентація:

Демонстрація бази даних у дії Обговорення логіки побудови

Розділ 3

ТРИВИМІРНЕ МОДЕЛЮВАННЯ

Тема 19 Програми для тривимірного моделювання. Етапи створення тривимірного зображення. Поняття моделі, візуалізація, віртуальний світ. Галузі використання 3D-моделей. Види моделей. Інтерфейс SketchUp, головні інструменти

Ключові поняття

3D-графіка
Тривимірне моделювання (3D-моделювання)
Візуалізація
Програми для 3D-моделювання

3D-графіка — це особливий різновид комп'ютерної графіки – сукупність методик та інструментів, що використовуються для створення малюнків тривимірних об'єктів.

Тривимірне моделювання (3D-моделювання) — це процес створення цифрової моделі об'єкта у трьох вимірах: висота, ширина, глибина.

Ключові ознаки 3D-моделі:

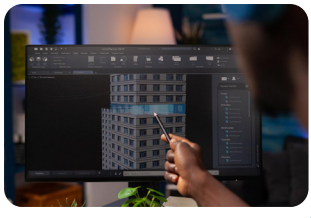
- має об'єм та форму;
- створюється у цифровому середовищі;
- доступна для перегляду з різних ракурсів.

Тривимірний малюнок легко можна відрізнити від двовимірного, бо в ньому є геометрична проєкція 3D-моделі на площину, що з'являється завдяки спеціальним програмам.

Галузі використання 3D-моделей

Архітектура

У галузі архітектури 3D-моделі допомагають проєктувати будинки, планувати інтер'єри та створювати візуалізації майбутніх споруд. Наприклад, так можна показати зацікавленим особам, який вигляд матиме школа ще до початку будівництва.



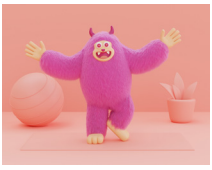
Медицина

У медицині тривимірні моделі застосовують для навчання і підвищення кваліфікації, планування операцій та виготовлення протезів. Наприклад, у такий спосіб хірургічна бригада може дослідити анатомічну модель серця конкретної людини у 3D перед проведенням складного хірургічного втручання.

Ігрова індустрія

В ігровій індустрії 3D-моделі є основою для створення персонажів, середовищ та об'єктів, які бачить гравець чи гравчиня у грі. Наприклад, усі персонажі та будівлі в грі Minecraft чи The Sims є 3D-моделями.

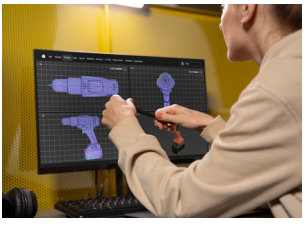
Мультфільми та кіно



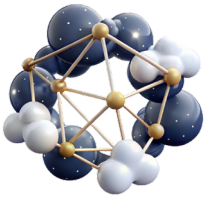
У кіно та мультфільмах моделі застосовують для створення візуальних ефектів. Наприклад, у фільмі «Аватар» практично всі фантастичні істоти були створені саме за допомогою 3D-моделювання.

Виробництво

У промисловості та виробництві інженери моделюють деталі машин та механізмів, щоб перевірити їхню сумісність ще до виготовлення. Наприклад, перед створенням нової моделі смартфона інженери будують його віртуальну 3D-модель і тестують її у цифровому середовищі.



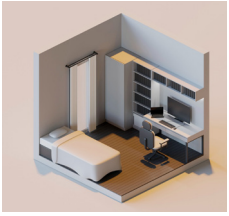
Освіта і наука



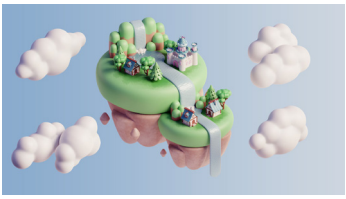
У науці та освіті 3D-моделі дають змогу краще вивчати складні об'єкти: молекули, органи, будову клітини або фізичні явища. Усі зацікавлені можуть віртуально «розібрати» автомобіль або вивчити внутрішню будову вулкана.

Модель — це абстрактне уявлення реального чи вигаданого об'єкта.

Візуалізація — процес перетворення математичної моделі у зображення, яке можна побачити.



Віртуальний світ — штучне цифрове середовище, створене з використанням 3D-моделей.



Етапи створення тривимірного зображення

1. *Створення ТЗ з урахуванням необхідних показників та наявних креслень.* У завданні вказують технічні дані: техніки, що використовуються, а також матеріали, розмір.

2. *Створення 3D-об'єкта із зазначеними характеристиками.* Для цього використовують професійні програми.

3. *Розробка реалістичної моделі.* На цьому етапі додають текстуру, виконують обробку з використанням растрових технологій. Далі підбір світла, вид джерел освітлення, глибина тіней. Обирають точку спостереження, залежно від типажу об'єкта: для окремих типів підходить вид з висоти зросту людини, а для інших — із висоти пташиного польоту.

4. *Рендеринг* (розробка реалістичного зображення). При обробці в програмі додають відблиски, відсвічування та інше. Створення чіткої презентації у форматі відео.

5. *Експорт* – збереження моделі у форматі, придатному для перегляду або друку.

Програми для 3D-моделювання

Blender – безкоштовна програма з відкритим кодом, яка містить широкий набір функцій для 3D-моделювання, скульптингу, рендерингу та анімації. Завдяки активній спільноті та постійним оновленням Blender є відмінним вибором як для початківців, так і для професіоналів.

Tinkercad – простий у використанні вебдодаток для 3D-моделювання, який ідеально підходить для новачків. Він дозволяє створювати моделі за допомогою базових геометричних форм та працює одразу в браузері, що робить його доступним для широкого кола користувачів.

SketchUp – зручна програма для швидкого 3D-моделювання, яка активно використовується в архітектурі та дизайні. Її простий інтерфейс дозволяє легко створювати 3D-моделі без необхідності глибоких технічних знань.

Види 3D-моделей

Фізичні моделі

Це моделі, які імітують форму та геометрію реальних об'єктів, без додаткової інформації про властивості матеріалу.

Приклад: у **SketchUp** можна змоделювати стілець або будинок, щоб побачити їхню форму з усіх боків.

Де використовується:

- У навчанні (наприклад, моделювання простих предметів на уроках).
- В архітектурі (первинні форми будівель до деталізації).

Інформаційні моделі

Ці моделі містять додаткові параметри — матеріали, текстури, вагу, міцність, теплопровідність тощо.

Де використовується:

- У будівництві (інформаційне моделювання будівель).
- У виробництві (моделювання деталей машин із точними матеріалами).
- В інженерії — аналіз поведінки конструкцій при навантаженнях.

Схематичні моделі

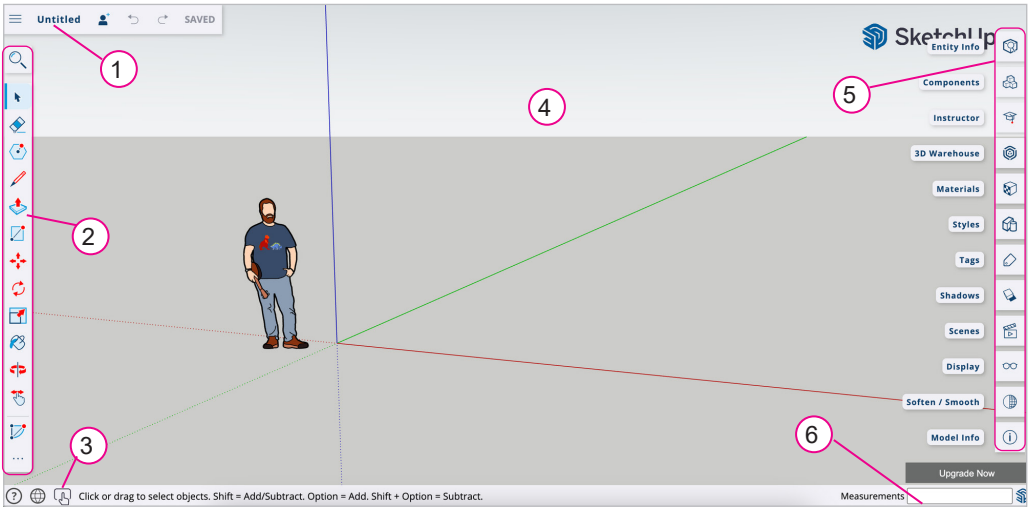
Це спрощені геометричні моделі, які показують загальну структуру, логіку або процес, але без деталей.

Приклад: блок-схема вентиляційної системи будівлі або спрощена 3D-модель маршруту труб.

Де використовується:

- У плануванні та аналізі.
- У технічній документації.
- У навчанні, коли потрібно пояснити принцип роботи системи (наприклад, рух електронів у колі).

SketchUp for Web — це браузерна версія популярного 3D-редактора, яка дозволяє створювати тривимірні моделі без встановлення програми. Все, що потрібно — комп'ютер та доступ до інтернету.



(1) **Панель зверху** — містить опції збереження, відкриття файлів, експорту моделі, вибору шаблону та налаштувань проєкту.

(2) **Панель інструментів ліворуч** (**Toolbar**) — містить основні інструменти для малювання та редагування об'єктів. Усі інструменти мають піктограми, які зручно розміщені вертикально.

(3) **Палець з курсором** – підказки щодо дій з мишею.

Показує коротку підказку щодо поточного активного інструмента.

Наприклад: **Click or drag to select objects** – натисніть або перетягніть, щоб вибрати об'єкти.

Якщо обрати інший інструмент — підказка зміниться відповідно.

④ **Робоча область** (**Modeling Window**) — головна зона, де ви створюєте та переглядаєте модель. Тут видно координатні осі (синя, червона, зелена), які допомагають орієнтуватися у просторі.

⑤ **Меню праворуч** (**Default Tray**) — відкривається при натисканні на піктограму «панелі». Там можна змінювати кольори, застосовувати текстури, налаштовувати шари, сцени тощо.

⑥ Це поле для точного введення розмірів або відстаней.

Наприклад, під час малювання прямокутника або переміщення об'єкта можна просто навести мишку й ввести розмір вручну (5m, 300cm, 12' 6) тощо — і натиснути **Enter.**

Підказки

Затискайте клавішу **Shift** + коліщатко миші для панорамування (**Pan**) – це дозволяє «рухати камеру» вліво, вправо, вгору чи вниз без обертання сцени.

Зберігайте проєкт вручну (**SAVE**).

Хоча автозбереження працює, краще періодично натискати **SAVE**, особливо перед закриттям вікна.

Спробуйте копіювати об'єкти з **Ctrl.**

Інструмент **Move** (**Переміщення**) + клавіша **Ctrl** (або **Option** на **Mac**) створює копію об'єкта.

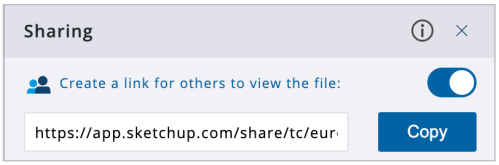
≡ (три смужки) — **Меню** (**Main Menu**)
Відкриває головне меню:
– **Створити новий проєкт**
– **Відкрити файл**
– **Імпортувати** / **Експортувати**
– **Налаштування** тощо

Стрілка вліво — скасувати дію (**Undo**)
Повертає на один крок назад.
Наприклад, видалили об'єкт — можна швидко повернути назад.
Стрілка вправо — повторити дію (**Redo**)
Якщо ви скасували дію помилково — ця кнопка поверне її назад.

Untitled — назва поточного проєкту
Натиснувши на назву, можна перейменувати файл (наприклад, Будинок 3D).

Додати учасника до свого проєкту
Поділитися проєктом із однокласниками (якщо ви вже зберегли проєкт)

SAVE (**Зберегти**) — зберегти проєкт вручну



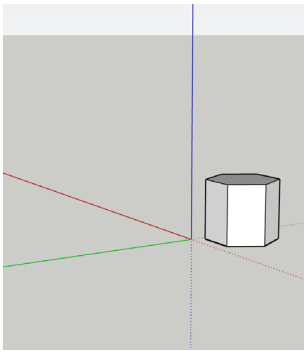
У вебверсії зберігання часто відбувається автоматично (автосейв), але натискання цієї кнопки примусово зберігає зміни.
Це важливо, якщо ви плануєте закрити вікно або перейти до іншого проєкту.

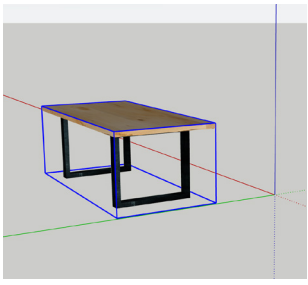
У **SketchUp** і будь-якому іншому 3D-редакторі об'єкти створюють у тривимірному просторі, який має три осі.

Червона вісь (X) позначає горизонтальний напрямок зліва направо. Вона допомагає визначити, наприклад, довжину об'єкта або будівлі.

Зелена вісь (Y) також є горизонтальною, але йде в напрямку вперед-назад. Зазвичай її використовують для створення глибини, наприклад, ширини кімнати.

Синя вісь (Z) — вертикальна. Вона визначає напрямок вгору-вниз і використовується для задання висоти об'єкта, наприклад, висоти стіни або даху.





Гарячі клавіші

R — Прямокутник
C — Коло
P — Push/Pull
M — Переміщення
S — Масштабування
Q — Обертання
L — Лінія
E — Гумка (Erase)
Space — Виділення

Select (Вибір) — використовується для вибору об'єктів.
Фішка: **Shift** + клацання — додає або знімає об'єкт з вибору.
Двічі клацання — вибір об'єкта + його поверхні; тричі — вибір усієї групи.

Eraser (Гумка) — стирає лінії та об'єкти.

Rectangle / **Polygon** / **Circle** — малюють базові фігури: прямокутник, багатокутник, коло.

Push/Pull (**Витягування**) — витягує 2D-форми у 3D-об'єкти.

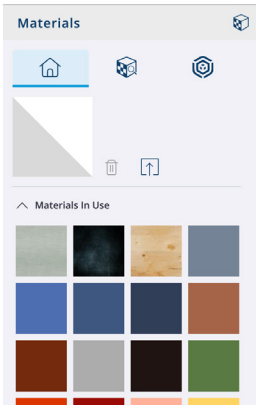
Line (**Лінія**) — малює прямі лінії. Основний інструмент для побудови.
Фішка: утримуй **Shift** для фіксації напрямку по осях (X, Y, Z).

Move (**Переміщення**) — пересуває об'єкти або копіює.
Фішка: утримуй **Ctrl** (**Cmd**) під час переміщення — створення копії.
Стрілками фіксуй рух по конкретній осі.

Rotate (**Обертання**) — обертає об’єкти навколо точки.
Фішка: введи кут повороту вручну (наприклад: 45).

Scale (**Масштабування**) — змінює розмір об’єктів.
Фішка: введи точне значення — наприклад, 2 → збільшити вдвічі.

Заливка кольором або текстурою.
Фішка: можна вибрати текстуру (наприклад, дерево, цегла) з бібліотеки.



Обертання камери навколо сцени.
Натисни коліщатко миші **+ Shift** — активується обертання.

Pan Tool (**Рука**) — переміщення вигляду камери вбік/вгору/вниз.
Фішка: натисни коліщатко миші, щоб швидко перейти в режим **Pan**.

Zoom Tool (**Збільшення**)
Збільшення або зменшення масштабу.
Прокручуй коліщатко миші — швидкий Zoom.

Три крапки внизу (**More**) — відкривають додаткові інструменти, які не вмістилися в панелі (наприклад, вимірювання, секції, компоненти).

Практична робота №19

Завдання 1. Створити сходинки.

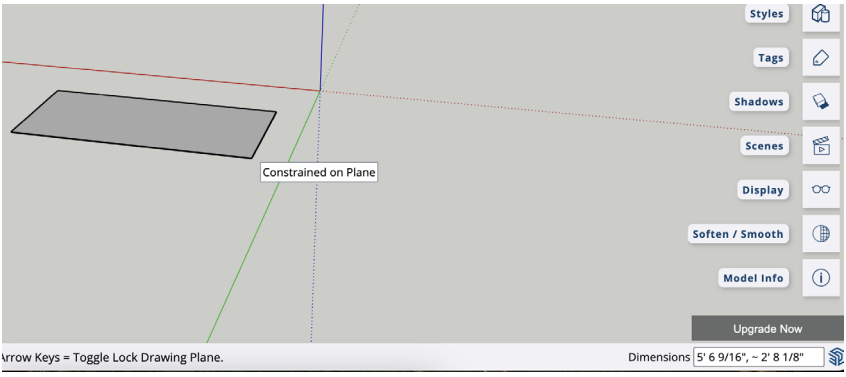
Хід роботи

1. Зайдіть у **SketchUp for Web**.

- Перейдіть за посиланням: **https://app.sketchup.com**
- Увійдіть у свій **Google-акаунт** або створіть новий.
- Натисніть **Start Modeling** або **+ New**, щоб створити новий проєкт.

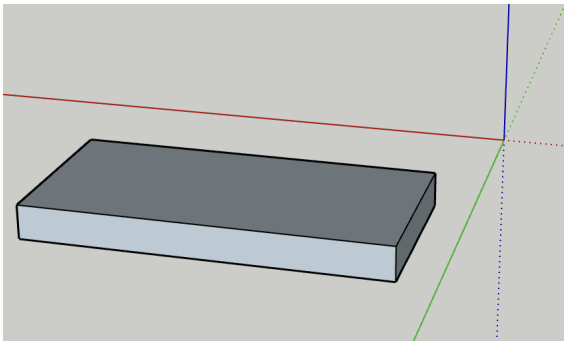
2. Створіть перший прямокутник.

- Оберіть інструмент **Rectangle** (**Прямокутник**).
- Намалюйте основу першого ступеня, наприклад, 100 см х 30 см.
- Для точних розмірів — після натискання введіть 100, 30 і натисніть **Enter**.



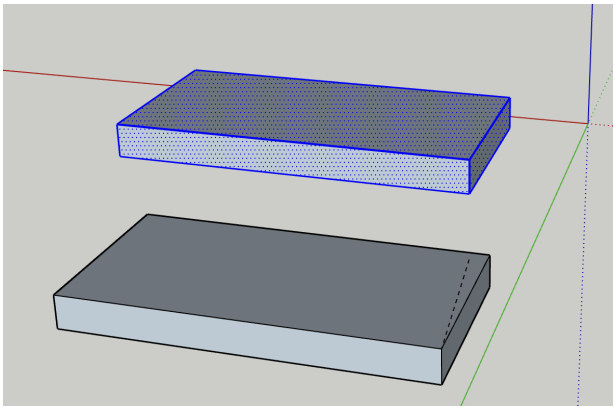
3. Витягніть форму.

- Виберіть інструмент **Push/Pull**.
- Натисніть на прямокутник і витягніть вгору на 15 см (або введіть 15 і **Enter**).



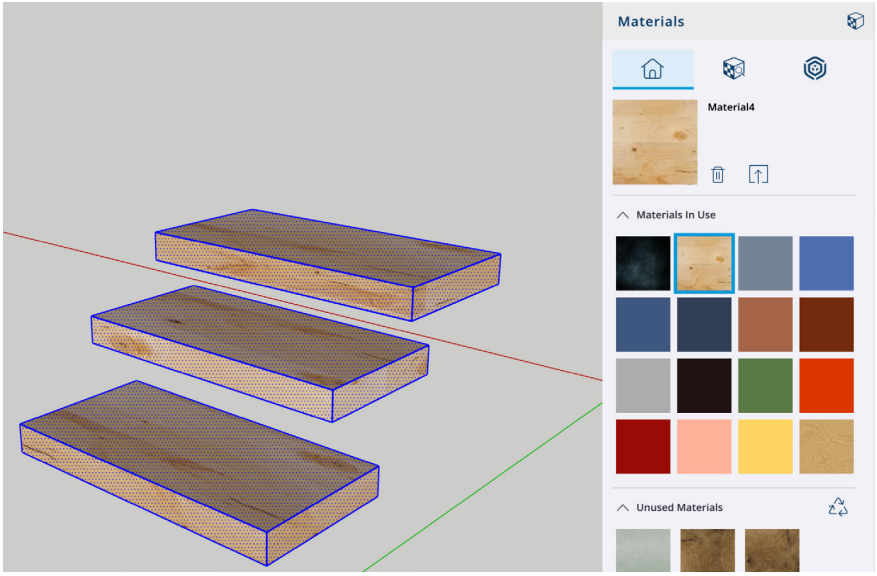
Створіть наступну сходинку.

- Оберіть **Move Tool**.
- Утримайте **Ctrl** (або **Option** на **Mac**), щоб зробити копію ступеня.
- Перетягніть копію трохи назад і вгору (наприклад, на 30 см назад і на 15 см вгору).
- Якщо з'являється зелена і синя підсвітка — ви все робите правильно по осях.

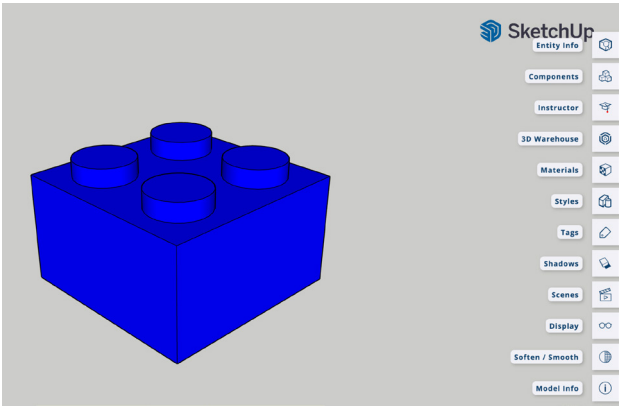


Повторіть для наступних сходинок.

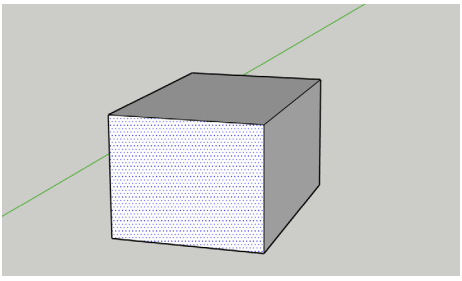
- Ще раз скопіюйте попередню сходинку.
- Створіть 3–5 ступенів.



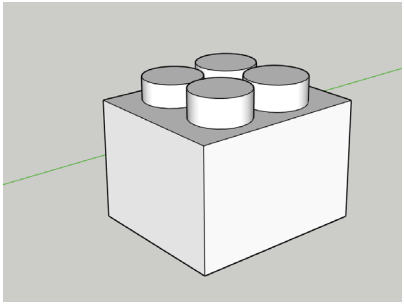
Завдання 2. Створити LEGO-подібну цеглинку з 4 круглими виступами.



1. Створіть основу (куб).
Інструмент: **Rectangle Tool**
Клацніть на площині й створіть прямокутник (наприклад, 4x4 см).
Push/Pull Tool → витягніть вгору (наприклад, 2 см).

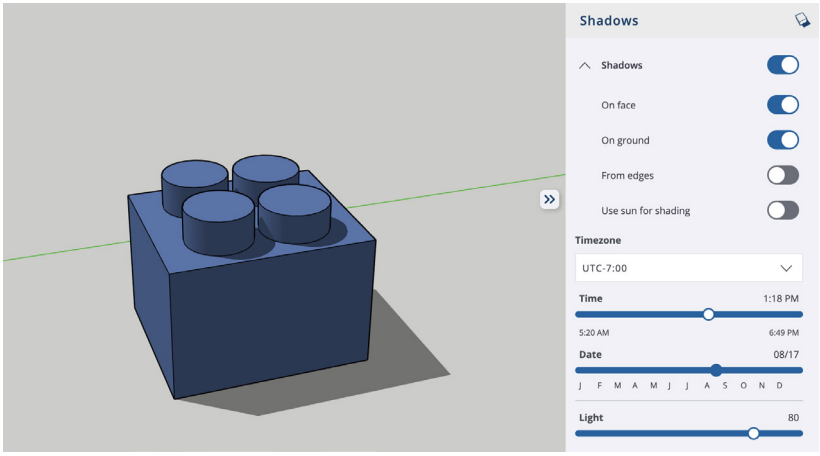


2. Створіть перший виступ.
Circle Tool → виберіть верхню грань куба.
Намалюйте коло (радіусом близько 0.6 см) у верхньому лівому куті.
Push/Pull Tool → витягніть угору (наприклад, 0.4 см).

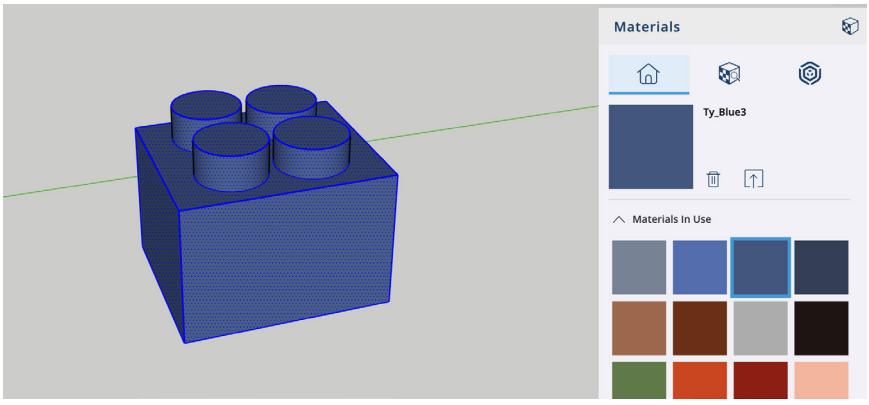


3. Зробіть копії виступу.
Виділіть круглий виступ (**Select Tool**).
Оберіть **Move Tool** → наведіть на центр циліндра.
Утримуйте **Ctrl** (або **Option** на **Mac**) — це задіює режим копіювання.
Потягніть копію в інший кут → клацніть.
У полі введення напишіть ***3** → натисніть **Enter** — отримаєте 3 додаткові копії = всього 4 виступи.

4. Змініть колір.
Відкрийте **Materials (праворуч)**.
Оберіть синій колір.
Залийте всю фігуру (куб + виступи).



Можна додати тінь.



Тема 20 **Основні інструменти SketchUp. Лінія, прямокутник, коло. Інструменти переміщення, обертання, масштабування. Орієнтація в просторі. Гіпотеза. Прогноз. Моделювання як спосіб наукового пізнання**

Ключові поняття

Моделювання
Інструменти трансформації
Основні інструменти створення фігур

Моделювання — це створення спрощеної копії об'єкта чи явища для його вивчення.

Гіпотеза: якщо зменшити висоту парти на 5 см, особам нижчого зросту буде зручніше сидіти.

Прогноз: зменшена висота дозволить зменшити навантаження на спину, і діти не будуть сутулитися.

Моделювання: створити 3D-модель стандартної парти та її зменшеного варіанту → порівняти розміщення стільця й рук учня на моделі → зробити висновок.

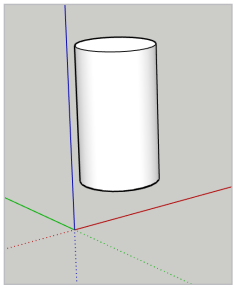
Основні інструменти створення фігур

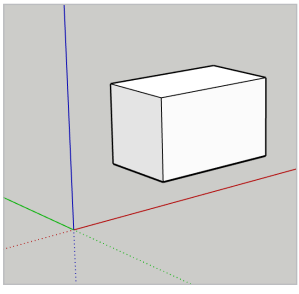
Line Tool (лінія) — для побудови відрізків і створення замкнених контурів.

Rectangle Tool (прямокутник) — простий спосіб побудови площини.

Circle Tool (коло) — створення кругів і бази для циліндрів.

Приклад: створення циліндра: коло + інструмент **Push/Pull**.





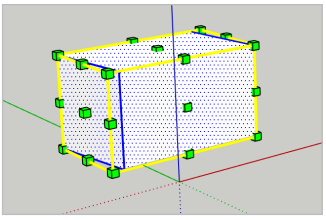
Інструменти трансформації

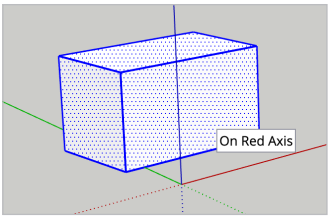
Move Tool — переміщення або копіювання (утримати **Ctrl**).

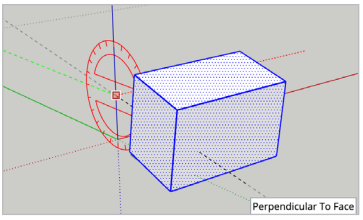
Rotate Tool — обертання об'єкта навколо осі.

Scale Tool — масштабування об'єкта в одному або кількох напрямках.

Комбінація клавіш: утримуйте **Ctrl** (**Windows**) або **Option (Mac)** під час переміщення — буде створено копію.







У **SketchUp** ви можете вмикати і налаштовувати тіні для своїх 3D-моделей, щоб надати їм більш реалістичного вигляду.

On face – тіні видно на поверхнях об'єкта.

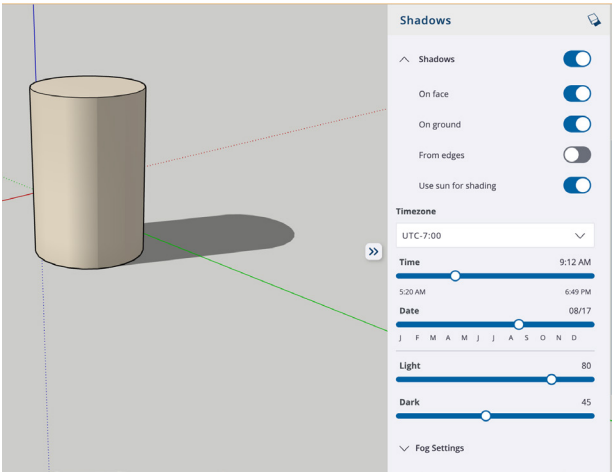
On ground – тіні падають на землю.

From edges – увімкнення тіней від країв об'єкта.

Use sun for shading – використовує сонячне освітлення, щоб відтворити реальні тіні залежно від часу доби.

Timezone / **Time** / **Date** – оберіть часовий пояс, а потім встановіть конкретний час і дату, щоб побачити, як змінюється положення тіні.

Light / **Dark** – регулювання яскравості світла й насиченості тіні.

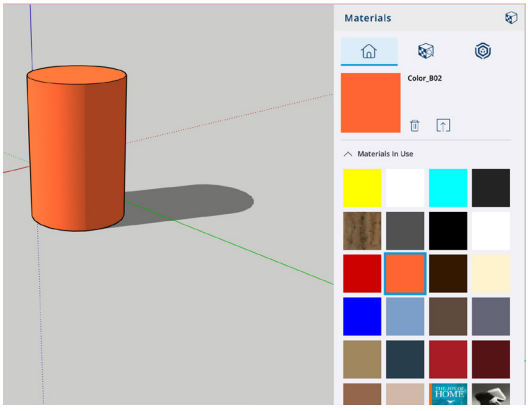


SketchUp дозволяє обирати й застосовувати різні текстури та кольори до будь-якої поверхні моделі.

Обирайте колір зі списку **Materials in Use**.

Використовуйте деревину, метал, скло, бетон або просто кольори.

Можна залити поверхню матеріалом одним клацанням за допомогою інструмента фарби (**Paint Bucket Tool**).



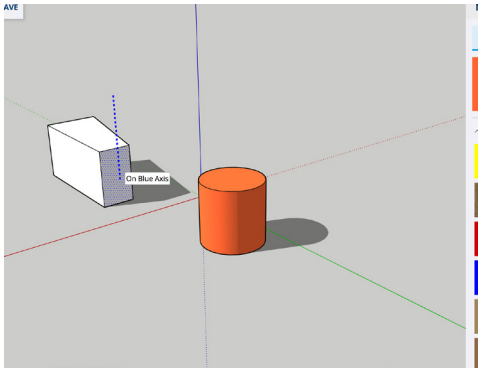
Орієнтація в просторі — це одна з базових навичок для роботи в **SketchUp**. Вона допомагає правильно розміщувати об'єкти, уникати помилок побудови та краще розуміти структуру моделі.

Підказки кольором

SketchUp підсвічує напрямки.

Коли ви ведете лінію вздовж червоної, зеленої або синьої осі — вона автоматично підсвічується відповідним кольором.

Це візуальна підказка, яка допомагає не збитися з просторового напрямку.



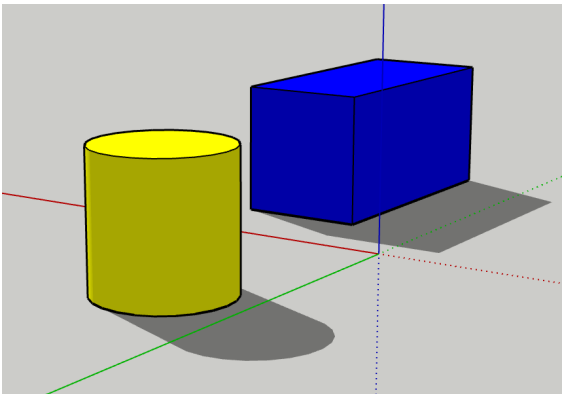
Приклад: при витягуванні форми вгору за допомогою інструмента **Push/Pull** або **Move** варто стежити, щоб напрям був по синій осі — тоді об'єкт збільшується чітко вгору.

Інструмент **Orbit** (**Огляд сцени**)

Допомагає «обертати камеру» навколо об'єкта, щоб подивитись на модель з різних боків.

Це ключовий інструмент для просторового розуміння моделі.

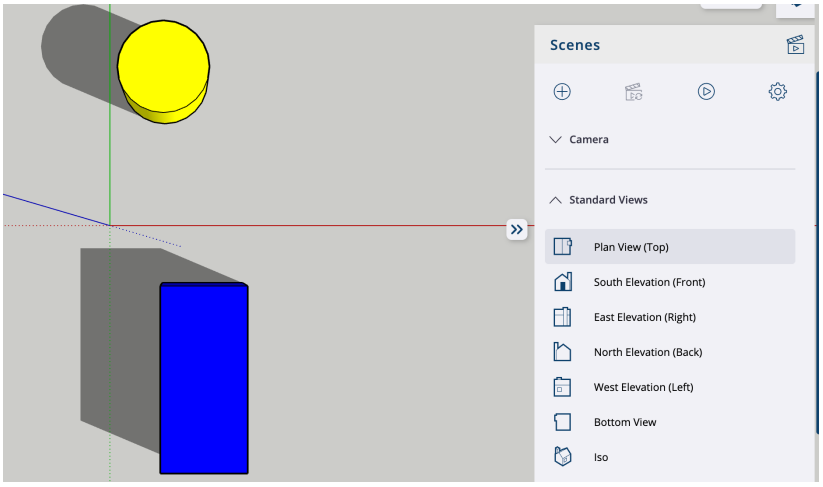
Утримуйте коліщатко миші та рухайте мишею — активується режим **Orbit**.



Панель інструментів «Views»

У **SketchUp** можна вибрати стандартні види: фронтальний, вигляд згори, зліва, перспектива тощо.

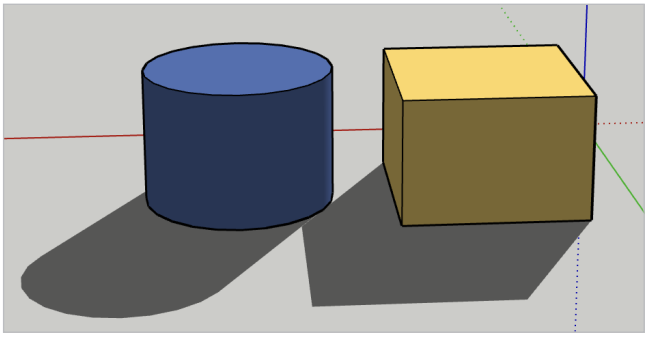
Це зручно для точного вирівнювання та моделювання.



Практична робота №20

Проста 3D-фігура з базових форм

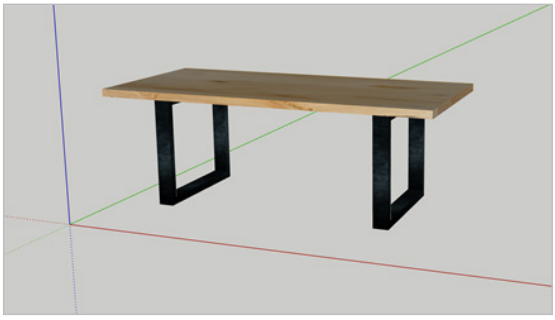
1. Створіть новий проєкт у **SketchUp.**
2. Накресліть прямокутник → витягніть його вгору інструментом **Push/Pull**.
3. Накресліть зверху коло → витягніть його в циліндр.
4. Додайте заливку та тінь.
5. Обертайте сцену інструментом **Orbit**.
   Результат: модель, що складається з цеглинки + циліндра.



Створення моделі стола

Рекомендовані розміри (можна адаптувати):

- Стільниця: 1400 мм × 700 мм × 40 мм
- Ніжка (2 шт.): 700 мм (висота), 20 мм (товщина), 500 мм (ширина)

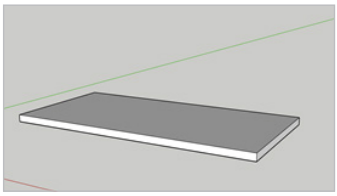


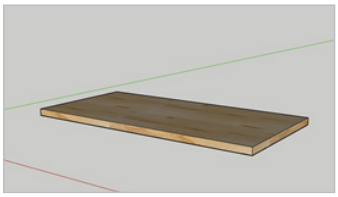
Створіть новий проєкт у **SketchUp.**

Створення стільниці

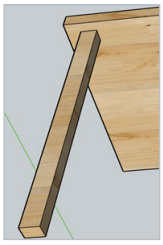
- Оберіть інструмент **Rectangle**.
- Намалюйте прямокутник на червоно-зеленій площині (наприклад, 1400 мм × 700 мм).

- Оберіть **Push/Pull** і витягніть прямокутник вгору на висоту стільниці (наприклад, 40 мм).
- Оберіть **Paint Bucket** → **Materials** → оберіть дерево і залийте поверхню.

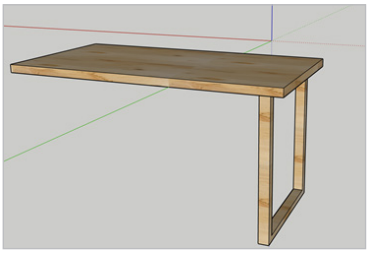




Створення ніжки



- Поверніться до інструмента **Rectangle**.
- Намалюйте прямокутник біля одного з кутів стільниці (наприклад, 50 мм × 20 мм).
- Витягніть його вниз на висоту ніжки (наприклад, 700 мм).
- Намалюйте ще один прямокутник (паралельно до першого) на підлозі (для з'єднувального елемента ніжки).
- З'єднайте вертикальні елементи перемичкою, утворюючи П-подібну форму.
- Виділіть усі частини ніжки, клацніть правою кнопкою миші → **Make Group**.



Копіювання ніжки

- Оберіть **Move Tool**, натисніть і утримуйте **Ctrl** (або **Option** на **Mac**).
- Перетягніть копію ніжки до протилежного краю стільниці.
- Переконайтесь, що вона вирівняна по осі.

Додавання матеріалів

- Оберіть інструмент **Paint**
- Для ніжок оберіть матеріал «метал» або темний колір.
- Для стільниці вже має бути обраний матеріал «дерево».

Робота з тінями

- Перейдіть до меню **Shadows** справа.
- Увімкніть **On ground** та **On face**, налаштуйте дату і час для реалістичного ефекту.
- За бажанням можна змінити інтенсивність світла.

Для точного введення розмірів після створення прямокутника або витягування — просто введіть значення з клавіатури та натисніть **Enter**.

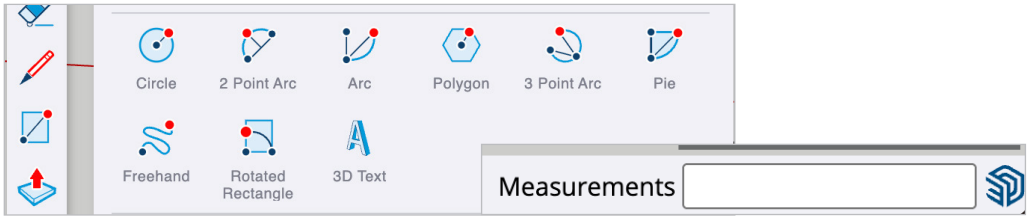
Використовуйте **Orbit** (обертання сцени) для контролю і перегляду моделі з різних боків.

Тема 21 **Робота з 3D-примітивами. Створення об'єктів за допомогою Push/Pull. Побудова елементарних форм (куб, коробка, стіл). Групування та компоненти. Створення груп і компонентів. Копіювання та редагування копій**

Ключові поняття

3D-примітиви
Створення груп і компонентів

3D-примітиви — це базові геометричні фігури, з яких будується складна 3D-модель.



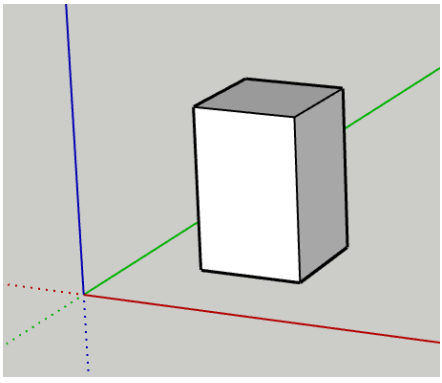
Повторне використання: примітиви можна перетворити на компоненти та копіювати.

Введення розмірів з клавіатури: наприклад, після створення прямокутника введіть 1000,600 — отримаєте стільницю розміром 1000x600 мм.

Інструмент Push/Pull (Витягування)

Цей інструмент перетворює площину (2D-фігуру) на об'ємний об'єкт (3D-примітив). Він є базовим для створення кубів, коробок, стільниць, стін тощо.

- Намалюйте прямокутник.
- Оберіть інструмент **Push/Pull**.
- Потягніть поверхню вгору — отримаєте паралелепіпед.
- Введіть точну висоту з клавіатури, натисніть **Enter**.



Якщо під час витягування площини натиснути клавішу **Ctrl** (або **Option** на **Mac**), то замість зміни форми створюється її копія — це зручно для повторюваних елементів на поверхні.

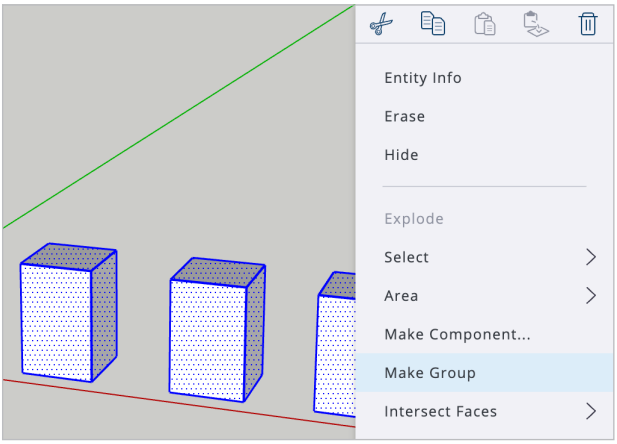
Якщо витягнути площину всередину тіла, **SketchUp** автоматично виріже порожнину в об'єкті. Цей прийом особливо корисний, наприклад, для створення дверцят, вікон або шухляд у моделях меблів.

Створення груп і компонентів

Група — об'єднання кількох елементів в один блок.

Компонент — розумна група, яка змінюється одночасно в усіх копіях.

Виділіть об'єкти / права кнопка мишки (контекстне меню) / **Make Group**.



- Легко переміщати, масштабувати без змін окремих частин.
- Права кнопка мишки (контекстне меню) / **Make Component.**
- Ідеально для повторюваних елементів (ніжки стола, вікна, плитка).

Практична робота №21

Створення стелажа з полицями

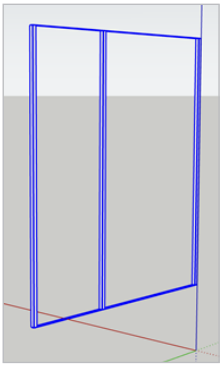
Рекомендовані розміри:

- Висота стелажа: 1500 мм
- Ширина: 1000 мм
- Глибина: 300 мм
- Кількість полиць: 5
- Товщина полиці: 20 мм
- Відстань між полицями: 300 мм

Створіть новий файл в **SketchUp Web**

Зайдіть на **app.sketchup.com** → **Start Modeling**.

Побудуйте одну вертикальну ніжку



- Використайте **Rectangle** → створіть основу 20×20 мм.
- Інструментом **Push/Pull** витягніть її на висоту 1500 мм.
- Зробіть групу (**Ctrl + G** / Права кнопка мишки / **Make Group**).

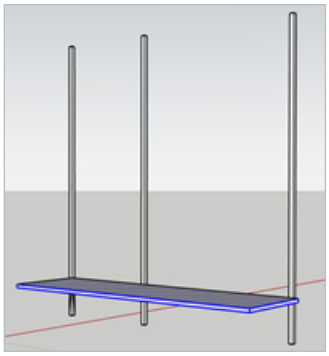
Скопіюйте інші 6 ніжок.

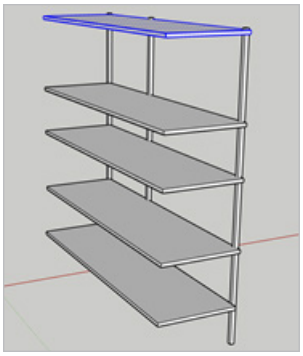
Оберіть **Move Tool**, затисніть **Ctrl** (або **Option**) → створіть копії для чотирьох кутів (за допомогою точного позиціювання по осях, наприклад, 800 мм та 300 мм).

Створіть першу полицю

- **Rectangle** → прямокутник 800×300 мм.
- **Push/Pull** → витягніть товщину полиці (наприклад, 20 мм).
- Зробіть групу.

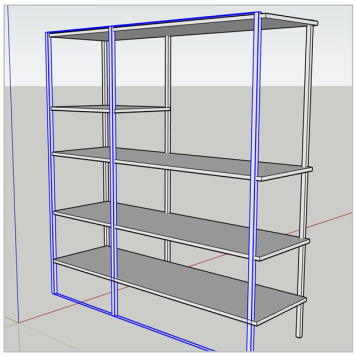
Звісно, полиці можете розмістити на свій вибір та змінити їхній розмір.





Як тільки розмістите всі полички — скопіюйте групу та розмістіть відповідно до поличок

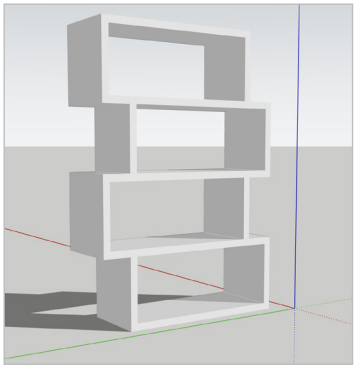
- Групуйте весь каркас стелажа
  Виділіть усі частини → **Права кнопка мишки** → **Make Group** / **Create Component**.
- Нанесіть матеріали
  - Перейдіть у праве меню **Materials**.
  - Оберіть дерево для полиць, чорний метал для ніжок.



Стелаж з виступаючими секціями

Рекомендовані розміри:

- Загальна висота: 1600 мм
- Ширина кожного блоку: 800 мм
- Висота блоку: 350 мм
- Глибина: 300 мм
- Товщина стінки: 30 мм
- Зсув кожного блоку вбік: 100 мм



Етапи виконання:

1. Створіть перший блок (рамку): **Rectangle** / **Push/Pull** / виріз всередині / знову **Push/Pull**.
2. Зробіть компонент блоку: **Права кнопка мишки** (контекстне меню) / **Make Component.**
3. Скопіюйте блок вгору тричі з кроком 400 мм: **Move Tool** + **Ctrl** / потім скопіюйте три рази.
4. Зсуньте блоки вліво/вправо по черзі.
   Орієнтуйтеся по осях (зсув ±100 мм).

- Нанесіть матеріал (білий або дерев'яний).
  Materials / оберіть відповідний колір або фактуру.
- Завжди працюйте по осях (червона, зелена, синя). Це допоможе уникнути перекосів і неточностей — модель буде «стояти рівно».
- Створюйте один елемент, а тоді копіюйте його. Не малюйте кожну секцію заново. Замість цього створіть компонент і використовуйте **Move** + **Ctrl**, потім повторіть три рази.
- Звертайте увагу на розміри при копіюванні.
  Орієнтуйтеся на точне розміщення за координатами (наприклад, зсув на 100 мм), щоб зберегти логіку конструкції.

Тема 22 **Створення інтер'єру (меблі, кімната). Побудова стін, підлоги, вікон. Моделювання елементів меблів. Екструдування форми об'єкта. Рендеринг тривимірної сцени. Що таке 3D Warehouse. Як завантажити готові моделі. Вставка в проєкт і адаптація**

Ключові поняття

Рендеринг
Екструдування форми об'єкта
Інтер'єр
3D Warehouse

Рендеринг — це процес, під час якого 3D-модель перетворюється у фотореалістичне зображення.

Тобто модель має вигляд справжнього фото.

Ми можемо створити візуально привабливу сцену за допомогою:

- матеріалів
- тіней
- точок огляду (**Scenes**)

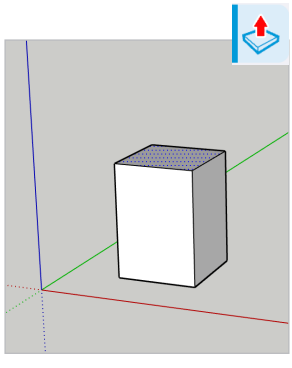
У **SketchUp** можна скористатися додатковими рендер-плагінами (**V-Ray**, **Enscape** тощо).

Екструдування форми об'єкта — це один із найважливіших процесів у 3D-моделюванні, і в **SketchUp** він реалізований за допомогою інструмента **Push/Pull** (**Виштовхування/Втягування**).

Екструдування (від англ. *extrude*) — це перетворення плоскої форми (наприклад, прямокутника чи кола) у тривимірний об'єкт шляхом витягування її вгору або вниз.

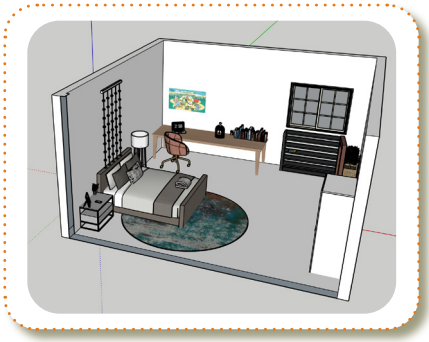
Push (**Втягування**) — вдавлює форму всередину (наприклад, створює отвір).

Pull (**Виштовхування**) — піднімає площину, формуючи 3D-фігуру.



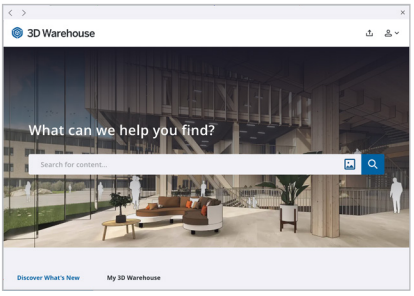
Інтер'єр — це внутрішній простір приміщення з усіма його елементами: стінами, підлогою, меблями, освітленням і декором.

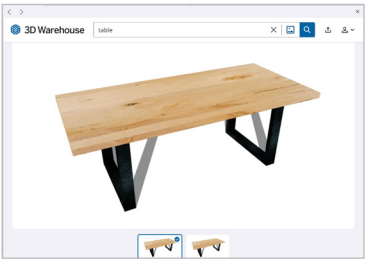
Основна мета інтер'єру — зробити простір зручним, функціональним та естетичним.



3D Warehouse — це велика онлайн-бібліотека, де зберігаються готові 3D-моделі для використання в проєктах **SketchUp**.

Це безкоштовний сервіс, який дозволяє завантажувати, переглядати та вставляти моделі у власні 3D-сцени.





Як користуватись 3D Warehouse в онлайн-версії SketchUp?

Відкрийте меню.

На верхній панелі натисніть значок трьох кубиків або кнопку **3D Warehouse**.

Введіть запит.

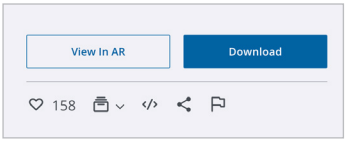
У полі пошуку напишіть назву потрібного об'єкта — англійською (наприклад, sofa, table, lamp).

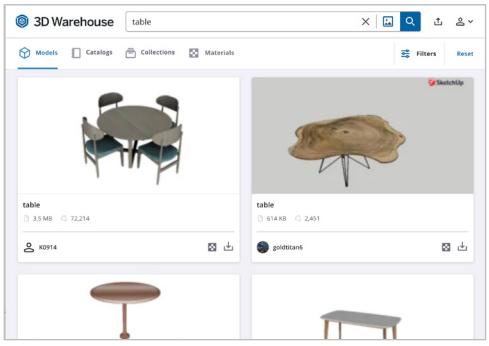
Перегляньте варіанти.

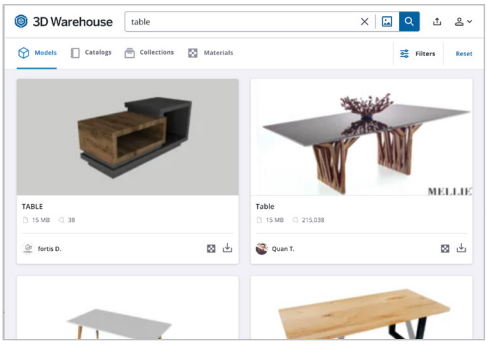
Сервіс відкриє мініатюри доступних моделей. Оберіть ту, яка найбільше підходить.

Завантажте.

Натисніть **Download** → підтвердіть, що хочете вставити модель у свій проєкт.







Що робити після вставлення моделі?

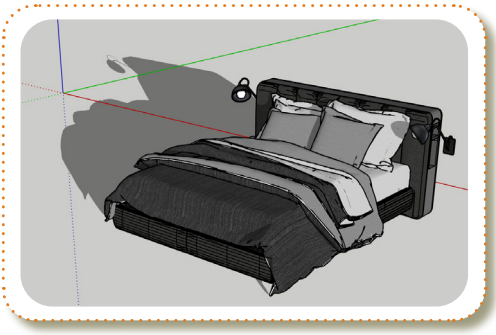
Модель автоматично з'являється в центрі сцени або там, де розміщений курсор.

Її можна:

- переміщати (**Move**);
- масштабувати (**Scale**);
- обертати (**Rotate**);
- перефарбовувати (**Materials**).

Поради:

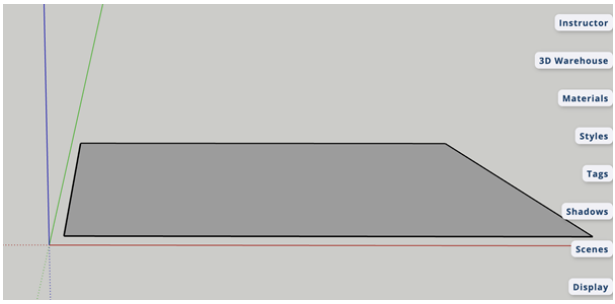
- Стежте, щоб розмір моделі відповідав масштабу вашого проєкту.
- Якщо об'єкт занадто складний, видаліть зайві деталі.



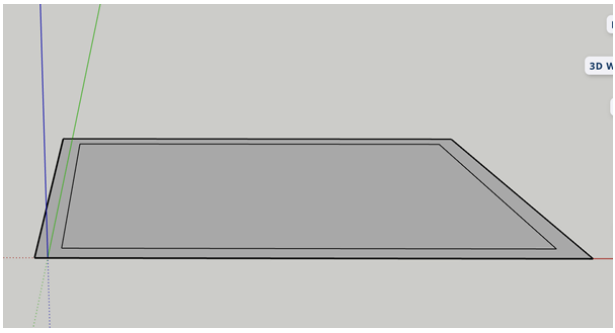
Практична робота №22

Завдання. Створення кімнати з елементами інтер'єру

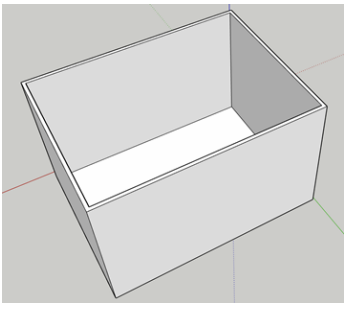
1. Відкрийте **SketchUp for Web** → натисніть **Start Modeling**.
2. **Rectangle Tool** — намалюйте основу кімнати (наприклад, прямокутник 4000 мм × 3000 мм).

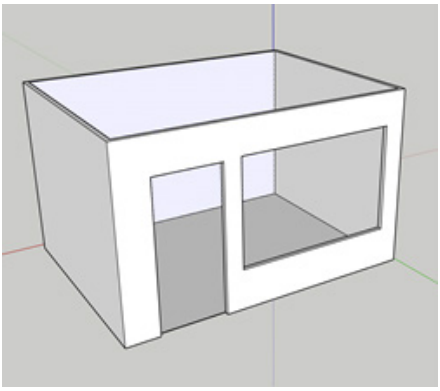


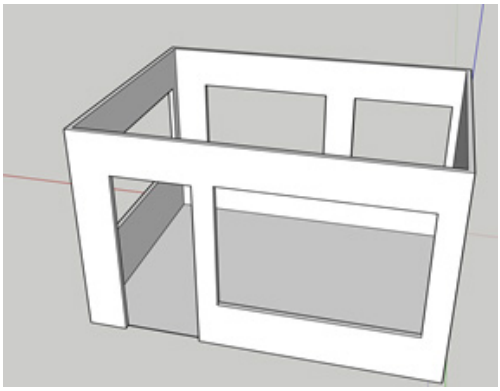
3. **Offset Tool** — оберіть створений прямокутник і зробіть внутрішній контур (наприклад, відступ на 200 мм).



4. **Push/Pull Tool**:
   - Виділіть проміжок між зовнішнім і внутрішнім прямокутником (тобто саме «стіни»).
   - Витягніть його вгору на висоту стін (наприклад, 2500 мм).
5. **Window Tool** або **Rectangle Tool** — створіть вікна й двері на стінах → виріжте їх за допомогою **Push/Pull**.







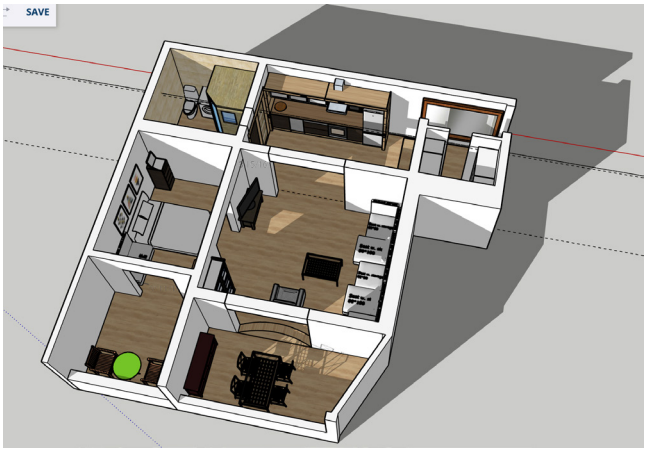
- Додайте інтер'єр у свою кімнату.
  Перейдіть до **3D Warehouse**:
  - Натисніть на іконку **3D Warehouse** (будиночок) на верхній панелі або зліва.
  - У рядку пошуку напишіть назву об'єкта англійською, наприклад: bed, chair, desk, bookshelf, lamp.
- Оберіть модель, що вам подобається → натисніть **Download** → **Load directly into your model**.
- Розмістіть об'єкт у кімнаті. Якщо він занадто великий або маленький:
  - Скористайтеся інструментом **Scale** (масштабування).
  - Поверніть або посуньте об'єкт за допомогою **Rotate** чи **Move Tool**.

  Розмістіть меблі логічно:
  - Не перекривайте двері та вікна.
  - Враховуйте, як людина буде пересуватись кімнатою.
- Завершення.
  Огляньте свою кімнату з різних ракурсів (іззовні та зверху).
  Збережіть проєкт в своєму обліковому записі в **SketchUp.**

Проєкт 1
Імпорт моделей із 3D Warehouse. Побудова інтер'єру кімнати

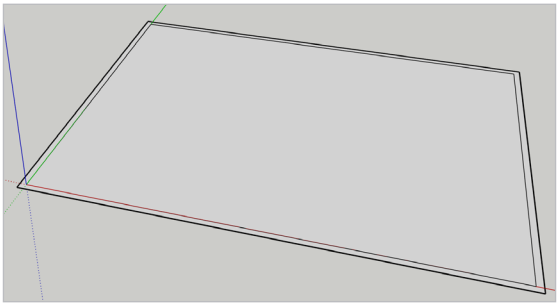
Індивідуальний проєкт

Завдання. Створити простий план будинку або квартири з перегородками, дверима, вікнами та базовим інтер'єром.



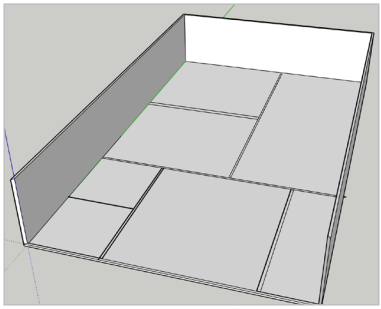
Крок 1. Створення основи будинку або квартири

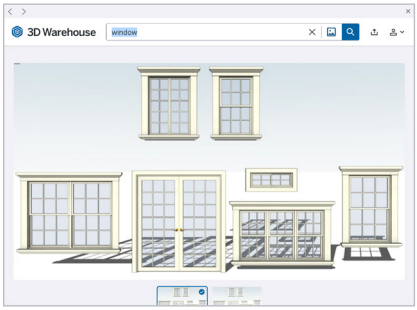
- Оберіть інструмент **Rectangle Tool** і накресліть основу приміщення — прямокутник (розмір можете обрати самі).
- За допомогою інструмента **Offset Tool** створюйте внутрішню лінію для стін (відступ 20 см).
- Підніміть стіни за допомогою **Push/Pull Tool** (висота: 280 см).



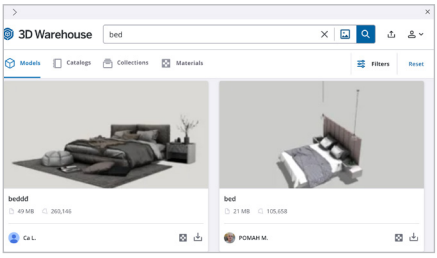
- Створення кімнат / зон
  - За допомогою **Line Tool** додайте лінії-перегородки, які ділять простір на кімнати.
  - Створіть дверні та віконні прорізи за допомогою **Rectangle Tool**, потім використайте **Push/Pull Tool**, щоб видалити їх.





- Додавання дверей і вікон
  - Відкрийте **3D Warehouse** → знайдіть **door** або **window**.
  - Завантажте обрані об'єкти одразу в модель.
  - Розмістіть їх у відповідних отворах. При потребі масштабуйте або поверніть.



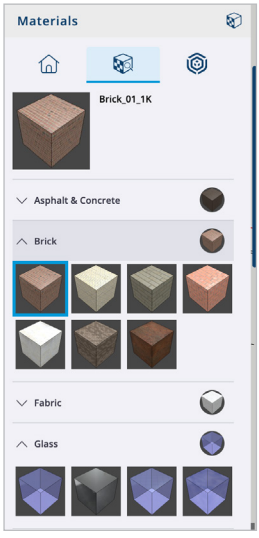
- Меблювання основних зон

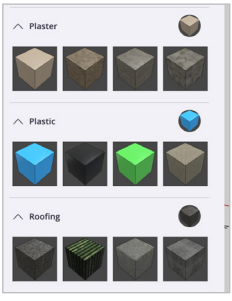
  Знайдіть та додайте:
  - Ліжко до спальні.
  - Стіл і стільці до кухні або вітальні.
  - Шафу, тумбу або поличку.

  Розмістіть об'єкти логічно.
- Додавання матеріалів

  Відкрийте панель **Materials** (праворуч).

  Оберіть колір чи текстуру для:
  - підлоги (наприклад, дерево або плитка);
  - стін (фарба, шпалери);
  - меблів (дерево, тканина тощо).
- Збережіть проєкт.





Індивідуальний проєкт

У вашій роботі мають бути використані інструменти:

- **Rectangle** — для побудови основи (підлоги або землі);
- **Push/Pull** — для створення стін або підняття об'єктів;
- **Offset** — для створення внутрішнього контуру стін;
- **3D Warehouse** — для вставлення готових моделей меблів або об'єктів довкілля (парти, дошки, лавки, дерева тощо);
- **Materials** — для надання матеріалів (дерево, бетон, скло тощо);
- **Move** / **Rotate** / **Scale** — для точного розміщення елементів;
- **Shadows** — для додавання тіней.

Завдання. Створити 3D-модель обраного простору (на вибір: кімната, клас або парк)

Ви повинні створити реалістичну 3D-модель простору, яку можна оглянути з різних ракурсів.

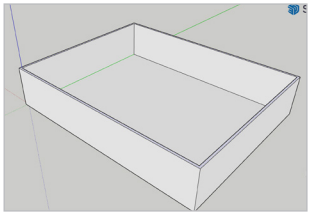
Обов'язково:

- є стіни, підлога, вікна або межі території;
- вставлено щонайменше 5 об'єктів з бібліотеки **3D Warehouse**;
- до всіх об'єктів застосовано матеріали;
- встановлене освітлення або тіні (опційно, якщо встигаєте).

↳ Побудова основи

Візьміть інструмент **Rectangle** (**Прямокутник**).

Створіть прямокутник — це буде підлога кімнати / площа парку (наприклад, 6м х 4м).

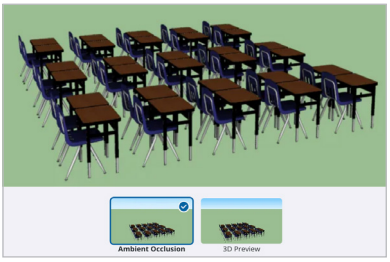


↳ Створіть стіни або межі

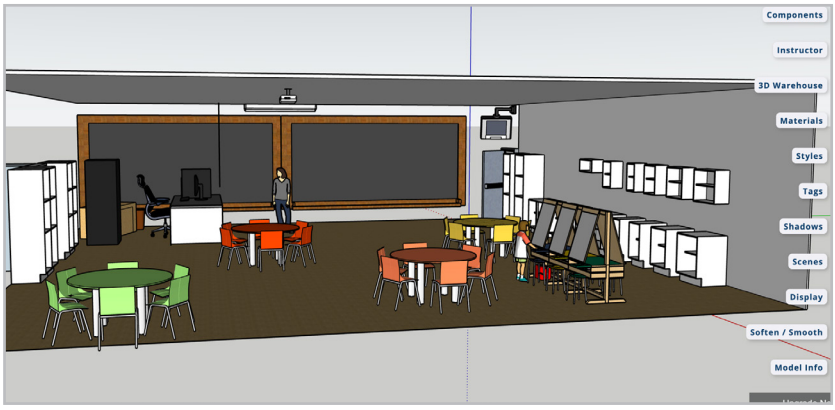
Виберіть інструмент **Offset** та натисніть на прямокутник, щоб створити контур стіни (завтовшки 0.2 м).

Потім виберіть інструмент **Push/Pull** та витягніть стіни вгору (на 2,5 м – для кімнати чи класу).

- Створіть вікна та двері
  Використайте **Rectangle**, щоб намалювати отвори на стіні.
  Потім **Push/Pull** — натисніть на отвір і протисніть до кінця стіни.
- Вставлення об'єктів з **3D Warehouse**
  Натисніть на кнопку **3D Warehouse** (праве меню).
  Перемістіть і відмасштабуйте об'єкти.



- Додайте матеріали
  Відкрийте **Materials** (праве меню).
  Оберіть матеріали: дерево, бетон, скло, трава.
- Додайте тіні (опційно)
  Відкрийте вкладку **Shadows**.
  Увімкніть **On face** / **On ground**.
  Встановіть час і дату для налаштування реалістичних тіней.
- Перевірте з різних ракурсів
  Використовуйте **Orbit**, **Pan** і **Zoom**, щоб перевірити модель.
  За потреби — скорегуйте розміщення об'єктів.
- Збережіть модель
  Назвіть файл за зразком: Прізвище_ім'я_9____.

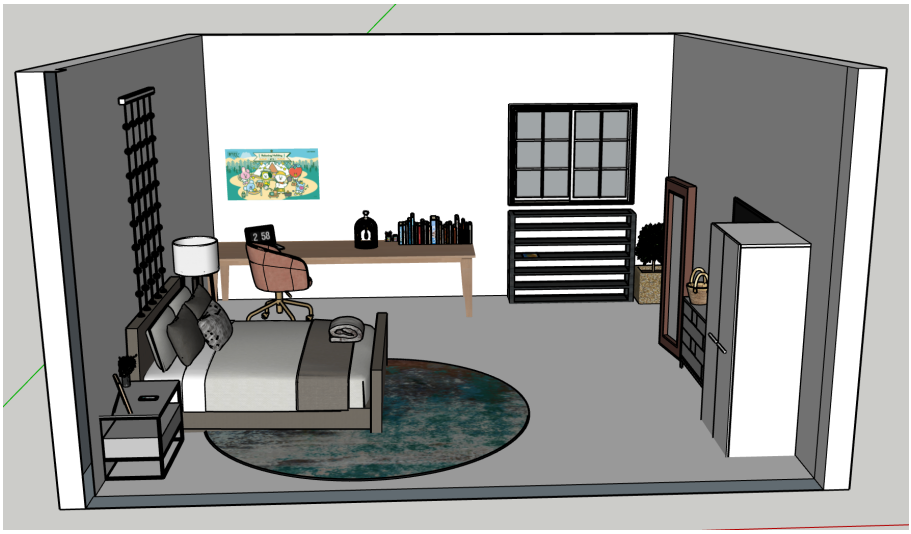


Індивідуальний проєкт

Завдання. Створити реалістичну 3D-модель класу/кімнати своєї мрії з урахуванням функціональності, дизайну та нестандартних елементів, які роблять простір унікальним.

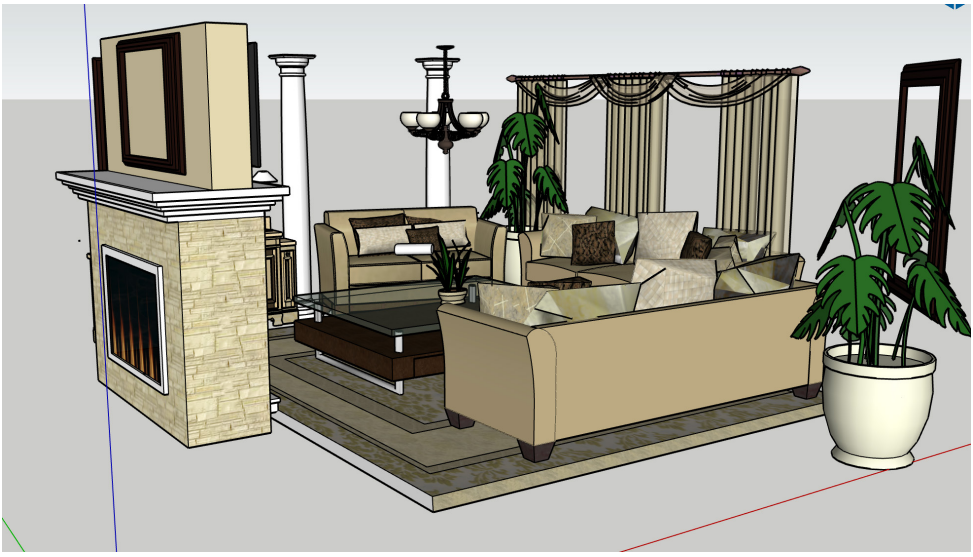
Структура проєкту

- Базова геометрія: стіни, підлога, стеля (мінімум 6м × 4м)
- Мінімум 3 матеріали різних типів (дерево, бетон, скло, метал, тканина тощо)
- Освітлення: мінімум 2 джерела (природне + штучне)
- Контрастна кольорова схема (не менше 4 кольорів)
- Реалістичні тіні та відображення



- Мінімум 8-10 об'єктів з **3D Warehouse**
- Меблі розташовані функціонально (не просто розкидані)
- Мінімум 2 предмети мають бути модифіковані (зміна розміру, кольору, матеріалу)

- Включити один нестандартний елемент (на кшталт підвісної полиці, динамічних стільців, голограми тощо)



- Перевірте з різних ракурсів
- Використовуйте **Orbit**, **Pan** і **Zoom**, щоб перевірити модель
- За потреби — скорегуйте розміщення об'єктів
- Збережіть модель
- Назвіть файл за зразком: Прізвище_ім'я_9____.

Практична робота: «Будинок моєї мрії. Інтер'єр та екстер'єр»

Індивідуальний проєкт

Це проєкт, у якому ви створюватимете 3D-модель свого ідеального будинку. На відміну від попередньої роботи (де ви проєктували один клас), тут працюватимете з ЦІЛИМ БУДИНКОМ — з його зовнішнім виглядом (фасадом, дахом, двором) та внутрішнім простором (кількома кімнатами, коридором, кухнею).

- **Зовні (екстер'єр):** фасад будинку, дах, вхідні двері, вікна, двір, доріжка, дерева та квіти
- **Усередину (інтер'єр):** мінімум 4 кімнати (спальня, вітальня, кухня, ванна) + коридор
- **Умеблювання:** кожна кімната має бути облаштована меблями й декором

- **Ландшафт:** клумби, дерева, чагарники, газон, доріжка, паркан
- Мінімум 5 матеріалів різних типів

Найголовніше! Ваш будинок не повинен бути копією чужого.
СТИЛЬ: Оберіть стиль — мінімалізм, класика, модерн, екостиль.

Тема 33 Повторення основ Python. Структура програми

Ключові поняття

Мова програмування Python
Програма та її структура
Модулі Python

Навіщо нам Python і чому повторюємо основи?

У 7–8 класах ви вже ознайомилися з мовою програмування **Python**:

- малювали черепашкою графіку;
- навчилися працювати зі змінними;
- створювали анімації;
- використовували умови й цикли;
- писали прості програми та робили цікаві проєкти.

Тепер у 9 класі ми підемо далі, але для цього спершу потрібно повторити основи та розібратися, як правильно будувати програму, щоб міцно закріпити знання і перейти до складніших тем, наприклад, списки, функції чи навіть створення ігор.

Python — це універсальний інструмент у вашому рюкзаку. Він простий, як говорити з другом, але потужний, як суперкомп'ютер.

Із **Python**:

- створюють вебсайти (наприклад, **Instagram** чи **YouTube**);
- аналізують дані (щоб прогнозувати погоду чи рекомендувати фільми на **Netflix**);
- керують роботами (як у фільмах про майбутнє);
- пишуть ігри (наприклад, частини **Minecraft**).

Уявіть: ви можете написати програму, яка порахує, скільки грошей заощадите на новий гаджет, або симулюватиме гру «Камінь-ножиці-папір» з комп'ютером!

Що таке програма?

Програма — це набір інструкцій, які комп'ютер виконує крок за кроком.

Це якби ви писали рецепт для приготування бутерброда: спочатку візьми хліб, потім намасти його маслом, зверху поклади сир... А програма — це ніби рецепт для комп'ютера.

Коли ви пишете код на **Python**, ви створюєте алгоритм для машини:

- вводимо дані;
- опрацьовуємо їх;
- виводимо результат.

Структура програми на Python

Кожна програма складається з кількох важливих частин.

Імпорт модулів (якщо потрібно). Це як додавання інгредієнтів. Наприклад, для графіки ми імпортували **turtle** та **tkinter**, для математичних обчислень — **math** та **random**.

Коментарі. Це нотатки для себе чи інших, які комп'ютер не виконує. Вони роблять програму зрозумілою, наче пояснення в шкільному зошиті.

- *Однорядковий коментар:* Починається з `#`. Наприклад: `# Це змінна для віку.`
- *Багаторядковий.* У лапках `""" Текст """`.

Приклад з життя. Уявіть, що готуєтеся до контрольної і пишете шпаргалку. Коментарі — це шпаргалка в коді, щоб через місяць чи більше вам було зрозуміло, чому написали саме так.

Приклад коду:

```
# Програма рахує вік собаки в людських роках
dog_age = 3  # Вік собаки в роках
human_age = dog_age * 7  # Кожний собачий рік = 7 людських
print("Собаці в людських роках:", human_age)  # Вивід
"""
Ця програма проста, але можна додати більше умов, наприклад,
для цуценят.
"""
```

Визначення змінних і даних.

Тут ми зберігаємо інформацію. Тобто змінні — це як коробки, в які розкладаєте речі. У 7-му та 8-му ми часто працювали зі змінними для графіки чи розрахунків. Змінна має ім'я (наприклад, `score`) і значення (наприклад, 10).

Як створювати: `змінна = значення`. Ім'я повинно бути без пробілів, починатися з малої літери (наприклад, `myAge`, не `my age`).

Типи даних

- **Цілі числа (int)**: для рахунку, як кількість яблук.
  Приклад: `apples = 5`.
- **Дробові числа (float)**: для вимірів, як зріст.
  Приклад: `height = 1.65`.
- **Рядки (str)**: текст, у лапках.
  Приклад: `name = "Оля"`.
- **Логічний (bool)**: Правда чи брехня (**True**/**False**).
  Приклад: `is_raining = True`.

Введення та виведення даних

Щоб програма була інтерактивною, використовуємо **input()** для введення і **print()** для виведення.

- **input("Запитання")**: чекає відповідь від користувача (завжди рядок, тому перетворюємо, наприклад, `int(input())`).
- **print(щось)**: показує на екрані.

Приклад з життя. Уявіть додаток для замовлення їжі. Користувач вводить ім'я, програма вітається з ним. А потім уже виконується функціонал програми.

```
name = input("Як тебе звати?") # Введення рядка
age = int(input("Скільки тобі років?"))  # Введення числа
print("Привіт,", name,"Тобі", age, "років.")
```


Операції та розрахунки


Умовні оператори (**if**, **else**, **elif**)


Цикли (**for**, **while**)

Операції та розрахунки.

Додавання, порівняння тощо.

Оператори — це інструменти для розрахунків, як у 8-му класі з арифметикою та логікою.

- **Арифметичні: +**, **-**, ***, */*, **//** (ціле ділення), **%** (залишок), **** (степінь).
- **Порівняння: ==** (дорівнює), **!=** (не дорівнює), **>**, **<**, **>=**, **<=**.
- **Логічні: and** (і), **or** (або), **not** (не).

Приклад з життя. Ви плануєте похід. Арифметика: скільки їжі потрібно взяти на всіх? Порівняння: чи вистачить грошей? Логіка: якщо дощ **and** холодно, то сидимо вдома.

```
# Програма для знаходження середнього балу
a = int(input("Оцінка з математики:"))
b = int(input("Оцінка з інформатики:"))
c = int(input("Оцінка з англійської:"))

avg = (a + b + c) / 3

print("Середній бал:", avg)
```

Умовні оператори (if, else, elif)

Оператор **if** використовують для розгалужень. Якщо умова **True**, виконуємо код.

- **if** умова: код
- **elif** інша умова: код (множинне розгалуження)
- **else**: код (якщо ніщо не підійшло)

Використовують вкладені умови: **if** всередині **if**.

Приклад коду:

```
#Гра "Вгадай число"
secret = 7
number = int(input("Вгадай число від 1 до 10:"))
if number == secret:
    print("Вгадав!")
elif number > secret:
    print("Занадто велике!")
else:
    print("Занадто маленьке!")
```

Цикли (for, while)

Використовуються, щоб повторювати дії багато разів.

Оператори для повторення: **while** (поки умова **True**) і **for** (для певної кількості).

- **while** умова: код — Як гра: поки не втомився, бігай.
- **for i in range** (кількість): код — Для повторення N разів.

Приклад коду:

```
# Малювання зірочок
# While: Поки i < 5
i = 0
while i < 5:
    print("*" * (i + 1))
    i += 1

# For: 5 разів
for j in range(5):
    print("Зірка номер", j + 1)
```

Модулі Python: ваші помічники в програмуванні

Модулі — це як набори інструментів, які додають нові можливості до **Python.**

У 7-му та 8-му класах ми вже використовували модулі **turtle** і **tkinter** для графіки, **math** для математичних функцій та **random** для випадкових чисел.

Щоб використовувати модуль, його потрібно імпортувати командою **import.** Уявіть, що це ніби відкрити коробку з новими фломастерами для малювання.

Модуль **math**

Математика стає простішою

Модуль **math** — це ваш калькулятор для складних обчислень. Він потрібен, коли хочете знайти квадратний корінь, округлити число чи використати константи, як число π (пі). У 8-му класі ми використовували його для математичних функцій, наприклад, при розв'язанні квадратних рівнянь.

Основні функції модуля math:

- **`math.sqrt(x)`** — квадратний корінь числа **`x`**.
  Наприклад, `math.sqrt(16)` повертає 4.
- **`math.pow(x, y)`** — підносить **`x`** до степеня **`y`**.
  Наприклад, `math.pow(2, 3) = 8`.
- **`math.pi`** — константа π (≈3.14159), корисна для обчислень із колами.
- **`math.floor(x)`** — округлює вниз (до меншого цілого).
  Наприклад, `math.floor(3.7) = 3`.
- **`math.ceil(x)`** — округлює вгору (до більшого цілого).
  Наприклад, `math.ceil(3.7) = 4`.

Приклад з життя. Ви плануєте вечірку і хочете спекти круглий торт. Щоб знати, скільки крему потрібно, обчислимо площу торта ($S = \pi r^2$).

```
from math import *

radius = float(input("Введи радіус торта (см):"))  # Наприклад,
15 см
area = pi * pow(radius, 2)  # Площа = π * r²
print("Площа торта:", round(area, 2), "см²")  # Округлення до
2 знаків
```

Модуль **random**

Додаємо випадковість

Модуль **random** робить програми непередбачуваними, як гра в кубик чи лотерею.

Основні функції модуля random:

- **`random.randint(a, b)`** — випадкове ціле число від **`a`** до **`b`** (включно).

- `random.choice(список)` — вибирає випадковий елемент зі списку.
- `random.random()` — випадкове число від `0` до `1` (дробове).
- `random.uniform(a, b)` — випадкове число з плаваючою точкою від `a` до `b`.

Малюємо зображення

Модуль **turtle** — це ваш віртуальний художник, який малює на екрані за командами. У 7-му класі ми створювали з черепашкою графіку, як-от зірки чи спіралі. Це чудовий спосіб візуалізувати програми!

Основні команди turtle:

- `turtle.forward(x)` — рух уперед на `x` пікселів.
- `turtle.left(кут)` — поворот ліворуч на кут градусів.
- `turtle.right(кут)` — поворот праворуч.
- `turtle.penup()` і `turtle.pendown()` — підняти/опустити перо (не малювати/малювати).
- `turtle.color("колір")` — змінити колір.

Створюємо віконні програми

Модуль **tkinter** дозволяє створювати програми з графічним інтерфейсом — кнопками, полями для введення, вікнами. Це ніби створювати додаток для телефона чи комп'ютера!

Основні елементи tkinter:

- `Tk()` — створює головне вікно.
- `Label(window, text="текст")` — текстова мітка.
- `Entry(window)` — поле для введення тексту.
- `Button(window, text="текст", command=функція)` — кнопка, яка викликає дію.
- `window.mainloop()` — запускає програму.

Приклади програм

Калькулятор вартості покупки

```
price = float(input("Введіть ціну товару:"))
k = int(input("Введіть кількість:"))

total = price * k
print("До оплати:", total, "грн")
```

Перевірка, чи можна вийти гуляти

```
weather = input("Яка погода? (сонце/дощ):")

if weather == "сонце":
    print("Йдемо гуляти!")
else:
    print("Беремо парасолю або сидимо вдома.")
```

Практична робота №33

Завдання 1. Напишіть програму, яка виводить ваше ім'я, прізвище та клас.

Завдання 2. Створіть програму, яка запитує у користувача його улюблений колір і виводить повідомлення: Твій улюблений колір — ...

Завдання 3. Напишіть програму, яка запитує температуру. Якщо температура > 0 → `"На вулиці тепло"`. Інакше → `"На вулиці холодно"`

Завдання 4. Використовуючи цикл, виведіть 10 разів повідомлення `"Я вчу Python!"`

Завдання 5. Напишіть програму для обчислення вартості поїздки, знаючи такі дані, як ціна квитка та кількість пасажирів.

Завдання 6. Створіть програму `"Шкільний дзвінок"`:

- користувач вводить номер уроку (1–7);
- програма виводить час, коли лунає дзвінок

(наприклад, 1-й урок — 8:30, 2-й — 9:20 і т. д.).

Завдання 7. Створіть програму `"Електронний щоденник"`:

- користувач вводить три оцінки;
- програма рахує середнє;
- якщо середнє ≥ 8 — вивести `"Молодець!"`, якщо менше — `"Треба підтягнутися"`.



Функції в мові Python. Створення власних команд для зручності програмування

Тема 34

Ключові поняття

Функції в Python
Оператор def
Повернення значень. Оператор return
Локальні та глобальні змінні
Вкладеність та рекурсія

Для чого потрібні функції?

У 8-му класі ви писали проєкти, наприклад, для розрахунку вартості піци. Там код іноді ставав довгим і заплутаним. А уявіть, що ви пишете велику гру або додаток — код може розростися на сотні рядків! Ось тут нам допоможуть функції.

Функції — це наче власні команди, які ви створюєте особисто. Функції дозволяють групувати код, повторювати його без копіювання і робити програми чистішими та зручнішими.

Уявіть, що готуєте обід. Щоб усе встигнути, ви розбиваєте процес на кроки. Наприклад, деякі з них:

- зробити підливу;
- відварити гарнір;
- нарізати овочі.

У програмуванні такі «кроки» називаються функціями — це «рецепти» всередині програми.

Проте деякі кроки виконуються багаторазово і для інших страв. Тому, аби щоразу заново не описувати, як нарізати овочі, ви просто кажете «нарізати овочі» — і це вже готова інструкція.

Що таке функція і який вигляд має

Із функціями ваша програма стає як конструктор: складаєте її з готових блоків.

Функції використовують у реальному світі:

- в **Instagram** — фільтри для фото;
- у грі **Minecraft** функції керують рухом персонажів;
- кнопка «ввімкнути світло» — це функція, яка керує світлом;
- у калькуляторі кнопка √ — функція для обчислення квадратного кореня.

У **Python** теж є вбудовані функції:

- `print()` – виводить інформацію на екран
- `input()` – отримує дані від користувача
- `len()` – обчислює довжину рядка чи списку
- `int()` – перетворює на ціле число

Функція — це блок коду, який виконує певну дію. Його можна викликати скільки завгодно разів, в будь-якому місці програми, не переписуючи знову всього коду.

Щоб створити свою функцію, у **Python** використовують ключове слово **def**.

Синтаксис:

```
def імя_функції(параметри):
    #Тіло функції
    #Дії, які виконує функція
    return результат
```

- **def** – означає «визначаємо функцію»;
- **ім'я_функції** – придумуємо самі. Повинно бути англійською мовою, з маленької літери, та має описувати, що саме вона робить (наприклад, `calculate_area` — обчислити площу). Не використовуйте пробілів, замість них робіть підкреслення _;
- **параметри** – це «вхідні дані», які ми передаємо для функції, як інгредієнти в рецепті. Їх пишуть у дужках. Якщо параметрів немає, дужки пусті ();
- **тіло функції** — код, який виконується. Він відступає на 4 пробіли (індентація — пам'ятаєте з циклів чи умов?);
- **return** – те, що функція повертає назад.

Приклад простої функції без параметрів:

```
def say_hello():  # Визначення функції
    print("Привіт, друзі! Готові програмувати?")  # Тіло
```

Щоб викликати функцію, достатньо звернутися до неї за ім'ям:

```
say_hello() #Виведе: "Привіт, друзі! Готові програмувати?"
```

Функції з параметрами

Параметри роблять функції гнучкими та універсальними — ви можете передавати різні значення, як у програмах, де ми вводили дані від користувача.

Параметри пишуть у дужках при визначенні, а при виклику — передаєте значення.

- Формальні параметри: У визначенні, наприклад, (name).
- Актуальні параметри: При виклику, наприклад, («Аня»).

Приклад коду: Функція для привітання з параметром

```
def greet(name):  # Параметр name
    print("Привіт,", name, "! Як твої справи?")
```

Виклики з різними значеннями

```
greet("Макс")  # "Привіт, Макс ! Як твої справи?"
greet("Катя")  # "Привіт, Катя ! Як твої справи?"
```

Функції з параметрами

Можна мати кілька параметрів, розділених комами.

Розширений приклад з кількома параметрами: Розрахунок площі прямокутника (як у математиці, але в коді).

```
def rectangle_area(length, width):  # Два параметри
    area = length * width
    print("Площа прямокутника:", area)
```

Виклики з різними значеннями

```
rectangle_area(5, 3)  # 15
rectangle_area(10, 2)  # 20
```

Повернення значень. Оператор return

Іноді функція не просто друкує щось, а повертає значення для подальшого використання. Для цього застосовуємо **return**. Після **return** функція закінчується.

- Без **return** функція повертає **None** (нічого).
- Із **return** ви можете присвоїти результат змінній і надалі використовувати її значення.

Приклад коду: Функція для додавання чисел

```
def add_numbers(a, b):  # Параметри a і b
    result = a + b
    return result  # Повертаємо значення
```

Присвоюємо результат функції змінній і виводимо її в консоль

```
sum1 = add_numbers(4, 5)  # sum1 = 9
print("Сума:", sum1)

sum2 = add_numbers(10, 20) + 5  # Можна використовувати в ви-
разах: 35
print("Сума з додатком:", sum2)
```

Повернення значень. Оператор return

Приклад з модулем **math**: Функція для обчислення гіпотенузи трикутника (за теоремою Піфагора).

```
from math import sqrt
def hypotenuse(a, b):
    return sqrt(a**2 + b**2)  # Повертаємо квадратний корінь

hyp1 = hypotenuse(3, 4)  # 5.0
print("Гіпотенуза:", hyp1)

hyp2 = hypotenuse(6, 8)  # 10.0
print("Гіпотенуза:", hyp2)
```

Локальні та глобальні змінні

Змінні у функціях — як речі в будинку: деякі видно тільки всередині приміщення *(локальні)*, напр., якась особиста річ у вашій кімнаті (тільки ви її бачите), а деякі — скрізь *(глобальні)*, напр., телевізор у вітальні (всі члени сім'ї можуть дивитися).

- **Локальні змінні:** створюються всередині функції, існують тільки там. Не впливають на код поза функцією.
- **Глобальні змінні:** створюються поза функціями, видно скрізь. Щоб змінити глобальну всередині функції, використовуйте **global**.

```
total = 0  # Глобальна змінна

def add_to_total(value):
    global total  # Використовуємо глобальну
    total = total + value  # Змінюємо її
    local_var = 10  # Локальна, видно тільки тут
    print("Локальна:", local_var)

add_to_total(5)
print("Глобальна total:", total)  # 5
print(local_var)  # Видасть помилку! Не видно поза функцією
```

Вкладеність та рекурсія

Функції можна викликати всередині інших функцій.

```
def add(a, b):
    return a + b

def square_sum(a, b):
    return add(a, b) ** 2

print(square_sum(2, 3))  # (2+3)^2 = 25
```

Рекурсія — це коли функція викликає себе як у дзеркалі, де відображення повторюється.

Корисно для задач як факторіал `(n! = n * (n-1)!)`. Але обережно: без умови зупинки буде нескінченний цикл!

Приклад з життя. Уявіть, що шукаєте ключі: перевіряєте кишеню, якщо немає — шукаєте в сумці, якщо немає — в кімнаті... Кожен крок викликає наступний.

```
def factorial(n):
    if n == 1:  # Умова зупинки
        return 1
    else:
        return n * factorial(n - 1)  # Виклик себе

print(factorial(5))  # 120 (5*4*3*2*1)
```

Практична робота №34

Завдання 1.
Створіть функцію **hello(name)**, яка вітає користувача за ім'ям. Викличте її кілька разів із різними іменами.

Завдання 2.
Напишіть функцію **square(x)**, яка повертає квадрат числа. Перевірте її роботу на числах 2, 5, 10.

Завдання 3.
Створіть функцію **area_triangle(a, h)**, яка обчислює площу трикутника за формулою:

$$S = \frac{1}{2} \cdot a \cdot h$$

Завдання 4. Напишіть функцію **even(number)**, яка перевіряє, чи параметр **number** є парним.

Завдання 5. Використання функцій у графіці **Turtle**. Створіть малюнок із кількох квіток, використовуючи функцію **flower()**.

- Імпортуйте бібліотеку **turtle** та створіть черепашку **t**

```
from turtle import *
створи функцію petal(), яка малює одну пелюстку
def petal():
    t.circle(50, 60)
    t.left(120)
    t.circle(50, 60)
    t.left(120)
```

- Створіть функцію **flower(n)**, яка малює квітку з **n** пелюсток

```
def flower(n):
    for i in range(n):
        petal()
        t.right(360 / n)
```

- Викличте функцію **flower(n)**

```
t.color("red")
flower(6)
зміни значення n
```

Додатково: параметр **color** внести у функцію flower(n) => flower(n, color)

```
flower(6, "red")
```



Списки. Оголошення, доступ, додавання, видалення елементів

Ключові поняття

Списки
Довжина списку len
Елементи списків
Зрізи списків
Методи списків

Що таке списки

Списки — це така чарівна коробка, в яку можете покласти багато речей: числа, слова, навіть інші списки.

Уявіть, що, збираючись у магазин, ви робите список необхідних покупок: «молоко», «хліб», «яблука»; можете додати «цукерки» чи видалити «хліб», якщо передумали, або перевірити, що на першому місці.

Чому списки корисні?

Вони гнучкі: можна змінювати розмір, швидко додавати/видаляти елементи, а ще — працюють з циклами та умовами, які ми вже знаємо.

У реальному світі списки використовують скрізь:

- у соцмережах (список друзів у **Instagram**);
- у магазинах (список товарів у кошику на **Rozetka**);
- в іграх (список рівнів у **Fortnite**).

Списки — це впорядкована колекція елементів, які можуть мати будь-яку кількість елементів і бути різними типами даних (числа, рядки, логічні значення чи навіть інші списки).

На відміну від звичайної змінної, яка тримає тільки одне значення (як склянка з одним напоєм), список — це як полиця з багатьма склянками: кожна має свій номер (індекс), і ви можете брати будь-яку.

Особливості списків

Впорядкованість

Змінність

Дублікати

Різноманітність

Впорядкованість. Елементи йдуть у послідовності, як у черзі в шкільній їдальні. Перший елемент — на позиції 0 (так, у **Python** нумерація починається з 0!).

Змінність. Списки можна змінювати — додавати, видаляти, змінювати елементи.

Дублікати. Можна мати однакові елементи, наприклад, два «яблука» в списку покупок.

Різноманітність. Списки можуть містити елементи різних типів.

Як створити список

Список створюють за допомогою квадратних дужок `[]`, а елементи розділяють комами.

Він буде змінною, що зберігає кілька значень одразу.

Порожній список:	`empty = [ ]` — як порожня коробка, куди потім складаєте речі.
З елементами:	`numbers = [1, 2, 3]` `fruits = ["яблуко", "банан", "груша", "персик"]`
З повтореннями:	`colors = ["червоний", "синій", "червоний"]`
Змішаний:	`mixed = [10, "кіт", True, 3.14]`
Вкладені списки:	`matrix = [[1, 2], [3, 4]]` — список списків, як таблиця

Приклади списків

Ситуація з життя	Список у Python
Уроки на тиждень	`lessons = ["математика", "інформатика", "історія", "англійська"]`
Температура за тиждень	`temps = [15, 17, 20, 22, 18, 16, 14]`
Продукти в холодильнику	`food = ["молоко", "хліб", "яйця", "сир"]`
Улюблені ігри	`games = ["Minecraft", "Brawl Stars", "Roblox"]`

Довжина списку

У **Python** список може містити багато елементів: числа, слова або навіть інші списки.

Іноді потрібно дізнатися, скільки елементів у списку — наприклад, скільки учнів у класі, скільки оцінок у журналі чи скільки покупок у кошику.

Для цього використовують вбудовану функцію `len()`.

```
fruits = ["яблуко", "апельсин", "груша"]
print(len(fruits))  # 3
```

Приклад 1. Скільки учнів у класі?

```
учні = ["Аня", "Богдан", "Катя", "Олег", "Марія"]
кількість = len(учні)
print("У класі", кількість, "учнів.")
# У класі 5 учнів.
```

Приклад 2.
Порожній список

```
покупки = []
print("У кошику товарів:", len(покупки))
# У кошику товарів: 0
```

Доступ до елементів списку

Щоб отримати окремий елемент списку, використовують індекс — номер елемента в списку.

Індексація починається з нуля! (перший елемент — `[0]`, другий — `[1]` тощо). Щоб доступитися: **`список[індекс]`**.

Позитивні індекси:

```
fruits = ["яблуко", "апельсин", "груша", "персик"]
print(fruits[0])   # яблуко
print(fruits[2])   # груша
```

Якщо спробувати звернутися до індекса, якого немає (наприклад, `fruits[10]`), отримаєте помилку **IndexError**.

Негативні індекси:

Python дозволяє рахувати з кінця:

- -1 — останній елемент
- -2 — передостанній і т.ін.

```
print(fruits[-1])   # персик
print(fruits[-2])   # груша
```

Зміна елемента:

Можна змінити значення будь-якого елемента, звернувшись до нього за індексом:

```
(fruits[1]) = "апельсин"
print(fruits) # ["яблуко", "апельсин", "груша", "персик"]
```

Пояснення: Ми просто записали нове значення замість старого — тепер на місці «банан» з'явився «апельсин».

Зрізи елементів списку

Зрізи **`(slices)`** — це спосіб взяти частину списку: `список[початок:кінець]`.

```
print(fruits[1:3])
# ['апельсин', 'груша']
```

З кроком: `numbers[0:10:2]` — кожний другий елемент.

Негативний крок:

```
print(fruits[::-1])
# ['персик', 'апельсин', 'яблуко']
```
— перевернути список

Приклад зрізу:

```
numbers = [0, 1, 2, 3, 4, 5]
even = numbers[0:6:2]  # [0, 2, 4]
reverse = numbers[::-1]  # [5, 4, 3, 2, 1, 0]
print("Парні:", even)
print("Навпаки:", reverse)
```

Додавання елементів у список

Є кілька способів додати нові елементи.

Метод append(): `append(елемент)` — додає в кінець. Швидко і просто.

```
fruits.append("ананас")
print(fruits)
# ["яблуко", "апельсин", "груша", "персик", "ананас"]
```

Метод insert(): `insert(індекс, елемент)`: вставляє на певну позицію, зсуваючи інші.

```
fruits.insert(1, "лимон")
print(fruits)
# ["яблуко", "лимон", "апельсин", "груша", "персик", "ананас"]
```

Оператор + — об'єднання списків: `список1 + список2` — створює новий список, поєднуючи два.

```
more_fruits = ["вишня", "диня"]
all_fruits = fruits + more_fruits
print(all_fruits)
```

Метод extend(): `extend(інший_список)`: додає елементи з іншого списку в кінець.

```
more_fruits = ["вишня", "диня"]
all_fruits = fruits + more_fruits
print(all_fruits)
```

Можна додавати з input:

```
fruits.append(input("Додай фрукт: "))
print(fruits)
```

Видалення елементів зі списку

Є кілька зручних способів:

- Метод **remove()** — видаляє за значенням: `fruits.remove("груша")`
  Якщо такого елемента немає — буде помилка.
- Метод **remove()** — видаляє за значенням: `fruits.pop(2)`
  Якщо індекс не вказати, **pop()** видаляє останній елемент.
- Оператор **del** — видаляє елемент або навіть увесь список:

```
del fruits[0]    # видаляє перший елемент
# або
del fruits       # видаляє повністю список
```

- Метод **clear()** — очищає список (залишає порожній)

```
fruits.clear()
print(fruits)  # []
```

Розширена інформація: **remove** — для значення, pop/del — для індексу. Якщо кілька однакових — **remove** видаляє тільки перше. Для видалення всіх — цикл (про це в наступних темах). Зрізи для видалення: **del список[1:3]** — видаляє шматок.

Комбінування з функціями та модулями

- Приклад коду з функцією:
  Наприклад, функція, яка приймає список і додає елемент.

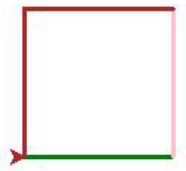
```
def add_to_list(my_list, item):
    my_list.append(item)
    return my_list  # Повертаємо оновлений список

friends = ["Аня", "Богдан"]
new_friends = add_to_list(friends, "Віка")
print("Друзі:", new_friends)  # ['Аня', 'Богдан', 'Віка']
```

- Приклад коду з модулями:

random.choice(список) — випадковий елемент, як у грі. В даному прикладі обираємо випадково колір зі списку.

```
import random

colors = ["червоний", "синій", "зелений"]
random_color = random.choice(colors)
print("Випадковий колір:", random_color)
```

Приклад коду з **turtle**:



```
from turtle import *
from random import *

t = Turtle()
t.width(3)

colors = ['red', 'green', 'blue', 'brown', 'orange', 'pink']

points = [[0, 0], [100, 0], [100, 100], [0, 100]]  # Вкладений
список
t.penup()
t.goto(points[0])  # [0, 0]
t.pendown()
for point in points[1:]:  # З 1-го елемента (про ітерацію —
пізніше)
    t.color(choice(colors)) # випадковий колір лінії зі списку
    t.goto(point)
t.goto(points[0])  # Замкнути квадрат
```

Практична робота №35

Завдання 1. Список покупок

1. Створіть список покупок: `["хліб", "молоко", "сир", "яблука"]`.
2. Виведіть перший і останній елементи.
3. Додайте до списку `"шоколад"`.
4. Замініть `"сир"` на `"йогурт"`.
5. Видаліть `"хліб"` зі списку.
6. Виведіть кінцевий список.

Завдання 2. Улюблені фільми

1. Створіть список із 5 назв улюблених фільмів.
2. Додайте ще один фільм у кінець списку.
3. Виведіть третій фільм у списку.
4. Видаліть будь-який фільм, який вже не подобається.
5. Порахуйте, скільки фільмів залишилось.

Завдання 3 (з random). Гра «Лотерея»

1. Створіть список чисел від 1 до 10 (або 10 імен).
2. Використайте **random.choice** для вибору переможця.
3. Видаліть його `pop`.
4. Додайте нового **append**.
5. Повторіть 3 рази.

Завдання 4 (з turtle).

1. Створіть список кольорів `["red", "green", "blue"]`.
2. Напишіть код чи функцію, яка малює коло з кольором за індексом.

Завдання 5.

1. Створіть список чисел `[1, 2, 3, 4, 5]`.
2. Використайте зрізи: виведіть парні позиції, переверніть список.
3. Додайте **extend** з іншим списком [6, 7].
4. Видаліть `pop(-1)`.
5. Очистіть **clear**.

Тема 36 Ітерація списком: цикл for, підрахунок, сумування

Ключові поняття

Ітерація списком
Вкладені списки
Підрахунок елементів списку
Сумування елементів списку

Чому ітерація робить списки «живими»

На попередньому уроці ми дізналися, що **список** — це спосіб зберігати багато значень в одній змінній.

Але часто потрібно **обробити всі елементи списку** — наприклад, порахувати суму, знайти середнє значення, перевірити кожен елемент.

Ось тут починається магія ітерації — це наче прогулянка по списку, де ви оглядаєте кожен елемент по черзі.

Ітерація — це процес перегляду елементів списку один за одним.

У **Python** для цього ідеально підходить цикл **for**, який ми вже вивчали в 7-му та 8-му класах для повторення дій певну кількість разів (наприклад, **for** та **in range(5)**). Тепер ми застосуємо його до списків!

З ітерацією ви можете підрахувати, скільки разів отримали 12 балів (підрахунок), або порахувати загальну суму кишенькових грошей за тиждень (сумування). Це робить програми розумнішими.

У реальному світі ітерацію використовують у додатках:

- у **Spotify** — для перегляду плейлиста пісень;
- у **Instagram** — для показу стрічки фото.

Що таке ітерація

Ітерація — це повторення дій для кожного елемента в колекції.

Інакше кажучи, комп'ютер бере перший елемент зі списку, виконує команду, потім переходить до другого, третього — і так далі, доки не обробить усі.

Чому це корисно? Бо без ітерації треба було б писати код для кожного елемента окремо: `print(список[0]), print(список[1])...`

Для великого списку — жах! З ітерацією один цикл робить усе. Це економить час і робить код чистішим, як функції.

Основні ідеї ітерації

- **Цикл for для списків:** **for** елемент **in** список — «для кожного елемента в списку роби щось».
- **Підрахунок:** рахуємо, скільки елементів задовольняють умову (наприклад, скільки оцінок > 10).
- **Сумування:** додаємо значення елементів (наприклад, сума чисел у списку).
- **Комбінування:** використовуємо **if** всередині **for** для умов, як у розгалуженнях у 7-8 класах.

Приклад простої ітерації:

```
friends = ["Аня", "Богдан", "Віка"]
for friend in friends:  # Для кожного друга в списку
    print("Привіт,", friend)  # Вивід для кожного
```

Цикл for для ітерації

Цикл **for** — це ваш «провідник» по списку.

Цикл **for** у **Python** дозволяє переглядати елементи списку один за одним і виконувати над ними потрібні дії. Уявіть, що список — це поїзд, а цикл **for** — це контролер, який заходить у кожен вагон і перевіряє, чи у всіх є квитки.

Синтаксис:

```
for елемент in список:
    команда(и)
```

Це означає: «Для кожного елемента у списку зроби такі дії».

Тут змінна (наприклад, **i**, **item**) по черзі стає кожним елементом списку.

Найпоширеніші способи використовувати for

Без індексів

Коли потрібні лише елементи списку (а не їхні позиції), найкращий спосіб — просто ітерувати значення.

```
fruits = ["яблуко", "банан", "вишня"]

for i in fruits:
    print("Я люблю", i)
```

```
Я люблю яблуко
Я люблю банан
Я люблю вишня
```

Змінна `i` послідовно приймає значення `"яблуко"`, потім `"банан"`, потім `"вишня"`.

Коли застосовувати: якщо потрібно просто виконати дії з кожним елементом списку (наприклад, вивести, підрахувати, перевірити тощо).

З індексами (enumerate)

Іноді потрібно знати не лише значення, а й позицію (індекс) елемента в списку.

У такому разі використовують вбудовану функцію **`enumerate()`**.

```
фрукти = ["яблуко", "банан", "вишня"]

for i, item in enumerate(фрукти):
    print(i, item)
```

```
0 яблуко
1 банан
2 вишня
```

Пояснення:

`enumerate()` автоматично додає до кожного елемента його номер (індекс).

`i` — індекс (починається з 0).

`item` — саме значення.

Якщо хочете, щоб нумерація починалася з 1, можна написати:

```
for i, item in enumerate(фрукти, 1):
    print(i, item)
```

З range() і len()

Іноді треба змінити елементи списку або звертатися до них за індексом.

У таких випадках використовують цикл із **`range()`** та **`len()`**:

```
fruits = ["яблуко", "банан", "вишня"]

for i in range(len(fruits)):
    print(i+1, 'фрукт:', fruits[i])
```

```
1 Фрукт: яблуко
2 Фрукт: банан
3 Фрукт: вишня
```

Пояснення:

`range(len(fruits))` створює послідовність чисел від 0 до 2 (довжина списку – 3).

На кожному кроці можете отримати доступ до елемента: `fruits[i]`.

Коли застосовувати: якщо потрібно змінювати елементи, видаляти їх або використовувати сусідні значення (наприклад, `список[i+1]`).

Вкладені списки

Списки можуть містити інші списки — це **вкладені списки** (наприклад, таблиці або матриці).

Щоб пройтися по всіх елементах, використовують **вкладені цикли**.

```
matrix = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]

for row in matrix:          # row — це підсписок
    for item in row:        # item — це окреме число
        print(item, end=" ")
    print()
```

```
1 2 3
4 5 6
7 8 9
```

Пояснення:

- Перший цикл перебирає кожен рядок (тобто підсписок).
- Другий цикл — кожен елемент у цьому рядку.

Коли застосовувати: якщо потрібно працювати з таблицями, сітками або будь-якими даними, організованими у вигляді рядків і стовпців (наприклад, розклад уроків, шахівниця, пікселі зображення).

Підрахунок елементів

Підрахунок — це ітерація з лічильником.

Підрахунок всіх елементів: для знаходження кількості усіх елементів використовують `len(список)`

Підрахунок з умовами

Іноді потрібно дізнатися, скільки разів певний елемент трапляється у списку. Це можна зробити вручну через цикл:

```
animals = ["кіт", "собака", "кіт", "папуга", "кіт"]
count = 0

for a in animals:
    if a == "кіт":
        count += 1

print("Кількість котів:", count)
```

```
Кількість котів: 3
```

Метод `count(елемент)`

```
grades = [10, 12, 9, 12, 11]
twelves = grades.count(12)
print("Кількість 12:", twelves)
```

```
Кількість: 12: 2
```

Сумування елементів

- Додавання за допомогою циклу

```
pocket_money = [50, 30, 40, 60]
total = 0
for money in pocket_money:
    total = total + money
print("Всього грошей:", total)
```

- Вбудована функція **sum()**

```
pocket_money = [50, 30, 40, 60]

total = sum(pocket_money)
print("Швидка сума:", total)
```

```
Всього грошей: 180
```

- Додавання з умовами

```
pocket_money = [50, 30, 40, 60]

big = 0
for money in pocket_money:
    if money > 40:
        big += money
print("Великі суми:", big)
```

```
Великі суми: 110
```

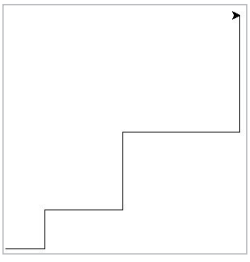
Приклад. Обчислення середнього значення

```
grades = [9, 10, 7, 8, 12]
total = 0

for i in grades:
    total = total + i

average = total / len(grades)
print("Середній бал =", average)
```

```
Середній бал = 9.2
```

Приклад з **turtle**:

```
from turtle import *

t = Turtle()

positions = [50, 100, 150]

for dist in positions:
    t.forward(dist)    # Рух уперед
    t.left(90)         # Поворот вліво
    t.forward(dist)    # Рух уперед
    t.right(90)        # Поворот праворуч
```



Приклад. Створення нового списку:

```
numbers = [2, 3, 4, 5]
squares = []

for n in numbers:
    squares.append(n ** 2)

print("Квадрати:", squares)
```

```
Квадрати: [4, 9, 16, 25]
```

Практична робота №36

Завдання 1. Підрахунок і сумування

1. Створіть список оцінок учнів:
   `grades = [12, 11, 10, 9, 8, 12, 7, 10, 11, 12]`
2. Використовуючи цикл **for**, знайдіть:
   - загальну суму оцінок;
   - середній бал учнів;
   - кількість учнів, які отримали 12 балів.
3. Виведіть всі результати на екран.

Завдання 2. Модифікація списку

Використовуючи цикл **for**, створіть новий список, де кожна оцінка збільшена на 1 бал, але не більш ніж 12. Виведіть новий список на екран.

Підказка: для обмеження використайте конструкцію `if new_mark > 12: new_mark = 12`.

Завдання 3. Список покупок

1. Створіть список покупок:
   `shopping = ["хліб", "молоко", "сир", "яблука", "банани"]`
2. За допомогою циклу **for** виведіть на екран повідомлення:
   `Я хочу купити <назва_товару> для кожного елемента списку.`

Завдання 4. Кількість від'ємних

Створіть список із додатних і від'ємних чисел. Знайдіть кількість від'ємних елементів списку.

Завдання 5. П'ять кіл

1. Створіть список чисел `[1,2,3,4,5]`.
2. За допомогою циклу **for** намалюйте коло для кожного числа, де радіус дорівнює числу, помноженому на 10.



Ключові поняття

Пошук у списках
Знаходження позиції елемента
Максимальний та мінімальний елементи списку

Що таке пошук у списках

У житті ми часто шукаємо щось найкраще або найгірше:

- найбільший бал у класі;
- найменшу ціну товару в магазині;
- найвищу температуру за тиждень;
- найпопулярнішу гру серед друзів.

У програмуванні для цього використовують пошук у списках. Ми можемо знайти максимум, мінімум або елементи, які відповідають певній умові.

Пошук дозволяє знаходити елементи в списку: чи є щось (наявність), де воно (індекс), яке найбільше/найменше (max/min), або шукати з умовами (наприклад, найбільше парне число). Це робить програми «розумними».

У реальному світі пошук використовують скрізь: у **Google** — шукають слова в текстах, у іграх як **Roblox** — знаходять особу, котра отримала максимум очок, у магазинах — мінімальну ціну товару.

Пошук — це процес знаходження елемента або інформації у списку.

У **Python** списки великі, тому ручний перегляд (список[0], список[1]...) — це нудно і неефективно. Пошук поєднується з ітерацією: цикл **for** «проглядає» елементи, а умови (**if**) допомагають знайти потрібне.

Перевірка наявності елемента

Щоб перевірити, чи існує елемент у списку, використовують ключове слово **in**:

if елемент **in** список (Повертає **True**/**False**. Швидко, без циклу)

```
products = ["молоко", "хліб", "яйця"]
if "яйця" in products:
    print("Можна спекти омлет!")  # Виведе це
else:
    print("Купи яйця!")

if "сир" not in products:
    print("Сиру немає.")  # Виведе це
```

Знаходження позиції елемента

Для знаходження індексу потрібного елемента використовують такі два способи:

- За допомогою циклу **for**

```
friends = ["Аня", "Богдан", "Віка", "Аня"]
positions = []

for i, friend in enumerate(friends):
    if friend == "Аня":
        positions.append(i)

print("Позиції Ані:", positions)
```

```
Аня на позиції: [0, 3]
```

- За допомогою методу **index(елемент)**

```
friends = ["Аня", "Богдан", "Віка", "Аня"]
position = friends.index("Аня")
print("Аня на позиції:", position)
```

```
Аня на позиції: 0
```

Метод **index** шукає лише перший індекс, де є елемент.

Знаходження максимуму та мінімуму

Щоб знайти максимум і мінімум у **Python**, є кілька способів залежно від даних, з якими ви працюєте. Ось основні підходи:

- Вбудовані функції **max()** і **min()**

  Python надає готові функції:

  - **max(список)** — повертає найбільший елемент
  - **min(список)** — повертає найменший елемент

```
numbers = [5, 2, 9, 1, 7, 6]
maximum = max(numbers)
minimum = min(numbers)
print("Максимум:", maximum)
print("Мінімум:", minimum)
```

```
Максимум: 9
Мінімум: 1
```

Приклад. Температура за тиждень.

```
temp = [-2, 0, 3, 5, -1, 4, 2]
print("Максимальна температура:", max(temp))
print("Мінімальна температура:", min(temp))
```

```
Максимальна температура: 5
Мінімальна температура: -2
```

Якщо потрібно знайти індекс максимального або мінімального елемента:

```
numbers = [5, 2, 9, 1, 7, 6]

max_index = numbers. index(max(numbers))
min_index = numbers. index(min(numbers))

print("Індекс максимуму:", max_index)
print("Індекс мінімуму:", min_index)
```

```
Індекс максимуму: 2
Індекс мінімуму: 3
```

Якщо потрібно знайти максимум і мінімум без вбудованих функцій, можна зробити це вручну, використовуючи цикл:

Пошук максимального

```
numbers = [5, 9, 3, 7, 2]

# Припускаємо, що перше число — найбільше
min_num = numbers[0]

for i in numbers:
    if i < max_num:
        max_num = i

print("Максимум:", max_num)
```

```
Максимум: 9
```

Пошук мінімального

```
numbers = [5, 9, 3, 7, 2]

# Припускаємо, що перше число — найменше
min_num = numbers[0]

for i in numbers:
    if i < min_num:
        min_num = i

print("Мінімум:", min_num)
```

```
Мінімум: 2
```

Пояснення:

- Цикл перевіряє кожне число у списку.
- Якщо знаходить більше — оновлює змінну **max_num**.
- Якщо знаходить менше — оновлює **min_num**.

Так працює логіка пошуку вручну.

Практична робота №37

Завдання 1. Максимум і мінімум

1. Створіть список оцінок учнів/учениць вашого класу з інформатики:
   ```
   grades = [9, 12, 7, 10, 11]
   ```
2. Знайдіть:
   - максимальний бал;
   - мінімальний бал;
   - індекс максимального і мінімального бала.
3. Виведіть результати на екран.

Завдання 2. Пошук за умовою

1. Створіть список чисел:
   ```
   numbers = [5, 8, 12, 3, 10, 7]
   ```
2. Виведіть усі числа, які більші за 7.
3. Знайдіть перше число, яке більше за 10, і виведіть його.

Завдання 3. Пошук із кількома умовами

1. Створіть список оцінок.
2. Виведіть усі оцінки, які не менші за 10 і не дорівнюють 12.

Завдання 4. Використання пошуку в графіці (**Turtle**)

1. Створіть список радіусів кіл:
   ```
   rad = [10, 50, 30, 70, 40]
   ```
2. Знайдіть найбільше коло (максимальний радіус).
3. Намалюйте усі кола та позначте найбільше червоним кольором.

Завдання 5.

1. Створіть список чисел.
2. Знайдіть елемент із найменшим значенням.
3. Знайдіть суму і кількість елементів, які стоять попереду нього.

Завдання 6. Улюблена гра

1. Створіть список комп'ютерних ігор:
   ```
   games = ["Minecraft", "Roblox", "Fortnite", "CS2", "GTA"]
   ```
2. Запитайте у користувача, яка його улюблена гра.
3. Зробіть перевірку, чи є ця гра у списку.



Тема 38 Сортування списків. Виведення за зростанням / спаданням

Ключові поняття

Сортування списків
Методи sort() і sorted()
Ручне сортування

Що таке сортування?

У житті ми постійно щось упорядковуємо:

- розкладаємо книжки за розміром або алфавітом;
- сортуємо фотографії за датою;
- дивимось ціни в інтернет-магазині від дешевих до дорогих.

У програмуванні це називається сортуванням — розташуванням елементів у певному порядку: за зростанням або спаданням.

Сортування — це процес переставляння елементів списку в певному порядку.

У **Python** списки можуть бути хаотичними, але після сортування вони стають організованими, як книги на полиці в бібліотеці.

Критерії сортувань:

За зростанням (**ascending**)

Від меншого до більшого (числа: 1, 2, 3; рядки: «Айдар», «Богдан», «Віка» — за алфавітом).

За спаданням (**descending**)

Від більшого до меншого (числа: 3, 2, 1; рядки: «Віка», «Богдан», «Айдар»).

Вбудовані методи сортування в Python

Python має прості інструменти для сортування:

- функція `sorted`(список) — повертає новий сортований список.
- метод `list.sort()` — сортує оригінальний список на місці (змінює його).

- `sorted`(список): не змінює оригіналу, корисно для копії.
- `list.sort()`: змінює список, економить пам'ять.
- Параметр `reverse=True`: для спадання.
- Параметр `key=функція`: для кастомного порядку (`key=len` — за довжиною, `key=str.lower` — ігнорує регістр).

Метод .sort()

Метод змінює сам список і розташовує його елементи в порядку за зростанням.

```
grades = [10, 7, 12, 9, 11]
grades.sort()
print(grades)
```

```
[7, 9, 10, 10, 11, 12]
```

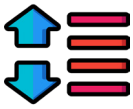
Щоб сортувати від більшого до меншого, треба вказати параметр **reverse=True**:

```
grades = [10, 7, 12, 9, 11]
grades.sort(reverse=True)
print(grades)
```

```
[12, 11, 10, 9, 7 ]
```

Функція sorted()

Функція не змінює початкового списку, а створює новий відсортований список.



```
numbers = [5, 2, 9, 1]
new_list = sorted(numbers)

print("Новий список:", new_list)
print("Старий список:", numbers)
```

```
Новий список:[1, 2, 5, 9]
Новий список:[5, 2, 9, 1]
```

Зручно, коли треба зберегти вихідний список без змін.

Як Python розуміє, яким чином сортувати

Python може сортувати:

- *числа* — за величиною;
- *рядки (тексти)* — за алфавітом, враховуючи регістр (спочатку великі літери, потім малі);
- *рядки (тексти)* — за алфавітом.

```
words = ["банан", "Яблуко", "Апельсин"]
sorted_words = sorted(words)
print("За алфавітом:", sorted_words)
```

Сортування відбувається за першими літерами, потім — за наступними.

```
За алфавітом:['Апельсин', 'Яблуко', 'банан']
```

Якщо потрібно проігнорувати регістр, а відсортувати лише за алфавітом, використовують параметр **key**:

```
words = ["банан", "Яблуко", "Апельсин"]
sorted_words = sorted(words, key=str.lower)
print("За алфавітом без регістру:", sorted_words)
```

```
За алфавітом без регістру:['Апельсин', 'банан', 'Яблуко']
```

Якщо ж потрібно сортувати за довжиною слова (кількість символів):

```
words = ["банан", "Яблуко", "Апельсин"]
by_length = sorted(words, key=len)
print("За довжиною:", by_length)
```

```
За довжиною:['банан', 'Яблуко', 'Апельсин']
```

Впорядкування списку вручну

Щоб краще зрозуміти процес сортування, можна зробити його вручну:

```
numbers = [5, 2, 8, 1, 9, 4, 5]

for i in range(len(numbers)):
    for j in range(i + 1, len(numbers)):
        if numbers[i] > numbers[j]:
            numbers[i], numbers[j] = numbers[j], numbers[i]

print(numbers)
```

Пояснення:

- Програма порівнює пари елементів.
- Якщо перше число більше від другого — вони міняються місцями.
- У результаті — список відсортовано.

Цікаво знати

- У великих програмах сортування використовується для обробки даних, створення звітів, таблиць, рейтингів, фільтрації тощо.
- У **Python** алгоритм сортування дуже ефективний і базується на методі **Timsort** (поєднання сортування вставкою і злиттям).

Практична робота №38

Завдання 1. Математичний рейтинг

1. Є список оцінок учня з алгебри: `marks = [12, 7, 10, 9, 8, 11]`
2. Відсортуйте оцінки за зростанням і за спаданням.
3. Виведіть обидва результати з поясненнями.

Завдання 2. Турнірна таблиця

1. Є список кількостей забитих голів командами: `goals = [3, 5, 2, 6, 4]`
2. Відсортуйте список за спаданням (від найкращого до найгіршого).
3. Визначте переможця — команду з найбільшою кількістю голів.
4. Виведіть результати турніру.

Завдання 3. Ціни в магазині

1. Є список з цінами деяких товарів: `prices = [150, 99, 320, 45, 200]`
2. Відсортуйте ціни за зростанням.
3. Виведіть найдешевший і найдорожчий товари.
4. Зробіть «знижку» — зменшіть усі ціни на 10% і знову відсортуйте.

Завдання 4. Алфавітний порядок

1. Створіть список: `animals = ["cat", "Elephant", "dog", "giraffe", "Ant", "Tiger", "jaguar"]`
2. Відсортуйте його в алфавітному порядку, враховуючи регістр.
3. Відсортуйте його в алфавітному порядку, не враховуючи регістру.
4. Відсортуйте його за довжиною слова.
5. Виведіть результати.

Завдання 5. Зоопарк

1. Створіть список тварин.
2. Відсортуйте його за довжиною назв тварин (від коротких до довгих і навпаки).
3. Додайте дві нові тварини, а потім видаліть одну за номером.

Ключові поняття

Генерація списків
Додавання елементів у циклі
Наповнення списку випадковими елементами

Що означає «генерувати список»?

На попередніх уроках ми створювали списки так:

```
numbers = [3, 5, 7, 9]
```

Але що робити, якщо нам потрібно скласти список із 100 чисел? Вводити все вручну — це довго, нудно і незручно.

Тому програмісти використовують генерацію списків.

Генерація дозволяє швидко заповнювати списки числами в порядку або випадково. Це робить програми динамічними.

У реальному світі генерацію використовують скрізь: у лотереях (випадкові числа), у прогнозах погоди (послідовності дат), у іграх як **Fortnite** (випадкові локації), у магазинах (генерація списків товарів за ціною).

Що таке генерація списків

Генерація списків — це автоматичне створення списків з елементами за певними правилами, без ручного перелічення.

У **Python** списки можна заповнювати вручну, використовуючи цикли, та генерувати з функцій чи модулів, щоб програми були гнучкими й ефективними.

Додавання елементів у циклі

Припустимо, користувач має ввести 5 чисел. Ми створюємо порожній список і заповнюємо його через цикл **for**:

```
numbers = []  # порожній список

for i in range(5):
    num = int(input("Введи число:"))
    numbers.append(num)

print("Отриманий список:", numbers)
```

```
Введи число: 5
Введи число: 7
Введи число: 4
Введи число: 8
Введи число: 2
Отриманий список [5, 7, 4, 8, 2]
```

Додавання невідомої кількості елементів (до введення «стоп»)

Коли ми не знаємо наперед, скільки елементів користувач введе, можна використати цикл **while** із командою зупинки повторення.

```
words = []

while True:
    word = input("Введи слово (або 'стоп' для завершення):")
    if word.lower() == "стоп":
        break
    words.append(word)

print("Отриманий список:", words)
```

```
Введи слово(або стоп для завершення): кіт
Введи слово: (або 'стоп' для завершення): пес
Введи слово: (або 'стоп' для завершення): кінь
Введи слово: (або 'стоп' для завершення): стоп
Отриманий список: ['кіт', 'пес', 'кінь']
```

Введення всіх елементів в одному рядку

Інколи вводити дані зручніше через пробіл.

```
words = input("Введи слово через пробіл:"). split()

print(words)
```

```
Введи числа через пробіл: лис вовк заєць
['лис', 'вовк', 'заєць']
```

```
numbers = [int(x) for x in input("Введи числа через пробіл:").
split()]

print(numbers)
```

```
Введи числа через пробіл: 5 7 4 3 8
[5, 7, 4, 3, 8]
```

Генерація за допомогою функції range()

Функція **range** створює послідовність цілих чисел, яку можна перетворити на список за допомогою **`list(range(...))`**. Це як ланцюжок намистинок — рівномірний і передбачуваний.

```
# Проста послідовність
days = list(range(1, 8))
print("Дні тижня:", days)
```

```
Дні тижня: [1, 2, 3, 4, 5, 6, 7]
```

```
# З кроком
even_numbers = list(range(0, 10, 2))
print("Парні числа:", even_numbers)
```

```
Парні числа: [0, 2, 4, 6, 8]
```

```
# Зворотний
countdown = list(range(10, 0, -1))
print("Зворотний відлік:", countdown)
```

```
Зворотний відлік: [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
```

Для найдопитливіших — приклад із абстракцією:

```
squares = [x * x for x in range(1, 6)]
print("Квадрати:", squares)
# [1, 4, 9, 16, 25]
```

Генерація за допомогою модуля random

Буває, що нам треба зробити не просто послідовність, а випадкові дані:

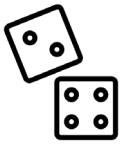
- випадкові оцінки;
- випадкові результати гри;
- випадкові кольори тощо.

Для цього використовується модуль **random**.

Основні функції для списків:

`random.randint(a, b)` випадкове ціле від **`a`** до **`b`** (включно).
`random.choice(список)` випадковий елемент зі списку.
`random.sample(список, k)` **`k`** унікальних випадкових елементів.
`random.shuffle(список)` перемішує елементи у списку.
`random.random()` випадкове дробове число від **`0`** до **`1`**.
`random.randrange(start, stop, step)` як **`range`**, але випадкове одне число.
`random.uniform(a, b)` випадкове дробове від **`a`** до **`b`**.

```
import random

grades = []
for i in range(10):
    grades.append(random.randint(6, 12))

print("Оцінки учнів:", grades)
```



```
Оцінки учнів: [11, 8, 8, 11, 11, 12, 8, 10, 8, 7]
```

```
import random

colors = ["червоний", "синій", "зелений"]
random_color = random.choice(colors)
print("Випадковий колір:", random_color)
```

```
Випадковий колір: зелений
```

Для найдопитливіших — **Python** дозволяє створювати списки «в один рядок»:

```
import random

numbers = [random.randint(1, 50) for i in range(6)]
print(numbers)
```

Ми можемо створити список чисел із певним діапазоном і випадково перемішати:

```
import random

numbers = list(range(1, 11))
random.shuffle(numbers)
print(numbers)
```

```
[1, 6, 10, 9, 2, 5, 3, 8, 7, 4]
```

Практична робота №39

Завдання 1. Мій список улюблених фруктів

1. Створіть порожній список **fruits**.

2. Запитайте у користувача, які його улюблені фрукти (5 назв), і додайте їх до списку.

3. Виведіть отриманий список.

Приклад:

```
Мій список фруктів: ['яблуко', 'банан', 'ківі', 'апельсин',
'груша']
```

Завдання 2. Список оцінок

1. Створіть порожній список **grades**.

2. Запитайте у користувача, які він отримав оцінки з `n` предметів, і додайте до списку як числа.

3. Після введення виведіть:

- усі оцінки;
- середній бал.

Завдання 3. Покупки у супермаркеті

1. Користувач вводить назви товарів, які хоче купити.

2. Введення триває, доки користувач не напише слова «стоп».

3. Усі товари зберігаються у список **cart**.

4. Після цього програма виводить усі покупки.

Завдання 4. Список чисел

Створіть список чисел від 1 до 20 за допомогою **range()** і виведіть його на екран.

Завдання 5. Парні числа

Створіть список парних чисел від 2 до 20 включно.

Завдання 6. Випадковий вибір героя

1. Створіть порожній список **heroes**.

2. Створіть список із 5 героїв, потім виведіть випадкового з них.

Практична робота №40

Тема: «Моя бібліотека»
Мета:

- Навчитися створювати, змінювати, сортувати й аналізувати списки у **Python**.
- Закріпити роботу з циклами, умовами та введенням даних.
- Розвивати алгоритмічне мислення на прикладі створення власного «додатку читача».

Завдання:

Створити свій мінідодаток «Моя бібліотека», який допомагає: зберігати список книг, котрі хочете прочитати; позначати вже прочитані книги; додавати нові книги; переглядати свої списки у зручному вигляді та інші корисні функції.

1. Створіть початкові списки за зразком або зі своїми назвами книг:
   `books_to_read` — список книг, які хочете прочитати;
   `books_read` — (пустий) список прочитаних книг.

```
# Початкові списки
books_to_read = ["Гаррі Поттер", "Мистецтво війни", "Голодні ігри", "Тореадори з Васюківки", "Ерагон", "Дюна", "Кобзар", "Володар мух", "Крадійка книжок", "Володар кілець" ]
books_read = []
```

Додаткова умова:

```
ratings — (пустий) список їхніх оцінок (за 10-бальною шкалою)
      ratings = []
```

2. Створити меню мінідодатку, використовуючи **while True**, з вибором функцій

```
📚 Вітаємо у додатку 'Моя бібліотека'!
======================================

Меню:
1 – Показати список книг для читання"
2 – Позначити прочитану книгу"
3 – Додати нову книгу
4 – Показати список прочитаних книг
5 – Вийти
👉 Оберіть дію:
```

```
print("Вітаємо у додатку 'Моя бібліотека'!")
print("=====================================")
while True:
    print("\nМеню:")
    print("1 – Показати список книг для читання")
    print("2 – Позначити прочитану книгу")
          …
    ch = input("Оберіть дію: ")
```

3. Реалізація

3.1. Виведення списку книг для читання

Описати дії при натисканні пункту 1 меню

```
if ch == "1":
    (ваш код)
```

За допомогою циклу вивести інформацію про список книг для читання. *Приклад виведення:*

```
📖 Книги для читання:
1. Гаррі Поттер
2. Мистецтво війни
3. Голодні ігри
4. Тореадори з Васюківки
5. Ерагон
6. Дюна
7. Кобзар
8. Володар мух
9. Крадійка книжок
10. Володар кілець
```

3.2. Відмітка прочитаної книги

Описати дії при натисканні пункту 2 меню

```
if ch == "2":
    (ваш код)
```

Дії цього пункту:

- запитує, яка книга прочитана (або за назвою, або за номером)
- цю книгу видаляє зі списку **books_to_read**
- цю книгу додає до списку **books_read**

```
Введіть назву книги: Гаррі Поттер
✅ Книгу 'Гаррі Поттер' додано до списку прочитаних.
```

3.3. Додавання нової книги

Описати дії при натисканні пункту 3 меню

```
if ch == "3":
    (ваш код)
```

Дії цього пункту:

- запитує, яку нову книгу бажаєте додати до списку читання;
- цю книгу додає до списку **books_to_read**

3.4. Показати список прочитаних книг

Описати дії при натисканні пункту 4 меню

```
if ch == "4":
        (ваш код)
```

Дії цього пункту: виводить список books_read

```
Прочитані книги:
1. Гаррі Поттер
2. Кобзар
```

Додаткова умова:

Описати дії, якщо список прочитаних книг пустий. (Один із варіантів реалізації — довжина цього списку дорівнює нулю).

```
Оберіть дію: 4
Ви ще нічого не прочитали.
```

3.5. Вихід з програми

Описати дії при натисканні пункту 5 меню, для виходу з програми.

Зразок:

```
elif ch == "5":
print("\n Дякуємо, що користувалися додатком 'Моя бібліоте-
ка'!")
print("Бажаємо приємного читання!")
break
```

Додаткова умова (творча):

Спробуйте удосконалити програму, додавши нові меню.

Наприклад: додайте пункт "Показати статистику":

- кількість прочитаних і непрочитаних книг;
- відсоток виконання читацького плану.

Ключові поняття

Рядки в Python
Конкатенація
Слайсинг
Методи рядків

Рядки в Python

Пам'ятаєте, що списки — це наче коробки з речами, які ми ітеруємо, шукаємо в них, сортуємо та генеруємо? Тепер уявіть, що замість чисел чи списків ви працюєте з текстом: словами, реченнями, повідомленнями.

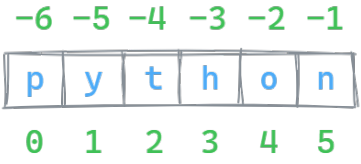
Ось тут на сцену виходять рядки **(strings)** у **Python** — це послідовності символів, як намистинки в намисті, де кожна намистинка — літера, цифра чи знак.

Рядки дозволяють програмам «говорити»: виводити привітання, імена, історії.

Це робить програми живими: наприклад, у грі можете вивести «Вітаю, гравцю!», у додатку з піцою, наприклад, — «Ваше замовлення: піца з сиром», або у вікні з **turtle** — написати текст на екрані. Без рядків програми були б нудними, як книга без слів!

У реальному світі рядки використовують скрізь:

- чатах (повідомлення в **Instagram**);
- пошуковиках (**Google** шукає слова);
- іграх (діалоги в **Minecraft**);
- додатках.



Рядок (string) — це незмінна послідовність символів, укладених у лапки.

Символами можуть бути літери, цифри, пробіли, знаки пунктуації тощо.

Приклади рядків у **Python**:

```
name = "Олександр"
city = 'Київ'
phrase = "Привіт, світе!"
number_str = "2025"
```

Рядки можуть містити будь-які символи, навіть емодзі: `smile = "😊"`

На відміну від списків (з попередніх тем), рядки не можна змінити на місці — щоб змінити, потрібно створити нові.

Основні ідеї


Незмінність


Довжина


Типи символів


Порівняння з іншими

Як створити рядок?

Оголошувати рядок просто: в одинарних ' ' або подвійних " " лапках. Для багаторядкових — потрійні """ """ або ''' '''.

Одинарні/подвійні лапки

Використовують для простих текстів. Якщо текст має ' — використовують ", і навпаки.

```
phrase1 = 'Привіт'
phrase2 = "Python — чудовий!"
```

Потрійні

Використовують для довгих текстів або абзаців, текстів з новими рядками, як у вірші.

```
message = '''Це приклад
багаторядкового
рядка у Python.'''
```

Форматування

Форматування рядків через **f-strings** (форматовані рядки) — це сучасний і дуже зручний спосіб вставляти значення змінних безпосередньо в текст.

Він з'явився в **Python 3.6** і зараз є стандартним підходом для роботи з рядками.

```
name = "Олександр"
print(f"Привіт, {name}!")
# Привіт, Олександр!
```

```
f"текст {змінна}"
```

Перед лапками ставимо **f** (або F), а всередині фігурних дужок {} пишемо назву змінної чи навіть вираз.

Форматування чисел

```
price = 12.3456
print(f"Ціна: {price:.2f} грн")
```

Виведе:

Ціна: 12.35 грн

Пояснення: `:.2f` — число з двома знаками після коми.

Конкатенація

```
name = 'Nazar'
print('Hello, ' + name)
# Hello, Nazar
```

Конкатенація — це операція з'єднання (об'єднання) рядків або інших послідовностей в один рядок.

Простими словами: конкатенація — це коли ми «склеюємо» два або більше текстів (рядків) разом.

Важливо:

- Конкатенація не виконує обчислень — вона не додає, а з'єднує текст.
- Якщо потрібно об'єднати числа, спочатку треба перетворити їх на текст.

Порожній рядок

```
"" — як порожня сторінка
```

Множення

```
laugh = "ха" * 5
print(laugh)
# хахахахаха
```

Множення рядків — це операція, при якій рядок повторюється кілька разів підряд.

Рядок — це теж список!

Уявімо, що рядок — це список символів, де кожен символ має свій порядковий номер (індекс).

Це означає, що ми можемо:

- звертатися до окремих символів;
- зробити зрізи (отримувати частини рядка);
- проходити по рядку циклом, як по списку.

Індексація починається з 0. Тобто перший символ має індекс 0, другий — 1, і т.д. Якщо з кінця, то індекси йдуть -1, -2 і т.д.

Приклад:

```
word = "Книга"

print(word[0])    # К  ← перший символ
print(word[1])    # н  ← другий символ
print(word[2])    # и
print(word[3])    # г
print(word[4])    # а
print(word[-1])   # а  ← останній символ
```

Робота з частинами рядка (слайсинг)

Ми можемо вирізати частину рядка за допомогою зрізів — як ножицями.

`рядок[початок:кінець:крок]`

Правило: зріз `[a:b:k]` бере символи від індексу **`a`** до **`b`**, але не включаючи **`b`**. Якщо потрібно, використовують крок **`k`**.

Приклад:

```
text = "Програмування"
print(text[2:8])    # ограму
print(text[0:7])    # Програм
print(text[7:])     # ування
print(text[:5])     # Прогр
print(text[::2])    # Пормвня (кожен другий символ)
print(text[::-1])   # яннавумаргорП
print(text[-5:-1]) # ванн
```

Основні методи рядків

Рядки в **Python** мають вбудовані методи — «команди», що допомагають з ними працювати.

Методи — це функції рядків, які викликаються: **рядок.метод()**. Вони повертають новий рядок (не змінюють оригіналу).

Зміна регістру:

`upper()` — великі літери
`lower()` — маленькі
`capitalize()` — перша велика
`title()` — кожне слово з великої

```
s = "мова Python"
print(s.lower())       # мова python
print(s.upper())       # МОВА PYTHON
print(s.capitalize())  # Мова python
print(s.title())       # Мова Python
```

Робота з пробілами:

Очищення:
`strip()` — видаляє пробіли з кінців,
`lstrip()/rstrip()` — зліва/справа.

```
text = "  привіт, світ!  "
clean = text.strip()  # привіт, світ!
print("Очищений:", clean)
```

Підрахунок:

`count(підрядок)` — рахує кількість входжень

```
text = "banana"
print(text.count("a"))  # 3
```

Рядки — це незмінні об'єкти

Це означає, що змінити окремий символ без створення нового рядка не можна.

Так не працює:

```
word = "кіт"
word[0] = "м"  # Помилка
```

Так можна:

```
word = "м" + word[1:]
print(word)  # міт
```

Порада: Пам'ятайте, що рядки — це текстові дані, тому будь-яке введення з клавіатури через **input()** теж є рядком!

```
age = input ("Введіть ваш вік:")
print (type(age))      # <class 'str'>
```

Практична робота №41

Завдання 1. Напишіть програму, яка запитує ім'я користувача та його улюблений предмет і виводить:

```
Привіт, [ім'я]! Тобі подобається [предмет]?
```

Додатково:

- Перетворіть ім'я так, щоб перша літера була великою **(capitalize())**
- Перетворіть предмет у великий регістр **(upper())**

Завдання 2. Маємо назву міста:

```
city = "Львів"
```

- Виведіть першу літеру міста
- Виведіть останню літеру
- Виведіть назву міста задом наперед

Завдання 3. Напишіть програму, яка запитує у користувача речення, яке має великі та маленькі літери. Порахуйте, скільки в реченні букв «а» (рахувати слід як велику «А», так і маленьку «а»).
Додатково: запитати в користувача, яку літеру він хоче порахувати.

Завдання 4. Напишіть програму, яка запитує у користувача три слова.

- Виведіть кількість букв кожного слова
- Виведіть найдовше слово

Завдання 5. Ви маєте рядок. Потрібно вивести рядок, що містить ті самі символи, але у зворотному порядку, а також частину рядка, що починається з n-го елемента (для обох рядків).

Завдання 6. Ви маєте рядок. Потрібно розділити його на два окремі рядки з однаковою кількістю символів (якщо символів непарна кількість, то середню літеру не використовувати). Виведіть отримані половини та рядок із заміненими місцями половинами.

Наприклад:

слово **Python**	слово **Informatika**
`Pyt`	`Infor`
`hon`	`atika`
`honPyt`	`atikaInfor`



Пошук у рядках
Заміна частини рядка
Методи split() та join()
Обхід рядка

Маніпуляції з рядками

Пам'ятаєте, що рядки — це «намиста» з символів, незмінні, але зручні для «розмов» у програмах? Тепер уявіть, що ви не просто дивитеся на це намисто, а маніпулюєте ним: шукаєте певну намистинку (пошук), замінюєте одну на іншу (заміна), розбираєте на частини (**split**) або збираєте з частин нове (**join**).

Ось так працюють маніпуляції з рядками — це інструменти, які дозволяють програмам «редагувати» текст, як у текстовому редакторі або грі з літерами.

Маніпуляції роблять програми потужнішими: наприклад, у чаті ви можете шукати слово в повідомленні, замінювати помилки (як автокорекція), розбивати речення на слова (для аналізу) або збирати слова в речення (для створення історії).

Без маніпуляцій рядки були б статичними, як напис на стіні — не можна змінити!

У реальному світі маніпуляції використовують скрізь:

- у пошуковиках (**Google** шукає слова)
- у чатах (заміна емодзі)
- в іграх (розбір команд у **Minecraft**)
- у додатках (розділення **email** на частини)

Пошук у рядках

Пошук — це процес знаходження підрядка (частини тексту) в рядку.

У **Python** для цього є методи **find**, **rfind**, **index**, які повертають індекс (позицію) знайденого, або -1/помилку, якщо немає.

find(підрядок)

Метод **find()** шукає, де вперше використовується певне слово або символ у рядку, і повертає індекс першого входження. Якщо не знаходить — повертає -1.

```
diary = "Мій щоденник: сьогодні я їв яблуко"
pos_apple = diary.find("яблуко")  # 27
print("Позиція 'яблуко':", pos_apple)
```

rfind(підрядок)

Працює як метод **find**, але шукає з кінця (останнє входження).

```
last_ya = diary.rfind("я")  # 21 (в "я їв")
print("Останнє 'я':", last_ya)
```

index(підрядок)

Метод **index()** робить те саме, але якщо слова немає, він видає помилку.

```
diary = "Мій щоденник: сьогодні я їв яблуко"
pos_d = diary.index("щоденник")  # 4
print("Позиція 'щоденник':", pos_d)
pos_orange = diary.index("апельсин")  # ValueError
print("Позиція 'апельсин':", pos_orange)
```

rindex(підрядок): як **index**, але з кінця.

пошук з параметрами: як **index**, але з кінця.

```
sub_pos = diary.find("я", 15, 25)  # Шукає 'я' з 15 до 25 — 21
print("'я' в зрізі:", sub_pos)
```

Перевірка наявності слова

Іноді не потрібно знати позицію — достатньо перевірити, чи є слово.

Для цього використовують оператор **in**: ➢

```
if "яблуко" in diary:
   print("Так, це слово є!")
else:
   print("Немає такого слова.")
```

Це дуже поширений спосіб: так теж можна перевірити, чи є **@** в email-адресі, чи є **https://** у посиланні.

Пошук чутливий до регістру: «Привіт» != «привіт». Для ігнору використовують **lower()** перед пошуком.

```
text = "Мене звати Олександр, я люблю Python."
word = "python"

if word.lower() in text.lower():
    print("Так, це слово є!")
else:
    print("Немає такого слова.")
```

Заміна частини рядка

Заміна — це зміна частини рядка на іншу частину. Метод **replace(старе, нове, count)** повертає новий рядок із замінами.

Параметри: старе — що замінити, нове — на що, **count** — скільки разів (за замовчуванням — всі).

Особливості: не змінює оригіналу, чутливий до регістру. Якщо **count=1** — тільки перше.

```
sentence = "Я люблю яблука, яблука смачні"
new_sentence = sentence.replace("яблука", "апельсини")
# 'Я люблю апельсини, апельсини смачні'
print(new_sentence)
```

```
first_replace = sentence.replace("яблука", "банани", 1)
print(first_replace)
# 'Я люблю банани, яблука смачні'
```

```
# Ігнор регістру: спочатку lower(), але replace не змінює регістр
low_text = sentence.lower().replace("яблука", "фрукти")
print(low_text)
 # 'я люблю фрукти, фрукти смачні'
```

Поділ рядка: метод .split()

Уявіть, що речення — це торт, який можна розрізати на шматочки — слова. Саме це робить метод **.split()**.

Split — розбиває рядок на список частин за роздільником.

split(роздільник, maxsplit): роздільник — за чим ділити (за замовчуванням пробіл), **maxsplit** — максимум поділів.

```
sentence = "Я люблю програмування і каву"
words = sentence.split()
print(words)
# ['Я', 'люблю', 'програмування', 'і', 'каву']
```

За замовчуванням розділювач — пробіл, але його можна задати самостійно ➢

```
fruits = "яблуко,груша,слива"
list_fruits = fruits.split(",")
print(list_fruits)
# ['яблуко', 'груша', 'слива']
```

Так можна легко створити список з одного рядка — наприклад, коли користувач вводить кілька слів через кому.

```
# З maxsplit
address = "місто:Київ,вулиця:Хрещатик,будинок:1"
parts = address.split(":", 1)
print(parts)
# ['місто', 'Київ,вулиця:Хрещатик,будинок:1'] (тільки 1 поділ)
```

Використовують також і такі методи:

- **rsplit(роздільник, maxsplit)**: як **split**, але з кінця.
- **splitlines()**: розбиває на рядки за **\n**.

```
# Splitlines
poem = '''Рядок один
Рядок два
Рядок три'''
lines = poem.splitlines()
print("Рядки вірша:", lines)
# ['Рядок один', 'Рядок
два', 'Рядок три']
```

З'єднання елементів у рядок

Метод **join()** робить зворотну дію до **split()** — він об'єднує список у рядок.

«роздільник».join(список)

Роздільник — за чим ділити (за замовчуванням пробіл), **maxsplit** — максимум поділів.

```
words = ['Я', 'люблю', 'Python']
sentence = " ".join(words)
print(sentence)
# Я люблю Python
```

Можна обирати будь-який розділювач:

```
words = ['Я', 'люблю', 'Python']
sentence = "-".join(words)
print(sentence)
# Я-люблю-Python
```

Приклад з **map** *для* **join**:

```
numbers = [1, 2, 3]
str_nums = "-".
join(map(str, numbers))
# '1-2-3' print(str_nums)
```

Комбінація методів

Реальна сила в тому, що ці методи можна поєднувати.

```
text = "Привіт, світ!"
clean_text = text.strip().replace("світ", "учень").upper()
print(clean_text)
# ПРИВІТ, УЧЕНЬ!
```

.strip() — прибирає зайві пробіли
.replace() — замінює слово
.upper() — робить великі літери

Приклад коду з функцією:

```
def censor(text, bad_word):
    replacement="*"*len(bad_word)
    return text.replace(bad_word, replacement)

message = "Це погане слово: фу"
censored = censor(message, "фу")  # 'Це погане слово: **'
print(censored)
```

Приклад з **random**:

```
import random

sentence = "Я їм фрукт"
fruits = ["яблуко", "банан", "апельсин"]
new_fruit = random.choice(fruits)
new_sentence = sentence.replace("фрукт", new_fruit)
print(new_sentence)  # Наприклад, 'Я їм банан'
```

Обхід рядка за допомогою циклу for

Іноді потрібно пройтись по кожному символу рядка: порахувати їх, знайти конкретні або змінити певні букви.

Для деяких із цих маніпуляцій уже є готові методи та функції. Проте все це можна робити «вручну».

У цьому допомагає цикл **for**:

```
text = "Hello"
for ch in text:
    print(ch)
```

```
H
e
l
l
o
```

Як це працює: **Python** бере перший символ рядка (**H**) і зберігає його у змінній **ch**.

Потім переходить до наступного (**e**), і так далі — поки не закінчиться рядок.

Обходячи посимвольно рядок, ми можемо розв'язувати велику кількість задач.

Але є ще одна хороша річ — це не просто «зачіпати» окремі символи, а й їхні індекси (порядкові номери).

```
text = "Python"

for i in range(len(text)):
    print(i + 1, 'літера – ', text[i])
```

```
1 літера – P
2 літера – y
3 літера – t
4 літера – h
5 літера – o
6 літера – n
```

Практичні приклади обходу рядків

Приклад 1. Підрахунок кількості символів **"a"** у рядку.

Такий підхід можна застосувати до аналізу будь-яких текстів — наприклад, порахувати кількість пробілів, розділових знаків, цифр у повідомленні.

```
text = "banana"
count = 0
for ch in text:
    if ch == "a":
        count += 1
print("Кількість 'a' =", count)
```

Приклад 2. Заміна певних символів.

Ми "обходимо" рядок символ за символом і формуємо новий рядок за власними правилами.

```
text = "Python"
new_text = ""
for ch in text:
   if ch == "o":
      new_text += "0"
   else:
      new_text += ch
print(new_text)
```

Приклад 3. Перевірка, чи є у рядку цифри.

```
text = "Пароль123"
for ch in text:
    if ch.isdigit():
        print("Знайдено цифру:", ch)
```

```
Знайдено цифру: 1
Знайдено цифру: 2
Знайдено цифру: 3
```

Метод **.isdigit()** перевіряє, чи символ є цифрою або чи весь рядок складається повністю із цифр.

Інші методи:

- **.isalpha()** — перевіряє, чи рядок складається лише з літер або чи символ є літерою
- **.isalnum()** — перевіряє, чи рядок складається лише з літер або цифр або чи символ є літерою або цифрою (без спеціальних знаків)
- **.isspace()** — перевіряє, чи символ є пробілом
- **.isupper()** — перевіряє, чи рядок складається лише з великих літер або чи символ є великою літерою
- **.islower()** — перевіряє, чи рядок складається лише з маленьких літер або чи символ є маленькою літерою

Практична робота №42

Завдання 1. Задано рядок. Не використовуючи методу **count**, виведіть кількість літер «п» у рядку.

Додатково: Виведіть частини рядків, які починаються з літер «п».

Завдання 2. Користувач вводить текст та слово з клавіатури. Полічіть, скільки разів слово використовується в тексті. Майте на увазі, що слово з різними регістрами – це одне слово.

Завдання 3. Користувач вводить рядок. Видаліть усі пробіли в рядку.

Завдання 4. Задано рядок, в якому є літери і цифри. Виведіть рядок без цифр.

Порядок виконання:

- створіть рядок із кількох слів, наприклад:

```
s = 'Сьогодні я пробіг 5 км за 30 хвилин і спалив 250 калорій!'
```

- створіть новий порожній рядок для результату, а також рядок цифр:

```
res = ''
numbers = '0123456789'
```

- за допомогою циклу **for** обходимо посимвольно по рядку **s** і перевіряємо, чи є цей символ у рядку **numbers**:

```
for i in s:
      if i in numbers:
            continue
      res = res + i
```

- виводимо результат

```
print('Рядок без цифр:', res)
```

Завдання 5. Задано рядок, в якому є великі та малі літери. Виведіть рядок без великих літер.

Завдання 6. Введіть речення з клавіатури. Визначіть, скільки слів у реченні.

Примітка: слова в реченні розділені одним пробілом. Кількість пробілів впливають на кількість слів у реченні.

Завдання 7. Задано речення. Підрахуйте кількість голосних у реченні **(а, е, і, о, у, я, ю, є, ї, и)** та виведіть рядок лише голосних літер цього речення.

Ключові поняття

Шифрування
Шифр Цезаря
Математична ідея шифру

Що таке шифрування?

Уявіть, що хочете передати своєму другові секретне повідомлення, але щоб ніхто не зрозумів, про що йдеться.

Наприклад, повідомлення «Зустрінемось о 18:00 біля школи» — не дуже підходить, бо його зміст зрозумілий.

Тому люди придумали шифрування:

Шифрування — це процес перетворення звичайного тексту (відкритого) на зашифрований (секретний код), щоб тільки той, хто знає ключ, міг прочитати.

Розшифрування — зворотний процес.

У реальному світі шифрування використовують скрізь:

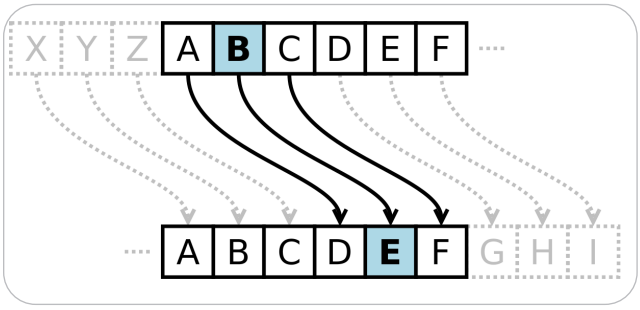
- у паролях (банківські **apps**)
- у чатах (**WhatsApp** шифрує повідомлення)
- в іграх (секретні коди в **Roblox**)
- в історії (**Енігма** в **WWII**)

Історія: Юлій Цезар і його шифр

Ми зосередимося на шифрі Цезаря — простому, але класичному методі шифрування.

Шифр Цезаря названий на честь Юлія Цезаря, римського імператора, який використовував його для таємних листів своїм генералам понад 2000 років тому!

Шифр Цезаря — це заміна із зсувом **(substitution cipher)**, де кожна літера алфавіту замінюється на іншу, зсунуту на фіксоване число позицій (ключ, наприклад, на 3: А→Г, Б→Д).



Отже:

ВІТАЮ → ДКХГБ

(бо В → Д, І → К,
Т → Х, А → Г, Ю → Б)

Як працює:	Алфавіт циклічний — після Я йде А. Для української: А→Г (зсув на 3 літери), Б→Д, ..., Я→В. Пробіли, знаки, цифри — не змінюються (або за правилами).
Ключ:	Число від 1 до 33: АБВГҐДЕЄЖЗИІЇЙКЛМНОПРСТУФХЦЧШЩЬЮЯ
Розшифрування:	Зсув у зворотний бік (ключ -3 або 30 для +3, бо 33-3=30).
Переваги для дітей:	Простий — як рахувати літери в абетці. Навчає циклів, рядків, модульної арифметики (зсув % 33).
Історичний контекст:	Цезар використовував зсув на 3 літери для латинського алфавіту (26 літер). У сучасному світі — базовий приклад, але слабкий (легко зламати).

Математична ідея шифру

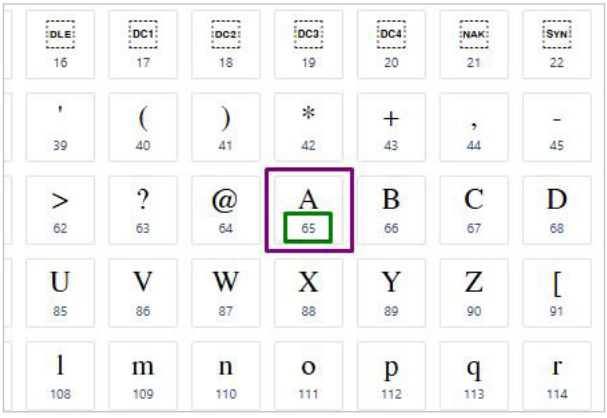
Кожен символ у комп'ютері — це не просто буква, а число, що зберігається у пам'яті та має свій номер (код) у таблиці **Unicode**.

Наприклад: А → 65 В → 66 С → 67 ... Z → 90

Якщо треба зсунути літеру, то просто додаємо ключ до коду:

новий_код = (старий_код + ключ)

А потім перетворюємо назад у літеру.



У **Python** існує функція **ord()** (від слова **ordinal** — порядковий), котра повертає числовий код символу згідно з таблицею **Unicode**.

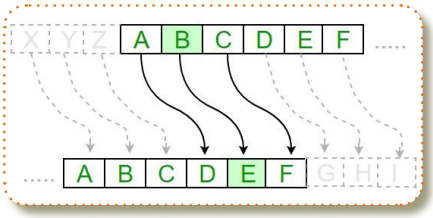
Для оберненої операції використовують функцію **chr()**. Вона робить протилежне: із числа створює символ.

```
print(ord('A'))    # 65
print(ord('a'))    # 97
print(ord('Я'))    # 1071
print(ord(' '))    # 32
```

```
print(chr(65))     # A
print(chr(1071))   # Я
```

Приклад у **Python**, *використовуючи таблицю* **Unicode**:

```
text = "HELLO"
key = 3
result = ""

for i in text:
  # Перетворюємо символ у число (код)
  code = ord(i)
  # Зсуваємо на 3 позиції
  new_code = code + key
  # Перетворюємо назад у символ
  new_ch = chr(new_code)
  result += new_ch

print("Зашифрований текст:", result)
```



Результат:

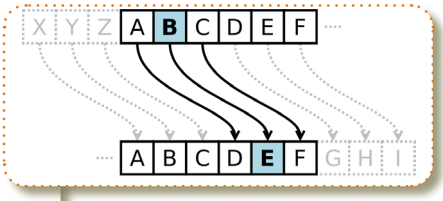
```
Зашифрований текст: KHOOR
```

Алгоритм:

1. Кожну літеру переводимо в число через **ord()**.
2. До цього числа додаємо фіксоване значення (наприклад, +3).
3. Отримуємо нове число й перетворюємо його назад у символ через **chr()**.

Розшифрування (декодування): щоб отримати початковий текст, потрібно зробити все навпаки — відняти ключ.

```
cipher = "KHOOR"
key = 3
original = ""

for i in cipher:
    new_code = ord(i) - key
    new_ch = chr(new_code)
    original += new_ch

print("Розшифрований текст:", original)
```



Результат:

```
Розшифрований текст: HELLO
```

Приклад у **Python**, *використовуючи рядок-алфавіт:*

```
alphabet = "АБВГҐДЕЄЖЗИІЇЙКЛМНОПРСТУФХЦЧШЩЬЮЯ"
text = input("Введіть текст для шифрування:").upper()
key = int(input("Введіть ключ (зсув):"))
result = ""

for ch in text:
    if ch in alphabet:
        index = alphabet.index(ch)
        new_index = (index + key) % len(alphabet)
        result += alphabet[new_index]
    else:
         result += i  # залишаємо пробіли чи розділові знаки
без змін

print("Зашифрований текст:", result)
```

Результат:

```
Введіть текст для шифрування: Вітаю
Введіть ключ (зсув): 3
Зашифрований текст: ДКЧҐБ
```

Практична робота №43

Завдання 1. Задано рядок. Подайте кожен символ рядка у вигляді коду таблиці **Unicode**.

Завдання 2. Створіть програму, яка запитує в користувача:

- текст для шифрування;
- ключ (зсув);

і виводить зашифрований текст.

Примітка: використовуючи таблицю **Unicode**, додатково зробіть меню:

```
1 – Зашифрувати текст
2 – Розшифрувати текст
0 – Вихід
```

Користувач обирає дію, вводить текст і ключ — програма виконує потрібну операцію.

Завдання 3. Секретне листування

Попросіть однокласника/однокласницю придумати ключ і зашифрувати для вас фразу.

Спробуйте розшифрувати її вручну або за допомогою програми.

(*Наприклад*, ключ = 4, текст = «СЬОГОДНІ ГАРНИЙ ДЕНЬ»).

Ключові поняття

Повідомлення з параметрами
Способи створення повідомлень з параметрами

Повідомлення з параметрами

У реальному житті майже всі сучасні програми формують повідомлення, які залежать від даних користувача:

- повідомлення про успішну оплату;
- електронний квиток, де вказано ім'я та час;
- листи-вітання: «Привіт, Олександре! 🎉 Ми раді бачити тебе знову!»;
- повідомлення у чаті: «Марія надіслала вам фото»;
- квитанції, сертифікати, запрошення, шкільні звіти — усюди є змінна інформація.

Такі повідомлення не пишуть вручну, а формують автоматично — вставляють змінні (параметри) в шаблон тексту.

Текстове повідомлення з параметрами (Параметризоване повідомлення) — це текст, у який програма підставляє певні значення (параметри).

Параметри — це «заповнювачі», які замінюються реальними даними (ім'ям, числом тощо). Це розширення маніпуляцій з рядками.

Наприклад:
```
"Вітаємо, [ім'я]! Ви отримали [кількість] балів!"
```

Якщо підставити:
```
ім'я = Олександр
кількість = 98
```

Отримаємо:
```
"Вітаємо, Олександр! Ви отримали 98 балів!"
```

Це і є параметризація тексту.

Параметризовані повідомлення роблять програми «дружніми» та розумними. Це наче створювати персоналізовані листівки: шаблон один, а ім'я та дата змінюються. Без параметрів мали б конкатенувати рядки **+ str(число)**, що незручно для складних текстів!

Основні способи створення текстів із параметрами

Python дозволяє робити це кількома способами.
Розглянемо поступово — від найпростішого до найзручнішого.

Спосіб 1. Конкатенація (об'єднання рядків)

```
name = "Олена"
score = 95
print("Вітаємо, " + name + "! Ви отримали " + str(score) +
"балів.")
```

- Знак + з'єднує рядки.
- Але потрібно перетворювати числа на текст **(str(score))**, тому цей метод не завжди зручний.

Спосіб 2. Форматування через f-рядки **(f-strings)**
Це найзручніший сучасний спосіб у **Python**.

```
name = "Олена"
score = 95
print(f"Вітаємо, {name}! Ви отримали {score} балів.")
```

- У рядку перед лапками ставимо літеру **f**.
- Усередині фігурних дужок **{ }** можна писати змінні або навіть вирази:

```
price = 120
tax = 0.2
print(f"До сплати: {price + price * tax} грн")
# До сплати: 144.0 грн
```

Спосіб 3. Метод **.format()**
Більш «класичний» спосіб, який часто використовують у старших версіях **Python**.

```
name = "Ігор"
subject = "математика"
print("Привіт, {0}! Твоя улюблена дисципліна – {1}.".format(name,
subject))
```

або коротше (заповнення автоматично, за порядком):

```
print("Привіт, {}! Твоя улюблена дисципліна –
{}.".format(name, subject))
```

Пояснення: `.format{name, subject}` — вказує порядок заповнення змінними місць, виділених фігурними дужками

Спосіб 4. Вирівнювання та формат чисел

Іноді потрібно зробити гарне табличне виведення:

```
name = "Марія"
grade = 11
avg = 10.389
print(f"Учениця: {name:10} | Клас: {grade:2} | Середній бал:
{avg.2f}")
```

Результат: `Учениця: Марія    | Клас: 11 |  Середній бал: 10.39`

Пояснення:

- `:10` — ширина поля (вирівняти у 10 символів);
- `:.2f` — число з двома знаками після коми.

Використання умов усередині шаблону

Можна навіть робити умовні повідомлення:

```
name = "Дмитро"
score = 80
if score >= 60:
    result = "склав"
else:
    result = "не склав"
print(f"{name} {result} тест із результатом {score} балів.")
```

Результат: `Дмитро склав текст із результатом 80 балів.`

Пам'ятай: якщо у програмі багато текстів, краще створювати шаблони, а не писати все вручну. Це економить час, зменшує кількість помилок і робить програму більш гнучкою.

Практична робота №44

Завдання 1. Створіть програму, яка вітає користувача/користувачку та використовує його/її особисті дані.

Програма повинна:

- дізнатися у користувача/користувачки:
  - його/її ім'я;
  - місто;
  - улюблене заняття (хобі);
- вивести повідомлення у вигляді:

```
Привіт, [ім'я] з міста [місто]! Ми знаємо, що ти любиш [хобі]. Це чудово!
```

Завдання 2. Створіть програму, яка формує квитанцію покупця.

Програма повинна:

- запитати у користувача:
  - ім'я покупця;
  - назву товару;
  - ціну одного товару;
  - кількість одиниць;
- обчислити загальну суму `(total = price * count)`;
- вивести повідомлення такого вигляду:

```
Покупець: [ім'я]
Товар: [назва]
Кількість: [кількість]
Сума до сплати: [сума] грн
```

Додатково:

- Додайте автоматичне вітання: `print(f"Вітаємо, {buyer}! Ви зробили чудову покупку!")`
- Обчисліть знижку 10% і відобразіть нову суму.
- Виведіть квитанцію у вигляді чека, з лініями розділення, наприклад:

```
------------------------------
КВИТАНЦІЯ
Покупець: Андрій
Товар: Підручник з інформатики
Кількість: 2
Сума до сплати: 640 грн
------------------------------
Дякуємо за покупку!
```

Завдання 3. Створіть програму «Гороскоп». Програма повинна згенерувати текст у вигляді:

`"Сьогодні [day], тобі пощастить у [luck]!"` з `random.choice` для `luck ["грі", "навчанні", "дружбі"]`.

Практична робота №45

Тема: «Розумний перевірник паролів»
Мета:

- Навчитися працювати з рядками та методами обробки тексту в **Python**.
- Ознайомитися з основами алгоритмів перевірки безпеки паролів.
- Розвивати критичне мислення та розуміння принципів кібербезпеки.

Завдання: створіть свій мінідодаток «Розумний перевірник паролів», який допоможе користувачу оцінити надійність введеного пароля.

Програма аналізує склад пароля (довжину, наявність цифр, літер, великих літер, символів) і дає оцінку його складності:

«легкий», «середній», «складний» або «дуже надійний».

Пояснення принципу роботи

1. Користувач вводить пароль.
2. Програма аналізує його:
   - довжину;
   - наявність цифр **(isdigit())**;
   - великих і малих літер **(isupper(), islower())**;
   - спеціальних символів (наприклад, @, #, !, % тощо).
3. Виводить повідомлення про рівень складності:

🔴 Дуже слабкий
🟡 Середній
🟢 Складний
🔵 Дуже складний
○ Невизначений

1. Створіть запит для введення паролю користувачем:

```
password = input("Введіть пароль:")
```

2. Створіть змінні-прапорці, які відповідатимуть за ініціалізацію існування маленьких літер, великих літер, символів тощо. За замовчуванням — **False** (немає таких).

```
# Ініціалізація змінних-прапорців
has_upper = False
has_lower = False
has_digit = False
has_symbol = False
```

3. Зробіть перевірку кожного символу в паролі та визначіть наявність малих літер, великих літер, цифр та спеціальних символів.

Приклад: перевірка, чи є великі літери у паролі:

```
password = input("Введіть пароль:")
has_upper = False   # спочатку вважаємо, що великих літер немає
for ch in password:
    if ch.isupper():       # якщо символ — велика літера
        has_upper = True   # змінюємо прапорець
        break              # далі перевіряти немає сенсу

if has_upper:
    print("✓ Пароль містить великі літери")
else:
    print("✖ Пароль не містить великих літер")
```

Аналогічно записати код для інших типів символів (чи є малі літери **has_lower**, чи є цифри **has_digit**, чи є спеціальні символи **has_symbol**)

Примітка для перевірки спеціальних символів:

```
if not ch.isalnum():  # якщо не літера і не цифра
```

4. Зробіть аналіз складності пароля:
   - якщо довжина пароля менша за 6 символів — «Пароль занадто короткий!»;
   - якщо пароль має маленькі літери, але не має великих літер, цифр та інших символів — «Слабкий пароль (лише малі літери)»;
   - якщо пароль має малі літери і цифри, але не має великих літер та інших символів, або пароль має малі й великі літери, але не має цифр та інших символів — «Середній пароль (малі літери + цифри) або (малі літери + великі літери)»;
   - якщо пароль має малі літери, великі літери і цифри, але не має інших символів — «Складний пароль (великі, малі літери + цифри)»;
   - якщо пароль має ще й спеціальні символи — «Дуже складний пароль (містить символи)»;
   - інакше: «Незвичайний пароль».

```
if len(password) < 6:
    print("🔴 Пароль занадто короткий!")
elif has_lower and not (has_upper or has_digit or has_symbol):
    print("🟡 Слабкий пароль (лише малі літери)")
elif [запишіть умову самостійно]:
    print("🟠 Середній пароль (малі літери + цифри)")
elif [запишіть умову самостійно]:
    print("🟢 Складний пароль (великі, малі літери + цифри)")
elif [запишіть умову самостійно]:
    print("🔵 Дуже складний пароль (включає символи)")
else:
    print("⚪ Незвичайний пароль, перевірте ще раз")
```

Приклади роботи програми

Введіть пароль: **qwerty**
🟠 Слабкий пароль (лише малі літери)
Введіть пароль: **qwerty123**
🟡 Середній пароль (малі літери + цифри)
Введіть пароль: **Qwerty123**
🟢 Складний пароль (великі, малі літери + цифри)
Введіть пароль: **Qwerty123!**
🔵 Дуже складний пароль (включає символи)

Додаткові творчі умови:

- Вивести детальну інформацію про пароль.
- Вивести рекомендацію, як поліпшити пароль (додати великі літери, спеціальні символи тощо).

Приклад:
Введіть пароль: **Qwerty1234**
Перевірка...
➦ Довжина: 10 символів
➦ Містить великі літери: ✔️
➦ Містить малі літери: ✔️
➦ Містить цифри: ✔️
➦ Містить символи: ❌
Ваш пароль — Складний (великі, малі літери + цифри) 🔒

Порада: додайте хоча б один спеціальний символ, щоб зробити пароль надійнішим!

Ключові поняття

Програмний проєкт
Ідея проєкту
Вибір теми проєкту

Програмний проєкт

Програмний проєкт — це повна програма, яку створюють від ідеї до готового продукту, використовуючи все вивчене: змінні, умови, цикли, функції, списки, рядки, обробку тексту тощо.

Це не просто код, а розв'язок проблеми або розвага, як гра чи додаток.

Проєктування важливе, бо розвиває креативність (придумуєте самі), вчить планувати (наче гра в шахи — думайте наперед), показує, як програмування допомагає в житті (наприклад, калькулятор для кишенькових грошей). Проєкти роблять вас «справжніми» програмістами — як у компаніях **Google**, де починають з ідей.

Що таке ідея проєкту?

Кожен великий винахід починається з маленької ідеї.

Із маленької думки («А що, якщо зробити гру про космос?») виростає велика програма, яка розв'язує проблему, розважає чи допомагає.

Це робить програмування не просто уроками, а справжньою пригодою: ви стаєте творцями, як у грі Minecraft, де створюєте свій світ.

Ідея — це задум, головна думка, навколо якої будується весь проєкт.

У програмуванні ідея проєкту — це коротка відповідь на запитання: «Що моя програма має робити і навіщо?»

Приклад:

- Якщо часто забуваєте виконати домашнє завдання, ідея може бути: «Створити застосунок, який нагадує про завдання зі школи».

- Якщо захоплюєтеся спортом — «Створити програму, яка підраховує кількість віджимань або кроків».

- Якщо любите читати — «Створити електронну бібліотеку з улюбленими авторами»

Ідея — це ще не код і навіть не готовий план, а лише насіння майбутнього проєкту.

Без неї не можна почати нічого створювати, бо саме вона визначає напрямок роботи.

Звідки беруться ідеї?

Ідеї нас оточують всюди. Треба лише навчитися їх помічати.

Найкращі ідеї з'являються тоді, коли ми помічаємо проблему або потребу, яку хочемо розв'язати.

Приклади джерел ідей

Повсякденне життя

- «Було б добре мати будильник, який вмикає улюблену пісню».
- «Хотілося б додаток, який нагадує пити воду».

Школа

- «Програма, яка допомагає тренуватись із математики».
- «Електронний словник англійських слів».

Хобі та інтереси

- «Трекер тренувань для гравця у футбол».
- «Симулятор акваріуму або догляду за тваринами».

Навколишнє середовище

- «Програма для сортування сміття».
- «Калькулятор витрат води або електроенергії».

Сім'я та друзі

- «Електронний календар сімейних свят».
- «Застосунок для вибору подарунків друзям».

Як вибрати тему для свого програмного проєкту?

Вибір теми — це один із найважливіших етапів.
Тема має бути цікавою, реальною і посильною для виконання.

Кроки вибору

Список ідей: запишіть 5-10 ідей.

Оцініть: за шкалою 1-5: цікавість (мені подобається?); складність (що можу зробити з вивченим?); корисність (допоможе мені/друзям?); час (встигну?).

Порівняйте: оберіть із найвищим балом; якщо рівно — оберіть найпростішу для старту.

Адаптуйте: якщо ідея велика — зробіть мініверсію (наприклад, не повний чат, а генератор відповідей).

Критерії вибору гарної теми

№	Критерії	Що це означає?	Приклад
1	Цікавість	Тобі справді хочеться цим займатись	Гра або сайт про улюблений вид спорту
2	Користь	Програма має розв'язувати якусь проблему або приносити користь	Щоденник для пакування домашнього завдання
3	Реалістичність	Її можна створити з тими знаннями, які вже маєш	Проста база даних, чат, словник
4	Оригінальність	Це щось нове або зроблене по-іншому	Гра, де потрібно тренувати логіку
5	Можливість розвитку	Проєкт можна розширювати далі	Спочатку проста програма, потім додати графіку чи вебверсію

Як народжується ідея проєкту?

1 Спостереження

Розгляньтесь довкола. Що вас дратує або що можна зробити зручнішим?

Приклад: у шкільному розкладі часто плутаються предмети → ідея створити «Розумний розклад».

2 Обговорення з іншими

Іноді найкраща ідея народжується після короткої розмови з однокласниками чи однокласницями.

Можна провести «мозковий штурм» у групі:

- Кожен пропонує свої ідеї.
- Усі обговорюють, що цікавіше і реальніше зробити.

3 Запис ідей

Усі ідеї потрібно записувати, навіть якщо вони здаються дивними.

Через деякий час саме така ідея може стати геніальною.

Як правильно обговорювати ідеї в групі

Коли проєкт роблять у групі, важливо навчитися висловлювати свої думки і слухати.

Поради для обговорення

1. Не критикуйте одразу чужі ідеї — краще спробуйте їх доповнити.
2. Кожна особа має право висловитися.
3. Обирайте ту ідею, яка найкраще поєднує інтереси всіх.
4. Не забувайте про реалістичність — не все, що звучить круто, легко зробити.

Приклад:

У групі троє учнів:

- Софія хоче зробити сайт про книжки.
- Денис — про фільми.
- Іван — про ігри.

Після обговорення вони вирішують створити спільний сайт «Медіагід: книги, фільми, ігри, які варті уваги» — усі задоволені, ідея стала цікавішою.

Як сформулювати ідею проєкту

Ідею потрібно записати у короткому реченні:

«Моя програма допомагає людям _____ , щоб _____».

Приклади:

- «Моя програма допомагає учням повторювати таблицю множення, щоб краще запам'ятати її».
- «Моя програма допомагає гравцям рахувати результати матчів».
- «Моя програма допомагає людям вести список прочитаних книжок».

Щоб не забути, що саме хочете зробити, варто створити паспорт ідеї проєкту:

Поле	Приклад
Назва проєкту	«Моя спортивна база»
Призначення	Зберігати інформацію про спортсменів/спортсменок
Кому корисно	Усім особам, котрі цікавляться спортом
Основні функції	Додавання, пошук і зміна даних
Очікуваний результат	Проста програма з меню на **Python**

Ідея програмного проєкту — це перший крок до створення справжньої програми.

Вона виникає з нашого досвіду, спостережень і бажання зробити світ трішки зручнішим.

Що чіткіше сформульована ідея — то легше потім її реалізувати.

Приклади можливих ідей шкільних проєктів

Напрямок	Приклад проєкту
Освіта	Програма-тренажер з математики або англійської
Хобі	Щоденник тренувань або фітнес-трекер
Культура	Віртуальна галерея художників
Ігри	Логічна гра «Вгадай слово» або «Лабіринт»
Суспільство	Електронний волонтерський календар подій
Особисте життя	Планувальник справ або трекер настрою
Довкілля	Програма, яка нагадує вимикати світло
Творчість	Генератор віршів або музичних ритмів

Практична робота №46

Завдання. «Моя ідея програмного проєкту».

Придумайте три ідеї програмних проєктів, які ви могли б створити. Для кожної ідеї коротко запишіть:

- Назву проєкту
- Проблему, яку він розв'язує
- Кому буде корисний
- Основні функції

Виберіть одну найкращу ідею і запишіть її у формі: «Мій проєкт — це програма, яка допомагає _____, щоб _____».

За бажанням — намалюйте або опишіть, який вигляд матиме інтерфейс програми (наприклад, головне меню, кнопки, кольори).

Додаткове завдання (для груп)

1. Об'єднайтесь у команди по 3–4 особи.
2. Кожен запропонує свою ідею.
3. Обговоріть усі й оберіть одну спільну.
4. Підготуйте коротку презентацію ідеї (2–3 слайди або усно перед класом).

Ключові поняття

Пріоритезація
Принцип Парето
Матриця Ейзенхауера
Метод "З'їж жабу"

Що таке пріоритезація?

У житті ми постійно стикаємося з вибором:
- Що зробити спочатку?
- Яке завдання головне, а яке може почекати?

Пріоритезація — це розташування завдань залежно від їхньої важливості, терміновості та складності в певному порядку (за пріоритетом).

Приклад із життя: ви прийшли зі школи і маєте зробити чимало: виконати домашнє завдання з математики, вигуляти собаку, пограти у гру, допомогти мамі.

Якщо спробувати зробити все одночасно — не вийде. Тому треба обрати пріоритет — тобто що зробити спочатку.

У програмуванні теж є пріоритезація.

Коли створюєте програмний проєкт, маєте десятки ідей:

- додати меню;
- зробити гарний інтерфейс;
- створити вхід користувача;
- зробити гру веселішою…

Але все одразу — неможливо.

Тож потрібно вирішити: «Що обов'язково потрібно зробити зараз, щоб програма працювала, а що можна додати потім, як покращення?»

Принцип Парето (правило 80/20)

Принцип Парето (або правило 80/20) — це ідея, що 80% результатів забезпечують 20% зусиль.

Суть принципу: 20% зусиль приносять 80% результату. А решта 80% зусиль — лише 20% результату.

Його відкрив італійський економіст Вільфредо Парето в 1896 році, спостерігаючи за горохом у саду: 20% стручків давали 80% гороху!

Потім він помітив, що 80% багатства Італії належало лише 20% населення.

Пізніше цей принцип почали застосовувати всюди — у бізнесі, навчанні, роботі, навіть у побуті.

Простими словами: не всі справи однаково важливі. Деякі приносять основну користь, інші — лише додаткову. Тому треба зосередитись на тих діях, які дають найбільший ефект.

Приклади:

Ситуація	20% дій	80% результату
Підготовка до контрольної	Повторив головні формули і задачі з підручника	Написав контрольну на «10»
Прибирання кімнати	Зібрав одяг і сміття	Кімната має чистий вигляд
Навчання програмування	Зрозумів основні команди Python	Уже можеш створювати прості програми
Підготовка проєкту	Спочатку створив базові функції	Програма працює, навіть без прикрас

Як застосовувати принцип Парето на практиці

1. **Складіть список усіх своїх справ або ідей.**
Наприклад: зробити дизайн, додати кнопки, створити меню, перевірити помилки, зробити авторизацію.

2. **Оцініть кожен пункт за важливістю.**
– Що дає найбільший результат?
– Що потрібно, аби програма взагалі працювала?

3. **Виділіть ті 20% завдань, без яких програма не має сенсу.**
Наприклад: «зберегти дані», «показати результати», «працювати без помилок». Почніть саме з них. А решту залиште на потім — коли буде час і бажання.

Приклад: у вас є ідея проєкту — «Моя бібліотека».
Ви хочете:

- зробити логотип;
- додати музику на фоні;
- створити форму входу;
- зробити пошук книг;
- вивести список книг;
- зберегти дані у файл.

Згідно з принципом Парето, 20% найважливішого:

- створити список книг;
- реалізувати пошук;
- додати можливість додавати нову книгу.

А решта (фон, логотип, кольори) — потім.

Інші способи пріоритезації

Іноді просто правила 80/20 недостатньо.
Тоді використовують інші методи — теж прості й зручні.

1. Метод «Eisenhower Matrix» (Матриця Ейзенхауера)

Це класичний метод, який ділить завдання на чотири категорії:

- **Важливі та термінові:** робіть негайно.
- **Важливі, але нетермінові**: плануйте виконання.
- **Неважливі, але термінові**: відкладіть або делегуйте.
- **Неважливі та нетермінові**: відкладіть або видаліть.

Цей метод допомагає не просто вибрати, що головне, а ще й правильно розподілити час.

Приклад:

Важливість / Терміновість	Приклад	Що робити
Важливе і термінове	Завтра контрольна — треба повторити	Зроби це зараз!
Важливе, але не термінове	Почати готуватись до олімпіади	Сплануй, коли зробиш
Неважливе, але термінове	Хтось кличе у гру, але треба вчитись	Відклади або делегуй
Неважливе і нетермінове	Безцільно гортати TikTok	Не роби взагалі

2. Метод «З'їж жабу»

Метод «З'їж жабу» (Eat That Frog) — це популярна техніка тайм-менеджменту, яка допомагає боротися з прокрастинацією та ефективно планувати свій день.

Фраза походить від вислову Браяна Трейсі: «Якщо зранку першою справою ви з'їсте живу жабу, то решта дня мине чудово — адже найгірше вже позаду».

Ідея методу: «Жаба» — це ваша найважливіша, але найнеприємніша справа.

Те завдання, яке ви постійно відкладаєте:

- важке;
- потребує зосередженості;
- має найбільше значення для результату.

Суть методу: зробіть цю справу першою — і день піде легко.

Алгоритм за методом «З'їж жабу»

- Складіть список справ на день або тиждень.
- Позначте найважливішу (або найскладнішу) справу — це ваша «жаба».
- Почніть день саме з неї, не відкладаючи.
- Після того, як "з'їсте жабу», виконайте решту завдань — вони здадуться простішими.

Приклад

- «Жаба» — це написати складне есе, підготувати презентацію або зробити проєкт.
- Після цього легше виконати дрібніші завдання: знайти картинки, оформити титулку тощо.

Порада: якщо маєте кілька «жаб», почніть із найбільшої і найслизькішої (тобто з тієї, яка принесе найбільше користі, якщо її зробити).

Поради для розумного планування

- Не намагайтеся зробити все одразу — краще трохи, але якісно.
- Якщо завдання велике — розбийте його на менші кроки.
- Починайте з найважчого — тоді далі буде легше.
- Не забувайте: іноді «менше — це більше».

Способи пріоритезації — це не просто правила, це спосіб мислення. Якщо навчитеся визначати головне, то:

- економитимете час;
- працюватимете ефективніше;
- створюватимете кращі проєкти.

Практична робота №47

Завдання 1. «Визначення пріоритетів мого проєкту».

1. Запишіть всі ідеї або функції, які плануєте додати до свого проєкту. (Мінімум 6–8 пунктів.)

Приклад:

- Головне меню
- Вхід користувача
- Пошук
- Список результатів
- Кнопка «Вихід»
- Музика
- Фон
- Логотип

2. Застосуйте принцип Парето.

↳ Визначте, які 20% функцій дадуть 80% користі.

↳ Виділіть їх окремо як основні.

Опишіть свої висновки:

– Що найважливіше у вашому проєкті?

– Що можна реалізувати пізніше?

– Як це допоможе працювати швидше?

Додаткове завдання (для груп):

- Порівняйте свої таблиці в групі.
- Обговоріть, чи збігаються у вас «топові» функції.
- Спробуйте спільно скласти «командний список пріоритетів».

Завдання 2. «Мій день у чотирьох квадратах»

1. Складіть список із 10–12 своїх справ (навчальних, домашніх, особистих).

Наприклад:

- вивчити параграф з інформатики;
- допомогти вдома;
- перевірити **Instagram**;
- зробити презентацію для уроку;
- погуляти з друзями.

2. Розподіліть їх у матрицю Ейзенгауера (намалюйте таблицю 2×2).

3. Позначте кольорами:

🔴 термінові важливі;

🟡 важливі, але нетермінові;

🔵 термінові, але неважливі;

○ неважливі й нетермінові.

4. Зробіть висновок: які справи займають найбільше часу, але не дають результату?
Додатково: створіть цю таблицю в **Google Таблицях** або **Excel**.

Завдання 3. «Алгоритм дій за матрицею»
Для кожного квадранта складіть короткий план дій, наприклад:
1. «Зроби негайно» → що саме треба зробити сьогодні.
2. «Плануй» → коли це зробиш.
3. «Делегуй» → кому можна передати.
4. «Видали» → чому не варто витрачати час.

Завдання 4. «Знайди свою жабу»

1. Перегляньте список завдань із попереднього пункту.
2. Визначте, яке завдання є вашою «жабою» — тобто:
   - важке;
   - вимагає зосередженості;
   - приносить найбільшу користь після виконання.
3. Напишіть короткий план:
   - Що саме потрібно зробити?
   - Коли почнете?
   - Які кроки допоможуть «з'їсти жабу»?
   - Яку винагороду собі дасте після виконання?

Ключові поняття

Структура проєкту
Модулі в проєкті
Етапи підготовки проєкту

Що таке структура проєкту?

Коли ми створюємо програму — навіть просту — вона складається не лише з коду. У неї є файли, дані, ресурси, зображення, коментарі тощо.

Ваш проєкт — це як будинок: ідея — це мрія про дім, пріоритезація — вибір, що будувати спочатку (фундамент чи дах), а структура проєкту — це план, як розмістити кімнати, двері та вікна, щоб будинок був зручним і міцним.

Структура проєкту — це спосіб організувати код на частини (модулі), як розділи в книзі, щоб програма була зрозумілою, легкою для змін і без «хаосу».

Що таке структура проєкту?

Уявіть, що зводите будинок LEGO:

- Є інструкція (план) — це наш головний файл **main.py**.
- Є окремі деталі — це модулі, функції, зображення, тексти.
- Є сховище деталей — це папка з ресурсами (**images**, **sounds**, **data**).
- Є етапи складання — це різні частини коду, які працюють разом.

Якщо все складено у правильному порядку — будинок стійкий.

Якщо ні — усе розвалюється.

Так відбувається і в програмуванні: коли код структурований, його легше розуміти, виправляти й розширювати.

Типова структура програмного проєкту

МІЙ ПРОЄКТ/

- main.py — головний файл програми, з якого все запускається
- helpers.py — модуль з допоміжними функціями
- data/ — вхідні або текстові дані
  - users.txt
- images/ — графічні ресурси (логотип, зображення)
  - logo.png
- tests/ — тестові файли для перевірки роботи програми
  - test_data.txt
- README.txt — короткий опис проєкту та інструкція користувача

Що таке модуль?

Модуль — це файл із розширенням **.py**, що містить змінні, функції або класи, які можна використовувати в інших програмах.

Модулі дозволяють:

- повторно використовувати код, не копіюючи його;
- розділяти роботу між учасниками команди;
- спрощувати читання і налагодження програми.

Тобто модуль — це спосіб розділити код на частини, щоб він був зручніший і зрозуміліший.

Види модулів

1. Вбудовані (стандартні): **`math`**, **`random`**, **`datetime`**, **`tkinter`**, **`turtle`**, ...
2. Власні: створені вами (наприклад, **`helpers.py`**, **`logic.py`**)
3. Зовнішні: завантажуються через **pip** (наприклад, **`requests`**, **`pandas`**, **`pygame`**)

Приклад:

Файл my_module.py:

```
def add(a, b):
    return a + b

def multiply(a, b):
    return a * b

def square(x):
    return x * x

name = 'Oleksandr'
country = 'Ukraine'
```

Файл main.py:

```
import my_module

number = 4

print("Квадрат числа:", my_module.square(number)) # 16
print("Сума чисел:", my_module.add(5, 3))       # 8
print("Добуток чисел:", my_module.multiply(4, 2)) # 8

print(f"My name {my_module.name}. I live in {my_module.country}")
# My name Oleksandr. I live in Ukraine
```

Як підключати модулі

Спосіб	Приклад	Пояснення
import module	import math	імпортує цілий модуль
from module import func	from math import sqrt	імпортує лише конкретну функцію
from module import *	from math import *	імпортує всі функції з даного модуля
import module as m	import math as m	дає коротку назву (псевдонім) модуля

Навіщо ділити програму на модулі?

Припустимо, ваша гра або проєкт має понад 500 рядків коду.

Якщо все міститиметься в одному файлі — знайти помилку або додати функцію буде складно.

А коли логіка збережена в одному модулі, графіка — в іншому, обробка даних — у третьому, тоді все зрозуміло, чітко та легко оновлювати.

Це і є принцип структурного програмування — коли програма поділена на логічні блоки.

Основні модулі типового навчального проєкту

Назва модуля	Призначення	Приклад
main.py	головний файл, який запускає програму	головне меню, старт гри
data.py	робота з файлами, даними, текстами	збереження результатів, зчитування бази даних
graphics.py	графіка, оформлення, візуалізація	малювання фігур у **Turtle**
logic.py	основні функції або алгоритми	перевірка правильності, підрахунок балів
settings.py	налаштування програми	кольори, розміри, константи

Етапи підготовки проєкту

1 **Вибір теми**
Визначте, що саме буде створюватися: гра, довідник, тест, симулятор, навчальний застосунок тощо.

2 **Обговорення мети**
Поясніть, для чого потрібен цей проєкт.
Яку користь він приносить користувачам або суспільству.

3 **Визначення функцій**
Опишіть, які дії повинна виконувати програма.
Наприклад: зберігати дані, рахувати, малювати тощо.

4 **Складання плану**
Продумайте етапи розробки і взаємозв'язки між частинами.
Що слід зробити спочатку, а що — пізніше.

5 **Розробка структури папок**
Створіть зрозумілу структуру.

6 **Створення базових файлів**
Підготуйте основу проєкту: **main.py**, **data.py**, **menu.py** — щоб почати з «каркаса».

7 **Покрокова розробка**
Розробляйте поетапно: спочатку — базові функції, потім — деталі та інтерфейс.

8 **Тестування**
Перевірте, чи все працює правильно: знайдіть помилки, протестуйте усі можливі сценарії.

9 **Документація**
Створіть короткий опис проєкту: як користуватися, що він робить, хто розробник.

Порада: гарний проєкт — це не лише код, а й структура, логіка та презентація!

Практична робота №48

Завдання 1. Планування проєкту

Визначте тему свого майбутнього проєкту. Це може бути:

- навчальний застосунок (тест, довідник, тренажер);
- гра;
- мініпрограма для підрахунків або органайзер.

Кроки:

1. Оберіть назву проєкту.
2. Сформулюйте його мету (що робитиме, для кого створений).
3. Опишіть 3–5 основних функцій програми.
4. Намалюйте (або напишіть) просту блок-схему структури проєкту.

Приклад:

Тема — «Калькулятор середнього бала».

Мета — допомогти учням швидко підраховувати свій середній бал за предметами.

Функції — введення оцінок, розрахунок середнього, збереження результату у файл.

Модулі — `main.py, calc.py, data.py, ui.py`.

Завдання 2. Створення структури папок

Створіть у своєму робочому каталозі базову структуру проєкту:

```
Мій_Проєкт/
│
├── main.py
├── data/
│   └── data.txt
├── modules/
│   ├── logic.py
│   └── helper.py
└── docs/
    └── readme.txt
```

Кроки:

1. Створіть відповідні папки у своїй робочій директорії.
2. У кожному файлі додайте коментар з описом його ролі.

`# main.py` — головний модуль, що керує програмою

`# logic.py` — тут буде основний алгоритм

`# helper.py` — допоміжні функції

Завдання 3. Підготовка базових модулів

Заповніть файли початковим кодом, щоб програма мала основу.

Приклад:

main.py

```
from modules.logic import start_program

print("🚀 Запуск програми!")
start_program()
```

modules/logic.py

```
def start_program():
    print("Програма працює!")
```

data/data.txt

```
(тут зберігатимуться дані користувачів)
```

Ключові поняття

Бібліотеки в Python
Бібліотеки math, random, time, turtle, tkinter

Що таке бібліотека?

Бібліотека — це набір готових функцій, команд і модулів, які розширюють можливості **Python**.

Наприклад:

- хочеш, щоб програма вибирала випадкове число — використай **random**;
- хочеш намалювати щось на екрані — є **turtle**;
- хочеш створити вікно, кнопки, текстові поля — допоможе **tkinter**;
- хочеш, щоб програма рахувала час або робила паузу — є **time**;
- хочеш використовувати математичні функції — є **math**.

Python має величезну стандартну бібліотеку, яка вже встановлена — достатньо лише її під'єднати.

Як під'єднати бібліотеку

Під'єднання виконується командою **import:**

```
import random
import time
```

Можна також під'єднати лише окремі частини:

```
from random import randint
from math import sqt
```

Можна також під'єднати цілу бібліотеку: `from turtle import*`

Бібліотека random — випадковість у програмі

Бібліотека **random** допомагає створювати випадкові події. Її часто використовують в іграх, тестах, симуляторах.

Основні функції random

Функції	Призначення	Приклад	Результат
`random()`	випадкове число	`random.random()`	0.4837
`randint(a,b)`	випадкове ціле число від `a` до `b`	`random.randint(1,6)`	4

`choice(список)`	випадковий елемент зі списку	`random.choice(["яблуко", "груша", "вишня"])`	`"груша"`
`shuffle(список)`	перемішує елементи списку	`random.shuffle(cards)`	список змінюється

Приклад: кидання кубика

```
import random

print("Кидаємо кубик...")
number = random.randint(1, 6)
print("Випало:", number)
```

Результат:
Кидаємо кубик...
Випало: 5

Бібліотека turtle — малювання за допомогою «черепашки»

turtle — це одна з найцікавіших бібліотек для початківців. Вона дозволяє малювати графіку за допомогою віртуальної «черепашки», яка рухається по екрану.

Черепашка — це маленький курсор, який залишає слід, коли рухається.

Рух черепашки

Команда	Опис
`t.forward(100)` / `t.fd(100)`	рух вперед на 100 пікселів
`t.backward(50)` / `t.bk(50)`	рух назад на 50 пікселів
`t.right(90)` / `t.rt(90)`	повернути праворуч на 90°
`t.left(45)` / `t.lt(45)`	повернути ліворуч на 45°
`t.goto(x, y)`	перемістити черепашку до координат `(x, y)`
`t.setx(x)` / `t.sety(y)`	встановити координату `x` або `y`
`t.setheading(angle)`	встановити напрямок (0° — вправо)

Контроль перо / пензель

Команда	Опис
`t.penup()` / `t.pu()`	підняти перо (не малює при русі)
`t.pendown()` / `t.pd()`	опустити перо (малює при русі)
`t.pensize(3)` / `t.width(3)`	товщина лінії
`t.pencolor("red")`	колір лінії
`t.fillcolor("yellow")`	колір заливки
`t.begin_fill()`	почати заливку
`t.end_fill()`	закінчити заливку

Малювання форм

Команда	Приклад
`t.circle(50)`	коло радіусом 50
`t.dot(10, "blue")`	цятка діаметром 10, синя
`t.stamp()`	залишає відбиток черепашки на полотні

Інші корисні команди

Команда	Опис
`t.speed(1)`	швидкість черепашки (1–10, 0 — миттєво)
`t.hideturtle()`	сховати черепашку
`t.showturtle()`	показати черепашку
`screen.bgcolor("lightblue")`	колір фону
`t.clear()`	очистити все малювання
`t.reset()`	очистити та повернути черепашку в центр
`screen.exitonclick()`	закрити вікно, клацнувши мишею

Приклад 1: малюємо квадрат

```
import turtle

t = turtle.Turtle()
t.color("blue")
t.pensize(3)

for i in range(4):
    t.forward(100)
    t.right(90)

turtle.done()
```

Приклад 2: зірка

```
import turtle

t = turtle.Turtle()
t.color("purple")
t.pensize(2)

for i in range(5):
    t.forward(150)
    t.right(144)

turtle.done()
```

Бібліотека time — робота з часом

Бібліотека **time** дозволяє керувати часом у програмі:

- робити паузи;
- вимірювати швидкість виконання;
- виводити поточний час.

Основні функції time

Функції	Призначення	Приклад
`sleep(x)`	пауза на x секунд	`time.sleep(2)`
`time()`	кількість секунд від 1970 року	`time.time()`
`ctime()`	поточний час у зручному форматі	`time.ctime()`

Приклад: імітація зворотного відліку

```
import time

for i in range(5, 0, -1):
    print(i)
    time.sleep(1)

print("Старт!")
```

Бібліотека tkinter — створення вікон і кнопок

tkinter — це стандартна бібліотека для створення віконних програм у **Python**.

З її допомогою можна зробити: кнопки, поля введення, написи, навіть прості графічні ігри.

Проста програма з вікном


```
from tkinter import *

window = Tk()
window.title("Моя перша програма")
window.geometry("300x200")

label = Label(window, text="Привіт, світе!", font=("Arial", 14))
label.pack(pady=20)

window.mainloop()
```

Програма з кнопкою

```
from tkinter import *

def say_hello():
    label.config(text="Привіт, програмісте!")

window = Tk()
window.title("Привітання")

label = Label(window, text="Натисни кнопку")
label.pack(pady=10)

button = Button(window, text="Привіт!", command=say_hello)
button.pack()

window.mainloop()
```

Взаємодія бібліотек

Бібліотеки можна поєднувати між собою.
Наприклад:

- у грі — **random** для випадкових подій;
- в **time** для затримки рухів;
- в **turtle** для візуалізації.

```
import turtle, random, time

t = turtle.Turtle()
t.color("green")

for i in range(5):
    t.forward(random.randint(50, 150))
    t.right(random.randint (45, 135))
    time.sleep(1)

turtle.done()
```

Черепашка рухається випадковими кроками, роблячи паузи між ними.

Практична робота №49

Завдання 1. Випадкове число

Створіть програму, яка запитує ім'я користувача й видає йому «щасливе число дня».

Завдання 2. Зворотний відлік

Програма запитує користувача, скільки секунд почекати, і робить зворотний відлік до нуля. Після цього виводить **Час минув!**.

Завдання 3. Малюємо фігуру

Черепашка малює фігуру зі сторін, кількість яких задає користувач. Кольори вибираються випадково.

Підказка: `random.choice(["red", "blue", "green"])`.

Завдання 4. Віконне привітання

Створіть вікно за допомогою **tkinter**, де буде:

- напис **Натисни кнопку**;
- кнопка **Привіт!**;
- після натискання змінюється текст на **Гарного дня, програмісте!**

Ключові поняття

Віджет Text
Віджет Listbox
Віджет Combobox

Основи Tkinter

Tkinter — це бібліотека **Python** для створення вікон, кнопок, текстових полів, списків та інших елементів.

Tk()	створює головне вікно
title()	назва вікна
geometry()	розмір вікна
mainloop()	запускає цикл програми, щоб вона працювала

```
import tkinter as tk

root = tk.Tk()
root.title("Моя програма")
root.geometry("400x300")
root.mainloop()
```

Ви уже знаєте такі віджети, як **Label**, **Entry**, **RadioButton**, **Checkbutton**, **Scale**. Проте існують й інші віджети, які дозволяють працювати зі списками та великими текстами (рядками).

Віджет Text

Text — це багаторядкове текстове поле, яке дозволяє користувачеві вводити, редагувати і відображати текст.

На відміну від **Entry**, який працює лише з одним рядком, **Text** дозволяє працювати з довгими текстами, абзацами та форматуванням.

Синтаксис:

```
text_widget = Text(parent, width=40, height=10)
text_widget.pack()
```

Пояснення параметрів

parent — батьківський елемент, зазвичай це вікно **(Tk())** або фрейм **(Frame)**
width — ширина текстового поля в символах
height — висота текстового поля в рядках
font — шрифт тексту, наприклад: font=(**Arial**, 12)
bg — колір фону
fg — колір тексту
wrap — перенесення рядка (NONE, CHAR, WORD)

Основні методи Text

Метод	Призначення	Приклад
get(start, end)	Отримати текст між позиціями	content=text_widget.get"1.0", END)
delete(start, end)	Видалити текст між позиціями	text_widget.delete("1.0", END)
insert(index, string)	Вставити текст у задану позицію	text_widget.insert("1.0", "Привіт!")
replace(start, and, string)	Замінити текст між позиціями	text_widget.replace("1.0", "1.5", "Hello"],
see(index)	Прокрутити до певної позиції	text_widget.see(END)
edit_ubdo()	Скасувати останню дію	text_widget.edit_undo()
edit_redo()	Повторити скасовану дію	text_widget.edit_redo()

Індексація в Text:

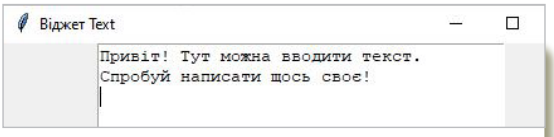
- Формат: «рядок.стовпець»
  - `"1.0"` — перший рядок, перший символ
  - `"2.5"` — другий рядок, шостий символ
- **END** — кінець тексту

Приклади використання:

```
text_widget.insert("1.0", "Початок тексту\n")
text_widget.insert(END, "Новий рядок")
content = text_widget.get("1.0", END)
text_widget.delete("1.0", END)
```

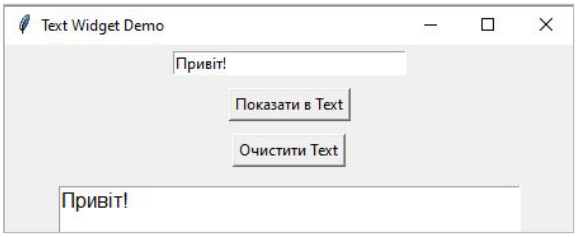
Віджет Text

Простий приклад:




```
import tkinter as tk

root = tk.Tk()
root.title("Віджет Text")

text_widget = tk.Text(root, height=5, width=40)
text_widget.pack()

text_widget.insert(tk.END, "Привіт! Тут можна вводити текст.\n
Спробуй написати щось своє!")

root.mainloop()
```

Програма, яка вводить текст, показує його у **Text** та очищає:




```
from tkinter import *

def show_text():
    content = entry.get()
    text_widget.delete("1.0", END)  # очищає Text
    text_widget.insert(INSERT, content)
def clear_text():
    text_widget.delete("1.0", END)

root = Tk()
root.title("Text Widget Demo")
root.geometry("400x300")

entry = Entry(root, width=30)
entry.pack(pady=5)

Button(root, text="Показати в Text", command=show_text).
pack(pady=5)
Button(root, text="Очистити Text", command=clear_text).
pack(pady=5)

text_widget = Text(root, width=40, height=10, font=("Arial", 12))
text_widget.pack(pady=10)

root.mainloop()
```

Застосування для рядків:

Можна зберігати введений текст у змінну:

`user_text = text_widget.get("1.0", tk.END)` # від першого рядка до кінця.

Віджет Listbox

Listbox — це віджет, який відображає список елементів у вигляді вертикального списку.

Користувач може:

- обирати один або кілька елементів;
- додавати та видаляти елементи програмно;
- прокручувати список за допомогою **Scrollbar**.

Синтаксис:

```
listbox = Listbox(parent, width=30, height=10,
selectmode=SINGLE)
listbox.pack()
```

Пояснення параметрів:

selectmode — режим виділення елементів:

SINGLE — один елемент;

BROWSE — схоже на **SINGLE**, але можна перетягувати виділення мишею;

MULTIPLE — кілька елементів можна виділяти клацанням;

EXTENDED — виділення за допомогою **Shift**/**Ctrl**.

Основні методи Listbox

Метод	Призначення	Приклад
`insert(index, element)`	Додадає елемент у список. index може бути 0, END або конкретний номер	`listbox.insert(END, "Mango")`
`delete(start, and=None)`	Видаляє елемент або діапазон елементів	`listbox.delete(0)` або `listbox.delete(0, END)`
`get(start, end=None)`	Отримати елементи списку	`items=listbox.get(0, END)`
`curselection()`	Повертає індекси виділених елементів (tuple)	`selected=listbox.curselection()`
`size()`	Повертає кількість елементів у списку	`n=listbox.size()`
`selection_set(start, end=None)`	Програмно виділити елемент/елементи	`listbox.selection_set(0)`
`selection_clear(start, end=None)`	Зняти виділення	`listbox.selection_clear(0, END)`

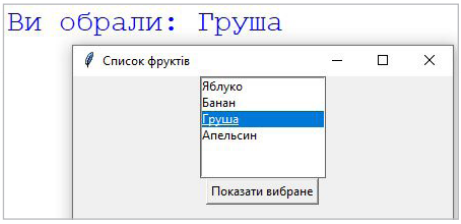
Індексація в Listbox:

- Індекси починаються з 0
- 0 — перший елемент
- **END** — останній елемент

Приклади використання:

```
# Додати елементи
listbox.insert(END, "Яблуко")
listbox.insert(END, "Банан")

# Отримати всі елементи
items = listbox.get(0, END)

# Виділити перший елемент
listbox.selection_set(0)

# Видалити виділений елемент
selected = listbox.curselection()
if selected:
    listbox.delete(selected[0])
```

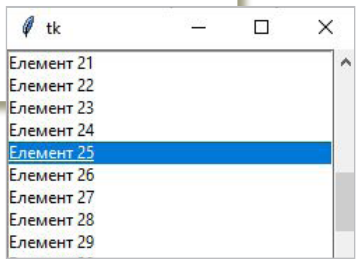
Простий приклад:




```
import tkinter as tk

root = tk.Tk()
root.title("Список фруктів")

fruits = ["Яблуко", "Банан", "Груша", "Апельсин"]
listbox = tk.Listbox(root, height=6)
listbox.pack()

for fruit in fruits:
    listbox.insert(tk.END, fruit)  # додаємо елементи у список

def show_selection():

    selection = listbox.get(listbox.curselection())
    print(f"Ви обрали: {selection}")

button = tk.Button(root, text="Показати вибране", command=show_
selection)

button.pack()
root.mainloop()
```

Використання **Scrollbar** з **Listbox**

```
import tkinter as tk

root = tk.Tk()

scrollbar = tk.Scrollbar(root)
scrollbar.pack(side=tk.RIGHT, fill=tk.Y)

listbox = tk.Listbox(root, yscrollcommand=scrollbar.set)
listbox.pack(side=tk.LEFT, fill=tk.BOTH, expand=True)

for i in range(50):
    listbox.insert(tk.END, f"Елемент {i+1}")

scrollbar.config(command=listbox.yview)

root.mainloop()
```



Застосування для списків:

Можна динамічно додавати і видаляти елементи списку.

Поради:

- Для списків з великою кількістю елементів завжди використовуйте **Scrollbar**.
- Можна комбінувати **Listbox** з іншими віджетами (**Button**, **Entry**) для динамічного додавання/видалення елементів.
- **Listbox** застосовуйте для простих списків, а для складних таблиць краще обирайте **Treeview** з **ttk**.

Віджет Combobox (ttk)

Combobox — комбінований список, де користувач може вибрати значення зі списку або ввести своє.

Синтаксис:

```
from tkinter import Tk
from tkinter import ttk

combobox = ttk.Combobox(parent, values=["Яблуко",
"Банан", "Вишня"], width=20)
combobox.pack()
```

Пояснення параметрів

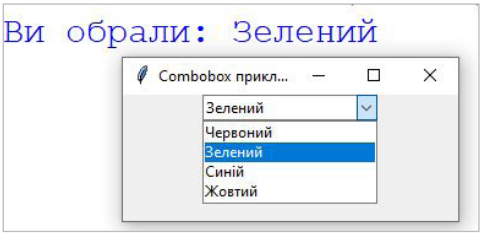
values — список елементів, які можна вибрати;
state — стан віджета:
"**readonly**" — можна лише вибирати зі списку, не редагуючи тексту;
"**normal**" — можна редагувати текст вручну;
"**disabled**" — віджет неактивний.

Основні методи Combobox

Метод	Призначення	Приклад
`get()`	Отримати обране значення	`selected=combobox.get()`
`set(value)`	Встановити значення комбобоксу	`combobox.set("Банан")`
`current(index)`	Виділити елемент за індексом	`combobox.current(0)`
`['values']`	Отримати або змінити список елементів	`combobox['values']= ["Апельсин", "Груша"]`
`bind("<<Combobox-Selected>>", callback)`	Викликати функцію при зміні вибору	`combobox.bind()`

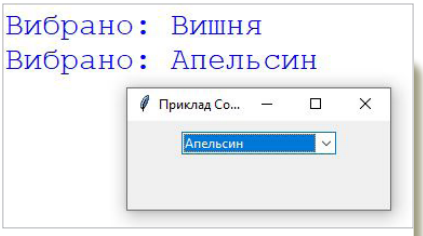
Віджет Combobox (ttk)

Простий приклад:




```
import tkinter as tk
from tkinter import ttk

root = tk.Tk()
root.title("Combobox приклад")

colors = ["Червоний", "Зелений", "Синій", "Жовтий"]

combo = ttk.Combobox(root, values=colors)
combo.pack()
combo.set("Оберіть колір")  # текст за замовчуванням

def show_color():
    print(f"Ви обрали: {combo.get()}")

button = tk.Button(root, text="Показати вибране", command=show_
color)

button.pack()

root.mainloop()
```

Приклад зі зміною функції




```
import tkinter as tk
from tkinter import ttk

def show_selection(event):
    print("Вибрано:", combobox.get())

root = tk.Tk()
root.title("Приклад Combobox")

fruits = ["Яблуко", "Банан", "Вишня", "Груша", "Апельсин"]

combobox = ttk.Combobox(root, values=fruits, width=20,
state="readonly")
combobox.pack(pady=10)
combobox.current(0)  # Виділити перший елемент

combobox.bind("<<ComboboxSelected>>", show_selection)

root.mainloop()
```

Практична робота №50

Завдання 1. Введення тексту

- Створіть вікно за допомогою **Tkinter**.
- Додайте віджет **Text**, куди користувач вводитиме інформацію:
  - Ім'я
  - Улюблене хобі
- Додайте кнопку **Привітати**
- При натисканні кнопки зчитуйте текст із віджета **Text**.
- Очистіть пробіли на початку та в кінці за допомогою `.strip()`.
- Виведіть у консоль повідомлення:

`Привіт, <ім'я>! Ми знаємо, що ти любиш <хобі>.`

Використайте методи рядків: `.strip()` для очищення пробілів.
Приклад функції:

```
def greet():
    text = text_widget.get("1.0", tk.END)  # отримати весь текст
    lines = text.split("\n")               # розділити на рядки
    if len(lines) >= 2:
        name = lines[0].strip()
        hobby = lines[1].strip()
        print(f"Привіт, {name}! Ми знаємо, що ти любиш {hobby}.")
    else:
        print("Будь ласка, введіть ім'я та хобі на двох рядках!")
```

Завдання 2. Список покупок

- Створіть **Listbox** із 5 товарами.
- Кнопка **Показати вибране** виводить у консоль назву виділеного продукту.
- Додатково: додати кнопку `"Очистити виділення"`.

Завдання 3. Вибір з **Combobox**

- Створіть **Combobox** з назвами міст.
- Кнопка **Показати місто** виводить вибране місто у консоль.
- Використати методи рядків `.upper()` або `.lower()` перед виведенням.

Ключові поняття

Тестування програми
Типи помилок
Обробка помилок try-except

Що таке тестування програми?

Тестування — це процес перевірки, чи працює програма правильно і стабільно.

Уявіть ситуацію: ви створили гру чи калькулятор. Тестування допомагає знайти:

- помилки в обчисленнях;
- ситуації, коли користувач вводить неправильні дані;
- небажану поведінку програми.

Підлітки часто тестують ігри або додатки на телефоні, щоб переконатися, що вони працюють без «глюків».

Типові помилки при введенні даних

Приклад: користувач повинен ввести число, але вводить текст.

```
age = int(input("Введіть свій вік: "))
```

Якщо ввести «дванадцять», програма зламається і покаже помилку:

```
Введіть свій вік: дванадцять
ValueError: invalid literal for int() with base: ' дванадцять'
```

Щоб уникнути цього, використовується **try-except**.

Обробка помилок: try-except

Синтаксис:

```
try:
    # спробувати виконати код
except ТипПомилки:
    # що робити, якщо сталася помилка
```

Приклад: перевірка числа.

```
try:
    age = int(input("Введіть свій вік: "))
    print(f"Вітаю! Тобі {age} років.")
except ValueError:
    print("Помилка! Будь ласка, введіть число.")
```

```
Введіть свій вік: дванадцять
Помилка! Будь ласка, введіть число.
```

Пояснення:

- **try** — намагається виконати код.
- Якщо сталася помилка **ValueError** (неправильне введення числа), виконується блок **except**.
- Програма не зламається, а виведе дружнє повідомлення користувачу.

Приклад: нескінченний цикл з перевіркою введення.

```
while True:
    try:
        num = int(input("Введіть число від 1 до 10: "))
        if 1 <= num <= 10:
            print(f"Дякую, ви ввели число {num}")
            break
        else:
            print("Число повинно бути від 1 до 10")
    except ValueError:
        print("Помилка! Введіть саме число")
```

Цей підхід дозволяє захистити програму від некоректних дій користувача.

Перевірка логіки та умов

При тестуванні треба перевіряти:

- Межі введення: числа мінімальні та максимальні.
- Неправильне введення: текст, порожній рядок, спеціальні символи.
- Нестандартні ситуації: нуль, негативні числа, великі числа.

Приклад: перевірка оцінки учня.

```
while True:
    try:
        mark = int(input("Введіть оцінку (1-12): "))
        if 1 <= mark <= 12:
            print(f"Оцінка {mark} записана!")
            break
        else:
            print("Оцінка повинна бути від 1 до 12")
    except ValueError:
        print("Помилка! Введіть число від 1 до 12")
```

Інші типи помилок

Помилка	Приклад
ZeroDivisionError	Ділення на нуль (10/0)
IndexError	Вихід за межі списку
NameError	Використання неіснуючої змінної
TypeError	Виконання операцій над несумісними типами
AttributeError	Виклик неіснуючого методу чи атрибута об’єкта
ImportError	import nonexist — спроба під’єднати неіснуючу бібліотеку
ModuleNotFoundError	Модуль не знайдено
FileNotFoundError	Файл не знайдено

Практична робота №51

Завдання 1. Вік користувача

- Напишіть програму, яка запитує вік користувача.
- Перевірте, чи введено число є від 5 до 100.
- Якщо користувач введе текст або число поза межами, програма повинна попросити повторити введення.

Завдання 2. Калькулятор

- Створіть програму, яка запитує у користувача **два числа** та обчислює їхню суму.
- Використайте **try-except**, щоб обробити помилки введення (наприклад, якщо введено текст замість числа).

Завдання 3. Ділення чисел

- Програма повинна запитувати дільник і ділене.
- Використайте **try-except** для обробки **ZeroDivisionError**.
- Якщо дільник = 0, вивести дружнє повідомлення: **На нуль ділити не можна!**.

Завдання 4. Перевірка списку

- Створіть список `fruits = ["яблуко", "банан", "груша"]`.
- Запитайте користувача, який фрукт він хоче вибрати.
- Використайте **try-except** для обробки **IndexError** або **ValueError** (наприклад, якщо користувач введе індекс, що не існує).

Завдання 5. Перевірка користувацьких даних

- Створіть програму, яка запитує:
  - Вік користувача
  - Оцінку за тест

Кроки:

1. Перевірте, що введено саме **число**.
2. Перевірте, що вік — **від 5 до 120**, а оцінка — **від 0 до 12**.
3. Якщо введено неправильне значення — повідомте користувача та запропонуйте спробувати ще раз.

Підказка: використайте цикл **while True** разом із **try-except.**

Добір тестових даних. Таблиця тестування. Оцінювання правильності: логіка, помилки, гіпотези

Теми 52-53

Ключові поняття

Помилка
Тестові дані
Логіка програм
Види помилок
Гіпотези при тестуванні програм

Коли ми пишемо програму, навіть якщо код здається правильним, завжди можуть з'явитися помилки.

Помилка — це ситуація, коли програма працює не так, як очікувалося.

Мета тестування:

- Перевірити, чи працює програма правильно для різних даних.
- Знайти слабкі місця програми.
- Запобігти збоям або некоректній роботі.

Приклад із реального життя:

Якщо робите мобільний додаток для калькулятора:

- Введення: 2 + 3 → очікуваний результат: 5 ✓
- Введення: 0 / 0 → очікуваний результат: помилка ×
- Введення: 123456789 * 987654321 → очікуваний результат: 121932631112635269 ✓

Тестування допомагає переконатися, що програма працює правильно в усіх випадках.

Добір тестових даних

Тестові дані — це конкретні вхідні значення, які ми даємо програмі для перевірки її роботи.

Види тестових даних

Коректні дані — програма має правильно обробити.

Некоректні дані — програма повинна вміти повідомити про помилку або не зламатися.

Крайові значення (граничні) — дуже маленькі або великі числа, порожні рядки.

Чому важливо підбирати різні дані:

- Користувачі поводяться по-різному.
- Програма може працювати для одних випадків, а для інших – «завалитися».

Приклад: Нехай маємо додаток для замовлення піци

№	Вхідні дані	Очікуваний результат	Коментар
1	піца Маргарита, 2	Сума = 200 грн	стандартне замовлення
2	піца Пепероні, 0	Повідомлення: “Введіть кількість >0”	перевірка некоректного введення
3	порожнє поле	Повідомлення: “Введіть назву піци”	граничний випадок

Таблиця тестування

Таблиця тестування — це зручний спосіб записати всі можливі варіанти введення та очікувані результати.

Навіщо потрібна таблиця:

- Легко перевірити програму на всіх випадках.
- Зручно для групового проєкту або на уроках.
- Можна швидко помітити, що програма пропускає якісь дані.

Приклад таблиці:

№	Вхід	Тип даних	Очікуваний результат	Реальний результат	Коментар
1	5 + 3	коректні	8	8	правильно
2	0 / 0	некоректні	помилка	помилка	правильно
3	-10 + 20	граничні	10	10	правильно
4	123456789 * 987654321	граничні	121932631112635269	?	перевірка великих чисел

Правила добору тестових даних

1. Враховувати всі типи даних (числа, рядки, від’ємні значення).
2. Обирати крайові значення – мінімум, максимум, нуль.
3. Перевіряти граничні та некоректні дані.
4. Використовувати як ручний, так і автоматичний контроль (через програму).

Логіка програми

Логіка програми – це спосіб, яким програма обробляє дані та приймає рішення, щоб отримати очікуваний результат.

Приклад із життя:

У комп'ютерній грі герой отримує 10 очок за кожен переможений рівень.

Якщо гравець проходить рівень: `score = score + 10`

Логіка програми: перевірка перемоги → додавання очок → оновлення рахунку.

Чому логіка важлива

Навіть якщо код синтаксично правильний, неправильна логіка призведе до неправильних результатів.

Наприклад, помилково віднімати очки замість додавання:

`score = score - 10`.

Види помилок

Синтаксичні (SyntaxError)

Порушення правил мови програмування.

Приклад: забута дужка, пропущена двокрапка.

```
print("Hello"  # SyntaxError
```

Логічні (Logical Error)

Код працює, але результат неправильний через помилку в алгоритмі.

Приклад: неправильно обчислені відсотки:

```
total = 100
part = 30
percent = total / part  # □ має бути part / total * 100
```

Помилки виконання (Runtime Error)

Помилки, які виникають під час роботи програми.

Приклад: ділення на нуль, робота з неіснуючим файлом.

```
x = 5 / 0  # ZeroDivisionError
```

Помилки введення (Input Error)

Користувач вводить неправильні дані.
Приклад: текст замість числа.

```
age = int(input("Введіть вік: "))   #
якщо введено "abc" → ValueError
```

Гіпотези при тестуванні програм

Гіпотеза – припущення, що відображає очікувану поведінку програми для конкретного набору даних.

Приклад із життя:
Ти міркуєш: «Якщо я введу число 5 у калькулятор, то програма покаже 25 після множення на 5».
Це гіпотеза, яку можна перевірити.

Чому важливо будувати гіпотези

- Допомагає структуровано перевіряти програму.
- Дозволяє шукати логічні помилки ще до тестування.
- Полегшує створення таблиць тестування.

Практичні роботи №52-53

Завдання 1. Калькулятор

1. Створіть програму, яка додає два числа, введені користувачем.
2. Складіть таблицю тестування для 5 різних варіантів:
   - два позитивні числа
   - одне від'ємне
   - два нулі
   - ділення на нуль
   - дуже велике число
3. Протестуйте програму і запишіть реальні результати у таблицю.
4. Складіть гіпотези для різних входів.
5. Виконайте програму і заповніть таблицю:

Вхідні дані	Гіпотеза	Реальний результат	Тип помилки	Коментар

Завдання 2. Додаток «Мій рейтинг книг»

1. Програма: користувач вводить назву книги й оцінку від 1 до 10.
2. Складіть таблицю тестування для:
   - правильного введення (назва + оцінка 7)
   - оцінки менш ніж 1
   - оцінки більш ніж 10
   - порожнього поля назви
3. Перевірте роботу програми на всіх випадках.
4. Складіть гіпотези.
5. Виконайте програму, заповніть таблицю та визначте, де потрібне виправлення логіки.

Завдання 3.

- Оберіть будь-який свій маленький проєкт (ігровий, спортивний, бібліотечний).
- Створіть 5-10 тестових випадків.
- Складіть таблицю тестування та перевірте програму.
- Складіть 5-10 гіпотез для різних входів.
- Виконайте тестування, знайдіть помилки і виправте їх.

Ключові поняття

Документація
Коментарі
Структурування коду

Що таке документація

Документація програми — це опис того, що робить програма, як вона працює і як її можна використовувати.

Приклад із життя: уявіть, що ви зібрали новий LEGO-набір. Без інструкції складно зрозуміти, де яка деталь і як вона з'єднується.

Так і з програмою: якщо в ній немає опису, навіть її автор чи авторка через місяць може забути, що саме малося на увазі.

Для чого потрібна документація:

- Іншим програмістам — щоб швидко зрозуміти, як працює код.
- Самому собі — щоб не забути логіки через певний час.
- Користувачу/користувачці — щоб знати, як працювати з програмою.

Коментарі у програмі

Коментар — це рядок у програмі, який не виконується, а лише пояснює роботу коду.

Як їх записують у **Python**:

```
# Це коментар, він не впливає на виконання програми
print("Привіт!")  # Це теж коментар — після команди
```

Види коментарів:

Однорядкові коментарі — починаються з #

```
# Обчислення площі прямокутника
a = 5
b = 10
s = a * b
print(s)
```

Багаторядкові коментарі

Багаторядкові коментарі — зручно використовувати для пояснення великих частин коду або як документацію функцій.

```
"""
Програма: Обчислення площі прямокутника
Автор: Софія К.
Дата: 28.10.2025
Опис: користувач вводить сторони, а програма обчислює площу
"""
```

Документування функцій

Коли ми створюємо функції, варто пояснювати, що вони роблять.

Для цього в **Python** є спеціальний коментар — **docstring** (документуючий рядок).

Приклад:

```
def add_numbers(a, b):
    """
    Функція для додавання двох чисел.
    Параметри:
      a — перше число
      b — друге число
    Повертає:
      суму чисел a та b
    """
    return a + b

print(add_numbers(5, 7))
```

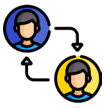
Якщо викликати `help(add_numbers)`, **Python** навіть покаже цей опис у вікні довідки!

```
12
Help on function add_numbers in module __main__:

add_numbers(a, b)
    Функція для додавання двох чисел.
    Параметри
       a – перше число
       b — друге число
    Повертає:
      суму чисел a та b
```

Структурування коду

Добре структурований код:

легко читати



зручно змінювати

не плутає того, хто його перевіряє

Основні правила структурування

1 Розділяйте код на логічні блоки

Наприклад:

- блок введення даних
- блок обчислень
- блок виведення результату

```
# 1. Ввід даних
name = input("Введіть своє ім'я:")

# 2. Обробка
message = f"Привіт, {name}!"

# 3. Виведення
print(message)
```

2 Залишайте порожні рядки між частинами програми для кращої читабельності

3 Надавайте змінним змістовні назви

✖ `x = 100` ✓ `speed_kmh = 100`

4 Використовуйте коментарі над важливими частинами коду, щоб пояснити логіку

Документація проєкту

Коли ви створюєте великий проєкт, важливо мати файл із описом програми (наприклад, **readme.txt** або r**eadme.md**).

У ньому можна зазначити:

- Назву проєкту
- Авторську групу
- Мету програми
- Які модулі використовуються
- Як запустити програму
- Приклади використання

Приклади з життя

Сфера	Документація
Книга	Передмова, зміст, пояснення автора/авторки
Техніка	Інструкція користувача
Гра	Меню допомоги, правила, опис керування
Програмування	Коментарі, README, інструкції

Документація — це мова спілкування програмістської спільноти зі світом.

Без неї навіть найкращий код — як мапа без підписів.

Приклад повної програми з документацією

```
"""
Програма: Калькулятор середнього бала
Автор(-ка): Марко І.
Дата: 28.10.2025
Опис: користувач вводить оцінки, а програма обчислює середнє значення
"""

# 1. Ввід даних
grades = input("Введіть оцінки через пробіл: ")

# 2. Обробка
grades_list = grades.split()
total = 0
for grade in grades_list:
    total += int(grade)
average = total / len(grades_list)

# 3. Виведення результату
print(f"Середній бал: {average:.2f}")
```

Практична робота №54

Завдання 1. «Моя перша документована програма»

1. Створіть будь-яку просту програму (наприклад, обчислення площі фігури).
2. Додайте:
   - Докстрінг з описом програми (назва, автор, мета).
   - Коментарі до кожного логічного блоку.
3. Перевірте, щоб код був охайно відформатований.

Завдання 2. «Коментатор коду»

У вас є код без коментарів:

```
a = int(input("Введіть перше число:"))
b = int(input("Введіть друге число:"))
if b != 0:
    print(a / b)
else:
    print("На нуль ділити не можна!")
```

Напишіть до нього документований варіант із поясненнями кожного рядка.

Завдання 3. «**README** до мого проєкту»

1. Виберіть один із своїх проєктів (наприклад, гру або тест).
2. Створіть короткий файл **readme.txt**, де опишіть:
   - Назву проєкту
   - Мету
   - Які модулі використані
   - Як запустити програму

Завдання 4 (вищий рівень).

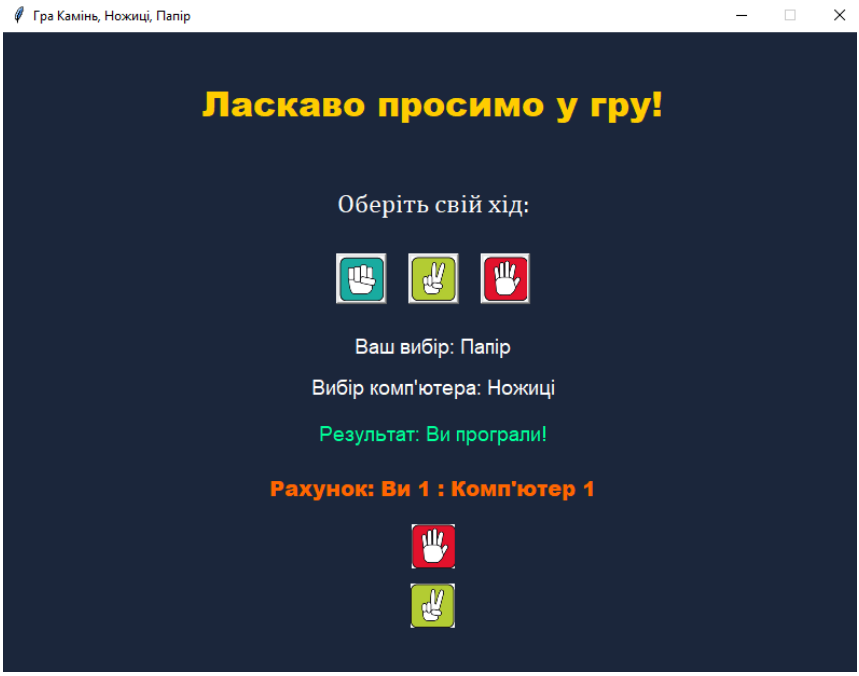
Візьміть більший проєкт (наприклад, гру або графічну програму з **Turtle** чи **Tkinter**).

1. Створіть **README.txt** за шаблоном.
2. Розділіть код на модулі — наприклад:
   - **main.py** — головна логіка;
   - **functions.py** — функції.
3. Додайте зрозумілі коментарі в кожному файлі.

1 етап. Організаційний

Завдання: Розробити програму з графічним інтерфейсом, яка дозволяє користувачеві:

- Обрати свій хід (Камінь, Ножиці або Папір).
- Порівняти свій вибір з комп'ютером і дізнатися результат (виграш, програш або нічия).
- Динамічно відображати рахунок гри.



Мета програми:

- Створити стартове вікно з привітанням та кнопками для вибору ходу.
- Показувати вибір гравця і комп'ютера у вигляді тексту та зображень.
- Відображати результат кожного раунду і оновлювати рахунок.
- Забезпечити зручний та інтуїтивно зрозумілий інтерфейс.

2 етап. Підготовчий

Мета програми:

Стартове вікно:

- Привітання користувача.
- Мітка з інструкцією.
- Кнопки для вибору ходу.

Вікно гри:

- Показ вибору гравця та комп'ютера.
- Зображення вибору.
- Мітка результату.
- Мітка рахунку.

Зображення:

Потрібно підготувати **png**-файли для кожного ходу (зображення однакового розміру):

- **rock.png** – камінь
- **scissors.png** – ножиці
- **paper.png** – папір

Масштабування через **subsample** для зручного відображення.

Збереження зображень до проєкту.

В інтернеті знайдіть тематичні зображення каменя, ножиць та паперу. Для цього в пошуковому рядку введіть ключові слова, віднайдіть потрібні зображення та збережіть їх у папці з проєктом і перейменуйте їх зручним іменем.


paper.png


rock.png


scissors.png

Зауваження! Візьміть до уваги, що розширення має бути **png** або **gif**. Для зручності — зберігайте зображення, роблячи скриншоти потрібного зображення, а також з прозорим фоном.

3 етап. Проєктний

Структура інтерфейсу:

Вікно гри:

- Колір фону: темний `(#1b263b)`.
- Мітки для відображення вибору гравця і комп'ютера.
- Мітка для результату раунду `(#00ff99)`.
- Мітка для рахунку `(#ff6600)`.
- Кнопки з картинками Камінь, Ножиці, Папір.

4 етап. 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.
- модуль **random** для випадкового вибору зі списку

```
from tkinter import *
from tkinter import messagebox
import random
```

Створення ігрового вікна:

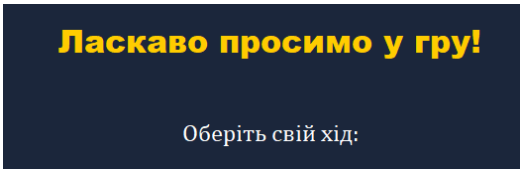
1. Задання розмірів вікна (`"700x650"`).
2. Колір фону: темний (`#1b263b`).
3. Фіксація розмірів (вимкнення масштабування).
4. Встановлення заголовка: `"Гра Камінь, Ножиці, Папір"`.
5. Запустити головний цикл **mainloop()**

Створити мітку для привітання **label_greeting** з такими властивостями: текст: `"Ласкаво просимо у гру!"`; шрифт: `Arial Black`, `24`; колір фону `"#1b263b"` і колір тексту `"#ffcc00"`.

```
label_greeting.pack(pady=40)
```

Створити мітку для інформації вибору **label_choose** з такими властивостями: текст: `"Оберіть свій хід:"`; шрифт: `"Cambria"`, `18`; колір фону `"#1b263b"` і колір тексту `"#ffffff"`.

```
label_choose.pack(pady=10)
```



Створити фрейм для розміщення кнопок із зображенням Каменя, Ножиць і Паперу за допомогою методу **grid**

```
button_frame = Frame(start, bg="#1b263b")
button_frame.pack(pady=20)
```

Імпортувати зображення для використання в кнопках і написах

```
rock_img = PhotoImage(file="rock.png").subsample(2, 2)
paper_img = PhotoImage(file="paper.png").subsample(2, 2)
scissors_img = PhotoImage(file="scissors.png").subsample(2, 2)
```

Зауваження! Масштабування зображення встановлюєте самостійно, залежно від розмірів вашого зображення.

Створити кнопки **rock_button**, **scissors_button**, **paper_button**, які будуть заповнені зображеннями та викликати функцію **play()** самої гри.

```
rock_button = Button(button_frame, image=rock_
img, command=lambda: play("Камінь"))
rock_button.grid(row=0, column=0, padx=10)

[scissors_button (ваш код)]
scissors_button.grid(row=0, column=1, padx=10)

[paper_button (ваш код)]
paper_button.grid(row=0, column=2, padx=10)
```

Вгорі вікна з кодом створити функцію **play()** та тимчасово заглушити її командою **print()**

```
from tkinter import *
from tkinter import messagebox
import random

# Функція гри
def play(player_choice):
    print()
```

Створити мітки **player_label**, **computer_label** з такими властивостями: текст: `"Ваш вибір: "` та `"Вибір комп'ютера: "` відповідно; шрифт: `'Arial'`, 14; колір фону `"#1b263b"` і колір тексту `fg="#ffffff"`.

```
player_label.pack(pady=5)
computer_label.pack(pady=5)
```

Створити мітку **result_label** з такими властивостями: текст: `"Результат: "`; шрифт: `'Arial'`, 14; колір фону `"#1b263b"` і колір тексту `fg="#00ff99"`.

```
result_label.pack(pady=10)
```

Створити мітку **score_label** з такими властивостями: текст: `"Рахунок: Ви 0 : Комп'ютер 0"`; шрифт: `'Arial Black'`, 14; колір фону `"#1b263b"` і колір тексту `fg="#ff6600"`.

```
score_label.pack(pady=10)
```

Додатково:

Створити мітки **player_img_label**, **computer_img_label**, які будуть заповнені зображеннями вибору гравця і комп'ютера.

```
player_img_label = Label(start, image=paper_img, bg="#1b263b")
player_img_label.pack(pady=5)

computer_img_label = Label(start, image=paper_img, bg="#1b263b")
computer_img_label.pack(pady=5)
```

Описати функцію **play()**, яка виконуватиме функції гри.

Спочатку ініціалізуємо дві змінні, які будуть відповідати за рахунок гравця і комп'ютера та список з елементами вибору:

```
# Ініціалізація рахунку
player_score = 0
computer_score = 0
elements = ["Камінь", "Ножиці", "Папір"]
```

Функція отримуватиме змінну **player_choice**, яку передає функція **lambda** при натисканні на кнопку, всередині функції задати глобальні змінні рахунків, які ініціалізували ззовні функції:

```
# Функція гри
def play(player_choice):
    global player_score, computer_score
```

Інші дії всередині функції:

- визначити вибір комп'ютера

```
computer_choice = random.choice(elements)
```

- описати порівняння вибору користувача та комп'ютера, а також визначити, хто переміг чи нічия. При перемозі — рахунок переможця збільшується на 1:

```
if player_choice == computer_choice:
        result = "Нічия!"
elif (player_choice == "Камінь" and computer_choice == "Ножиці") or \
             (player_choice == "Ножиці" and computer_choice == "Папір") or \
              (player_choice == "Папір" and computer_choice == "Камінь"):
        result = "Ви виграли!"
        player_score += 1
    else:
        result = "Ви програли!"
        computer_score += 1
```

вивести результати в мітки для результатів:

```
player_label.config(text=f"Ваш вибір: {player_choice}")
     computer_label.config(text=f"Вибір комп'ютера: {computer_
choice}")
    result_label.config(text=f"Результат: {result}")
       score_label.config(text=f"Рахунок: Ви {player_score} :
Комп'ютер {computer_score}")
```

5 етап. Тестувальний

Перевірка функціоналу:

- Кожен хід правильно порівнюється з комп'ютером.
- Рахунок оновлюється після кожного раунду.
- Результат відображається коректно (**Ви виграли**, **Ви програли**, **Нічия**).
- Зображення вибору відображаються правильно.
- Інтерфейс не змінює розміру і залишається зручним.

Додатково

6 етап. Оптимізація та розширення

Можливі розширення:

- Зміна зображень в мітках, щоб було видно вибір комп'ютера та гравця.
- Анімація для перемоги чи поразки.
- Збереження статистики гравця у файл.
- Зміна стилів кнопок і шрифтів для більш яскравого дизайну.

7 етап. Документація

Створити документацію у файлі **Readme.txt**

- `play(player_choice)` – обробляє вибір гравця, визначає результат, оновлює рахунок і зображення.
- Змінні **player_score** та **computer_score** зберігають поточний рахунок.
- Зображення зберігаються у словнику **images** для зручного виклику.
- Всі кольори підібрано для темного стилю гри та яскравих акцентів.

1 етап. Організаційний

Кожен із нас хоча б раз бачив телевізійну гру «Хто хоче стати мільйонером?».

Кожне з питань оцінюється у певну кількість грошей. За кожну правильну відповідь гравець отримує щораз більше балів, але якщо помиляється — гра завершується.

Завдання. Розробити програму з графічним інтерфейсом, яка дозволяє користувачеві:

- проходити гру-тест з питаннями різного рівня складності;
- отримувати гроші за правильні відповіді (0, 100, 500, 1000, 2000, 5000, …);
- створювати власні запитання у спеціальному вікні за допомогою віджетів **Text**, **Listbox**, **Combobox**;
- редагувати та видаляти запитання зі списку.

Програма повинна:

- рахувати віртуальні гроші (бали);
- динамічно відображати результат гри.

Мета програми

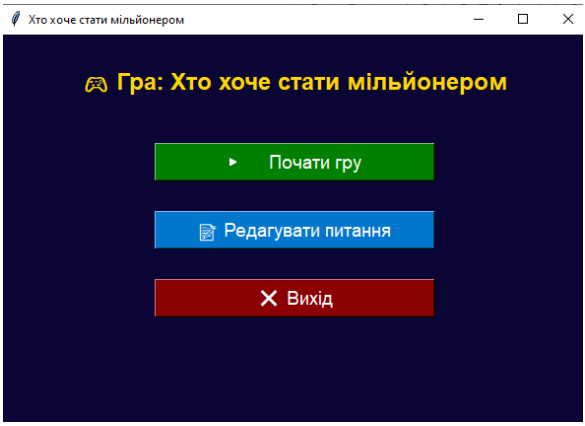
- Створити зручну інтерактивну гру «Хто хоче стати мільйонером?», з різними вікнами.
- Зробити можливість відповісти на серію запитань, кожне з яких оцінюється у певну кількість грошей.
- Продемонструвати використання **tkinter** і бібліотеки **random**.
- Навчитися працювати з віджетами **Text**, **Listbox**, **Button**, **Label**, **Combobox**.
- Закріпити знання з тем «Списки» і «Випадкові числа».

2 етап. Підготовчий

Структура проєкту

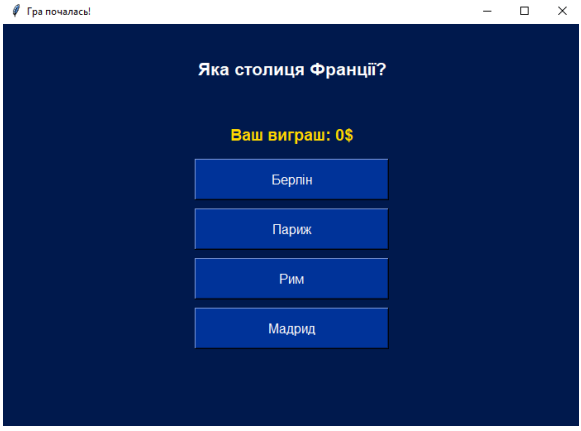
↳ Стартове вікно:

- Привітання гравця.
- Кнопка «Почати гру».
- Кнопка «Редагувати запитання».
- Кнопка «Вихід».



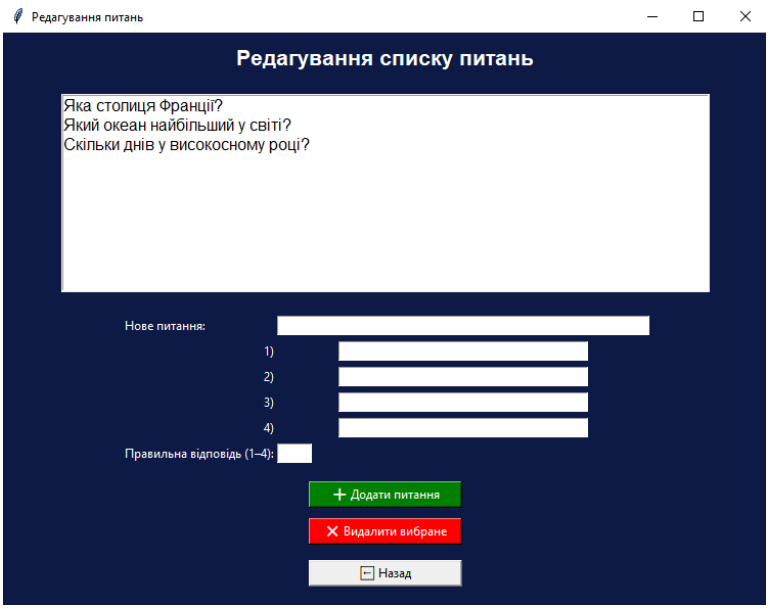
Вікно гри:

- Мітка для відображення поточного запитання.
- Мітка із сумою виграшу.
- Вибір варіантів відповіді.



Вікно редагування запитань:

- Мітка із заголовком
- **Listbox** для перегляду списку запитань.
- Поле **Text** для введення тексту запитання.
- Чотири поля **Entry** для варіантів відповіді.
- **Combobox** для вибору правильної відповіді.
- Кнопка **Додати запитання**.
- Кнопка **Видалити вибране запитання**.
- Кнопка **Назад**.



3 етап. Планування

Щоб гра працювала логічно, розділимо її на три частини.

Частина	Опис
1. Меню	Вибір дії: грати, редагувати або вийти
2. Редактор	Вікно з Text, Entry і Listbox для створення питань
3. Гра	Основна логіка вікторини, кнопки з варіантами відповідей

Основні змінні:

- **questions** — список усіх запитань (повноцінна гра має 15 запитань);
- **options** — варіанти відповідей (список у списку);
- **answers** — індекси правильних відповідей;
- **money_levels** — список грошових винагород.

Необхідні бібліотеки:

```
import tkinter as tk
from tkinter import messagebox
import random
```

4 етап. Проєктний

Структура інтерфейсу:

Стартове вікно:

- Розмір: 600x400 пікселів.
- Колір фону: темно-синій (#0a0535).
- Мітка привітання: текст жовтий (gold).
- Мітка інструкції: білий (#ffffff).
- Кнопки для вибору.

Вікно гри:

- Розмір: 700x500 пікселів.
- Колір фону: темно-синій (#00194d).
- Мітка запитання: білий (#ffffff).
- Мітка виграшу (gold).
- Кнопки з відповідями.

Вікно редагування запитань:

- Розмір: 750x600 пікселів.
- Колір фону: темно-синій (#00194d).
- Мітка заголовку: білий (#ffffff).
- Віджети **Listbox**, **Combobox**, **Entry**, **Button** — у фреймах
- Кнопки з відповідями.

5 етап. Реалізація коду

Створення інтерфейсу

Підключення модулів:

```
from tkinter import *
from tkinter import messagebox
import random
```

Задання початкових даних грі:

Список запитань. Кожен елемент — це окреме запитання.
Наприклад:

```
questions = [
    "Яка столиця Франції?",
    "Який океан найбільший у світі?",
    "Скільки днів у високосному році?",
    "Що потрібно комп'ютеру для роботи?",
    "Який учений відкрив закон всесвітнього тяжіння?"
]
```

Список варіантів відповідей. Для кожного запитання є 4 варіанти.

```
options = [
    ["Берлін", "Париж", "Рим", "Мадрид"],
     ["Індійський", "Атлантичний", "Тихий", "Північний Льодо-
витий"],
    ["364", "365", "366", "367"],
    ["Папір", "Миша", "Скальпель", "Молоток"],
   ["Архімед", "Ейнштейн", "Галілео Галілей", "Ісаак Ньютон"]
]
```

Список індексів правильних відповідей. Тут зберігаються номери правильних варіантів (0–3). Наприклад, для першого запитання правильна відповідь — варіант №1 (тобто `"Париж"`).

```
answers = [1, 2, 2, 1, 3]
```

Рівні виграшу. Після кожного правильного запитання сума збільшується.

```
money_levels = [0, 100, 500, 1000, 2000, 5000, 10000, 20000,
30000, 40000,50000, 75000, 150000, 200000, 500000, 1000000]
```

Створення стартового вікна main:

1. Задання розмірів вікна (`"600x400"`)
2. Колір фону: `#0a0535`
3. Фіксація розмірів (вимкнення масштабування).
4. Встановлення заголовка: `"Хто хоче стати мільйонером?"`.
5. Запустити головний цикл **mainloop()**

Створити мітку для заголовку **title_label** з такими властивостями: текст: `"Гра: Хто хоче стати мільйонером"`; шрифт: `"Arial Black"`, 18, `"bold"`; колір фону `"#0a0535"` і колір тексту `"gold"`.

```
title_label.pack(pady=30)
```

Створити кнопки головного меню:

```
Button(main, text="▶️ Почати гру", font=("Arial", 14), width=25,
bg="green", fg="white", command=start_game).pack(pady=15)

Button(main, text="📝 Редагувати питання", font=("Arial", 14),
width=25, bg="#0077cc", fg="white", command=open_edit_window).
pack(pady=15)

Button(main, text="❌ Вихід", font=("Arial", 14), width=25,
bg="darkred", fg="white", command=main.destroy).pack(pady=15)
```

Вгорі вікна з кодом створити функції `start_game()`, яка закриває головне вікно та викликає функцію `open_game_window()`, що створюватиме вікно з грою. А функція `open_edit_window()` закриває головне вікно та викликає функцію `open_question_editor()`, яка створюватиме вікно з редагуванням питань.

```
# ----------------------------
# Функції
# ----------------------------

def start_game():
    main.withdraw()
    open_game_window()

def open_edit_window():
    main.withdraw()
    open_question_editor()
```

withdraw() – приховує це вікно з екрана, не закриваючи його. Потім можна повернути його за допомогою **`main.deiconify()`**.

Вікно залишається у пам'яті, просто його не видно користувачу.

Третя кнопка — викликає функцію **destroy**, яка закриває вікно

Створення вікна гри `game_win` — код пишеться всередині функції `open_game_window() :`

```
def open_game_window():
    game_win = Toplevel()
    game_win.title("Гра почалась!")
        ... ваш код …
```

Toplevel() — те саме, що й **Tk()**, тільки викликається тоді, коли вікна приховуються, а не закриваються.

1. Задання розмірів вікна (`"700x500"`)
2. Встановлення заголовка: `"Гра почалась!"`.
3. Колір фону: `#00194d`
4. Фіксація розмірів (вимкнення масштабування).

Створити мітку для запитання **`question_label`** з такими властивостями: текст: `""`; шрифт: `"Arial"`, `16`, `"bold"`; колір фону `"#00194d"` і колір тексту `"white"`.

```
question_label.pack(pady=40)
```

Створити мітку для поточного виграшу **money_label** з такими властивостями: текст: `"Ваш виграш: 0$"`; шрифт: `"Arial"`, `14`, `"bold"`; колір фону `"#00194d"` і колір тексту `"gold"`.

```
money_label.pack(pady=10)
```

Створити три змінні — списки:

```
buttons = []
current_question = [0]
balance = [0]
```

`buttons = []` — це список кнопок.

Коли ми створюємо кнопки з варіантами відповідей (A, B, C, D), нам зручно зберегти їх у список, щоб потім легко:

- змінювати текст кнопок;
- вимикати або вмикати кнопки;
- перевіряти, яку натиснув користувач.

`current_question = [0]` — це номер поточного запитання.

Ми починаємо гру із запитання №0 (першого). Коли користувач відповідає правильно, то збільшуємо його: `current_question[0] += 1`

`balance = [0]` — це поточна сума виграшу (рахунок гравця).

Цікаво знати:

Чому в квадратних дужках (`[0]`)? Бо якщо ми оголосимо просто `current_question = 0`, а потім будемо змінювати його всередині вкладеної функції (наприклад, коли натискають кнопку), **Python** сприйме це як локальну змінну, і вона не зміниться у головному коді.

А список (`[0]`) можна змінювати всередині будь-якої функції — тому це зручний «обхідний шлях» для **Tkinter**.

Запускаємо цикл, який створюватиме 4 кнопки з варіантами відповідей:

```
for i in range(4):
            b = Button(game_win, text="", width=25, height=2,
font=("Arial", 12), bg="#003399", fg="white",
                       command=lambda i=i: check_answer(i))
        b.pack(pady=5)
        buttons.append(b)
```

Пояснення:

Ми маємо 4 варіанти відповіді для кожного запитання (`A`, `B`, `C`, `D`).

Цей цикл повторюється 4 рази, а змінна `i` приймає значення `0`, `1`, `2`, `3`.

Тобто:

- при `i = 0` — створюється кнопка для першого варіанта;
- при `i = 1` — для другого;
- при `i = 2` — для третього;
- при `i = 3` — для четвертого.

`b = Button(game_win, ...)` — це створення кнопки. Вона відображатиметься у вікні `game_win` (тобто у вікні гри).

Параметри:

- `text=""` — поки що порожній, текст додається пізніше (запитання ще не показане);
- `width=25, height=2` — розміри кнопки;
- `font=("Arial", 12)` — шрифт;
- `bg="#003399", fg="white"` — кольори (синя кнопка з білим текстом);
- `command=lambda i=i: check_answer(i)` — головне! 👉 коли гравець натискає кнопку, викликається функція `check_answer(i)` — тобто передається номер варіанту (0, 1, 2 або 3).

`lambda i=i: check_answer(i)`

Це анонімна функція, яка «запам'ятовує» поточне `i`. Якщо написати просто `command=lambda: check_answer(i)`, то всі кнопки передавали б те саме значення — останнє `i` (тобто 3). Тому робимо `lambda i=i`, щоб кожна кнопка мала власне `i`.

`buttons.append(b)`

Додаємо кнопку `b` у список `buttons`, щоб потім легко змінювати її текст у функції `show_question()`. Це дозволяє швидко оновити всі 4 кнопки для нового запитання без створення нових елементів.

Перед цим циклом опишемо функцію `show_question()`. Ця функція показує наступне запитання та варіанти відповідей:

```
def show_question():
        if current_question[0] < len(questions):
            q = questions[current_question[0]]
            question_label.config(text=q)
            opts = options[current_question[0]]
            for i, b in enumerate(buttons):
              b.config(text=opts[i], state=NORMAL, bg="#003399")
        else:
                  messagebox.showinfo("Вітаємо!", f"Ви виграли
{balance[0]}$ 🎉")
            game_win.destroy()
            main.deiconify()
```

Пояснення:

`if current_question[0] < len(questions):`

`current_question[0]` — номер поточного запитання (починається з 0);

`len(questions)` — кількість усіх запитань у грі.

Якщо ще залишились запитання — показуємо наступне. Якщо запитань уже немає — гра завершується (пункт нижче).

```
q = questions[current_question[0]]
question_label.config(text=q)
```

Беремо запитання за поточним індексом зі списку **questions**.

Виводимо його у напис **(Label)** на екрані гри.

opts — список із 4 варіантів для цього запитання (наприклад: `["Тихий", "Індійський", "Атлантичний", "Льодовитий"]`).

buttons — список 4 кнопок, створених раніше.

За допомогою **enumerate()** одночасно беремо:

`i` — номер варіанта (0–3)

`b` — саму кнопку

І оновлюємо:

`text=opts[i]` → підпис на кнопці

`state=NORMAL` → робимо активною

`bg="#003399"` → повертаємо стандартний синій колір

Таким чином усі 4 кнопки отримують нові варіанти відповідей.

Після циклу викликаємо функцію **show_question()**

Кожна зі створених кнопок викликає функцію **check_answer(i)**. Ця функція перевіряє правильність відповіді на запитання та оновлює виграш або завершує гру у випадку помилки:

```
def check_answer(i):
        correct = answers[current_question[0]]
        if i == correct:
            balance[0] = money_levels[current_question[0]+1]
          money_label.config(text=f"Ваш виграш: {balance[0]}$")
            current_question[0] += 1
            show_question()
        else:
             messagebox.showerror("Помилка", f"Неправильно! Ви
виграли {balance[0]}$.")
            game_win.destroy()
            main.deiconify()
```

Отримуємо правильну відповідь:

`correct = answers[current_question[0]]`

`answers` — список із номерами правильних варіантів (0–3).

`current_question[0]` — номер поточного запитання.

`correct` — правильний варіант для цього запитання.

Перевіряємо, яку кнопку натиснув користувач:

`if i == correct:`

`i` — це номер кнопки, яку натиснув гравець (0–3).

Якщо він збігається з `correct` → відповідь правильна ✓

Інакше → неправильна ✖

Якщо відповідь правильна:

`money_levels` — список рівнів виграшу, наприклад: `[0, 100, 500, 1000, 2000, 5000]`

`current_question[0] + 1` → наступний рівень призу (бо перше питання — за 100$, друге — за 500$ тощо)

Оновлюємо напис: `"Ваш виграш: 500$"`

Переходимо до наступного запитання (`current_question[0] += 1`)

Викликаємо `show_question()` → відображаємо нове запитання. Таким чином після кожної правильної відповіді гра плавно переходить далі.

Якщо відповідь неправильна:

Виводимо віконце з повідомленням про помилку ✖

Гра завершується `(game_win.destroy())`

Повертаємось до головного меню `(main.deiconify())`

Створення вікна редагування запитань **`edit_win`** — код пишеться всередині функції **`open_question_editor()`** :

```
def open_question_editor():
    edit_win = Toplevel()
    edit_win.title("Редагування питань")
```

1. Задання розмірів вікна (`"750x600"`)
2. Встановлення заголовка: `"Редагування запитань"`.
3. Колір фону: `"#0d1a45"`

Створити мітку для заголовка

```
Label(edit_win, text="Редагування списку запитань", font=("Arial", 16, "bold"), bg="#0d1a45", fg="white").pack(pady=10)
```

Створити поле для списку запитань **listbox**

```
listbox = Listbox(edit_win, width=70, height=10, font=("Arial", 12))
listbox.pack(pady=10)
```

Створити фрейм, в якому будемо розміщувати віджети для створення запитань і варіантів відповідей

```
frame = Frame(edit_win, bg="#0d1a45")
frame.pack(pady=10)
```

До фрейму додати та розмістити мітки та поля в такому вигляді, використовуючи **grid**:


```
Label(frame, text="Нове питання:", bg="#0d1a45", fg="white").grid(row=0, column=0, sticky=W)
    q_entry = Entry(frame, width=60)
    q_entry.grid(row=0, column=1, pady=3)
```

```
    Label(frame, text="1)", bg="#0d1a45", fg="white").grid(row=1,
column=0, sticky=E)
     opt1 = Entry(frame, width=40)
     opt1.grid(row=1, column=1, pady=3)

    Label(frame, text="2)", bg="#0d1a45", fg="white").grid(row=2,
column=0, sticky=E)
     opt2 = Entry(frame, width=40)
     opt2.grid(row=2, column=1, pady=3)

    Label(frame, text="3)", bg="#0d1a45", fg="white").grid(row=3,
column=0, sticky=E)
     opt3 = Entry(frame, width=40)
     opt3.grid(row=3, column=1, pady=3)

    Label(frame, text="4)", bg="#0d1a45", fg="white").grid(row=4,
column=0, sticky=E)
     opt4 = Entry(frame, width=40)
     opt4.grid(row=4, column=1, pady=3)

    Label(frame, text="Правильна відповідь (1-4):", bg="#0d1a45",
fg="white").grid(row=5, column=0, sticky=E)
     correct_var = Entry(frame, width=5)
     correct_var.grid(row=5, column=1, sticky=W, pady=3)
```

Додати 3 кнопки:

- викликає функцію **`add_question()`**, яка додає запитання та відповіді до списків

```
Button(edit_win, text="➕ Додати питання", command=add_question,
bg="green", fg="white", width=20).pack(pady=5)
```

- викликає функцію **`delete_question()`**, яка видаляє зі списку обране запитання та відповідні до нього відповіді

```
Button(edit_win, text="❌ Видалити вибране", command=delete_
question, bg="red", fg="white", width=20).pack(pady=5)
```

- викликає **lambda** — функцію, яка закриває це вікно і повертає головне

```
Button(edit_win, text="⬅️ Назад", command=lambda: (edit_win.
destroy(), main.deiconify()), width=20).pack(pady=10)
```

Після цього викликаємо функцію `refresh_list()`

Перед **Label** із заголовком описуємо функцію `refresh_list()`. Ця функція оновлює список запитань у віджеті **Listbox**, щоб він завжди відображав актуальні дані (тобто — всі поточні запитання з масиву `questions`).

```
def refresh_list():
        listbox.delete(0, END)
        for q in questions:
            listbox.insert(END, q)
```

Опишемо функцію `delete_question()`, яка видаляє вибране запитання зі списку

```
def delete_question():
        index = listbox.curselection()
        if index:
            i = index[0]
            questions.pop(i)
            options.pop(i)
            answers.pop(i)
            refresh_list()
```

Опишемо функцію `add_question()`, яка додає запитання та варіанти відповіді у список, використовуючи `try-except`

```
def add_question():
        q = q_entry.get().strip()
          if q and opt1.get() and opt2.get() and opt3.get() and
opt4.get():
            questions.append(q)
             options.append([opt1.get(), opt2.get(), opt3.get(),
opt4.get()])
            try:
                correct = int(correct_var.get()) - 1
                if 0 <= correct <= 3:
                    answers.append(correct)
                    refresh_list()
                    q_entry.delete(0, END)
                    opt1.delete(0, END)
                    opt2.delete(0, END)
                    opt3.delete(0, END)
                    opt4.delete(0, END)
                else:
           messagebox.showerror("Помилка", "Номер правильної від-
повіді має бути 1–4.")
            except:
                  messagebox.showerror("Помилка", "Вкажіть номер
правильної відповіді (1–4).")
        else:
         messagebox.showwarning("Помилка", "Заповніть усі поля!")
q = q_entry.get().strip()
```

Отримуємо текст запитання з поля **Entry** (віджет для введення тексту). `.strip()` прибирає зайві пробіли з початку і кінця.

```
if q and opt1.get() and opt2.get() and opt3.get() and opt4.get():
```

Перевіряємо, чи всі поля заповнені (запитання і 4 варіанти відповідей).

```
  questions.append(q)
            options.append([opt1.get(), opt2.get(), opt3.get(),
opt4.get()])
```

Додаємо нове запитання до списку **questions**.

І чотири відповіді — у список **options** (вкладений список).

```
try:
    correct = int(correct_var.get()) - 1
```

Зчитуємо номер правильної відповіді (очікується 1, 2, 3 або 4).

Віднімаємо 1, бо в **Python** індексація починається з нуля.

```
if 0 <= correct <= 3:
                answers.append(correct)
                refresh_list()
```

Якщо введений номер правильний (1–4), додаємо його до списку `answers`.

`refresh_list()` — оновлює **Listbox**, щоб одразу показати нове запитання.

```
q_entry.delete(0, END)
                opt1.delete(0, END)
                opt2.delete(0, END)
                opt3.delete(0, END)
                opt4.delete(0, END)
```

Очищуємо поля після додавання, щоб можна було вводити наступне запитання.

6 етап. Тестувальний

Перевірити:

- Відображення стартового, ігрового та редакторського вікна.
- Додавання й видалення запитань працює.
- При правильній відповіді — рахунок оновлюється.
- При неправильній — гра завершується.
- Виграш відповідає рівню складності запитання.

Додатково:

7 етап. Оптимізація та розширення

Можливі розширення:

Реалізувати підказки («50 на 50», «Допомога друга»).

8 етап. Документація

Створіть документацію у файлі `Readme.txt`

Назва: Хто хоче стати мільйонером?

Призначення: навчальна гра для перевірки знань

Основні файли: `millionaire.py`

Використані віджети: **Label**, **Button**, **Text**, **Entry**, **Listbox**, **Combobox**

Модулі: **tkinter**, **random**, **ttk**, **messagebox**

Автор: (ПІБ учня)

Рік: 2025

Розділ 5

БЕЗПЕКА ЦИФРОВИХ СИСТЕМ

Теми 61-62 **Цифрові інновації: користь і виклики. Інформація та інформаційні процеси у різних галузях. Захист особистих даних та конфіденційність у цифрових середовищах. Безпека цифрових систем**

Ключові поняття

Цифрове середовище
Основні складові цифрового середовища
Цифрові інновації
Інформаційна безпека
Загрози безпеці в інтернеті

Цифрове середовище

Цифрове середовище — це віртуальний простір, що охоплює всі цифрові пристрої, мережі, онлайн-ресурси, де люди спілкуються, навчаються, працюють.

Цифрове середовище робить наше життя більш зручним і доступним. Воно дозволяє нам швидко знаходити інформацію, спілкуватися з іншими людьми, працювати та навчатися в онлайн-режимі, а також розв'язувати різноманітні завдання.

Цифрове середовище дозволяє нам оточувати себе новими можливостями, але також вимагає від нас обережності та вміння користуватися цими технологіями відповідально та безпечно.

У цифровому середовищі ми користуємося різними програмами і вебсайтами, щоб шукати інформацію, спілкуватися з іншими людьми, грати в ігри, редагувати фотографії і відео, а також виконувати багато інших завдань.

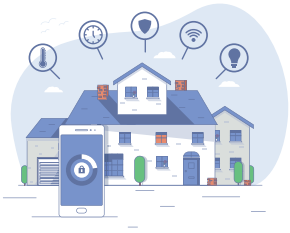
Основні складові цифрового середовища

- Апаратне забезпечення (комп'ютери та цифрові пристрої)
- Мережева інфраструктура (інтернет)
- Програмне забезпечення (програми й додатки)
- Цифрові платформи та соціальні мережі
- Цифрові дані та інформаційні ресурси
- Користувачі та їхня цифрова компетентність
- Бази даних та інформаційні системи
- Інтернет речей (IoT) та смартпристрої

Цифрові інновації — це нові технології та рішення, які змінюють наше життя за допомогою комп'ютерів, інтернету, смартфонів, ШІ тощо.

Приклади:

- штучний інтелект (чат-боти, перекладачі, рекомендації на **YouTube/TikTok**);
- розумні будинки та «розумні» пристрої (лампи, колонки, годинники);
- онлайн-освіта (курси, платформи, відеоуроки);
- безконтактні платежі **Apple Pay/Google Pay;**
- 3D-друк, дрони, робототехніка.



Виклики та ризики

- **Інформаційне перевантаження:** занадто багато новин, контенту, реклами.
- **Втрата приватності**: наші дані збирають соцмережі, сайти, додатки.
- **Залежність від гаджетів**: тікток/ігри замість сну, навчання, спорту.
- **Фейки та маніпуляції**: неправдиві новини, пропаганда, deepfake-відео.
- **Кіберзлочинність**: шахрайство, віруси, крадіжка паролів і грошей.

Як це працює в різних сферах

Бізнес і банки

- інтернет-банкінг, оплата онлайн
- аналіз покупок (які товари популярні)
- реклама, таргетинг (тобі показують саме ті оголошення, які «цікавлять»)

Освіта

- електронні щоденники, журнали
- платформи для навчання (**Google Classroom**, **Moodle, Zoom**, відеоуроки)
- тести онлайн, автоматична перевірка

Медицина

- електронні медичні картки
- діагностичні системи (аналіз знімків, аналіз аналізів)
- онлайн-консультації з лікарем

Особисті дані

Особисті дані — інформація, за якою можна впізнати конкретну людину:

- прізвище, ім'я, по батькові;
- дата народження, номер телефону;
- e-mail;
- домашня адреса;
- фото, відео, голос;
- логіни, паролі,
- дані банківських карток;
- геолокація (де ти зараз перебуваєш).

Чому це важливо?

Якщо особисті дані потраплять до шахраїв, вони можуть:

- отримати доступ до акаунтів (**Instagram**, **Telegram**, ігри);
- вкрасти гроші з картки;
- оформити кредити на ваше ім'я;
- шантажувати (наприклад, погрожувати оприлюднити особисті фото).

Інформаційна безпека

Інформаційна безпека — стан захищеності інформаційного середовища суспільства, який забезпечує конфіденційність, доступність і цілісність даних.

Проблема захисту даних від втрати, викрадення, спотворення або пошкодження потребує посиленої уваги, оскільки зростає роль інформаційно-комунікаційних технологій у сучасному суспільстві.

Розкриття конфіденційної інформації у Мережі

Розкриття конфіденційної інформації у Мережі — повідомлення повного власного імені чи імен членів родини, адреси проживання і навчального закладу, номерів телефонів та іншої приватної інформації в інтернеті.

Загрози безпеці в інтернеті

Загрози безпеці в інтернеті — це потенційні ризики та небезпеки, які можуть виникнути під час користування інтернетом.

Віруси та інші шкідливі програми

Віруси, черви, троянські коні та інші види шкідливого програмного забезпечення можуть інфікувати ваш комп'ютер або пристрій. Вони можуть пошкодити вашу операційну систему, вкрасти особисті дані або перешкоджати нормальній роботі комп'ютера.

Доступ до облікових записів

Зловмисники можуть намагатися взяти під контроль ваші облікові записи, такі як пошта, соціальні мережі чи онлайн-банкінг, і використовувати їх для шкідливої діяльності.

Крадіжка ідентичності

Зловмисники можуть намагатися вкрасти вашу ідентичність, використовуючи ваші особисті дані для злочинних цілей.

Фішинг

Атаки фішингу — це спроби обману користувачів, зазвичай через електронну пошту або вебсайти, з метою витягнути особисті або фінансові дані, такі як паролі чи номери кредитних карток.

Спам

Спам — це небажані повідомлення, які надсилаються користувачам без їхньої згоди. Спам може містити небезпечні посилання або вкладення.

Основні правила безпечної поведінки в інтернеті

- Повідомте, якщо вас хтось скривдив.
  Ніколи не пізно розповісти дорослим, якщо вас хтось образив, не намагайтеся розібратися в цьому самостійно. Зверніться до батьків або вчителів.
- Не відкривайте сумнівних повідомлень
  Не відкривайте листів електронної пошти, файлів або вебсторінок, отриманих від людей, яких ви реально не знаєте або не довіряєте.
- Нікому не давайте своїх логіна та пароля, персональних даних, за винятком дорослих вашої родини.
- Дотримуйтесь мережевого етикету
  Завжди будьте ввічливими в електронному листуванні, і ваші кореспонденти будуть ввічливими з вами.
- Користуйтеся тільки перевіреними Wi-Fi-мережами.
  Не відправляйте важливих даних (дані кредитних карток, онлайн-банкінгу тощо) через публічні та незахищені Wi-Fi-мережі.
- Перевіряйте сертифікат безпеки сайтів у вигляді замка в адресному рядку браузера

Завжди поводьтеся у мережі так, як би ви хотіли, щоб поводилися з вами!

Як слід захищати цифрові системи

1 Паролі та двофакторна автентифікація (2FA)

Пароль — базовий механізм автентифікації користувача

Надійний пароль: достатня довжина, складність, унікальність

Заборонено використовувати один пароль для всіх сервісів

2FA: пароль + додатковий фактор (код, токен, біометрія)

2FA зменшує ризик несанкціонованого доступу навіть при компрометації пароля

Двофакторна автентифікація (2FA) — це механізм, коли до пароля додається другий фактор: одноразовий код із SMS, кодувальний застосунок, фізичний ключ або біометричні дані (відбиток пальця, Face ID). Таким чином, для входу потрібна не лише інформація (пароль), а й щось, чим володіє користувач фізично (телефон, токен). Це суттєво підвищує рівень захисту облікового запису.

2 Шифрування

Криптографічний захист (шифрування)

Шифрування — перетворення даних у вигляд, непридатний для читання без ключа

Розрізняють симетричне та асиметричне шифрування

Використовується під час передавання даних (протокол HTTPS, месенджери)

Захищає конфіденційність інформації від перехоплення й несанкціонованого доступу

3 Резервні копії (backups)

Резервна копія — дублікати важливих даних, збережені окремо від основного носія

Застосовується для відновлення інформації після збоїв, атак або випадкового видалення

Рекомендовано використовувати хмарні сервіси та/або окремі фізичні носії

Принцип: критично важливі дані не повинні існувати лише в одному примірнику.

Резервне копіювання — це систематичне створення копій даних, які зберігаються на іншому носії або в іншому середовищі (наприклад, у хмарному сховищі або на зовнішньому диску).

Мета — забезпечити можливість відновлення інформації у випадку:

- фізичної поломки пристрою;
- вірусної атаки чи шифрувальника (ransomware);
- помилкового видалення файлів користувачем;
- технічних збоїв.

4 Антивірус та мережевий екран (фаєрвол)

Захист від шкідливого програмного забезпечення

Шкідливе ПЗ (malware): віруси, трояни, шпигунські програми, ransomware

Антивірусні програми виявляють і блокують шкідливий код

Мережевий екран (фаєрвол) фільтрує мережевий трафік і контролює підключення

Спільна робота антивірусу та фаєрвола знижує ризики несанкціонованого доступу

Захист особистого цифрового простору

- створюйте надійні, складні паролі та не встановлюйте однакового пароля для декількох ресурсів
- не пересилайте у месенджерах конфіденційну інформацію та фото документів
- завантажуй програми та додатки лише з офіційних джерел
- надавайте перевагу інтернету від мобільного оператора перед загальнодоступною мережею Wi-Fi
- пам'ятайте, що найкращий метод захисту своїх інтернет-акаунтів – двофакторна автентифікація. Вмикайте її всюди, де є така можливість

Цифрові інновації дають величезні можливості: спрощують роботу, навчання, комунікацію, відкривають доступ до глобальних ресурсів.

Водночас вони створюють нові ризики: інформаційне перевантаження, маніпуляції, кібератаки, втрату конфіденційності.

Безпека цифрових систем базується на комплексі заходів: надійній автентифікації, шифруванні, резервному копіюванні, захисті від шкідливого ПЗ та регулярному оновленні програмного забезпечення.

Практичні роботи №61-62

«Цифрові інновації та безпека в обраній сфері»

Завдання

- дослідити, як цифрові інновації використовуються в одній конкретній сфері (освіта, медицина, бізнес, соцмережі тощо);
- показати, які дані збираються, які є ризики та як їх захищають (паролі, шифрування, резервні копії, 2FA і т.д.).

Середовище виконання: **Canva**

Хід роботи

Оберіть сферу, яку хочете дослідити, наприклад:

- освіта
- медицина
- банківська справа / онлайн-платежі
- соціальні мережі та месенджери
- онлайн-ігри / геймінг

Знайдіть 1–2 конкретні цифрові інновації у своїй сфері (наприклад: електронний щоденник, онлайн-банкінг, телемедицина, чат-бот, система рекомендацій у TikTok).

Для кожної інновації коротко запишіть у чернетці:

- Яке завдання вона розв'язує?
- Які дані збирає про користувача? (ПІБ, контактні дані, історія покупок, місцезнаходження тощо)
- Які є ризики? (витік даних, несанкціонований доступ, маніпуляції, залежність і т.д.)
- Які засоби захисту використовуються / рекомендуються? (паролі, 2FA, шифрування, резервні копії, антивірус, налаштування приватності).

Підготуйте інформацію для показу в презентації, використовуючи **Canva**.

Збережіть презентацію (посилання або PDF) та підготуйтеся коротко презентувати 2–3 основні ідеї.

Ключові поняття

Цифровий слід
Конфіденційність

Цифровий слід — це сукупність усіх даних про людину, які залишаються під час користування інтернетом та цифровими пристроями.

- Пости, вподобайки, коментарі
- Пошукові запити
- Історія перегляду сайтів
- Дані з додатків, геолокація, історія покупок
- Технічні дані: IP-адреса, пристрій, браузер

Кожна дія онлайн залишає слід.
Дані зберігаються довше, ніж нам здається.
Від цифрового сліду залежить наша репутація й безпека.

Види цифрового сліду

Активний цифровий слід

Усе, що користувач створює свідомо
Публікації, фото, відео, сторіз
Коментарі, відгуки, листування

Пасивний цифровий слід

Дані, які збираються автоматично
Cookies, геолокація, час відвідування сайту
Інформація про пристрій, мережу, поведінку на сайті

Навіщо компаніям наш цифровий слід?

- Персоналізація сервісів (рекомендації, підказки)
- Таргетована реклама
- Аналітика поведінки користувачів
- Поліпшення продуктів і сервісів
- Формування профілів користувачів (interest profile, risk profile тощо)

Компанії збирають дані про користувачів, щоб персоналізувати сервіси: показувати саме ті фільми, товари чи дописи, які нам можуть сподобатися.

Ризики, пов'язані з цифровим слідом

- Втрата конфіденційності (надмірний обсяг даних про людину)
- Репутаційні ризики (старі пости, фото, жарти)
- Кібербулінг, переслідування, доксинг (поширення особистих даних)
- Крадіжка особистості (identity theft)
- Маніпуляція поведінкою через персоналізований контент та рекламу
- Компанії збирають дані про користувачів, щоб персоналізувати сервіси: показувати саме ті фільми, товари чи дописи, які нам можуть сподобатися.

Конфіденційність

Конфіденційність — це право людини контролювати, хто, коли й у якому обсязі має доступ до її особистої інформації.

- Не вся інформація про людину повинна бути публічною
- Людина має право знати, які дані про неї збирають
- Важливо розуміти, кому ми ці дані добровільно передаємо

Що не варто робити публічним

- Точну домашню адресу, розклад пересування
- Дані документів (паспорт, ІПН, учнівські/студентські квитки)
- Фото квитків, банківських карток, медичних документів
- Чужі дані без їхньої згоди (контакти, фото, історії)

Управління цифровим слідом

- Перегляд і видалення старих постів, фото, коментарів
- Перевірка налаштувань приватності в соцмережах
- Відмова від зайвих дозволів у мобільних додатках
- Видалення непотрібних акаунтів і сервісів

Важливо час від часу «чистити» свій цифровий слід, оскільки все, що ми колись викладали онлайн, може з'явитися в найнесподіваніший момент — під час вступу, працевлаштування чи навіть конфлікту з однолітками.

Очищення цифрового сліду — це спосіб взяти під контроль свою онлайн-історію та свідомо вирішувати, яку інформацію про себе ви залишаєте у відкритому доступі.

Як захистити конфіденційність

- Використовувати складні паролі й двофакторну автентифікацію
- Не переходити за підозрілими посиланнями, не відкривати сумнівних вкладень
- Уважно читати, які дозволи запитує додаток (камера, мікрофон, геолокація)
- Обмежувати коло людей, які бачать сторіз, пости, геотеги

Цифровий слід формується постійно, навіть коли ми про це не думаємо

Практична робота №63

Які дані збирає сайт? Аналіз політики конфіденційності
Формат — доповідь (резюме) в **Canva**

Оберіть один популярний сайт або сервіс (наприклад, **Google**, **YouTube**, інтернет-магазин, освітню платформу).

Знайдіть внизу сторінки посилання типу «Конфіденційність», «Privacy policy», «Політика щодо файлів cookie».

Прочитайте (не всю, а ключові блоки) і випишіть у документ:

- які категорії даних збирають (облікові, контактні, технічні, поведінкові тощо);
- для чого їх використовують (реклама, аналітика, поліпшення сервісу, безпека);
- кому можуть передавати дані (партнери, рекламодавці, служби безпеки).

Складіть коротке резюме (10–12 речень):

- що саме дізнається сайт про користувача;
- які моменти вас насторожили;
- чи готові ви користуватися цим сервісом, знаючи це.

Представте інформацію класу.
Надайте доступ за покликанням педагогу.

Електронна демократія. Електронне урядування. Цифровий підпис

Тема 64

Цифрова трансформація держави
Електронне урядування
Електронні послуги
Електронний цифровий підпис

Сучасна держава активно використовує цифрові технології для взаємодії з громадянами.

Електронна демократія, електронне урядування та цифровий підпис є ключовими елементами цієї трансформації.

Цифрова трансформація держави — це систематичне впровадження інформаційно-комунікаційних технологій у роботу органів влади.

Її мета — зробити публічні послуги більш доступними, прозорими та ефективними.

Громадянин отримує можливість взаємодіяти з державою дистанційно, без фізичної присутності в установах.

Основними інструментами електронної демократії є електронні петиції, онлайн-консультації, публічні обговорення та сервіси «бюджету участі».

Вони дають можливість громадянам висувати пропозиції, коментувати проєкти рішень і контролювати використання публічних коштів.

Електронна демократія в Україні реалізується через:

- платформи електронних петицій (місцеві та загальнодержавні);
- публічні обговорення проєктів рішень органів влади онлайн тощо.

Електронне урядування

Електронне урядування забезпечує цифрову інфраструктуру та електронні послуги держави.

Електронна демократія використовує ці можливості для участі громадян у прийнятті рішень.

Електронне урядування втілюється через:

- єдині портали державних послуг та мобільні застосунки;
- Електронні реєстри (населення, бізнесу, нерухомості тощо);

- онлайн-сервіси: реєстрація ФОП, довідки, витяги, соцвиплати, запис до черги.

Переваги

- Доступність послуг 24/7 незалежно від місця проживання.
- Менше черг, менше суб'єктивного впливу чиновників.
- Прозорість рішень та руху публічних коштів.

Ризики

- Залежність від стабільності ІТ-систем та кіберзахисту.
- Можливість витоку персональних даних у разі слабкої безпеки.

Електронні послуги — це адміністративні та інші публічні послуги, які надаються в онлайн-форматі через спеціальні портали чи додатки.

Громадянин може подати заяву, отримати довідку, зареєструвати бізнес або сплатити податки дистанційно.

Електронний цифровий підпис

Електронний цифровий підпис (або скорочено — **ЕЦП**) за правовим статусом прирівняний до власноручного підпису або печатки.

ЕЦП — це дані в електронній формі, отримані за результатами криптографічного перетворення, які додаються до інших даних або документів і забезпечують їх цілісність та ідентифікацію автора.

За допомогою послуг ЕЦП можна підписувати електронні документи, користуватися електронними послугами, реєструватися на державних порталах тощо. Документи, підписані за допомогою ЕЦП, мають таку ж саму юридичну силу, як і звичайні.

Цифровий підпис дає змогу:

- Підписувати заяви, договори та інші документи без паперу й особистої присутності.
- Подавати електронну звітність, реєструвати бізнес, оформлювати послуги онлайн.
- Підтверджувати свою особу в державних та банківських сервісах.

Закритий ключ цифрового підпису прирівнюється до особистого підпису та печатки.

Передавання ключа або пароля іншим особам означає передавання права підпису від вашого імені.

Базується на парі ключів: закритий (підписує) і відкритий (перевіряє).

Електронне урядування створює цифрову інфраструктуру та портали послуг.

Електронна демократія використовує ці платформи для участі громадян у прийнятті рішень.

Цифровий підпис забезпечує довіру, ідентифікацію та юридичну силу дій громадян онлайн.

Практична робота №64

Дослідницька пошукова робота
Маршрут однієї е-послуги

Завдання:

1. Оберіть одну послугу з цього списку:
- реєстрація місця проживання;
- реєстрація ФОП;
- отримання витягу / довідки (про несудимість, місце проживання тощо);
- запис до електронної черги (ЦНАП, лікар, садок тощо).

2. Знайдіть в інтернеті інформацію:
- як ця послуга надається онлайн (через який портал / застосунок);
- які кроки потрібно зробити (крок 1, 2, 3…);
- чи потрібен електронний підпис і на якому етапі;
- які дані користувач має ввести.

3. У **Canva** створіть один візуальний маршрут (схему на 1–2 слайдах):
- ліва частина: «Як це було раніше (паперовий формат)» — коротко: прийти в установу, заповнити бланк, стояти в черзі, забрати документ;
- права частина: «Як це працює зараз онлайн» — 4–6 кроків із піктограмами (зайти на портал, обрати послугу, заповнити дані, підписати е-підписом, отримати результат).

4. Додайте внизу 3–4 речення висновку:
- у чому вигода для громадянина;
- які є ризики або складнощі;
- чи вважаєте, що онлайн-формат кращий.

Представте свою візуальну карту класу.
Доступ за покликанням надайте педагогу.

Соціальні аспекти масштабного збору та аналізу даних. Великі дані. Криптовалюти, блокчейн. NFT. Кодування та шифрування даних. Системи числення

Тема 65

Ключові поняття

Дані
Великі дані
Криптовалюта
Кодування та шифрування даних
Система числення

Великі дані

Дані — це відомості про об'єкти, події, процеси, подані у формі, придатній для зберігання й опрацювання.

У цифрову епоху дані стали стратегічним ресурсом, який можна аналізувати, продавати й використовувати для прийняття рішень.

Масштабний збір даних відбувається через сайти, додатки, сенсори, соціальні мережі, платіжні системи.

Масштабний збір даних створює детальні профілі користувачів: інтереси, звички, фінансову поведінку.

Великі дані — це масиви різнорідних даних надзвичайно великого обсягу, які надходять з високою швидкістю.

Характеризуються властивостями 3V: *volume* (обсяг), *velocity* (швидкість надходження), *variety* (різноманітність).

Для їх опрацювання застосовують спеціалізовані платформи, алгоритми машинного навчання та хмарні обчислення.

Великі дані активно використовуються в різних сферах життя суспільства для прийняття рішень, прогнозування та оптимізації процесів.

- Аналіз поведінки користувачів у соцмережах і онлайн-магазинах.
- Оптимізація транспортних систем, енергомереж, міської інфраструктури.
- Медичні дослідження, прогнозування епідемій, персоналізована медицина.

Великі дані збираються й аналізуються у цифровій формі, закодованій у певних системах числення.

Для захисту таких даних застосовують шифрування.

Криптовалюти, блокчейн. NFT

Криптовалюта — це цифрові гроші, які існують у вигляді записів у блокчейні й не мають фізичної форми.

Транзакції підтверджуються учасниками мережі за допомогою криптографічних алгоритмів.

Переваги: децентралізація, прозорість операцій, швидкі міжнародні перекази.

NFT — це унікальні криптографічні токени в блокчейні, які засвідчують право власності на конкретний цифровий об'єкт.

На відміну від криптовалют, NFT не є взаємозамінними: кожен токен має свій ідентифікатор.

Використовуються для продажу цифрового мистецтва, колекційних предметів, ігрових активів, але викликають дискусії щодо реальної цінності та спекуляцій.

Блокчейн — це розподілений реєстр транзакцій, який зберігається одночасно на багатьох комп'ютерах мережі.

Дані організовано в блоки, що послідовно з'єднані криптографічними хешами.

Кодування та шифрування даних

Кодування — це подання даних у певному форматі або системі запису (наприклад, кодування тексту в ASCII чи Unicode).

Основна мета кодування — забезпечити коректне зберігання й передавання інформації між системами.

Формати тексту: UTF-8, Unicode.

Коди символів (таблиці ASCII, emoji-коди).

Кодування зображень і відео (JPEG, MP4).

Шифрування — це перетворення даних у вигляд, незрозумілий без ключа, з метою захисту від несанкціонованого доступу.

Шифрування здійснюється за допомогою криптографічних алгоритмів та ключів.

Захищені сайти з протоколом HTTPS.

Шифрування листування у месенджерах (end-to-end).

Захист Wi-Fi мереж паролем.

Система числення

Система числення — це спосіб запису чисел за допомогою обмеженого набору символів та правил їх використання.

Комп'ютери працюють у двійковій системі числення, де використовуються лише цифри 0 і 1.

Двійкові цифри відповідають двом станам електронних елементів (наприклад, «є сигнал» / «немає сигналу»).

Усі дані — числа, текст, зображення, звук — у пам'яті комп'ютера подаються послідовностями бітів (0 і 1).

Сучасні технології дають змогу збирати й аналізувати колосальні обсяги даних, створювати криптовалюти, блокчейн-системи та нові цифрові сервіси.

Але разом із технічним прогресом зростає відповідальність за етичне зберігання й використання цих даних, повагу до приватності та прав людини.

Практична робота №65

Дослідницька робота

Як про мене збирають дані онлайн

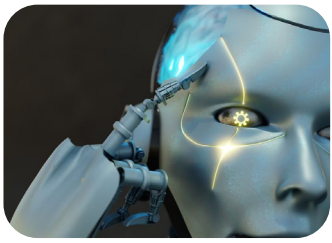
1. Оберіть 2–3 онлайн-сервіси, якими реально користуєтеся: це можуть бути, наприклад, **Google** / **YouTube**, **Instagram**, **TikTok**, **Google Maps**, онлайн-магазин тощо.
2. Для кожного сервісу знайдіть інформацію про збір даних. Зайдіть у налаштування профілю та розділ **Конфіденційність / Privacy**.
3. Зробіть таблицю в **Google Docs** або **Google Sheets** з такими колонками:
   - «Назва сервісу»
   - «Які дані про мене збирають»
   - «Для чого це можуть використовувати» (реклама, рекомендації, аналітика, безпека тощо)
   - «Можливі ризики для користувача» (втрата приватності, таргетинг, витік даних, маніпуляції).
4. Заповніть таблицю для обраних сервісів. Пишіть своїми словами, не копіюйте довгі фрагменти з сайту.
5. Зробіть один слайд у **Canv**а з назвою «Мій цифровий профіль у сервісах», де коротко підсумуйте:
   - Які типи даних про вас збирають найчастіше.
   - Що вас найбільше здивувало або занепокоїло.
   - 2–3 речення висновку: що ви змінили б у своїх налаштуваннях чи поведінці онлайн.

Підготуйтеся коротко представити свою роботу (1–2 хвилини): назвіть обрані сервіси, один приклад з таблиці та свій головний висновок.

Ключові поняття

Штучний інтелект
Генерація зображень
Робота із текстом, мовою, перекладом

Штучний інтелект (ШІ, штучний розум, *англ. artificial intelligence*, **AI)** — це здатність обчислювальних систем виконувати завдання, які зазвичай співмірні з можливостями людського інтелекту (навчання, міркування, розв'язання питань, сприйняття та ухвалення рішень).



Штучний інтелект (ШІ) — це галузь комп'ютерних наук, яка займається розробкою систем, здатних виконувати завдання, що вимагають інтелектуальних здібностей. ШІ прагне створити комп'ютерні програми та алгоритми, які здатні розв'язувати складні завдання, подібні до тих, які розв'язує людський мозок.

Чому взагалі з'явився ШІ

Накопичення величезних обсягів даних.

Людина не може самостійно опрацьовувати ті обсяги інформації, які генерують інтернет, камери, фінансові системи, медицина тощо. Потрібні були системи, які здатні автоматично аналізувати й узагальнювати дані.

Складність реальних задач.

Є багато задач, які неможливо розв'язати звичайною логікою «якщо А — зроби Б». Наприклад:

- розпізнати, що на фото саме кіт, а не собака;
- зрозуміти, що людина сказала на записі, навіть якщо в неї акцент чи шум у кімнаті;
- побачити на медичному знімку, чи є ознаки хвороби;

Для таких задач неможливо просто написати список правил типу: «якщо в людини такі-то риси обличчя — це точно вона».

Тому й з'явився інший підхід: система дивиться на багато прикладів (тисячі фото, записів, знімків), шукає в них спільні ознаки й сама вчиться

помічати закономірності, замість того щоб ми вручну прописували всі правила.

Розвиток обчислювальної техніки.

Ідеї ШІ існували ще в середині XX ст., але тільки поява потужних процесорів, графічних карт і хмарних обчислень зробила можливим навчання складних моделей на мільйонах прикладів.

Де межа можливостей ШІ

- ШІ не має свідомості, емоцій чи «власної волі».
- Він не «розуміє» світ, як людина, а працює зі структурами даних та ймовірностями.
- Його сила — у швидкому аналізі великих масивів інформації й пошуку закономірностей, які людям важко помітити.
- Його слабкість — залежність від якості даних, можливість помилок.

Машинне навчання — це підхід у ШІ, коли комп'ютер не просто виконує готову програму, а вчиться на прикладах.

Поступово модель налаштовує свої параметри так, щоб менше помилятися.

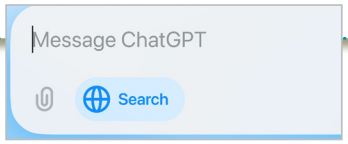
Де ми стикаємося із ШІ щодня

- Стрічка рекомендацій у YouTube, TikTok, Instagram.
- Автоматичний переклад текстів і субтитрів.
- Фільтри спаму у пошті.
- Пропозиції товарів в інтернет-магазинах.
- Навігаційні програми і прогноз часу в дорозі.
- Розпізнавання облич і відбитків для розблокування телефона.

Чат-боти та текстові моделі

ChatGPT (OpenAI)

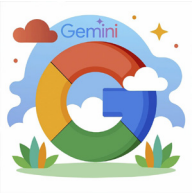
- *Що вміє:* відповідати на запитання, пояснювати теми, генерувати тексти, ідеї, коди.
- *Доступ:* є безкоштовна версія через браузер (потрібен акаунт).



ChatGPT

Google Gemini

Що вміє: відповіді на запитання, пояснення, генерація тексту, зображень (частково).
Доступ: безкоштовно через Google-акаунт.



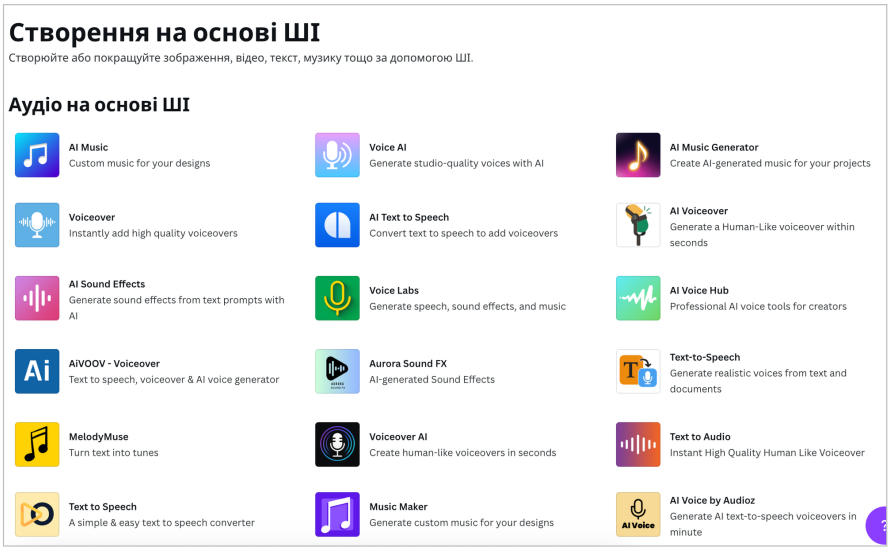
Gemini

Генерація зображень

Canva

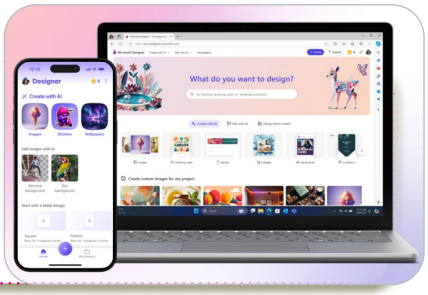
Canva (вбудований ШІ)

Інструменти: Text to Image, Magic Media, Magic Design.
Доступ: має безкоштовний тариф, частина AI-опцій — із лімітами, але спробувати можна.



Microsoft Designer

Генерація картинок за текстовим описом.
Доступ: безкоштовно з акаунтом **Microsoft**, працює у браузері.

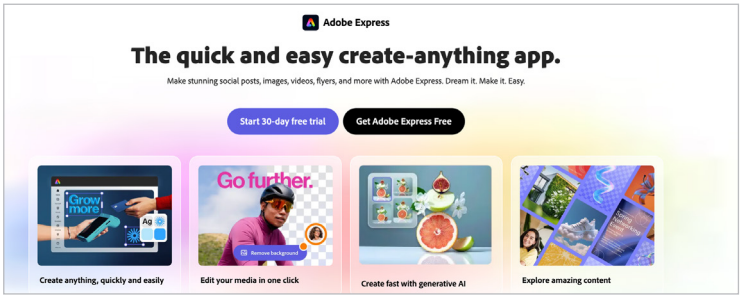


Adobe Express

Adobe Express

Генерує зображення за підказкою, можна далі редагувати.

Доступ: є безкоштовний план, базові можливості відкриті.



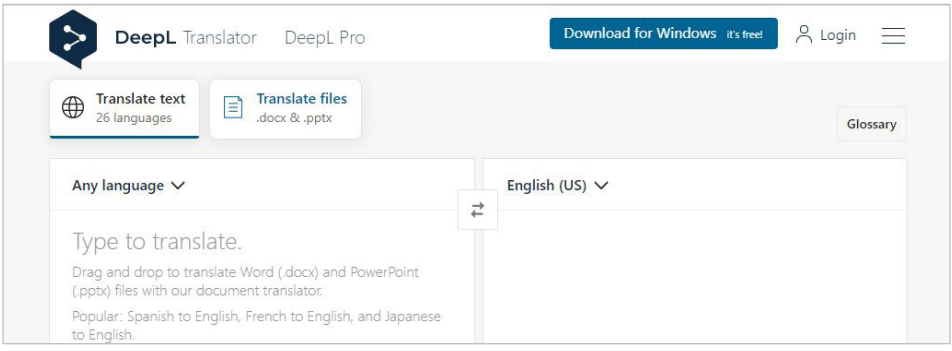
Робота із текстом, мовою, перекладом

DeepL Translator

Якісний машинний переклад + пояснення.

Доступ: безкоштовно з обмеженнями, у браузері.

DeepL



Grammarly / LanguageTool

Перевірка граматики й стилю англійських текстів.

Доступ: безкоштовні плани.

Практичні роботи №66-67

Практична робота №1

Один запит — різні відповіді

1. Оберіть одне навчальне запитання з інформатики або математики. Приклади:
- «Що таке двійкова система числення?»
- «Що таке алгоритм?»
- «Чим відрізняється змінна від константи?»

2. Сформулюйте це запитання одним реченням українською мовою.

3. Виберіть два різні ШІ-сервіси із поданих які запропонує вчитель.
- **ChatGPT**
- **Google Gemini**
- **Microsoft Copilo**t або інші

4. Введіть абсолютно однакове запитання у кожен із двох чат-ботів окремо.
- Не змінюйте формулювання між сервісами.
- Збережіть або скопіюйте обидві відповіді.

5. Порівняйте відповіді двох ШІ-систем. Поміркуйте і зробіть висновок.
- Яка відповідь була зрозумілішою для вас? Чому?
- Де було більше прикладів або пояснень?
- Чи є розбіжності в інформації (різні визначення, пропущені деталі)?
- Чи помітили помилки або неточності в якійсь із відповідей?

6. Створіть один слайд у **Canva** з коротким підсумком.
- У верхній частині: напишіть своє запитання.
- Ліворуч: коротко запишіть сильні сторони першого сервісу (2–3 пункти).
- Праворуч: сильні сторони другого сервісу (2–3 пункти).
- Внизу: зробіть власний висновок у 2–3 реченнях:

  Який сервіс вважаєте кориснішим саме для навчання і чому.
  Чи можна повністю довіряти відповіді ШІ без перевірки.

7. Підготуйтеся усно пояснити свій слайд класу:
- яке було запитання;
- чим відрізнялися відповіді;
- який висновок зробили щодо використання ШІ для навчання.

Практична робота №2

Крок 1. Створення таблиці для порівняння

1. Відкрийте **Google Docs** або **Google Sheets**.
2. Створіть таблицю з колонками:

№	Фрагмент тексту	Google Translate	DeepL	Чат-бот ШІ

3. Таблицю поки залиште порожньою — заповните її після перекладу.

Крок 2. Пошук тексту про українські символи.

1. Відкрийте браузер і знайдіть невеликий текст (5–7 речень) українською мовою на тему, пов'язану з Україною, наприклад:
   - про державні символи України (прапор, герб, гімн);
   - про національні символи (калина, вишиванка, тризуб, верба тощо).

Використовуйте **надійні джерела**: енциклопедії, освітні портали, офіційні сайти.

2. Скопіюйте один абзац (5–7 речень) у свій документ — це буде оригінальний текст.

Крок 3. Переклад фрагмента трьома способами.

Перекладіть обраний фрагмент тексту про українські символи трьома різними способами англійською мовою:

- через **Google Translate**;
- через інший онлайн-перекладач (наприклад, **DeepL**);
- за допомогою чат-бота ШІ (**ChatGPT** / **Gemini** / **Copilot** – за вказівкою вчителя).

Збережіть усі три варіанти перекладу (у тому ж документі), щоб потім їх порівнювати.

Крок 4. Заповнення таблиці та порівняння.

1. Розбийте свій текст на **3–5 речень або логічних фрагментів**.
2. Для кожного речення/фрагмента заповніть рядок таблиці.

Крок 5. Висновок.

Напишіть короткий висновок (**5–7 речень**), у якому дайте відповідь на запитання:

- Який сервіс, на вашу думку, переклав найкраще цей текст і чому?
- Чи були місця, де переклад був неточним або дивним?
- Чи можна повністю довіряти перекладу ШІ без перевірки людиною?
- У яких випадках онлайн-перекладачі справді корисні, а де без знання мови не обійтися?

Правові відносини у сфері інформаційних технологій. Навчання у цифрову епоху. Порівняння сервісів електронного навчання. Укладання списку електронних сервісів підтримки навчання

Теми 68-70

Ключові поняття

Правовідносини у сфері ІТ-права
Академічна доброчесність
Цитування

Правовідносини у сфері ІТ-права — це особливий вид правових відносин, які виникають, змінюються та припиняються між різними суб'єктами з приводу набуття, здійснення та захисту прав інтелектуальної власності, права на інформацію та інших прав в інформаційному середовищі.

Мета — захистити права людини, інтелектуальну власність, персональні дані, інформаційну безпеку.

«Закон України "Про захист персональних даних"».
«Закон України "Про авторське право і суміжні права"».
«Закон України "Про інформацію"»

Права користувача в цифровому середовищі	Обов'язки користувача
Право на приватність та захист персональних даних.	Дотримуватися авторського права: не копіювати чужих матеріалів без дозволу і посилання.
Право на доступ до інформації.	Не поширювати фейкову інформацію, мову ворожнечі, кібербулінг.
Право на авторство власних текстів, фото, відео, проєктів.	Обережно ставитися до персональних даних інших людей, не викладати без згоди.

Приклади порушень прав підлітків:

- викладення фото без згоди;
- створення фейкових сторінок з чужим ім'ям;
- цькування в чатах.

Академічна доброчесність — це сукупність етичних принципів та визначених законом правил, якими мають керуватися учасники освітнього процесу під час навчання, викладання та провадження наукової діяльності.

1. Самостійне виконання навчальних завдань
2. Посилання на джерела інформації
3. Надання достовірної інформації про результати власної діяльності
4. Відповідальність за власні вчинки і чесне здобуття оцінок

Порушенням академічної доброчесності вважаються:

Плагіат

Академічний плагіат — оприлюднення наукових результатів, отриманих іншими особами, як результатів власного дослідження та/або відтворення опублікованих текстів інших авторів без зазначення авторства.

Списування

Списування — виконання письмових робіт із залученням зовнішніх джерел інформації, крім дозволених для використання, зокрема під час оцінювання результатів навчання.

Хабарництво

Хабарництво — надання учасником освітнього процесу коштів, майна, послуг, пільг чи будь-яких інших благ з метою отримання неправомірної переваги в освітньому процесі.

Необ'єктивне оцінювання

Необ'єктивне оцінювання — свідоме завищення або заниження оцінки результатів навчання здобувачів освіти.

Фальсифікація результатів досліджень

Фальсифікація — свідома зміна чи модифікація вже наявних даних, що стосуються освітнього процесу чи наукових досліджень.

Здавання іспитів підставними особами

Цитування

Цитування — пряме використання першоджерела з посиланням на нього.

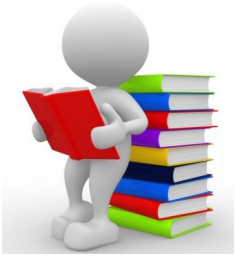
Пряме – дослівне цитування автора

Непряме – виклад думок, ідей автора своїми словами

Як правильно цитувати?

завжди використовувати лапки

повністю цитувати текст



вказати в дужках прізвище автора

цитата супроводжується посиланням на джерело

Академічна доброчесність — це чесність у навчанні: самостійне виконання завдань, коректні посилання на джерела, відсутність плагіату.

Навчання перестало бути прив'язаним тільки до школи чи університету: ми можемо вчитися через інтернет, з будь-якої точки світу.

З'явилися онлайн-курси, вебінари, платформи, мобільні додатки, які доповнюють або частково замінюють традиційні уроки.

Зростає роль цифрових навичок: робота з платформами, хмарними сервісами, онлайн-курсами, відеоконференціями.

Важливо розвивати критичне мислення: не все, що є в інтернеті, — якісне й правдиве.

Виклики навчання в цифрову епоху

Перевантаження інформацією: дуже багато курсів і матеріалів, складно обрати те, що справді якісне.

Відволікання: поруч із корисними платформами — соцмережі, ігри, розважальний контент, які заважають зосередитися.

Особливості навчання в цифровій епосі

Змішане навчання *(blended learning)* поєднання традиційних уроків та онлайн-ресурсів.

Цифрові навички як основа уміння працювати з платформами e-learning; завантажувати і здавати завдання онлайн; працювати в хмарних документах.

Типи сервісів електронного навчання

LMS (системи керування навчанням): **Google Classroom**, **Moodle**, **Microsoft Teams**

LMS (Learning Management System) – це система, у якій в одному місці зібрані:

- курси й уроки;
- матеріали (файли, відео, посилання);
- завдання та тести;
- оцінки й прогрес учнів.

Приклади: **Google Classroom**, **Moodle**, **Microsoft Teams** для освіти.

Платформи онлайн-курсів: **Prometheus, Coursera, EdEra** та інші

Платформа онлайн-курсів – це сайт або сервіс, де розміщені готові курси різних авторів / університетів:

- відеолекції;
- конспекти;
- тести й завдання;
- сертифікати після проходження.

Приклади: **Prometheus**, **Coursera**, **EdEra**, **Udemy**.

Відеосервіси: **YouTube**, освітні відеобібліотеки

Відеосервіси – це платформи, де зберігаються й програються відео, зокрема навчальні:

- уроки;
- розбирання задач;
- експерименти, демонстрації.

Приклади: **YouTube** (освітні канали), **Всеукраїнська школа онлайн.**

Тестові та ігрові платформи: **Kahoot!, Quizizz, Classtime, LearningApps, Wordwall**

Тестові та ігрові платформи – це сервіси для створення інтерактивних завдань:

- тести, вікторини, опитування;
- ігрові форми навчання (квізи, пазли, обертові колеса).

Приклади: **Kahoot!**, **Quizizz**, **Classtime**, **Wordwall**, **LearningApps**.

У цифрову епоху ми не просто користуємося технологіями, а щодня вступаємо в правові відносини, захищаємо свої дані й відповідаємо за власну цифрову поведінку.

Електронні платформи, курси й онлайн-інструменти можуть стати потужною підтримкою, якщо використовувати їх свідомо й відповідально.

Практичні роботи №68-70

Мій цифровий навчальний простір

Хід роботи

Завдання. Створіть документ або таблицю з назвою «Мій цифровий навчальний простір».

Зробіть таблицю:

Вид діяльності	Що саме я роблю онлайн	Який сервіс використо-вую	Як часто (рідко/іноді/часто)	Переваги	Які є труднощі

Заповніть таблицю для різних ситуацій:

- отримання/здавання домашніх завдань;
- пошук теорії;
- підготовка до контрольних;
- виконання проєктів;
- додаткове самонавчання (мови, IT тощо).

Вибір двох сервісів електронного навчання.

Оберіть 2 сервіси, які реально використовуєте або про які говорили на уроці, наприклад:

Google Classroom, **Moodle**, **Microsoft Teams**, **Prometheus**, **Coursera**, **EdEra**, **Kahoot!**, **Quizizz**, **Wordwall**, **LearningApps**

Критерій	Сервіс №1	Сервіс №2
Призначення (для чого)		
Основні можливості		
Мова інтерфейсу		
Доступність (безкоштов-ний?)		
Переваги		
Недоліки		

Запишіть у таблицю усі сервіси, якими користуєтеся для навчання, і розподіліть їх за категоріями, наприклад:

- для уроків і домашніх завдань;
- для конспектів і нотаток;
- для тестів і тренування навичок;
- для проєктів і презентацій;
- для саморозвитку (мови, ІТ, хобі).

Підготуйте презентацію в **Canva** та презентуйте однокласникам.

ЗМІСТ

ЖИТТЯ З РОЗУМНИМИ ПРИСТРОЯМИ

ВПОРЯДКУВАННЯ ДАНИХ

ТРИВИМІРНЕ МОДЕЛЮВАННЯ

ПРОГРАМНІ ПРОЄКТИ

БЕЗПЕКА ЦИФРОВИХ СИСТЕМ

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 9 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.22,59.
Умовн. фарбо-відб. 90,36. Обл.-вид. арк. 13,01.

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