МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
«ДНІПРОВСЬКА ПОЛІТЕХНІКА»

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інформаційних технологій та комп’ютерної інженерії

Д.В. Іванов

НАВЧАЛЬНА КОМП’ЮТЕРНА ПРАКТИКА

Методичні рекомендації
для здобувачів ступеня бакалавра
спеціальності 126 Інформаційні системи та технології

Дніпро
НТУ «ДП»
2025

Іванов Д.В.

Навчальна комп’ютерна практика [Електронний ресурс] : методичні рекомендації для здобувачів ступеня бакалавра спеціальності 126 Інформаційні системи та технології / Д.В. Іванов ; М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». – Дніпро : НТУ «ДП», 2025. – 55 с.

Автор:
Д.В. Іванов.

Затверджено науково-методичною комісією спеціальності 126 Інформаційні системи та технології (протокол №5 від 23.04.2025) за поданням кафедри інформаційних технологій та комп’ютерної інженерії (протокол №14 від 23.04.2025).

Методичні рекомендації призначені для надання допомоги здобувачам ступеня бакалавра спеціальності 126 Інформаційні системи та технології під час проходження навчальної комп’ютерної практики, підготовки та захисту звіту.

Подано основні вимоги щодо організації і проведення передатестаційної практики, змісту і структури звіту, його оформлення та захисту, критерії та порядок оцінювання.

Відповідальний за випуск завідувач кафедри інформаційних технологій та комп’ютерної інженерії В. В. Гнатушенко, д-р техн. наук, проф.

ЗМІСТ

ВСТУП

Навчальна комп’ютерна практика є невід’ємною частиною освітнього процесу підготовки фахівців за спеціальністю 126 Інформаційні системи та технології. Вона спрямована на формування у студентів практичних навичок, необхідних для проєктування, розробки, впровадження та супроводу інформаційних систем, а також для ефективного використання сучасних інформаційно-комунікаційних технологій у професійній діяльності.

Проведення навчальної комп’ютерної практики спрямовано на закріплення теоретичних знань, отриманих студентами за час навчання в університеті, набуття та вдосконалення практичних навичок і вмінь за спеціальністю 126 Інформаційні системи та технології. Види, обсяг і зміст практик визначається освітньо-професійною програмою підготовки, що відображається у навчальних планах і графіках навчального процесу. Програму складено відповідно до «Положення про проведення практики здобувачів вищої освіти Національного технічного університету «Дніпровська політехніка».

Назва практики «комп’ютерна» визначає зміст завдань, які студент повинен уміти вирішувати після проходження цієї навчальної практики.

У процесі проходження практики студенти знайомляться з сучасним прикладним програмним забезпеченням, набувають досвіду аналізу та обробки даних, моделювання процесів, автоматизації розрахунків, створення графічних матеріалів та оформлення результатів роботи згідно з вимогами стандартів. Особлива увага приділяється самостійній роботі з сучасними інструментами розробки та середовищами програмування, а також дотриманню вимог стандартів і нормативних документів у сфері ІТ.

Комп’ютерні інформаційні системи і технології – це базова технологічна основа створення інформаційних систем (ІС), що дозволяють реалізувати їх функціональні можливості у різноманітних предметних галузях діяльності майбутніх фахівців певної спеціальності. Практичні завдання, що їх отримують студенти на робочих місцях, повинні ґрунтуватися на використанні у роботі сучасних ІС. При реалізації програмних додатків для обробки і інтеграції різноманітних даних потрібно використовувати програмні середовища з мовами програмування Python, SQL, C++, Java та інші.

В освітньо-професійній програмі «Інформаційні системи та технології» здійснено розподіл програмних результатів навчання за організаційними формами освітнього процесу. Зокрема, до освітнього компонента «Навчальна комп'ютерна практика» віднесено такі програмні результати навчання:

ПР3	Використовувати базові знання інформатики й сучасних інформаційних систем та технологій, навички програмування, технології безпечної роботи в комп'ютерних мережах, методи створення баз даних та інтернет-ресурсів, технології розроблення алгоритмів і комп’ютерних програм мовами високого рівня із застосуванням об’єктно-орієнтованого програмування для розв’язання задач проектування і використання інформаційних систем та технологій

ПР6	Демонструвати знання сучасного рівня технологій інформаційних систем, практичні навички програмування та використання прикладних і спеціалізованих комп’ютерних систем та середовищ з метою їх запровадження у професійній діяльності

Методичні рекомендації містять організаційні вказівки щодо проходження практики, опис завдань, методичні настанови до їх виконання, а також критерії оцінювання результатів. Практика сприяє формуванню фахових компетентностей, розвитку аналітичного мислення, вмінь працювати у команді, самостійно приймати рішення та застосовувати отримані знання в умовах реальних виробничих завдань.

1. Мета та завдання навчальної комп’ютерної практики

У процесі проходження навчальної комп’ютерної практики здобувачі набувають компетентностей, що являють собою спеціальну структуровану сукупність знань, умінь, навичок і ставлень, що визначені освітньою програмою бакалаврів спеціальності 126 Інформаційні системи та технології.

Навчальна комп’ютерна практика сприяє поглибленню і закріпленню теоретичних знань і практичних умінь, отриманих у процесі вивчення протягом перших двох курсів навчальних дисциплін «Об’єктно-орієнтоване програмування», «Бази даних в інформаційних системах», «Комп’ютерні мережі», «Архітектура комп'ютерів», «Програмування комп’ютерних систем мовою Python».

Мета навчальної комп’ютерної практики: сформувати у здобувачів професійні компетентності шляхом опанування практичними навичками програмування мовою Python з елементами ООП, роботою з БД та роботою з мережею з відповідними інформаційними системами та технологіями.

Завдання практики:

- закріпити базові знання з інформатики й сучасних інформаційних систем та технологій, навички програмування, технології безпечної роботи в комп'ютерних мережах, методи створення баз даних та інтернет-ресурсів, технології розроблення алгоритмів і комп’ютерних програм мовами високого рівня із застосуванням об’єктно-орієнтованого програмування для розв’язання задач проектування і використання інформаційних систем та технологій;
- демонструвати знання сучасного рівня технологій інформаційних систем, практичні навички програмування та використання прикладних і спеціалізованих комп’ютерних систем та середовищ з метою їх використання у професійній діяльності.

Практика проводиться на ІІ курсі, після закінчення теоретичного навчання у весняному семестрі. Наприкінці семестру кафедра ІТКІ проводить зі студентами збори, на яких розглядаються всі питання організації і проходження практики.

Навчальна практика проводиться на кафедрі інформаційних технологій та комп’ютерної інженерії і в лабораторіях НТУ «Дніпровська політехніка», що відповідають вимогам освітньої-професійної програми (ОПП) підготовки студентів за спеціальністю 126 Інформаційні системи та технології, які володіють необхідним і достатнім обладнанням.

Тривалість навчальної практики 4 тижні (6,0 кредитів ЄКТС) та визначається затвердженим навчальним планом за спеціальністю 126 Інформаційні системи та технології.

До кінця терміну перебування на практиці студент повинний завершити складання звіту.

2. ПРАКТИЧНА РОБОТА №1. Робота з файлами в мові програмування Python

2.1. Мета роботи

Вивчити теоретичні відомості щодо роботи з файлами. Навчитися розрізняти та використовувати їх функції та особливості. Вивчити основні методи роботи з файлами. Навчитися здійснювати операції читання та запису для файлів у мові Python.

2.2. Теоретичні відомості

2.2.1. Загальні відомості

Взаємодія з файлами в мові програмування Python дає можливість зберігати інформацію, яка була оброблена програмою, в постійній пам'яті комп'ютера та мати можливість отримати доступ до цієї інформації в будь який час.

Файл – це впорядкована множина даних, що зберігається на диску та може бути використаним програми для збереження даних або ж і є самою програмою.

Існує два типи файлів: текстові та бінарні файли. Текстові файли – це файли в яких інформація представлена у текстовому вигляді, до прикладу – це файли з розширенням txt, csv, html та багато інших. *Бінарні файли* – це файли дані в яких зберігаються в бінарному вигляді.

2.2.2. Робота із текстовими файлами

У мові програмування Python існують вбудовані функції для роботи з файлами. Перед початком роботи з файлом його потрібно відкрити з допомогою функції open(file_path, mode). У цій функції є два вхідних параметри file_path та mode, file_path це шлях до файлу, а mode – це режим, в якому буде відкритий файл, список всіх доступних режимів можна знайти в таблиці 2.1.

Таблиця 2.1 – Список доступних режимів відкриття файлу

Режим	**Опис**
r	Відкриття на читання
w	Відкриття на запис, вміст файлу видаляється, якщо файла не існує буде створено новий.
x	Відкриття файлу на запис, якщо файлу не існує то буде викинуте виключення.
a	Відкриття файлу на дозаписування файлу вміст буде дописано в кінець файлу.
b	Відкриває файл в бінарному вигляді.
t	Відкриття в текстовому режимі.
+	Відкриття на читання та запис.

Режими можуть бути об'єднані, до прикладу 'rb' – читання файлу в двійковій формі. По замовчуванню рівний 'rt' тобто читання в текстовому форматі. Приклад відкриття файлу, який знаходиться поряд із кодом програми.

Приклад:

```
f = open('text.txt')
```

Після відкриття файлу на читання будуть доступні декілька способів для зчитування даних з файлу.

Перший спосіб це використання методу *read* цей метод зчитує всі дані з файлу. Приклад використання методу нижче.

Приклад:

```
f = open('text.txt')
data = f.read()
print(data)
```

Результат:

```
Hello world.
Hello people.
```

Вивід – це вміст файлу.

Ще одним способом може бути метод *readlines* даний метод зчитує всі рядки з файлу та повертає у вигляді списку. Приклад використання нижче.

Приклад:

```
f = open('text.txt')
data = f.readlines()
print(data)
```

Результат:

```
['Hello world.\n', 'Hello people.']
```

Також для зчитування файлу можна використати цикл **for** передавши в якості даних об’єкт відкритого файлу. Таким чином файл буде зчитуватись по рядку.

Приклад:

```
f = open('text.txt')
for data in f:
    print(data)
```

Результат:

```
Hello world.

Hello people.
```

Після завершення роботи із файлом його потрібно закривати, щоб зробити доступним для інших процесів. Це робиться з допомогою методу **close.** Приклад роботи з даним методом зображений нижче.

Приклад:

```
f = open('text.txt')
for data in f:
    print(data)
f.close()
```

Для того щоб відкрити файл для запису використовується модифікатор ‘w’. Після відкриття файл на запис буде доступний метод для запису у файл.

Приклад:

```
f = open('text.txt', 'w')
f.write("Hello students")
f.close()
```

Результат:

```
Hello students
```

2.2.3. Робота із бінарними файлами

Для роботи із бінарними файлами можна використати пакет *pickle* який доступний в стандартній бібліотеці Python.

Для запису у файл бінарних даних потрібно відкрити файл для цього використовуючи режим ‘wb’.

Приклад:

```
import pickle

data = "string for writing"
f = open('text.data', 'wb')
pickle.dump(data, f)
f.close()
```

Результат:

```
€EOT•SYNNULNULNULNULNULNULNULDC2string for writing".
```

В бінарні файли можна записувати не тільки рядки в бінарній формі, а й різні складні структури даних наприклад словники.

Приклад:

```
import pickle

data = {
    "one": 1,
    "two": 2,
}
f = open('text.data', 'wb')
pickle.dump(data, f)
f.close()
```

Результат:

```
€EOT•NAKNULNULNULNULNULNULNUL}"(ETXone"KSOHETXtwo"KSTXu.
```

Для того щоб зчитувати бінарні дані із файлу його подібно відкрити в режимі читання бінарних даних та використати метод **load**.

Приклад:

```
import pickle

f = open('text.data', 'rb')
data = pickle.load(f)
f.close()

print(data)
```

Результат:

```
{'one': 1, 'two': 2}
```

2.3. Програма роботи

1) Ознайомитися з теоретичними відомостями, щодо роботи із текстовими та бінарними файлами.

2) Виконати ознайомчі завдання із теоретичних відомостей.

3) Виконати завдання згідно вашого варіанту, який був виданий викладачем (таблиця 2.2). Додати коментарі до коду.

4) Оформити звіт.

Таблиця 2.2 – Варіанти завдань

№	Завдання
1	Написати програму, яка запитуватиме в користувача особисті дані (прізвище, ім’я та по-батькові, дату народження, вік, місто проживання та номер телефону) та записуватиме їх у файл.
2	Створити програму, яка буде запитувати в користувача бали з декількох предметів, зберігати їх в словнику та записувати його у файл, а друга програма зчитує дані із файлу та виводити їх на екран.
3	Створити текстовий файл в якому рядки чисел розділені пробілами (мінімум 10 рядків). Знайти суму чисел в кожному рядку та зберегти його в словник, де ключ – це номер рядка, а значення – сума чисел. Вивести даний словник на екран.
4	Написати програму, яка видаляє рядок із файлу за його номером, нумерація рядків починається з 1.
5	Написати програму, яка буде зчитувати дані із файлу та виводити їх на екран сортуючи за зростанням.
6	Створити текстовий файл в якому рядки чисел розділені пробілами (мінімум 20 рядків). Знайти різницю чисел в кожному рядку та зберегти його в словник, де ключ – це номер рядка, а значення – сума чисел. Вивести даний словник на екран.
7	Створити програму, яка буде запитувати в користувача дані про нього, зберігати їх в словнику та записувати його у файл, а друга програма зчитує дані із файлу та виводити їх на екран.
8	Написати програму, яка видаляє рядок із файлу за його номером, нумерація рядків починається з 5.
9	Написати програму, яка буде зчитувати дані із файлу та виводити їх на екран сортуючи по алфавіту.
10	Створити текстовий файл в якому рядки чисел розділені пробілами (мінімум 30 рядків). Знайти середнє значення в кожному рядку та зберегти його в словник, де ключ – це номер рядка, а значення – сума чисел. Вивести даний словник на екран.
11	Створити програму, яка буде запитувати в користувача назви предметів у даному семестрі, зберігати їх в словнику та записувати його у файл, а друга програма зчитує дані із файлу та виводити їх на екран.
12	Написати програму, яка видаляє рядок із файлу за його номером, нумерація рядків починається з 10.
13	Написати програму, яка буде зчитувати дані із файлу та виводити їх на екран сортуючи за спаданням.
14	Створити текстовий файл в якому рядки чисел розділені пробілами (мінімум 15 рядків). Знайти мінімальне число в кожному рядку та зберегти його в словник, де ключ – це номер рядка, а значення – сума чисел. Вивести даний словник на екран.

15	Створити програму, яка буде запитувати в користувача назві факультетів університету, зберігати їх в словнику та записувати його у файл, а друга програма зчитує дані із файлу та виводити їх на екран.
16	Написати програму, яка видаляє рядок із файлу за його номером, нумерація рядків починається з 15.
17	Написати програму, яка буде зчитувати дані із файлу та виводити їх на екран віднімаючи від кожного значення мінімальне серед усіх значень.
18	Створити текстовий файл в якому рядки чисел розділені пробілами (мінімум 10 рядків). Знайти максимальне число в кожному рядку та зберегти його в словник, де ключ – це номер рядка, а значення – сума чисел. Вивести даний словник на екран.
19	Створити програму, яка буде запитувати в користувача бали про хобі, зберігати їх в словнику та записувати його у файл, а друга програма зчитує дані із файлу та виводити їх на екран.
20	Написати програму, яка видаляє рядок із файлу за його номером, нумерація рядків починається з 20.
21	Написати програму, яка буде зчитувати дані із файлу та виводити їх на екран віднімаючи від кожного значення середнє серед усіх значень.
22	Створити текстовий файл в якому рядки чисел розділені пробілами (мінімум 7 рядків). Знайти мінімальне число в кожному рядку та зберегти його в словник, де ключ – це номер рядка, а значення – сума чисел. Вивести даний словник на екран.
23	Створити програму, яка буде запитувати в користувача прізвища викладачів по дисциплінам в поточному семестрі, зберігати їх в словнику та записувати його у файл, а друга програма зчитує дані із файлу та виводити їх на екран.
24	Написати програму, яка видаляє рядок із файлу за його номером, нумерація рядків починається з 25.

2.4. Контрольні питання

1) Що таке файл?
2) Які існують типи файлів?
3) Що таке текстові і бінарні файли?
4) Які існують вбудовані функції для роботи з файлами?
5) Назвіть режими файлів.
6) Охарактеризуйте пакет *pickle*.
7) Методи роботи з файлами.

3. ПРАКТИЧНА РОБОТА №2. Об'єктно-орієнтоване програмування та його принципи

3.1. Мета роботи

Вивчити теоретичні відомості, що до принципів об’єктно-орієнтованого програмування. Навчитися працювати з класами та об’єктами. Навчитись писати код в об’єктно-орієнтованому стилі.

3.2. Теоретичні відомості

3.2.1. Загальні відомості

Об’єктно-орієнтоване програмування (ООП) – це парадигма програмування в якій компоненти програми розроблені на основі зв’язані між собою об’єктів, де характеристики та взаємодія об’єктів описуються в класах.

3.2.2. Класи та об’єкти.

Клас – в об’єктно-орієнтованому програмування можна представити у вигляді креслення або карти об’єкту. Дивлячись на клас можна зрозуміти як побудований об’єкт, зрозуміти які характеристики має об’єкт, побачити його логіку роботи та зрозуміти як він взаємодіє із іншими об’єктами.

Розглянемо детальніше, що таке клас на прикладі автомобіля. З точки зору ООП автомобіль – це абстрактне поняття, яким можна описати велику кількість конкретних марок та моделей автомобілів, тобто об’єктів або їх ще називають екземплярами класу.

Таким чином *об’єкт* – це конкретний екземпляр певного класу, тобто у випадку з автомобілями – це може бути автомобіль Mazda 6 2021 року випуску.

В мові програмування Python класи можна створити з допомогою ключового слова **class**. Приклад створення класу зображений у прикладі:

Приклад:

```
class Car:
    pass
```

Використовуючи ключове слово **pass** можна пропустити оголошення класу.

Для оголошення характеристик класу в ООП існують атрибути або в англомовній термінології *properties*. Їх прийнято оголошувати в конструкторі класу.

Конструктор класу – це метод, який визивається під час створення екземпляру (про це детальніше згодом). В конструктор можна передати аргументи по аналогії із функціями в Python, та використати їх для оголошення характеристик класу. Перший аргумент **self -** це екземпляр описаного класу Car, який автоматично передається в якості першого аргументу у всі методи класу.

Таким чином можна отримати клас Car з трьома атрибутами які оголошуватимуть виробника, модель. Та рік випуску автомобіля.

Приклад:

```
class Car:
    def __init__(self, manufacturer = None, model = None, age = None):
        self.manufacturer = manufacturer
        self.model = model
        self.age = age
```

Для обробки даних або виконання якихось інших дій в класах існують методи. Методи аналоги функцій, але для класів. Також в них першим аргументом передається екземпляр об'єкту описаного класу, його прийнято іменувати **self.** Оголосимо методи для запуску, зупинки та руху автомобіля, в та реалізуємо їх. Для прикладу в якості їхньої реалізації виведемо дію яку вони реалізовуватимуть.

Приклад:

```
class Car:
    def __init__(self, manufacturer = None, model = None, age = None):
        self.manufacturer = manufacturer
        self.model = model
        self.age = age

    def start(self):
        print('Start {} car'.format(self.manufacturer))

    def stop(self):
        print('Stop {} car'.format(self.manufacturer))

    def move(self):
        print('Mode {} car'.format(self.manufacturer))
```

Таким чином було створено клас, який описує автомобіль.

Для створення екземпляра класу необхідно присвоїти його змінній, синтаксис подібний до виклику функції, у дужках можна передати аргументи, які приймає конструктор класу.

Приклад:

```
mazda_car = Car('Mazda', '6', '2021')
```

Для доступу до атрибутів об'єкту та методів використовується символ крапки.

Приклад:

```
mazda_car = Car('Mazda', '6', '2021')
print(mazda_car.manufacturer)
mazda car.start()
```

Результат:

```
Mazda
Start Mazda car
```

В класах мови програмування Python також існують так звані «чарівні» методи з допомогою них можна описати логіку роботи для роботи об’єкту із циклами, з математичними операторам або з функцією print та багато іншого, так як це реалізовано в інших типах даних в Python адже клас – це є також опис свого типу даних. Такі методи зазвичай починаються та закінчується двома нижніми підкресленнями «__». Розглянемо приклад який перетворюватиме об’єкт класу Car в типу рядок. Для цього треба реалізувати метод **__str__**.

Приклад:

```
class Car:
    def __init__(self, manufacturer = None, model = None, age = None):
        self.manufacturer = manufacturer
        self.model = model
        self.age = age

    def __str__(self):
        return "Manufacturer {},\nModel: {},\nAge: {}".format(
            self.manufacturer, self.model, self.age)

mazda_car = Car('Mazda', '6', '2021')
print(mazda_car)
```

Результат:

```
Manufacturer Mazda,
Model: 6,
Age: 2021
```

3.3. Програма роботи

1) Ознайомитися з теоретичними відомостями, що об’єктноорієнтованого програмування.

2) Виконати ознайомчі завдання із теоретичних відомостей.

3) Виконати завдання згідно вашого варіанту, який був виданий викладачем (таблиця 3.1). Додати коментарі до коду.

4) Оформити звіт.

Таблиця 3.1 – Варіанти завдань

№	Завдання
1	Створити клас студента та заповнити поля цього класу із клавіатури, вивести дані на екран.
2	Написати клас, який буде зберігати в атрибуті та змінювати баланс користувача. В класі має бути два методи: для збільшення балансу та зменшення балансу.
3	Створити клас викладача та заповнити поля цього класу із клавіатури, вивести дані на екран.

4	Написати клас, який буде зберігати в атрибуті та змінювати зарплату співробітника. В класі має бути два методи: для збільшення зарплати та зменшення зарплати.
5	Створити клас кафедри та заповнити поля цього класу із клавіатури, вивести дані на екран.
6	Написати клас, який буде зберігати в атрибуті та змінювати стипендію студента. В класі має бути два методи: для збільшення стипендії та зменшення стипендії.
7	Створити клас факультету та заповнити поля цього класу із клавіатури, вивести дані на екран.
8	Написати клас, який буде зберігати в атрибуті та змінювати кількість студентів в групі. В класі має бути два методи: для збільшення кількості студентів та зменшення кількості студентів.
9	Створити клас гуртожитку та заповнити поля цього класу із клавіатури, вивести дані на екран.
10	Написати клас, який буде зберігати в атрибуті та змінювати баланс користувача. В класі має бути метод для збільшення балансу.
11	Створити клас автомобілів та заповнити поля цього класу із клавіатури, вивести дані на екран.
12	Написати клас, який буде зберігати в атрибуті та змінювати кількість автомобілів. В класі має бути два методи: для збільшення кількості автомобілів та зменшення кількості автомобілів.
13	Створити клас літаків та заповнити поля цього класу із клавіатури, вивести дані на екран.
14	Написати клас, який буде зберігати в атрибуті та змінювати стипендію студента. В класі має бути метод для збільшення.
15	Створити клас мотоциклів та заповнити поля цього класу із клавіатури, вивести дані на екран.
16	Написати клас, який буде зберігати в атрибуті та змінювати кількість студентів на факультеті. В класі має бути два методи: для збільшення кількості студентів на факультеті та зменшення кількості студентів на факультеті.
17	Створити клас косметики та заповнити поля цього класу із клавіатури, вивести дані на екран.
18	Написати клас, який буде зберігати в атрибуті та змінювати зарплату банківських співробітників. В класі має бути два методи: для збільшення зарплати банківських співробітників та зменшення зарплати банківських співробітників.
19	Створити клас футболістів та заповнити поля цього класу із клавіатури, вивести дані на екран.
20	Написати клас, який буде зберігати в атрибуті та змінювати кількість літаків. В класі має бути два методи: для збільшення кількості літаків та зменшення кількості літаків.

21	Створити клас боксерів та заповнити поля цього класу із клавіатури, вивести дані на екран.
22	Написати клас, який буде зберігати в атрибуті та змінювати кількість мотоциклів. В класі має бути два методи: для збільшення кількості мотоциклів та зменшення кількості мотоциклів.
23	Створити клас продуктових магазинів та заповнити поля цього класу із клавіатури, вивести дані на екран.
24	Написати клас, який буде зберігати в атрибуті та змінювати кількість дипломів з відзнакою. В класі має бути два методи: для збільшення кількості дипломів з відзнакою та зменшення кількість дипломів з відзнакою.

3.5. Контрольні питання

1) Що таке об’єктно-орієнтоване програмування?
2) Що таке клас?
3) Які особливості можна зрозуміти з вигляду класу?
4) Що таке об’єкт?
5) Як створюються класи?
6) Наведіть приклад створення класу.
7) Як запустити оголошення класу?
8) Що таке атрибути?
9) Що таке конструктор класу?
10) За що відповідає self?
11) Як отримати доступ до атрибутів об’єкту та методів?
12) За що відповідає __str__?

4. ПРАКТИЧНА РОБОТА №3. Робота з базою даних із Python-програми

4.1. Мета роботи

Ознайомитися з організацією та розробити сховища даних з використанням парадигми ООП.

4.2. Теоретичні відомості

Реляційна база даних – це набір таблиць із даними.

Таблиця – це прямокутна матриця, що складається з рядків та стовпців. Таблиця визначає відношення (*relation*).

Рядок – запис, що складається з полів-стовпців. У кожному полі може міститися деяке значення або спеціальне значення NULL (порожньо). У таблиці може бути довільна кількість рядків. Для реляційної моделі порядок розташування рядків не важливий і його не визначено.

Кожний стовпець у таблиці має власне ім'я й тип.

При програмуванні на *Python* доступ до бази даних не складніший за доступ до інших джерел даних (файлів, мережевих об'єктів). Для демонстрації обрано СКБД *SQLite*, що працює як під Unix, так і під Windows. Крім установлення власне *SQLite* (сайт http://sqlite.org) та модуля з'єднання з *Python* (http://pysqlite.org), останнє необхідно для старих версій *Python*, у версії 2.6 ця СКБД йде як убудований модуль, якогось додаткового налаштування виконувати не потрібно, тому що *SQLite* зберігає дані бази в окремому файлі. Тому відразу братися за створення таблиць, занесення до них даних та виконання запитів не можна. Обрана СКБД (у силу своєї "легкості") має одну істотну особливість – за одним невеликим винятком, СКБД *SQLite* не звертає уваги на типи даних (вона зберігає всі дані у вигляді рядків), тому модуль розширення sqlite3 для *Python* виконує додаткову роботу з перетворення типів.

Схематично робота з базою даних може виглядати приблизно так:

- під'єднання до бази даних (виклик *connect*() з отриманням об'єкта-з'єднання);
- створення одного або декількох курсорів (виклик методу об'єкта-з'єднання *cursor*() з отриманням об'єкта-курсора);
- виконання команди або запиту (виклик методу *execute*() або його варіантів);
- отримання результатів запиту (виклик методу *fetchone*() або його варіантів);
- завершення транзакції або її відкочування (виклик методу об'єкта-з'єднання *commit*() або *rollback*());
- коли всі необхідні транзакції виконано, підключення закривається викликом методу *close*() об'єкта-з'єднання.

Спочатку потрібно імпортувати модуль sqlite3 і створити зв'язок з базою даних. Можете передати назву файлу або просто використовувати спеціальну рядок ": memory:" для створення бази даних в пам'яті. У нашому випадку, створюємо його на диску в файлі під назвою mydatabase.db.

```
import sqlite3

conn = sqlite3.connect("mydatabase.db")  # або :memory: щоб зберегти в RAM
cursor = conn.cursor()

# Створення таблиці
cursor.execute("""CREATE TABLE albums
                  (title text, artist text, release_date text,
                  publisher text, media_type text)
                  """)
```

Далі створюємо об'єкт cursor, який дозволяє взаємодіяти з базою даних і додавати записи, крім усього іншого. Тут використовуємо синтаксис SQL для створення таблиці під назвою альбоми з п'ятьма наступними полями: title, artist, release_date, publisher і media_type. SQLite підтримує тільки п'ять типів даних: null, integer, real, text і blob. Давайте напишемо цей код і вставимо деякі дані в нашій новій таблиці. Запам'ятайте, якщо запускаєте команду CREATE TABLE, при цьому база даних вже існує, отримаєте повідомлення про помилку.

```
# Додаємо дані в таблицю
cursor.execute("""INSERT INTO albums
                  VALUES ('Glow', 'Andy Hunter', '7/24/2012',
                  'Xplore Records', 'MP3')"""
               )

# Зберігаємо зміни conn.commit()

# Вставляємо список з п'яти наборів даних використовуючи безпечний метод
"?"
albums = [('Exodus', 'Andy Hunter', '7/9/2002', 'Sparrow
Records', 'CD'),
        ('Until We Have Faces', 'Red', '2/1/2011',
'Essential Records', 'CD'),
        ('The End is Where We Begin', 'Thousand Foot
Krutch', '4/17/2012', 'TFKmusic', 'CD'),
        ('The Good Life', 'Trip Lee', '4/10/2012', 'Reach
Records', 'CD')]

cursor.executemany("INSERT INTO albums VALUES (?,?,?,?,?)", albums)
conn.commit()
```

Тут використовували команду INSERT INTO SQL щоб вставити запис в нашу базу даних. Зверніть увагу на те, що кожен об'єкт знаходиться в одинарних лапках. Це може ускладнити роботу, якщо потрібно вставити рядки, які містять одинарні лапки. У будь-якому випадку, щоб зберегти запис у базі даних, потрібно створити її. Наступна частина коду показує, як додати кілька записів за раз за допомогою методу курсора executemany. Зверніть увагу на те, що

використовуємо знаки питання (?), Замість рядків заміщення (%) щоб вставити значення. Зверніть увагу, що використання рядка заміщення небезпечно, так як може стати причиною появи атаки ін'єкцій SQL. Використання знака питання набагато краще, а використання SQLAlchemy тим більше, так як він робите все необхідне, щоб уберегти вас від правки вбудованих одинарних лапок на те, що SQLite в змозі приймати.

4.2.1. Редагування і видалення записів

Можливість оновлювати записи у вашій базі даних це ключ до того, щоб ваші дані велися акуратно, і був повний порядок. Якщо не можете редагувати дані, тоді ваша база стане марною досить скоро. Іноді вам, в тому числі, потрібно буде видаляти і рядки. Створимо новий скрипт в тій же директорії, де був створений попередній

```
import sqlite3

conn = sqlite3.connect("mydatabase.db") cursor = conn.cursor()
 sql = """ UPDATE albums
SET artist = 'John Doe'
WHERE artist = 'Andy Hunter'
"""  cursor.execute(sql) conn.commit()
```

Тут використовували команду SQL UPDATE, щоб оновити таблицю альбомів. Тут можете використовувати команду SET, щоб змінити поле, так що в нашому випадку змінимо ім'я виконавця на John Doe в кожного запису, де поле виконавця зазначено для Andy Hunter. Зверніть увагу на те, що якщо не підтвердите зміни, то вони не будуть внесені в базу даних. Команда DELETE настільки ж проста.

```
import sqlite3

conn = sqlite3.connect("mydatabase.db") cursor = conn.cursor()

sql = "DELETE FROM albums WHERE artist = 'John Doe'"
 cursor.execute(sql) conn.commit()
```

Видалення ще простіше, ніж оновлення. У SQL це займає всього дві строчки. В даному випадку, все, що потрібно зробити, це вказати SQLite, з якої таблиці видалити (albums), і яку саме запис за допомогою пункту WHERE. Таким чином, було виконано пошук запису, в якій присутня ім'я "John Doe" в поле виконавців, після чого ці дані були видалені.

4.2.2. Основні запити SQLite

Запити в SQLite дуже схожі на ті, які використовуєте в інших базах даних, таких як MySQL або Postgres. Використовуємо звичайний синтаксис SQL для виконання запитів, після чого об'єкт cursor виконує SQL. Ось кілька прикладів:

```
import sqlite3
 conn = sqlite3.connect("mydatabase.db") # conn.row_factory
= sqlite3.Row cursor = conn.cursor()
 print("Список всіх записів артиста Red:") sql = "SELECT *
FROM albums WHERE artist=?" cursor.execute(sql, [("Red")])
print(cursor.fetchall())  # or use fetchone()

print("Список всіх записів в таблиці:")
for row in cursor.execute("SELECT rowid, * FROM albums ORDER
BY artist"):     print(row)

print("Results from a LIKE query:")
sql = "SELECT * FROM albums WHERE title LIKE 'The%'" cursor.execute(sql)

print(cursor.fetchall())
```

Перший запит, який виконали, називається SELECT *, що означає, що хочемо вибрати всі записи, які підходять під передане ім'я виконавця, в нашому випадку це "Red". Далі виконуємо SQL і використовуємо функцію fetchall () для отримання результатів. Також можете використовувати функцію fetchone () для отримання першого результату. Зверніть увагу на те, що тут є прокоментований розділ, пов'язаний з таємничим row_factory. Якщо не прокоментуєте цей рядок, результат повернеться, так як об'єкти Row, подібні словників Python і дають доступ до полів рядків точнісінько, як і словник. У будь-якому випадку, не можете виконати призначення пункту, використовуючи об'єкт Row. Другий запит дуже схожий на перший, але повертає кожну запис в базі даних і впорядковує результати по імені артиста в порядку зростання. Це також показує, як можемо зробити цикл результати видачі. Останній запит показує, як команда LIKE використовується при пошуку часткових фраз. У нашому випадку, шукали по всій таблиці заголовки, які починаються з артикля The. Знак відсотка (%) є підстановлювальний оператором.

4.3. Порядок виконання роботи і опрацювання результатів

Приклад 4.1. Збереження даних в базі даних:

```
import sqlite3

# Клас для що описує працівників підприємства class Worker():
   def __init__(self, **kwargs):
      self.name = kwargs.get('name')        self.position = kwargs.get('position')
self.work_from_date = kwargs.get('work_from_date')        self.birth_date =
kwargs.get('birth_date')
    def get_data(self):        return self.name, self.position,
self.work_from_date, self.birth_date
```

```
#  створення екземпляра класу працівник з використанням непозиційних параметрів у конструкторі
w1 = Worker(name='Petrov Vadym', position='manager', work_from_date='09-12-2019', birth_date='18-02-1990')

conn = sqlite3.connect("staff_db.db") cursor = conn.cursor()

# Створення таблиці після першого запуску скрипта потрібно закоментувати
cursor.execute("""CREATE TABLE staff
                  (name text, position text, work_from_date text,
                   birth_date text)
               """)
# Створення першого запису в таблиці "вручну" cursor.execute("""INSERT INTO staff
                  VALUES ('Ivanov Ivan', 'director', '07-10-2019',
                  '2-12-1980')"""
               )
# Створення першого запису в таблиці із екземпляра класу Worker
cursor.execute("""INSERT INTO staff
                  VALUES (?,?,?,?)""", w1.get_data()
               ) # Зберігаємо зміни
conn.commit()

# Вставляємо список з трьох працівників
all_workers = [('Borysov Mykola', 'ingeneer', '23-11-1976', '04-12-2019'),
               ('Pavlyik Inna', 'secretary', '12-09-1991', '22-01-2020'),
               ('Kolodych Leonid', 'ingeneer', '16-08-1986', '13-01-2020')]

cursor.executemany("INSERT INTO staff VALUES (?,?,?,?)", all_workers)
conn.commit()

#  Виведення на екран всіх записів
print("Записи в таблиці бази даних у вигляі списка:") sql = "SELECT * FROM staff" cursor.execute(sql) print(cursor.fetchall())

# Редагування запису для конкретного працівника sql = """
UPDATE staff
SET position = 'main ingeneer'
WHERE name = 'Kolodych Leonid'
```

```
"""  cursor.execute(sql) conn.commit()

# Виводимо список всіх інженерів print("Список всіх ingeneer:")

sql = "SELECT * FROM staff WHERE position=?" cursor.execute(sql,
[("ingeneer")])
print(cursor.fetchall())

#  Виведення на екран всіх записів

print("Список всіх записів в таблиці:")
for row in cursor.execute("SELECT rowid, * FROM staff ORDER
BY name"):
    print(row)
```

4.3. Програма роботи

1) Ознайомитися з теоретичним матеріалом.

2) Створити та запустити на виконання приклад 4.1. Додати в таблицю ще 3 записи.

3) Виконати завдання згідно варіанту, який був виданий викладачем (таблиця 4.1).

Таблиця 4.1 – Варіанти завдань

№	Завдання
1	Додати до створеного класу метод для запису даних у базу даних.
2	Додати до створеного класу метод для видалення записів із бази даних.
3	Додати до створеного класу метод для пошуку записів по заданому критерію.
4	Додати до створеного класу метод для оновлення записів у базі даних.
5	Додати до створеного класу метод для зміни структури таблиці в базі даних.
6	Додати до створеного класу метод для видалення об’єкта із бази даних.
7	Додати до створеного класу метод для видалення всіх рядків у таблиці без видалення самої таблиці.
8	Додати до створеного класу метод для додавання нових записі у базу даних.
9	Додати до створеного класу метод для оновлення існуючих записі у базі даних.
10	Додати до створеного класу метод для надання прав доступу до бази даних.
11	Додати до створеного класу метод для відкликання прав доступу до бази даних.

12	Додати до створеного класу метод для створення індексу для прискорення пошуку.
13	Додати до створеного класу метод для видалення індексу для прискорення пошуку.
14	Додати до створеного класу метод для пошуку кількості записів у таблиці бази даних.
15	Додати до створеного класу метод для пошуку суми значень записів у таблиці бази даних.
16	Додати до створеного класу метод для середнього значення записів у таблиці бази даних.
17	Додати до створеного класу метод для мінімального значення записів у таблиці бази даних.
18	Додати до створеного класу метод для максимального значення записів у таблиці бази даних.
19	Додати до створеного класу метод для групування записів із зазначенням поля таблиці бази даних.
20	Додати до створеного класу метод для фільтрації груп у таблиці бази даних.
21	Додати до створеного класу метод для задання псевдоніму для стовпця або таблиці в базі даних.
22	Додати до створеного класу метод для створення віртуальної таблиці у базі даних.
23	Додати до створеного класу метод для створення унікального ідентифікатору рядка таблиці у базі даних.
24	Додати до створеного класу метод для створення шаблону пошуку (%, _) у таблиці бази даних.

4.4. Контрольні запитання.

1) Дайте визначення поняттю "реляційна база даних".
2) Дайте визначення поняттю "таблиця".
3) Дайте визначення поняттю "рядок".
4) Які СКБД підтримує Python?
5) Що таке курсор (cursor)?

5. ПРАКТИЧНА РОБОТА №4. Розроблення програмного забезпечення з графічним інтерфейсом мовою Python

5.1. Мета роботи

Ознайомитися з організацією графічного інтерфейсу на основі бібліотеки Tkinter.

5.2. Теоретичні відомості

Tkinter – це пакет для Python, призначений для роботи з бібліотекою Tk. Бібліотека Tk містить компоненти графічного інтерфейсу користувача (graphical user interface – GUI), написані на мові програмування Tcl.

Під графічним інтерфейсом користувача (GUI) маються на увазі всі ті вікна, кнопки, текстові поля для введення, скролери, списки, радиокнопки, прапорці та ін., Які бачите на екрані, відкриваючи ту чи іншу програму. Через них взаємодієте з програмою і керуєте нею. Всі ці елементи інтерфейсу разом будемо називати віджетами (widgets).

В даний час майже всі програми, які створюються для кінцевого користувача, мають GUI. Рідкісні програми, які передбачають взаємодію з людиною, залишаються консольними. У попередніх двох курсах ми писали тільки консольні програми.

Існує безліч бібліотек GUI. Tk далеко не найпопулярніша, хоча з її використанням написано чимало проектів. Однак по ряду причин вона була обрана для Python за замовчуванням. Установчий файл Пітона зазвичай вже включає пакет tkinter в складі стандартної бібліотеки поряд з іншими модулями.

Не вдаючись в подробиці, Tkinter можна охарактеризувати як перекладач з мови Python на мову Tcl. Пишете програму на Python, а код модуля tkinter у вас за спиною переводить ваші інструкції на мову Tcl, який розуміє бібліотека Tk.

Додатки з графічним інтерфейсом користувача подієвоорієнтовані. Повинні мати уявлення про структурний і бажано об'єктно-орієнтованому програмуванні. Подієвоорієнтоване орієнтоване на події. Тобто та чи інша частина програмного коду починає виконуватися лише тоді, коли трапляється ту чи іншу подію.

Подієво-орієнтоване програмування базується на об'єктноорієнтованому і структурному. Навіть якщо ми не створюємо власних класів та об'єктів, то все-одно ними користуємося. Всі віджети - об'єкти, породжені вбудованими класами.

Події бувають різними. Спрацював часовий чинник, хтось клікнув мишкою або натиснув Enter, почав вводити текст, перемкнув радіокнопки, прокрутив сторінку вниз і т. ін. Коли трапляється щось подібне, то, якщо був створений відповідний обробник, відбувається спрацьовування певної частини програми, що приводить до якого-небудь результату.

Tkinter імпортується стандартно для модуля Python будь-яким із способів: import tkinter, from tkinter import *, import tkinter as tk. Можна імпортувати окремі класи, що робиться рідко. В даному курсі буде в основному використовуватися from tkinter import *.

Щоб написати GUI-програму, треба виконати приблизно наступне:

- Створити головне вікно.
- Створити віджети і конфігурувати їх властивостей (опцій).
- Визначити події, тобто те, на що буде реагувати програма.
- Визначити обробники подій, тобто те, як буде реагувати програма.
- Розташувати віджети в головному вікні.
- Запустити цикл обробки подій.

Послідовність не обов'язково така, але перший і останній пункти завжди залишаються на своїх місцях. Подивимося все це в дії.

Існують і інші способи створення інтерфейсів програм, написаних мовою Python. Серед них зокрема і PyQt – інтерфейс Python до бібліотеки Qt.

В сучасних операційних системах будь користувальницький додаток укладено в вікно, яке можна назвати головним, так як в ньому розташовуються всі інші віджети. Об'єкт вікна верхнього рівня створюється від класу Tk модуля tkinter. Змінну, зв'язувану з об'єктом, часто називають root (корінь):

root = Tk ()

Клас Tk є базовим і викликається за допомогою конструктора tkinter.Tk(screenName=None, baseName=None, className='Tk', useTk=1), що створює віджет верхнього рівня, який зазвичай є головним вікном додатку.

Tkinter є бібліотекою, орієнтованою на події, в яких є головний цикл оброблення подій. Для запуску такого циклу використовується метод mainloop, а для виходу – метод quit.

До віджетів бібліотеки tkinter належать:

– Toplevel – вікно верхнього рівня, що зазвичай використовується під час створення багатовіконних програм;
– Button – кнопка;
– Label – мітка (напис без можливості редагування);
– Entry – віджет, що дозволяє ввести один рядок тексту;
– Text – віджет для введення багаторядкового тексту;
– Listbox – перелік, з якого можна обрати один або декілька елементів;
– Frame – рамка для організації віджетів всередині вікна;
– Checkbutton – віджет, що дозволяє вибрати деякий пункт у вікні;
– Radiobutton – дозволяє обрати тільки один пункт у вікні;
– Scale – віджет, що дозволяє вибрати значення з деякого діапазону;
– Scrollbar – віджет, що дозволяє прокручувати інший відмет;
– Menu – віджет для створення меню, що спливає або випадає;
– Menubutton – кнопка з меню;
– Canvas – основа для виведення графічних примітивів;
– Message – віджет аналогічний Label, що дозволяє розташовувати багаторядкові тексти;
– Progressbar – віджет, який відображає статус довготривалих операцій;
– Separator – вертикальна або горизонтальна полоса розподілу;
– Sizegrip – віджет, який дозволяє користувачу змінити розмір вікна верхнього рівня;

– Treeview – ієрархічна колекція пунктів. Усі ці класи віджетів є підкласами класу Widget.

5.3. Порядок виконання роботи і опрацювання результатів.

Приклад 5.1. Демонстрація розміщення віджетів на головному вікні

```
from tkinter import * from tkinter import ttk
 root = Tk() root.title('Example')
content = ttk.Frame(root, padding=(3, 3, 12, 12)) frame = ttk.Frame(content, borderwidth=5,
            relief="sunken", width=200, height=100) namelbl = ttk.Label(content, text="Name") name =
ttk.Entry(content) onevar = BooleanVar() twovar = BooleanVar()
threevar = BooleanVar()
one = ttk.Checkbutton(content, text="One",                variable=onevar, onvalue=True) two =
ttk.Checkbutton(content, text="Two",                variable=twovar, onvalue=True) three =
ttk.Checkbutton(content, text="Three",                variable=threevar, onvalue=True) ok =
ttk.Button(content, text="OK") cancel = ttk.Button(content, text="Cancel") content.grid(column=0, row=0,
sticky=(N, S, E, W)) frame.grid(column=0, row=0, columnspan=3, rowspan=2,          sticky=(N, S, E, W))
namelbl.grid(column=3, row=0, columnspan=2, sticky=(N,                                W), padx=5)
name.grid(column=3, row=1, columnspan=2, sticky=(N, E,
                                W), pady=5, padx=5)
one.grid(column=0, row=3) two.grid(column=1, row=3) three.grid(column=2, row=3)
ok.grid(column=3, row=3) cancel.grid(column=4, row=3) root.columnconfigure(0, weight=1)
root.rowconfigure(0, weight=1) content.columnconfigure(0, weight=3)
content.columnconfigure(1, weight=3) content.columnconfigure(2, weight=3)
content.columnconfigure(3, weight=1) content.columnconfigure(4, weight=1)
content.rowconfigure(1, weight=1) root.mainloop()
```

Приклад 5.2. Відкриття текстового файлу.

```
from tkinter import * import tkinter.filedialog
def LoadFile(ev):
   fn = tkinter.filedialog.Open(root, filetypes=[('*.txt files', '.txt')]).show()    if fn == '':
      return
   textbox.delete('1.0', 'end')
   textbox.insert('1.0', open(fn, 'rt').read())
root = Tk()
panelFrame = Frame(root, height=20, bg='blue') textFrame = Frame(root, height=40, width=50)
panelFrame.pack(side='top', fill='x')
textFrame.pack(side='bottom', fill='both', expand=1) textbox = Text(textFrame, font='Arial 12',
wrap='word') scrollbar = Scrollbar(textFrame) scrollbar['command'] = textbox.yview
textbox['yscrollcommand'] = scrollbar.set textbox.pack(side='left', fill='both', expand=1)
scrollbar.pack(side='right', fill='y') loadBtn = Button(panelFrame, text='Open') loadBtn.bind("<Button-
1>", LoadFile)


loadBtn.place(x=10, y=1, width=40, height=20) root.mainloop()
```

Приклад 5.3. Додавання подій при натисненні на кнопки

```
from tkinter import *
 def triangle():    canvas.coords(r, (0, 0, 0, 0))
   canvas.itemconfig(t,  fill='yellow',  outline='white')          canvas.coords(t,  (50,  200,  340,  200,  110,  60))
text.delete(1.0, END)
   text.insert(1.0, 'Зображення трикутника')    text.tag_add('title', '1.0', '1.end')    text.tag_config('title',
font=('Times', 14), foreground='blue')
 def rectangle():    canvas.coords(t, (0, 0, 0, 0, 0, 0))
```

```
   canvas.itemconfig(r, fill='blue', outline='white')     canvas.coords(r, (80, 50, 320, 200))     text.delete(1.0,
END)
   text.insert(1.0, 'Зображення прямокутника')     text.tag_add('title', '1.0', '1.end')     text.tag_config('title',
font=('Times', 14), foreground='black')
 win = Tk()
b_triangle = Button(text="Трикутник", width=15, command=triangle)
b_rectangle = Button(text="Прямокутник", width=15, command=rectangle)
 canvas = Canvas(width=400, height=300, bg='#fff') text = Text(width=55, height=5, bg='#fff',
wrap=WORD) t = canvas.create_polygon(0, 0, 0, 0, 0, 0)
 r = canvas.create_rectangle(0, 0, 0, 0) b_triangle.grid(row=0, column=0)
b_rectangle.grid(row=1, column=0) canvas.grid(row=0, column=1, rowspan=10)
text.grid(row=11, column=1, rowspan=3) win.mainloop()
```

5.4. Програма роботи

1) Ознайомитися з теоретичним відомостями.

2) Створити та запустити на виконання приклади 5.1-5.3.

3) Виконати завдання згідно вашого варіанту, який був виданий викладачем (таблиця 5.1).

Таблиця 5.1 – Варіанти завдань

№	Завдання
1	Модифікуйте приклад 5.2: додайте в меню кнопку Save, яка дозволяє зберегти модифікований текстовий файл.
2	Модифікуйте приклад 5.3: додайте кнопку для малювання кола.
3	Модифікуйте приклад 5.3: додайте кнопку для очищення полотна.
4	Модифікуйте приклад 5.3: додайте кнопку для малювання трикутника.
5	Модифікуйте приклад 5.3: додайте кнопку для малювання квадрата.
6	Модифікуйте приклад 5.3: додайте кнопку для малювання прямокутника.
7	Модифікуйте приклад 5.3: додайте кнопку для малювання п’ятигранника.
8	Модифікуйте приклад 5.3: додайте кнопку для малювання шестигранника.
9	Модифікуйте приклад 5.3: додайте кнопку для малювання довільної фігури.
10	Модифікуйте приклад 5.3: додайте кнопку для малювання трапеції.
11	Модифікуйте приклад 5.2: додайте в меню кнопку Save As, яка дозволяє перезберегти модифікований текстовий файл.
12	Модифікуйте приклад 5.3: додайте кнопку для малювання паралелограма.
13	Модифікуйте приклад 5.3: додайте дві кнопки: для малювання кола та квадрату.
14	Модифікуйте приклад 5.3: додайте кнопку для малювання ромба.
15	Модифікуйте приклад 5.3: додайте дві кнопки: для малювання трапеції та квадрату.
16	Модифікуйте приклад 5.3: додайте кнопку для малювання прямокутного трикутнику.

17	Модифікуйте приклад 5.3: додайте кнопку для малювання рівностороннього трикутника.
18	Модифікуйте приклад 5.3: додайте дві кнопки: для малювання трикутника та квадрату.
19	Модифікуйте приклад 5.3: додайте кнопку для малювання рівнобедреного трикутника.
20	Модифікуйте приклад 5.3: додайте дві кнопки: для малювання трикутника та трапеції.
21	Модифікуйте приклад 5.2: додайте в меню кнопку Exit, яка дозволяє завершити процес редагування та вийти.
22	Модифікуйте приклад 5.3: додайте дві кнопки: для малювання трикутника та прямокутника.
23	Модифікуйте приклад 5.3: додайте кнопку для малювання восьмигранника.
24	Модифікуйте приклад 5.3: додайте дві кнопки: для малювання кола та трикутника.

5.5. Контрольні запитання.

1) Для чого використовують ГІ?
2) Назвіть основні елементи управління (віджети) бібліотеки tkinter.
3) Який метод віджету Tk() дозволяє задати мінімальні розміри вікна.
4) Який метод віджету Tk() дозволяє задати максимальні розміри вікна?
5) Який метод віджету Tk() дозволяє задати заголовок вікна?
6) Для чого використовується віджет Label ()?
7) Для чого використовується віджет Button ()?
8) Для чого використовується віджет Canvas ()?

6. ПРАКТИЧНА РОБОТА №5. Побудова графіків математичних функцій у мові Python

6.1. Мета роботи

Набуття навичок роботи з бібліотекою Matplotlib для візуалізації даних.

6.2. Теоретичні відомості

6.2.1. Matplotlib

Matplotlib – бібліотека на мові програмування Python для візуалізації даних двовимірною 2D графікою (3D графіка також підтримується). Отримувані зображення можуть бути використані як ілюстрації в публікаціях. Зображення, які генеруються в різних форматах, можуть бути використані в інтерактивній графіці, наукових публікаціях, графічному інтерфейсі користувача, веб-додатках, де потрібно будувати діаграми (англ. plotting).

Бібліотека Matplotlib побудована на принципах ООП, але має процедурний інтерфейс pylab, який надає аналоги команд MATLAB.

Пакет підтримує багато видів графіків і діаграм:

- Графіки (line plot)
- Діаграми розсіювання (scatter plot)
- Стовпчасті діаграми (bar chart) і гістограми (histogram)
- Секторні діаграми (pie chart)
- Діаграми «Стовбур-листя» (stem plot)
- Контурні графіки (contour plot)
- Поля градієнтів (quiver)
- Спектральні діаграми (spectrogram)

Набір підтримуваних форматів зображень, векторних і растрових, можна отримати з словника FigureCanvasBase.filetypes. Типові підтримувані формати: EPS, EMF, JPEG, PDF, PNG, PostScript, RGBA, SVG, SVGZ, TIFF.

6.2.2. Налаштування вигляду графіків

Крім того, щоб просто побудувати криву, було б добре її назвати, позначити осі, вивести легенду (це особливо стане в нагоді, якщо будувати кілька графіків). Крім того, інколи потрібно змінити вигляд самої кривої, межі її побудови (рис. 6.1).

```
from numpy import *
import matplotlib.pyplot as plt
 t = linspace(0, 3, 51) y = t * exp(-t ** 2)
plt.plot(t, y, 'r-.', label='t*exp(-t^2)')
plt.axis([0, 1, -0.05, 0.5])  # задання [xmin, xmax, ymin, ymax]
plt.xlabel('t')   # позначення вісі абсцис plt.ylabel('y')   # позначення е вісі ординат
plt.title('Графік функції')   # назва графіка plt.legend()   # вставка легенди (тексту в label)
plt.show()
```

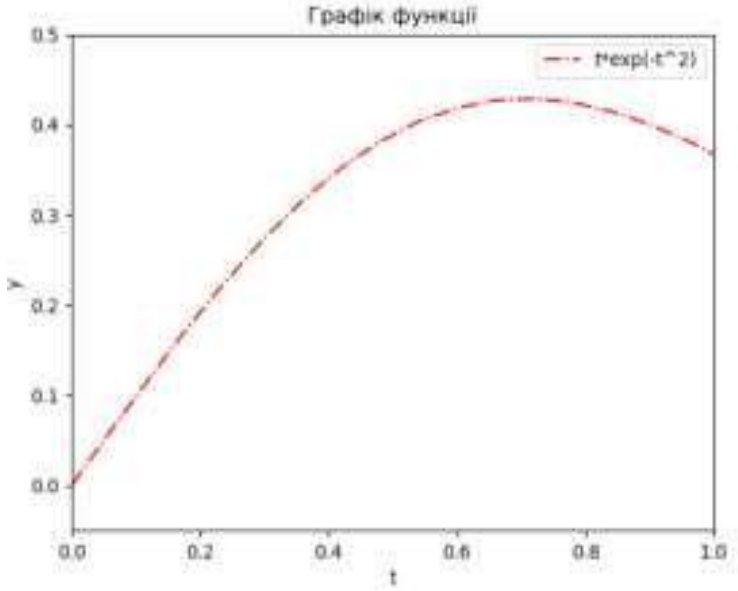


Рисунок 6.1 – Результат налаштування вигляду графіків

Крім зазначених нововведень, позначених в коментарях, в аргументах функції plot()бачимо два нових. Останній задає текст легенди графіка. Строковий аргумент r-. відповідальний за те, що змінився вигляд кривої. За замовчуванням цей аргумент b- що означає синю (blue) суцільну лінію. Допустимі наступні значення приведені в табл. 6.1.

Таблиця 6.1 – Допустимі значення функції plot()

b, blue	синій колір
c, cyan	блакитний колір
g, green	зелений колір
k, black	чорний колір
r, red	червоний колір
w, white	білий колір
y, yellow	жовтий колір
-	суцільна лінія
--	штрихова лінія
-.	штрих-пунктирна лінія
:	пунктирна лінія

За замовчуванням легенда розташовується в правому верхньому куті, але можна її і перенести за рахунок аргументу loc. Цьому аргументу можна привласнювати і чисельне значення, але звичайно легше сприймається рядок. У таблиці нижче приводяться можливі варіанти (рис. 6.2).

Таблиця 6.2 – Варіанти розміщення легенди

Місце	String	Code
кращий варіант	best	0
вгорі справа	upper right	1
вгорі зліва	upper left	2
внизу справа	lower left	3
внизу зліва	lower right	4
справа	right	5
посередині зліва	center left	6
посередині справа	center right	7
посередині внизу	lower center	8
посередині вгорі	upper center	9
посередині	center	10

6.2.3. Маркери

Тут також показано як можна об'єднувати відразу три графіка в одній інструкції. Крім того, видно, що можна не тільки використовувати маркери (y1) або лінії (y2), але і об'єднувати їх разом (y3). Найбільш часто в наукових дослідженнях і журналах призводять графіки, що відрізняються один від одного саме маркерами, тому і в matplotlib для їх позначення є безліч способів:

. точковий маркер;
, точки, розміром з піксель; o кола;
v трикутники носом вниз;
^ трикутники носом догори;
> трикутники дивляться вправо; < трикутники дивляться вліво;
s квадрати; p п’ятикутники; * зірочки; h шестикутники;
H повернені шестикутники;
+ плюси; x хрестики; D ромби; d вузькі ромби;
| вертикальні зарубки.

6.2.4. Додаткові аргументи plot()

Отже, в один аргумент можемо поставити відразу три параметра: першим вказуємо колір, другим – стиль лінії, третім – тип маркера. Однак вже така нотація може у людини незнайомої з нею, викликати подив. Крім того, вона не дозволяє розділяти параметри лінії і маркера, тому існує варіант з використанням keywords (табл. 6.3) – також це дозволяє щедра функція plot().

Можна також вносити зміни у відмітки на осях координат. Робиться це за допомогою функцій xticks() і yticks(), в які передаються один або два списки значень: або просто список згаданих значень, або їх же, але спочатку ті місця, на які вони встають:

```
x = [5, 3, 7, 2, 4, 1]
plt.xticks(range(len(x)), ['a', 'b', 'c', 'd', 'e', 'f'])  plt.yticks(range(1, 8, 2))
```

Таблиця 6.3 – Варіанти використання keywords

Keyword argument	Що міняє
color або c	колір лінії
linestyle	стиль лінії, використовуються позначення, показані вище
linewidth	товщина лінії у вигляді float-числа
marker	вид маркера
markeredgecolor	колір краю (edge) маркера
markeredgewidth	товщина краю маркера
markerfacecolorh	колір маркера
markeresize	Розмі маркера

Для нанесення сітки існує команда: plt.grid(True).

Для того, щоб одну або декілька осей виставити в логарифмічному масштабі застосовуються команди plt.semilogx() и plt.semilogy().

6.2.5. Збереження файлу

```
from numpy import *
import matplotlib.pyplot as plt
 t = linspace(0, 1, 51) y = t  * exp(-t ** 2) plt.plot(t, y)
plt.savefig('name_of_file.png', dpi=200)
```

Файл зберігається в тій же директорії з ім'ям і розширенням, зазначеним в першому аргументі. Другий необов'язковий аргумент дозволяє «на льоту» змінювати роздільну здатність картинки, що зберігається у файл.

Буває так, що дивитися на картинки в уже налаштованій програмі не потрібно і потрібно їх саме зберігати на майбутнє, щоб переглянути і порівняти їх всі разом. Тоді нам не потрібно запускати вікно перегляду результатів. Для цього до колишніх інструкцій посилаємо головному модулю повідомлення: matplotlib.use('Agg')

6.2.6. Текст, примітки

Крім тексту в назвах, підписах до осей, легенди, можна безпосередньо вставляти його в графік за допомогою простої функції text(x, y, text), де x і y координати, а text текстовий рядок. Ця функція вставляє текст відповідно до координат даних. Існує можливість вводити і в координатах графіка, в яких за (0, 0) приймається нижній лівий кут, а за (1, 1) правий верхній. Це робиться за допомогою функції figtext(x, y, text).

Текстові функції, звичайно вставляють текст в графік, але часто буває потрібно саме вказати, виділити якийсь екстремум, незвичайну точку. Це легко зробити за допомогою приміток – функції annotate('annotation', xy = (x1, y1), xytext = (x2, y2)). Тут замість annotation ми пишемо текст примітки, замість (x1,

y1) координати цікавої точки, замість (x2, y2) координати, де ми хочемо вставити текст.

6.3. Порядок виконання роботи і опрацювання результатів

Для початку необхідно інсталювати пакети numpy і matplotlib. Для цього зручно використовувати pip – це інсталятор пакетів для мови Python.

Установка pip.

Завантажуємо Пітон скрипт get-pip.py (https://bootstrap.pypa.io/get-pip.py). При зберіганні переконайтесь, що зберегли файл із розширенням .py, а не .txt.

У командій стрічці (краще PowerShell) запустіть даний файл з допомогою Python інтерпретатора:

python \path\to\get-pip.py

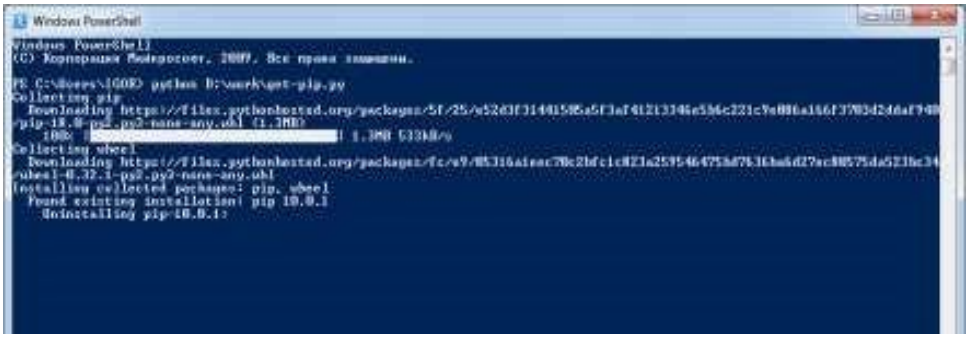


Рисунок 6.2 – Інсталяція пакетів

В даному випадку файл get-pip.py лежить в директорії D:/work.

Після запуску даної команди маєте отримати повідомлення про успішну інсталяцію pip пакет менеджера. Інакше – потрібно розбиратись: або шлях до файла вказано не правильно, або Python не у видимих для командної стрічки шляхах.

Для остаточноготесту, що pip встановлено правильно просто запускаєте в командній стрічці (PowerShell) команду pip. Без аргументів дана команда видасть документацію по використанню.

Тепер маючи Python і Pip черга за numpy і matplotlib:

pip install numpy pip install matplotlib pip install numpy

Приклад 6.1. Набір точок

```
import matplotlib.pyplot as plt
plt.plot([1, 3, 2, 4]) plt.show()
```

Функція plot() будує графік, а функція show() його показує. Аргумент, що приймається функцією plot() – це послідовність y-значень. Інший, який ми опустили, що стоїть перед y – це послідовність x-значень. Оскільки його немає, графік генерується для чотирьох зазначених y, список з чотирьох x: [0, 1, 2, 3].

Приклад 6.2. Функція

```
from numpy import *  # для використання функцій exp та linspace
import matplotlib.pyplot as plt
  def f(t):     return t ** 2 * exp(-t ** 2)

t = linspace(0, 3, 51)  # 51 точка між 0 та 3 y = f(t) plt.plot(t, y)
plt.show()
```

Приклад 6.3. Налаштування вигляду графіків

```
import matplotlib.pyplot as plt
 t = linspace(0, 3, 51) y = t ** 2 * exp(-t ** 2)
plt.plot(t, y, 'g--', label='t^2*exp(-t^2)') plt.axis([0, 3, -0.05, 0.5]) # [xmin, xmax, ymin, ymax] plt.xlabel('t')  # позначення вісі абсцис plt.ylabel('y')  # позначення е вісі ординат plt.title('My first normal plot')  # назва графіка plt.legend()  # вставка легенди (тексту в label) plt.show()
```

Приклад 6.4. Декілька кривих на одному графіку

```
from numpy import *
import matplotlib.pyplot as plt
 t = linspace(0, 3, 51) y1 = t ** 2 * exp(-t ** 2) y2 = t ** 4 * exp(-t ** 2) y3 = t ** 6 * exp(-t ** 2)
plt.plot(t, y1, 'g^',  # маркери із зелених трикутників  t, y2, 'b--',  # синя штриховая  t, y3, 'ro-')  # червоні круглі маркери
# з'єднані суцільною лінією plt.xlabel('t') plt.ylabel('y')
plt.title('Plotting with markers') plt.legend(['t^2*exp(-t^2)',  't^4*exp(-t^2)',
        't^6*exp(-t^2)'],  # список легенди  loc='upper left')  # положення легенди
plt.show()
```

6.4. Програма роботи.

1) Ознайомитися з основами візуалізації даних засобами бібліотеки Matplotlib.

2) Створити та запустити на виконання приклади 6.1-6.5, спробувати змінити параметри побудови графіків, модифікувати всі приклад так, щоб зберегти у файл отримані графіки.

3) Зобразити 2d графік функції згідно вашого варіанту, який був виданий викладачем (таблиця 6.4) та зберегти у .png файл.

Таблиця 6.4 – Варіанти завдань

№	Завдання
1	Y(x)=x*sin(5*x), x=[-2...5]
2	Y(x)=1/x*sin(5*x), x=[-5...5]
3	Y(x)=2^x*sin(10x), x=[-3...3]
4	Y(x)=x^(1/2)*sin(10*x), x=[0...5]
5	Y(x)=15*sin(10*x)*cos(3*x), x=[-3...3]

6	Y(x)=5*sin(10*x)*sin(3*x), x=[0...4]
7	Y(x)=sin(10*x)*sin(3*x)/(x^2), x=[0...4]
8	Y(x)=5*sin(10*x)*sin(3*x)/(x^(1/2)), x=[1...7]
9	Y(x)=5*cos(10*x)*sin(3*x)/(x^(1/2)), x=[0...5]
10	Y(x)=-5*cos(10*x)*sin(3*x)/(x^(1/2)), x=[0...10]
11	Y(x)=-5*cos(10*x)*sin(3*x)/(x^x), x=[0...5]
12	Y(x)=5*sin(10*x)*sin(3*x)/(x^x), x=[0...8]
13	Y(x)=x^sin(10*x), x=[1...10]
14	Y(x)=-x^cos(5*x), x=[0...10]
15	Y(x)=x^cos(x^2), x=[0...10]
16	Y(x)=cos(x^2)/x, x=[0...5]
17	Y(x)=10*cos(x^2)/x^2, x=[0...4]
18	Y(x)=(1/x)*cos(x^2+1/x), x=[1...10]
19	Y(x)=sin(x)*(1/x)*cos(x^2+1/x), x=[-2...2]
20	Y(x)=5*sin(x)*cos(x^2+1/x)^2, x=[1...10]
21	Y(x)=5*sin(1/x)*cos(x^2+1/x)^2, x=[1...4]
22	Y(x)=5*sin(1/x)*cos(x^2)^3, x=[-4...4]
23	Y(x)=cos(x^4)/x, x=[0...1]
24	Y(x)=sin(10^x)*sin(3*x)/(x^2), x=[0...1]

6.5. Контрольні запитання.

1) Які засоби мова Python надає для роботи з 2D графікою?
2) Які бібліотеки призначені для роботи з графікою?
3) Яким чином можна відобразити графік функції?
4) Яким чином можна зберегти зображення у файл?
5) Яким чином побудувати декілька кривих на одному графіку?
6) За допомогою якого параметру можна змінити колір графіка?
7) Як додати легенду до графіка? 16.6.7. Як зберегти графік функції у файл?

7. ПРАКТИЧНА РОБОТА №6. Побудова 3D графіків. Робота з mplot3d Toolkit

7.1. Мета роботи

Набуття навичок роботи з тривимірною графікою засобами мови програмування Python.

7.2. Теоретичні відомості

7.2.1. Гістограми

Для побудови гістограм (діаграм у вигляді набору стовпчиків) в Matplotlib використовуються функція bar і barh, які будують вертикальні або горизонтальні гістограми відповідно. Ці функції, як і інші функції малювання, імпортуються з модуля pylab. Функції bar і barh мають безліч необов'язкових параметрів з додатковими настройками, ми розглянемо тільки найбільш часто використовувані можливості для налаштування зовнішнього вигляду гістограм.

Функції bar і barh мають два обов'язкових параметра:

- Список координат розташування стовпчиків по осі X для bar або по осі Y для barh.
- Значення, що задають висоту (довжину) стовпчиків. Довжини цих двох списків повинні бути рівні.

```
import matplotlib.pyplot as plt import numpy as np
 data1 = 10 * np.random.rand(5) data2 = 10 * np.random.rand(5)
data3 = 10 * np.random.rand(5) locs = np.arange(1, len(data1) + 1)
width = 0.27
plt.bar(locs, data1, width=width)
plt.bar(locs + width, data2, width=width, color='red') plt.bar(locs + 2 * width, data3,
width=width, color='green') plt.xticks(locs + width * 1.5, locs) plt.show()
```

7.2.2. 3D графіка

Matplotlib дозволяє будувати 3D графіки. Імпортуємо необхідні модулі для роботи з 3D:

```
import matplotlib.pyplot as plt from mpl_toolkits.mplot3d import Axes3D
```

У бібліотеці доступні інструменти для побудови різних типів графіків.

Для побудови лінійного графіка використовується функція plot ().

*Axes3D.plot(self, xs, ys, *args, zdir='z', **kwargs)*

- *xs*: *1D* масив
- *x* координати.
- *ys*: *1D* масив
- *y* координати.

- *zs*: скалярне значення або *1D* масив
- *z* координати. Якщо переданий скаляр, то він буде присвоєний всім точкам графіка.
- *zdir: {'x', 'y', 'z'}*
- Визначає вісь, яка буде принята за *z* напрямок, за замовчуваням: *'z'*.
- ***kwargs*
- Додаткові аргументи, аналогичні тим, що використовуються в функції *plot()* для побудови двовимірних графіків.

7.2.3. Каркасна поверхня

Для побудови каркасної поверхні використовується функція *plot_wireframe(). plot_wireframe(self, X, Y, Z, *args, **kwargs)*

- *X, Y, Z: 2D* масиви

Дані для побудови поверхні.

- *rcount, ccount: int* Максимальна кількість елементів каркаса, яке буде використано в кожному з напрямків. Значення за замовчуванням: 50.
- *rstride, cstride: int* Ці параметри впливають на величину кроку, з яким будуть братися елементи рядка / стовпця з переданих масивів. Параметри rstride, cstride і rcount, ccount є взаємовиключними.
- ***kwargs* Додаткові параметри.

7.2.4. Поверхня

Для побудови поверхні використовується функція *plot_surface().*

*plot_surface(self, X, Y, Z, *args, norm=None, vmin=None, vmax=None, lightsource=None, **kwargs)*

7.3. Порядок виконання роботи і опрацювання результатів

Приклад 7.1. Побудова тривимірного графіка

```
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D import numpy as np
import matplotlib.pyplot as plt

mpl.rcParams['legend.fontsize'] = 10

fig = plt.figure() ax = fig.gca(projection='3d')
theta = np.linspace(-4 * np.pi, 4 * np.pi, 100) z = np.linspace(-2, 2, 100)
r = z**2 + 1
x = r * np.sin(theta) y = r * np.cos(theta)
ax.plot(x, y, z, label='параметрична крива') ax.legend() plt.show()
```

Приклад 7.2. Побудова графіка функції z=sin(0.3x)*cos(0.75y)

```
import pylab
```

```
from mpl_toolkits.mplot3d import Axes3D import numpy
 def makeData():     x = numpy.arange(-10, 10, 0.5)     y =
numpy.arange(-10, 10, 0.5)     xgrid, ygrid = numpy.meshgrid(x, y)

   zgrid = numpy.sin(xgrid * 0.3) * numpy.cos(ygrid * 0.75)     return xgrid, ygrid, zgrid
 if __name__ == '__main__':     x, y, z = makeData()
    fig = pylab.figure()     axes = Axes3D(fig)
   axes.plot_surface(x, y, z, rstride=1, cstride=1)     pylab.show()
```

Приклад 7.3. Побудова поверхні

```
from mpl_toolkits.mplot3d import Axes3D import matplotlib.pyplot as
plt import numpy as np

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Вхідні дані
u  = np.linspace(0, 2 * np.pi, 100)
v  = np.linspace(0, np.pi, 100) x = 10 * np.outer(np.cos(u), np.sin(v)) y = 10 *
   np.outer(np.sin(u), np.sin(v)) z = 10 * np.outer(np.ones(np.size(u)), np.cos(v))

# Побудова поверхні
ax.plot_surface(x, y, z, color='b') plt.show()
```

Приклад 7.4. Анімована побудова графіка функціїї

```
from math import  * import matplotlib.pyplot as plt import
matplotlib.animation as animation
  def data_gen(t=0):
    cnt = 0     while cnt < 1000:
       cnt += 1         t += 0.1
       yield t, sin(2*pi*t) * exp(-t/10.)
 def init():
    ax.set_ylim(-1.1, 1.1)     ax.set_xlim(0, 10)     del
xdata[:]     del ydata[:]
    line.set_data(xdata, ydata)     return line,
 fig, ax = plt.subplots() line, = ax.plot([], [], lw=2) ax.grid()
xdata, ydata = [], []
 def run(data):     # update the data     t, y = data
xdata.append(t)     ydata.append(y)     xmin, xmax =
ax.get_xlim()     if t >= xmax:
       ax.set_xlim(xmin,  2*xmax)              ax.figure.canvas.draw()
line.set_data(xdata, ydata)

    return line,
 ani = animation.FuncAnimation(fig, run, data_gen, lit=False, interval=10,
repeat=False, init_func=init) plt.show()
```

7.4. Програма роботи

1) Ознайомитися з принципами побудови тривимірних та анімованих графіків.

2) Створити та запустити на виконання приклади 7.1-7.4.

3) Зобразити гістограму частоти появи літер у певному тексті (текст зчитується із текстового файла) та зберегти у .png файл.

4) Додати анімацію до побудованого в практичній роботі №6 графіка згідно вашого варіанту, який був виданий викладачем (таблиця 7.1).

Таблиця 7.1 – Варіанти завдань

№	Завдання
1	Y(x)=x*sin(5*x), x=[-2...5]
2	Y(x)=1/x*sin(5*x), x=[-5...5]
3	Y(x)=2^x*sin(10x), x=[-3...3]
4	Y(x)=x^(1/2)*sin(10*x), x=[0...5]
5	Y(x)=15*sin(10*x)*cos(3*x), x=[-3...3]
6	Y(x)=5*sin(10*x)*sin(3*x), x=[0...4]
7	Y(x)=sin(10*x)*sin(3*x)/(x^2), x=[0...4]
8	Y(x)=5*sin(10*x)*sin(3*x)/(x^(1/2)), x=[1...7]
9	Y(x)=5*cos(10*x)*sin(3*x)/(x^(1/2)), x=[0...5]
10	Y(x)=-5*cos(10*x)*sin(3*x)/(x^(1/2)), x=[0...10]
11	Y(x)=-5*cos(10*x)*sin(3*x)/(x^x), x=[0...5]
12	Y(x)=5*sin(10*x)*sin(3*x)/(x^x), x=[0...8]
13	Y(x)=x^sin(10*x), x=[1...10]
14	Y(x)=-x^cos(5*x), x=[0...10]
15	Y(x)=x^cos(x^2), x=[0...10]
16	Y(x)=cos(x^2)/x, x=[0...5]
17	Y(x)=10*cos(x^2)/x^2, x=[0...4]
18	Y(x)=(1/x)*cos(x^2+1/x), x=[1...10]
19	Y(x)=sin(x)*(1/x)*cos(x^2+1/x), x=[-2...2]
20	Y(x)=5*sin(x)*cos(x^2+1/x)^2, x=[1...10]
21	Y(x)=5*sin(1/x)*cos(x^2+1/x)^2, x=[1...4]
22	Y(x)=5*sin(1/x)*cos(x^2)^3, x=[-4...4]
23	Y(x)=cos(x^4)/x, x=[0...1]
24	Y(x)=sin(10^x)*sin(3*x)/(x^2), x=[0...1]

7.5. Контрольні запитання.

1) Яким чином можна відобразити гістограму?

2) Який метод використовується для побудови поверхні? Які її параметри?

3) Який модуль бібліотеки matplotlib дозволяє будувати анімовані графіки?

8. ПРАКТИЧНА РОБОТА №7. Парсинг web-сторінок засобами мови Python

8.1. Мета роботи

Познайомитися з принципами розробки та парсингу сайтів. Ознайомитися із принципами роботи із HTML та CSS. Вивчити структуру DOM документів. Навчитись працювати із Django-web framework.

8.2. Теоретичні відомості

8.2.1. HTML та DOM

Всі сучасні сайти в інтернеті використовують HTML як мову розмітки. *HTML* – це мова гіпертекстової розмітки, самий базовий блок інтернет сторінки, який визначає наповнення та структуру документа.

HTML використовує «*markup*» розмітку для відображення тексту, зображень та іншого.

Після того як браузер зробить HTTP запит на сервер та отримає HTML код сторінки, браузер аналізує його та будує DOM-дерево.

DOM – це об'єктна модель документа, яку браузер створює в пам'яті комп'ютера на основі HTML, який отримав із сервера.

Браузер побудувавши DOM-дерево використовує його для відображення сторінки та надає зручний API для використання його в JavaScript для отримання із нього інформації та модифікації.

HTML сторінки складаються із тегів, які можуть буть вкладені один в одного, атрибутів та коментарів.

В документах для створення розмітки окремі теги розміщуються в середині інших, та для деяких тегів можуть бути використані атрибути (рис. 8.1).

```
<html>
  <head>
    <title>Заголовок сторінки</title>
  </head>
  <body>
    <h1 style="color: red;">Заголовок</h1>
    <div>
      Контент сторінки
    </div>
  </body>
</html>
```

Рисунок 8.1 – Приклад HTML документу

8.2.2. Парсинг

Parsing або *scraping* – конвертація даних призначена для перегляду людиною у web-браузера структуровані дані.

Для парсингу необхідно завантажити html сторінку та розпарсити її перетворивши в розпаршені структуровані дані. Для вирішення першої задачі можна використати бібліотеку *requests* та його метод *get* (рис. 8.2).

```
import requests
r = requests.get('https://nuwm.edu.ua/')
responce = r.text

print(r.status_code)
print(responce)
```

Рисунок 8.2 – Процес парсингу

Даний приклад завантажить web-сторінку та відобразить її на екрані терміналу.

Далі необхідно знайти необхідний нам елемент для прасингу, для цього можна перейти на необхідний нам сайт, для прикладу візьмемо головну сторінку сайту Національного технічного університету «Дніпровська політехніка» https://nmu.org.ua/. Знайдемо елемент із якого ми будемо парсити посилання тобто тег “a”. Для цього натиснемо правою кнопкою миші та виберемо пункт “Переглянути/Дослідити” або в англомовній версії “Inspect” (рис. 8.3) .

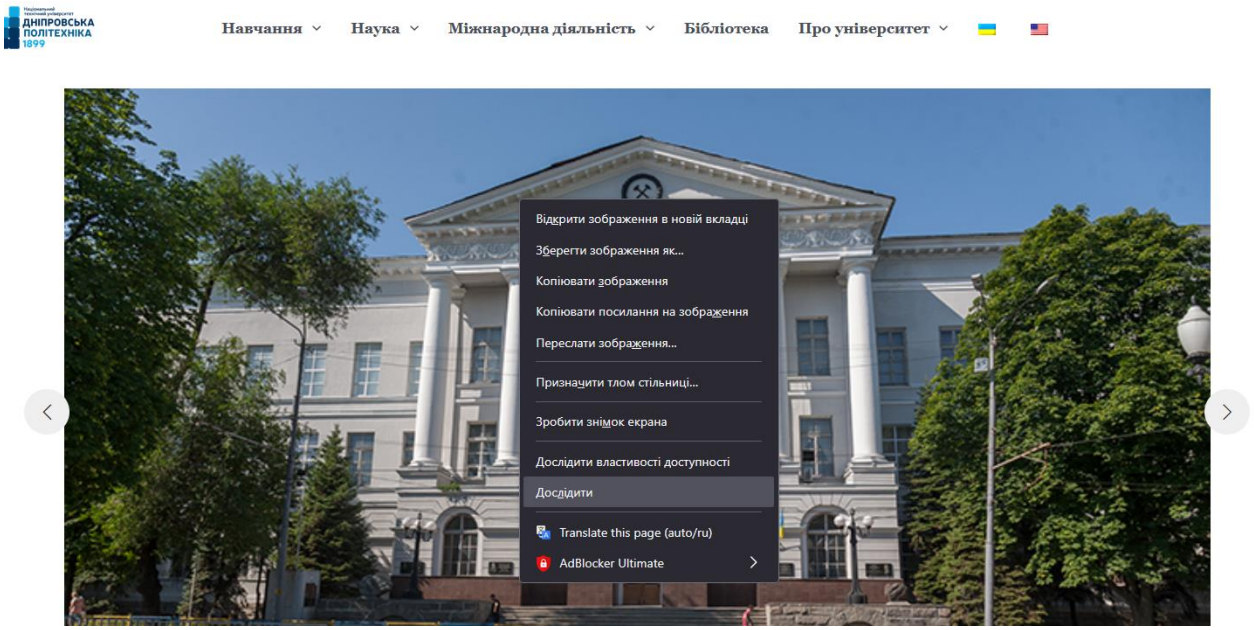


Рисунок 8.3 – Приклад перегляду даних сайту https://nmu.org.ua/

Далі потрібно знайти унікальний шлях до елементу в меню, яке відкрилось. Для цього потрібно вибирати тег та клас або id елемента (рис. 8.4).

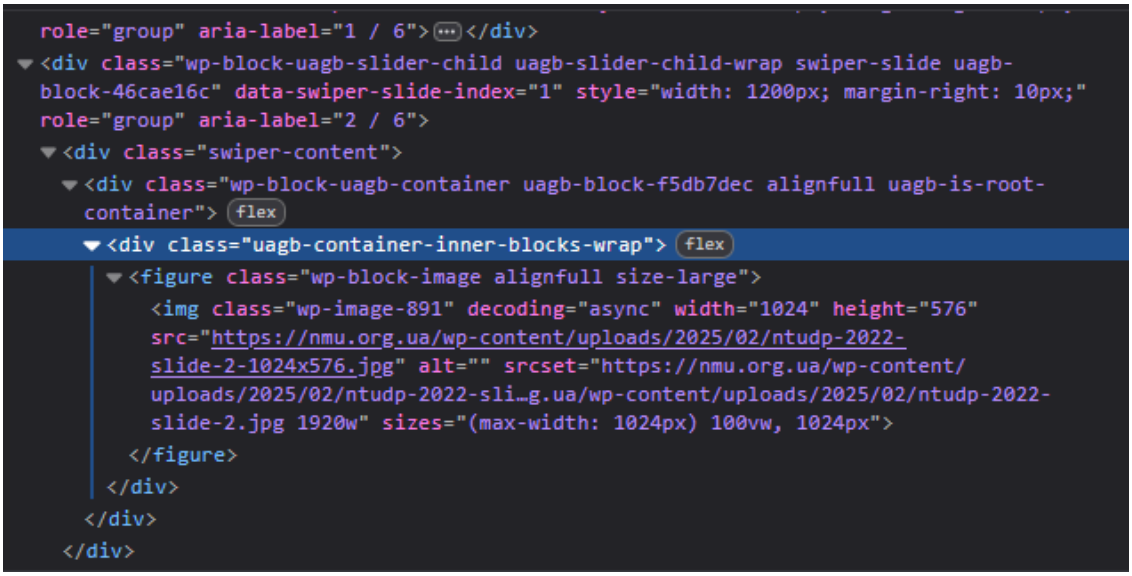


Рисунок 8.4 – Приклад знаходження id елементу

Далі використовуючи бібліотеку bs4 знайдемо всі посилання які знаходяться в обраному слайдері (рис. 8.5).

```
import requests
from bs4 import BeautifulSoup

r = requests.get('https://nmu.org.ua/' )
responce = r.text

soup = BeautifulSoup(responce)
for item in soup.find_all('div', {'class': 'carousel-item'}):
    print(item.find('a')['href'])
```

Рисунок 8.5 – Використання бібліотеки bs4

8.2.3. Django Для розробки сайтів часто використовується фреймворк Django.

Для початку роботи з Django його необхідно встановити. Для цього потрібно запустити команду:

pip install Django==3.2.3

Після встановлення потрібно створити проект для Django за допомогою: *django-admin startproject mysite* .

Створивши проект змінимо декілька налаштувань, для цього потрібно внести зміни у файл *mysite/settings.py*

У файлі *settings.py* знайдіть рядок, що містить TIME_ZONE і замініть його на ваш часовий пояс:

TIME_ZONE = 'Europe/Kiev'

Дуже часто сайти використовують бази даних для цього у файлі *settings.py* (рис. 8.6).

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}
```

Рисунок 8.6 – Використання бази даних

Щоб створити базу даних для нашого блогу, запустимо наступне в консолі: *python manage.py migrate*.

Після конфігурування та створення бази даних можна запустити веб-сервер для розробки командою: *python manage.py runserver 0:8000*

Далі у браузері необхідно відрити сторінку за посиланням http://127.0.0.1:8000/, де буде відображатись базова сторінка проекту.

8.3. Порядок виконання роботи

Завдання 1: Написати парсер який буде витягувати посилання із сайту за вашим вибором.

Завдання 2: Створити, налаштувати та запустити проект Django. У звіт вставити файл конфігурації, вивід команди *python manage.py migrate* та сркіншот стандартної сторінки проекту Django в браузері.

8.4. Програма роботи

1) Ознайомитися з теоретичними відомостями, щодо парсингу web-сторінок та розробки сайтів.

2) Виконати ознайомчі завдання із теоретичних відомостей.

3) Виконати завдання згідно вашого варіанту, який був виданий викладачем (таблиця 8.1). Додати коментарі до коду.

4) Оформити звіт.

Таблиця 8.1 – Варіанти завдань

№	Завдання
1	https://vnz.org.ua/
2	https://abit-poisk.org.ua/
3	https://abiturients.info/
4	https://pstu.edu/uk/
5	https://uu.edu.ua/
6	https://knu.ua/
7	https://knute.edu.ua/
8	https://knutd.edu.ua/
9	https://university.kse.ua
10	https://www.dnu.dp.ua/
11	http://umsf.dp.ua/
12	https://ust.edu.ua/
13	https://www.dsau.dp.ua/

14	https://duan.edu.ua/
15	https://dduvs.edu.ua/
16	https://dmu.edu.ua/
17	https://www.knu.edu.ua/
18	https://itstep.edu.ua/
19	https://medinstytut.lviv.ua/
20	https://dspu.edu.ua/
21	https://lpnu.ua/
22	https://khmnu.edu.ua/
23	http://ldfa.lviv.ua/
24	https://lma.edu.ua/

8.5. Контрольні питання

1) Що таке HTML?
2) Що таке DOM?
3) Навіщо слугує *markup*?
4) З чого складаються HTML сторінки?
5) Наведіть приклад найпростішого HTML документу.
6) Що таке *Parsing*?
7) В чому різниця між *parsing* і *scraping*?
8) Наведіть алгоритм парсингу сторінки.
9) Для початку роботи з Django потрібно…?
10) Як створити проект для Django?

9. Критерії оцінювання

Після закінчення терміну практики здобувачі вищої освіти звітують про виконання програми практики і відповідних методичних рекомендацій. Форма звітності здобувача вищої освіти про проходження практики – письмовий звіт. Звіт рецензує й затверджує керівник практики від кафедри.

Звіт здобувачів вищої освіти з навчальної комп'ютерної практики приймає керівник практики від кафедри. Керівник практики від кафедри приймає залік у здобувачів вищої освіти в університеті протягом перших двох тижнів семестру після закінчення навчальної комп'ютерної практики.

Оцінювання результатів практики здобувачів проводиться за 100-бальною шкалою з обов'язковим переведенням бальних оцінок до інституційної шкали. Оцінка за практику вноситься до заліково-екзаменаційної відомості і залікової книжки здобувача вищої освіти за підписом керівника практики від кафедри (табл. 9.1).

Таблиця 9.1 – Шкали оцінювання навчальних досягнень здобувачів НТУ «ДП»

Рейтингова	**Інституційна**
90…100	відмінно / Excellent
74…89	добре / Good
60…73	задовільно / Satisfactory
0…59	незадовільно / Fail

При оцінюванні проходження практики враховуються:

- повнота виконання вимог програми практики і відповідних методичних рекомендацій;
- зміст і якість оформлення робочих записів, добірки графічних і текстових матеріалів роботи, що представляється, а також усього звіту в цілому;
- поводження студента під час проходження практики.

Студент, що не виконав програму практики і відповідним чином одержав незадовільну оцінку при захисті звіту, направляється на практику ще раз в період канікул, а при відсутності такої можливості – відраховується з НТУ «Дніпровська політехніка».

Студенту, який не виконав програму практики з поважних причин, може бути надано право проходження практики повторно протягом наступного навчального року за індивідуальним графіком. Студент, який вдруге отримав негативну оцінку з практики, відраховується з університету.

Складовою загальної суми балів захисту практики є:

1) сума балів за зміст звіту окремо за кожним структурним розділом;

2) бали за дотримання вимог щодо змісту;

3) бали безпосередньо за захист звіту.

Система оцінювання знань студентів та звіту з навчальної практики наведена у табл.9.2.

Таблиця 9.2 – Складові загальної суми балів результатів практики

Оцінка за зміст звіту	Оцінка за дотримання вимог	Оцінка за захист звіту	Сума
40	20	40	100

Під час захисту керівник практики уважно розглядає зміст звіту, виставляє бали за зміст кожного розділу, після чого задає студентові усні запитання, які дозволяють оцінити розуміння студентом приведених у звіті положень. Виставлена загальна сума балів переводиться у традиційну оцінку і заноситься у відповідні документи як підсумкова оцінка з практики.

Шкала балів, які враховуються при виставленні підсумкової оцінки з практики, наведена в таблиці 9.3.

Таблиця 9.3 – Критерії оцінювання результатів виконання практики

№ з/п	*Результат виконання*	*Рейтингова кількість балів*	*Інституційна оцінка*
1.	- повне та вичерпне викладення матеріалу, яке використовувалося студентом під час опрацювання відповідного розділу; - повний склад необхідних додатків, які вимагаються відповідним розділом практики; - актуальність і достовірність поданої у звіті інформації; - дотримання вимог щодо змісту та оформлення структурних частин звіту.	90…100	відмінно/ Excellent
2.	- неповне викладення матеріалу або неповна відповідність змісту роботи вимогам практики (75-90% охоплення зазначених у вимогах до практики); - неповний склад додатків, які вимагаються відповідним розділом звіту (75-90% необхідних додатків); - звіт містить незначні відхилення від правил оформлення; - студент при захисті дав невпевнені відповіді на запитання викладача або помилявся.	74…89	добре/Good

3.	- неповне викладення матеріалу або неповна відповідність змісту роботи вимогам завдання з практики (50-75% охоплення зазначених у вимогах до практики); - неповний склад додатків, які вимагаються відповідним розділом звіту (50-75% необхідних додатків); - звіт містить значні відхилення від правил оформлення; - звіт поданий до захисту несвоєчасно.	60…73	задовільно /Satisfactory
4.	- одночасно присутні два чи більше критеріїв, що відповідають незадовільній оцінці; - неповне викладення матеріалу або неповна відповідність змісту роботи вимогам завдання з практики (менше 50% охоплення зазначених у вимогах до практики); - неповний склад додатків, які вимагаються відповідним розділом звіту (менше 50% необхідних додатків); - недостовірність поданої у ПЗ інформації. - звіт про виконання відсутній; - студент не з'являвся на захист.	0…59	незадовільно/ Fail

Кінцева кількість балів з визначених в таблиці 9.3 діапазонів встановлюється з урахуванням якості захисту виконаного звіту та вірних відповідей на поставлені керівником практики запитання, відповідності сформованим компетентностям навчання.

10. Техніка безпеки

На початку навчальної комп'ютерної практики здобувачі вищої освіти проходять інструктаж з техніки безпеки та охорони праці, ознайомлюються з правилами внутрішнього розпорядку підприємства, порядком отримання документації та матеріалів.

При проходженні практики студенти зобов'язані:

1) пройти індивідуальний інструктаж з охорони праці та безпеки життєдіяльності під час роботи у комп'ютерному класі;

2) дотримуватися інструкцій з охорони праці для працівників своєї професії;

3) при зміні робочого місця пройти додатковий інструктаж;

4) виконувати всі рекомендації з охорони праці, які надає керівник практики;

5) під час дії карантинних заходів, студент повинен дотримуватися всіх правил поведінки та безпеки, встановлених або рекомендованих Міністерством охорони здоров’я України, а також внутрішніми розпорядженнями та наказами керівництва НТУ “Дніпровська політехніка”.

Оскільки основна робота студента зосереджена за робочим місцем, що обладнано комп’ютером, який під’єднаний до електричної мережі, то перш за все потрібно виконувати всі норми та правила з техніки безпеки і охорони праці.

При багаторазовому порушенні студентом правил техніки безпеки та охорони праці, питання про подальше використання техніки комп’ютерного класу розглядається керівництвом кафедри та факультету з прийняттям відповідного рішення.

Список використаних джерел

1. Соколова Н.О. Методичні рекомендації до виконання курсової роботи з дисципліни «Об’єктно-орієнтоване програмування» для здобувачів ступеня бакалавра спеціальності 126 Інформаційні системи та технології / Н.О. Соколова ; М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». – Дніпро : НТУ «ДП», 2023. – 34 с.
2. Каштан В.Ю. Комп’ютерні мережі [Електронний ресурс]: методичні рекомендації до виконання лабораторних робіт для здобувачів ступеня бакалавра спеціальності 126 Інформаційні системи та технології / уклад.: В.Ю. Каштан, Я.В. Панферова, В.С. Зарічний ; М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». – Дніпро : НТУ «ДП», 2024. – 86 с.
3. Програмування комп’ютерних систем мовою Python [Електронний ресурс]: методичні рекомендації до виконання лабораторних робіт для здобувачів ступеня бакалавра спеціальності 126 Інформаційні системи та технології / уклад.: В.Ю. Каштан ; М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». – Дніпро : НТУ «ДП», 2025. – 76 с.
4. Соколова Н.О. Бази даних в інформаційних системах [Електронний ресурс] : конспект лекцій з дисципліни «Бази даних в інформаційних системах» для здобувачів ступеня бакалавра освітньо-професійної програми «Інформаційні системи та технології» зі спеціальності 126 Інформаційні системи та технології / уклад. Н. О. Соколова ; М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». – Дніпро : НТУ «ДП», 2024. – 233 с.
5. Каштан В.Ю. Комп’ютерні мережі (частина 1): навч. наоч. посіб. / В.Ю. Каштан, М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». – Електрон. дані. – Дніпро : НТУ «ДП», 2023. – Ч.1.– 371 с.
6. Каштан В.Ю. Програмування комп’ютерних систем мовою Python. Частина 1: навч. наоч. посіб. / В.Ю. Каштан, В.В. Гнатушенко, Д.В. Сущевський, Є.О. Обиденний ; М-во освіти і науки України, Нац. техн. ун-т «Дніпровська політехніка». – Електрон. дані. – Дніпро : НТУ «ДП», 2023. – 189 с.
7. Бенгтсон Р. Python для аналізу даних / Р. Бенгтсон ; пер. з англ. – Київ. : Видавництво “Піраміда”, 2023. — 528 с.
8. Шилдт Г. Основи програмування. Python / Г. Шилдт. — Львів : Видавництво Старого Лева, 2022. – 304 с.
9. Парахоняк, О. Ю. Основи об'єктно-орієнтованого програмування мовами Java та Python : навч. посіб. / О. Ю. Парахоняк. – Львів : ЛНУ імені Івана Франка, 2022. – 288 с.
10. Gagolewski, M. Minimalist Data Wrangling with Python [Електронний ресурс] / M. Gagolewski. – 2022. – Режим доступу: https://datawranglingpy.gagolewski.com
11. Walters, G. Python GUI Programming with PAGE: Create professional-looking GUIs for Python applications efficiently and effectively. – Delhi : BPB Publications, 2023. – 320 p. – ISBN 978-93-5551-234-5.

12. Pajankar, A. Hands-on Matplotlib: Learn Plotting and Visualizations with Python 3 / A. Pajankar. – 1st ed. – New York : Apress, 2022. – 220 p. – ISBN 978-1-4842-7410-1.

13. Mastering 3D Data Visualization with Python's mplot3d Library [Електронний ресурс] // Adventures in Machine Learning. – 2023. – Режим доступу: https://www.adventuresinmachinelearning.com/mastering-3d-data-visualization-with-pythons-mplot3d-library

14. Vincent, W. S. Django for Professionals: Production websites with Python & Django / W. S. Vincent. – [S.l.] : Independently published, 2023. – 300 с. – ISBN 978-1081582162.

15. Кулик, Ю. В. Платформа для обміну навчальними матеріалами з використанням фреймворку Django / Ю. В. Кулик. – Київ : Національний авіаційний університет, 2024. – 86 с. – Режим доступу: https://er.nau.edu.ua/handle/NAU/66235.

16. Парфьонов, Ю. Є. Питання міграції схеми бази даних під час супроводу веб-застосунків на базі фреймворку Django // Системи обробки інформації. – 2024. – № 2 (177). – С. 494–501. – DOI: 10.30748/soi.2024.177.07.

17. Chapagain, A. Hands-On Web Scraping with Python: Extract quality data from the web using Python libraries / A. Chapagain. – Birmingham : Packt Publishing, 2023. – 350 с. – ISBN 978-1837636211. – Режим доступу: https://www.amazon.com/Hands-Web-Scraping-Python-techniques/dp/1837636214.

18. Sahin, K. Python Web Scraping Tutorial: Step-By-Step Guide [Електронний ресурс] / K. Sahin. – 2025. – Режим доступу: https://www.scrapingbee.com/blog/web-scraping-101-with-python/.

ДОДАТКИ

Додаток 1. Вимоги до оформлення звітності

Звіт оформлюється на аркушах формату А4 та включає титульний аркуш (зразок оформлення представлений у додатку Б) та опис роботи за представленим нижче зразком.

У разі захисту звіту з виконання роботи при аудиторній формі навчання, звіт друкується та надається викладачу на розгляд.

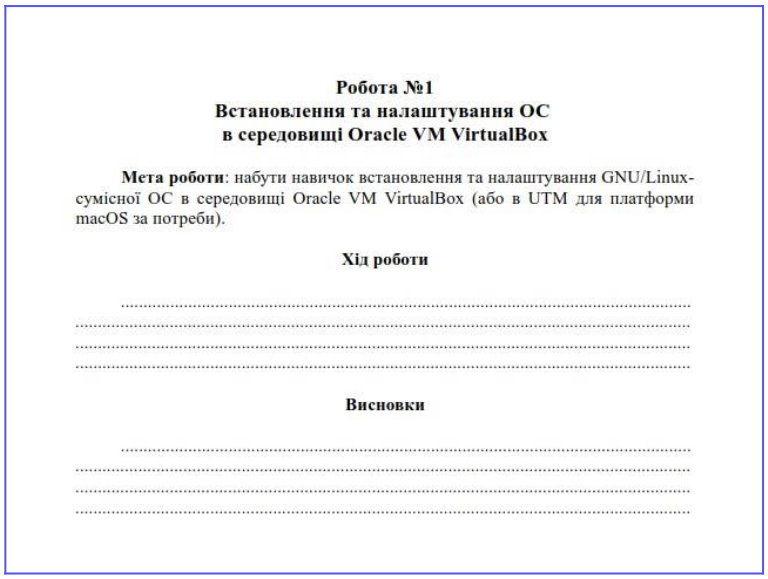
Для захисту виконаної роботи у дистанційній (online) формі навчання, звіт подається до захисту в електронному форматі PDF (Portable Document Format) та надсилається викладачеві через засоби online-спілкування для розгляду.

Опис виконаної роботи в звіті розпочинається з наступного аркуша. Вгорі сторінки, по центру, вказується номер роботи та її тема. Після цього вказується мета роботи. Через рядок по центру сторінки йдуть слова *Хід роботи* і далі через рядок розпочинається основний текст опису виконаної роботи.

Після завершення опису роботи, через рядок, йде частина, яка присвячена висновкам по проведеній роботі. Вона починається зі слова *Висновки*, розміщеного по центру листа та через рядок починається текст з висновками по роботі.

Приклад загального вмісту опису роботи представлений на рис. А.1.

Написання основного тексту ходу роботи та висновків йде **від третьої особи**!!! Тобто, є невірним використання у звіті висловів, на кшталт: “В ході роботи я розробив алгоритм розгалуження”. Замість такого вислову вірно буде написати: “В ході роботи розроблений алгоритм розгалуження”. Або замість вислову, на кшталт: “Під час виконання роботи ми опанували використання оператору розгалуження”, вірно буде написати: “Під час виконання роботи опановано використання оператора розгалуження”.



Робота №1
Встановлення та налаштування ОС
в середовищі Oracle VM VirtualBox

Мета роботи: набути навичок встановлення та налаштування GNU/Linux-сумісної ОС в середовищі Oracle VM VirtualBox (або в UTM для платформи macOS за потреби).

Хід роботи

.....
.....
.....
.....

Висновки

.....
.....
.....
.....

Рис. А.1. Приклад загального подання опису роботи

Текст звіту оформляється шрифтом Times New Roman з розміром 14, міжрядковий інтервал 1 – 1,5. Абзацний відступ – 5 символів.

Шрифтом з розміром 10 оформлюють тексти кодів програм або командних скриптів, якщо такі є в роботі. При цьому бажано використовувати один з наступних моноширинних шрифтів: Source Code Pro, Monospace, Consolas, DejaVu Sans Mono, Courier New, Monospaced, Menlo, SF Mono або подібні.

Вирівнювання тексту в звіті – по ширині сторінки.

Вирівнювання рисунків та підписів до них – по центру сторінки.

Вирівнювання таблиць (якщо такі будуть) – по центру сторінки.

Якщо код програми не є великим за обсягом, то його можна подати у вигляді знімку з екрану, причому фон повинен бути білого кольору, а текст коду програми повинен добре виділятися. Приклад такого знімку з екрану представлений на рис. А.2. Такий рисунок потрібно розмістити по центру сторінки звіту та надати по центру підпис для нього. Наприклад:

Рисунок 1 – Код програми за пунктом 1 завдання роботи №3

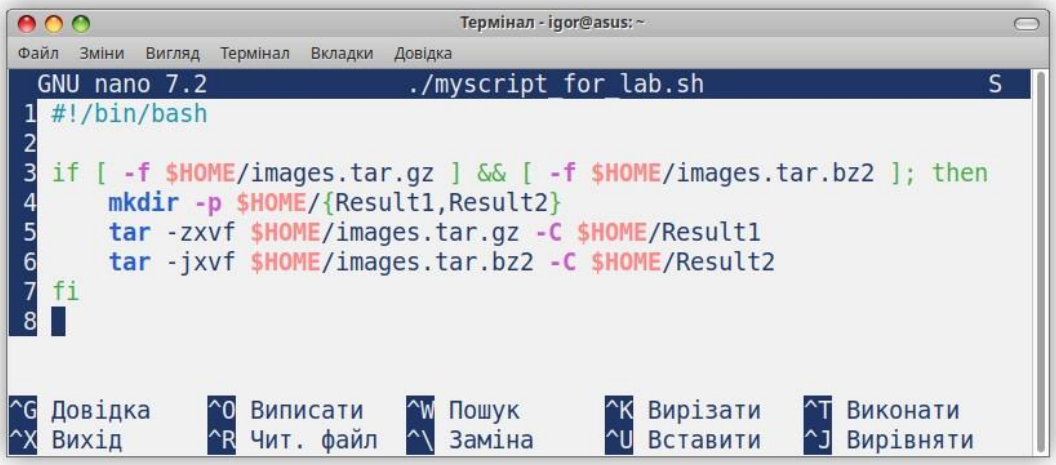


Рис. А.2. Приклад знімка з екрану вмісту
командного *bash*-скрипту для звіту

Якщо в тексті звіту наводиться рисунок, то на нього обов'язково повинно бути посилання з тексту та пояснення. Якщо поданий текст коду програми або командного скрипту, то при описі вказують номер рядку, коментуючи дії, які виконані в рядку під цим номером.

В роботах є завдання, які містять варіанти. При отриманні варіанту виконання від викладача, цей номер варіанту повинен бути відображений на початку опису ходу роботи. Якщо варіант завдання містить математичний вираз для підрахунку, то такий вираз також повинен бути присутнім у звіті. Для створення математичного виразу потрібно скористатися певними

інструментами: Microsoft Equation або Mathtype. Математичний вираз розміщують по центру сторінки та, при необхідності, нумерують. Наприклад:

$$RMSE_{3D} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(\left(x_i - x_i^0\right)^2 + \left(y_i - y_i^0\right)^2 + \left(z_i - z_i^0\right)^2\right)} \quad (1)$$

При складанні тексту висновків, потрібно вказати на результати щодо досягнення поставленої мети роботи. Вказати особливості виконання роботи та складнощі. При цьому доречно залучати вислови на кшталт: “В роботі розглянуті...”, “...виконані...”, “Отриманий результат дозволив...”, “Використання умов дозволило...” та ін.

Додаток 2. Зразок титульного аркуша

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НТУ «ДНІПРОВСЬКА ПОЛІТЕХНІКА

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інформаційних технологій та комп'ютерної інженерії

ЗВІТ

про проходження _____ практики
(навчальної комп'ютерної)

Виконавець: _____ *П.І. Іванчук*
(підпис)

Керівники	Прізвище, ініціали	Оцінка	Підпис
Доцент кафедри інформаційних технологій та комп'ютерної інженерії	Д.В. Іванов		

Дніпро
20___

Навчальне видання

Іванов Денис Валерійович

НАВЧАЛЬНА КОМП'ЮТЕРНА ПРАКТИКА

Методичні рекомендації
для здобувачів ступеня бакалавра
спеціальності 126 Інформаційні системи та технології

Видано в авторській редакції.

Електронний ресурс.
Підписано до видання 28.04.2025. Авт. арк. 3,9.

Національний технічний університет «Дніпровська політехніка».
49005, м. Дніпро, просп. Дмитра Яворницького, 19.