Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»



Звіт

З лабораторної роботи №7

З дисципліни «Кросплатформні засоби програмування»

На тему:

**ДОСЛІДЖЕННЯ БАЗОВИХ КОНСТРУКЦІЙ МОВИ PYTHON**

Виконав: Тупісь М. Н.

ст. гр.KI-304

Прийняв:

Іванов Ю.С.

Львів – 2025

**Мета**: ознайомитися з базовими конструкціями мови python та оволодіти навиками написання й автоматичного документування простих консольних програм мовою python.

**Теоретичні відомості**

Python програми – це набір \*.py файлів. Кожен файл – це окремий модуль. Модуліпідключаються за допомогою ключового слова import після якого вказується назвафайлу без розширення py. Щоб доступитися з одного модуля до іншого вони мають бути водному каталозі. Якщо модуль лежить в іншому каталозі, то щоб до нього доступитисяцей каталог має бути оформлений як пакет. Для цього достатньо у нього поміститипорожній файл \_\_init\_\_.py. При підключені пакету виконується вміст файлу \_\_init\_\_.py,який може містити визначення імен і інші необхідні для роботи з пакетом конфігураційнідії.

Python — це високорівнева, інтерпретована, динамічно типізована та кросплатформна мова програмування, яка завдяки своїй простоті та універсальності широко використовується в науці, бізнесі та розробці програмного забезпечення.

* У Python змінні не потребують явного оголошення типу — він визначається автоматично при присвоєнні значення. Основні вбудовані типи даних:
* **Числові**: int (цілі), float (дійсні), complex (комплексні);
* **Логічний тип**: bool (True, False);
* **Текстові**: str (рядки);
* **Структури даних**: list (списки), tuple (кортежі), set (множини), dict (словники).

Мова підтримує стандартні арифметичні (+, -, \*, /, //, %, \*\*), логічні (and, or, not) та оператори порівняння (==, !=, >, <, >=, <=).

Для прийняття рішень у програмі використовуються оператори:

if умова:

# блок коду

elif інша\_умова:

# інший блок

else:

# альтернативний блок

Python підтримує два основні види циклів:

* for — перебір елементів колекцій або послідовностей;
* while — виконання блоку коду, доки умова є істинною.
* Керування циклами здійснюється за допомогою break, continue, pass.

Функції визначаються ключовим словом def. Вони дозволяють організувати код у модульні блоки, які можна повторно викликати з різними аргументами.

У Python немає класичних масивів, як у C чи Java, замість них використовуються **списки**. Для створення багатовимірних структур можна застосовувати вкладені списки, у тому числі **зубчасті (нерівномірні) матриці**, де кількість елементів у різних рядках може відрізнятися.

Хід роботи

Завдання: Написати та налагодити програму на мові python згідно варіанту.

Варіант: 3



Лістинг програми мовою python:

import sys

def generate\_shaded\_array(size: int, fill\_char: str) -> list[list[str]]:

    """

    Генерує "зубчатий" масив у формі перевернутого трикутника.

    Args:

        size (int): Розмір матриці.

        fill\_char (str): Символ для заповнення.

    Returns:

        list[list[str]]: Згенерований зубчатий масив.

    """

    jagged\_array = []

    mid\_row = (size + 1) // 2

    for i in range(size):

        if i < mid\_row:

            count = size - 2 \* i

            row = [fill\_char] \* count

            jagged\_array.append(row)

        else:

            jagged\_array.append([])

    return jagged\_array

def print\_array\_to\_console(array: list[list[str]]):

    """

    Виводить масив на консоль, центрує кожен рядок.

    Args:

        array (list[list[str]]): Масив для виведення.

    """

    size = len(array)

    for row in array:

        padding = (size - len(row)) // 2

        print("  " \* padding, end="")

        print(" ".join(row))

def main():

    """

    Основна функція програми.

    """

    try:

        size\_input = input("Введіть розмір квадратної матриці (n > 0): ")

        if not size\_input.isdigit():

            print("Помилка: Розмір матриці має бути цілим числом.", file=sys.stderr)

            return

        size = int(size\_input)

        if size <= 0:

            print("Помилка: розмір матриці має бути додатнім числом.", file=sys.stderr)

            return

        fill\_char = input("Введіть один символ-заповнювач: ")

        if len(fill\_char) != 1:

            print("Помилка: потрібно ввести рівно один символ.", file=sys.stderr)

            return

        # --- Генерація та вивід масиву ---

        shaded\_array = generate\_shaded\_array(size, fill\_char)

        print("\nЗгенерований зубчатий масив:")

        print\_array\_to\_console(shaded\_array)

    except Exception as e:

        print(f"Сталася помилка під час виконання програми: {e}", file=sys.stderr)

if \_\_name\_\_ == "\_\_main\_\_":

    main()

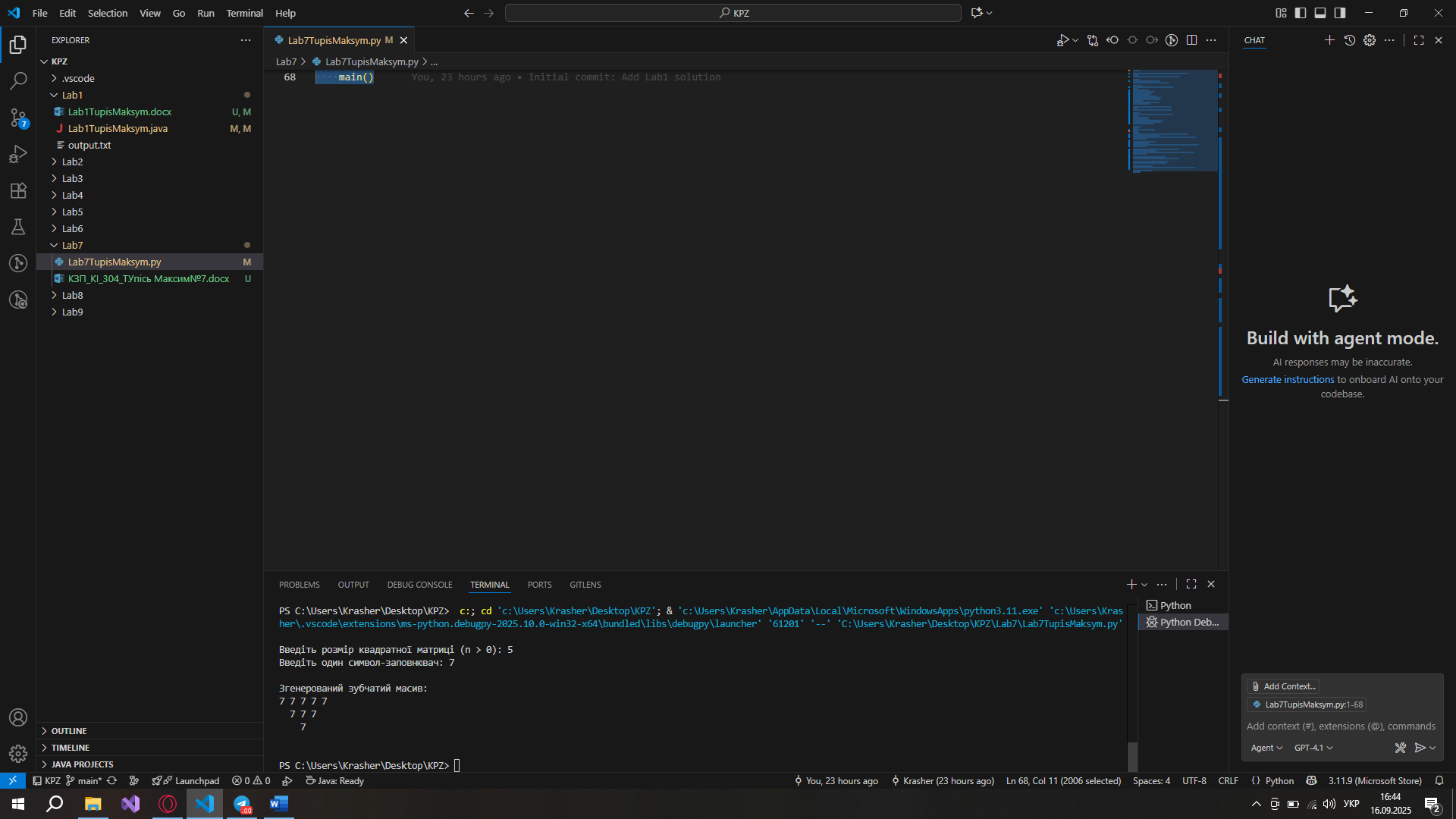


Рис.1 Приклад роботи програми

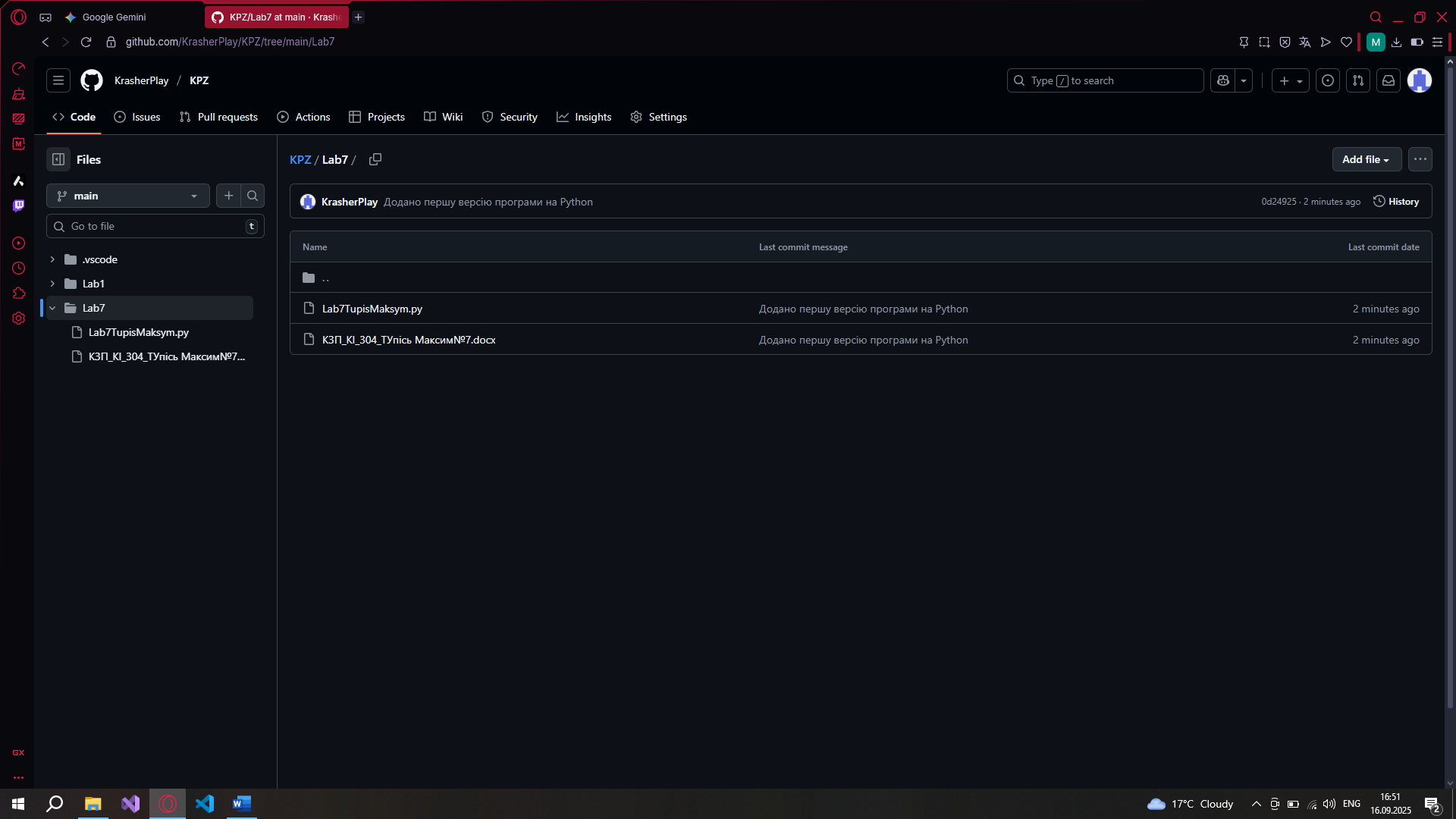


Рис.2 Git hub репозиторій з файлами програми

**Висновок:** У ході виконання лабораторної роботи було досліджено основні базові конструкції мови програмування python. Зокрема, було розглянуто оголошення та використання змінних різних типів даних, оператори присвоєння, арифметичні та логічні операції, умовні конструкції (if, switch) та циклічні оператори (for, while, do-while). Практична реалізація прикладів дозволила закріпити знання щодо синтаксису мови та особливостей її застосування для розв’язання простих алгоритмічних задач.Окрему увагу було приділено роботі з масивами, у тому числі багатовимірними та так званими зубчастими матрицями (jagged arrays), які в python реалізуються як масиви масивів. На практиці було досліджено способи їх ініціалізації, доступу до елементів та відмінності від класичних прямокутних матриць. Це дозволило зрозуміти гнучкість і потужність такої структури даних, а також її зручність для роботи з нерівномірними таблицями або структурами, що мають різну кількість елементів у рядках.Таким чином, у процесі роботи було сформовано базові практичні навички програмування на python, отримано досвід використання основних операторів, циклів і умовних конструкцій, а також опановано принципи створення та застосування зубчастих матриць. Це є важливим кроком для подальшого вивчення об’єктно-орієнтованих можливостей мови та розробки більш складних програмних систем.