Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»



Звіт

З лабораторної роботи №8

З дисципліни «Кросплатформні засоби програмування»

На тему:

**ФАЙЛИ ТА ВИКЛЮЧЕННЯ У PYTHON**

Виконав: Тупісь М.Н.

ст. гр.KI-304

Прийняв:

Іванов Ю.С.

Львів – 2025

**Мета**: оволодіти навиками використання засобів мови Python для роботи з файлами.

**Теоретичні відомості**

Файли — це важливий інструмент для збереження даних. У Python роботу з файлами можна розділити на кілька ключових етапів: відкриття, читання/запис і закриття.

**Відкриття файлу**

Щоб почати роботу з файлом, його потрібно відкрити. Це можна зробити за допомогою функції open(), яка приймає два основних аргументи: ім'я файлу (шлях до нього) та режим роботи.

Основні режими:

* **'r'** (read) — **режим читання**. Файл відкривається лише для зчитування інформації. Це режим за замовчуванням.
* **'w'** (write) — **режим запису**. Якщо файл існує, його вміст буде **повністю стерто**. Якщо файлу немає, він буде створений.
* **'a'** (append) — **режим додавання**. Дані будуть додаватися в кінець файлу, не стираючи його вміст. Якщо файлу немає, він буде створений.
* **'x'** (exclusive creation) — **режим виключного створення**. Файл буде створено, але якщо він уже існує, виникне помилка FileExistsError.
* **'t'** (text) — **режим текстовий** (за замовчуванням). Дані будуть оброблятися як рядки.
* **'b'** (binary) — **режим бінарний**. Дані будуть оброблятися як байти (наприклад, для роботи з зображеннями, аудіо, тощо).

Наприклад: file = open('example.txt', 'w')

Після відкриття файлу можна виконувати операції читання або запису:

* **file.write(data)** — записує рядок або байти у файл.
* **file.read()** — зчитує весь вміст файлу і повертає його як рядок.
* **file.readline()** — зчитує один рядок.
* **file.readlines()** — зчитує всі рядки файлу і повертає їх у вигляді списку.

**Виключення —** це події, що виникають під час виконання програми і порушують її нормальний хід. Наприклад, спроба відкрити неіснуючий файл або ділення на нуль. Обробка виключень дозволяє зробити програму надійною.

Основна конструкція для обробки виключень — це блок try...except

Блок **finally** виконується завжди, незалежно від того, чи виникла помилка в блоці try. Він часто використовується для звільнення ресурсів, наприклад, для закриття файлу або з'єднання з базою даних, якщо оператор with не використовується.

Хід роботи

Завдання: Написати та налагодити програму на мові Python згідно варіанту. Програма має:

задовольняти наступним вимогам:

• програма має розміщуватися в окремому модулі;

• програма має реалізувати функції читання/запису файлів у текстовому і

двійковому форматах результатами обчислення виразів згідно варіанту;

• програма має містити коментарі.

Варіант: 23

23. y=ctg(8x)/x

Лістинг програми мовою python:

//file main.py

# file: main.py

import solution

# Імена файлів для збереження результатів

txtFile = "Lab8/result.txt"

binFile = "Lab8/result.bin"

# Виклик основної функції

solution.solution(txtFile, binFile)

# Вивід вмісту текстового файлу

print(f"\nВміст файлу {txtFile}:")

print(solution.read\_txt(txtFile))

# Вивід вмісту двійкового файлу

solution.read\_binary(binFile)

//file solution.py

# file: solution.py

from math import tan

import struct

# Основна функція обчислення

# @filenameResultTxt - текстовий файл для запису результату

# @filenameResultBin - двійковий файл для запису результату

def solution(filenameResultTxt, filenameResultBin):

try:

# Ввід даних користувачем

x = float(input("Ця програма обчислює y = ctg(8x) / x\nВведіть x: "))

# Перевірка, щоб уникнути ділення на нуль

if x == 0:

print("Помилка: x не може дорівнювати 0 (ділення на нуль).")

return

# Обчислення ctg(8x)/x

y = (1 / tan(8 \* x)) / x

# Запис у текстовий файл

with open(filenameResultTxt, "a", encoding="utf-8") as file:

file.write(f"Результат функції y=ctg(8x)/x при x={x} дорівнює {y}\n")

# Запис у двійковий файл (float - 4 байти)

with open(filenameResultBin, "ab") as file:

file.write(struct.pack("f", y))

print(f"Результат успішно збережено у файли {filenameResultTxt} та {filenameResultBin}")

except Exception as e:

print(f"Виникла помилка: {e}")

# Функція для читання текстового файлу

def read\_txt(filename):

try:

with open(filename, "r", encoding="utf-8") as file:

return file.read()

except FileNotFoundError:

return "Файл не знайдено."

# Функція для читання двійкового файлу

def read\_binary(filename):

print(f"\nВміст двійкового файлу {filename}:")

try:

with open(filename, "rb") as file:

while True:

data = file.read(4) # читаємо по 4 байти (тип float)

if not data or len(data) < 4:

break

number = struct.unpack("f", data)[0]

print(number)

except FileNotFoundError:

print("Файл не знайдено.")

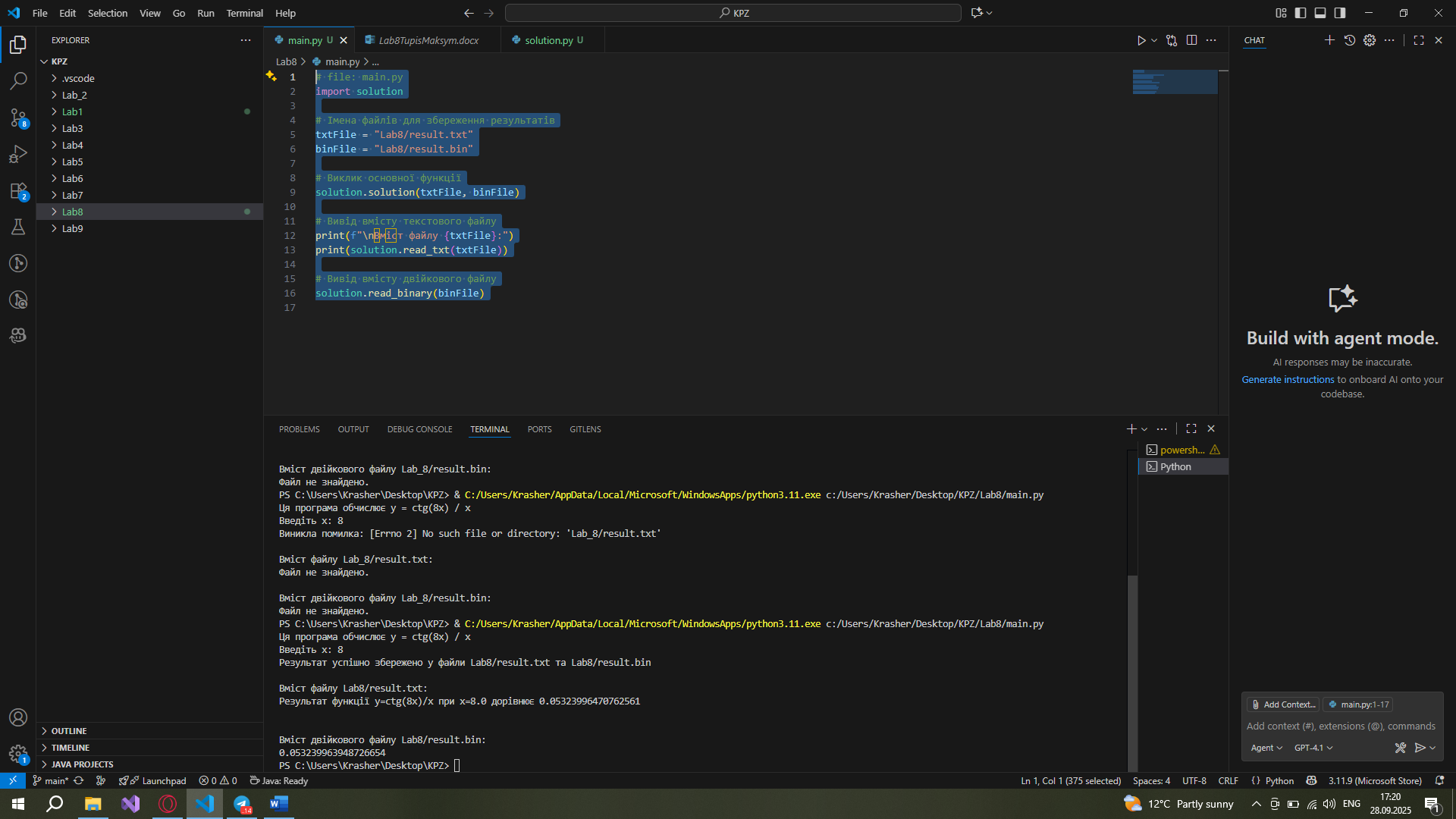


Рис.1 Приклад роботи програми

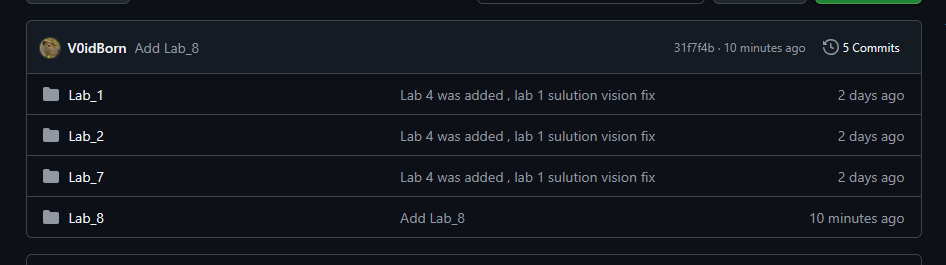


Рис.2 Git hub репозиторій з файлами програми

**Висновок:** ВисновокУ ході виконання лабораторної роботи було успішно засвоєно теоретичні знання та набуто практичні навички роботи з файлами та обробки виключень у Python. Робота з файлами дозволяє зберігати, зчитувати та оновлювати дані, що робить програми більш гнучкими та функціональними. Вивчення різних режимів роботи з файлами ('r', 'w', 'a') та використання контекстного менеджера with забезпечило ефективну та безпечну взаємодію з файловою системою.Особлива увага була приділена механізму обробки виключень. Застосування конструкцій try...except дозволило розробити стійкий до помилок код, який коректно реагує на неочікувані ситуації, такі як відсутність файлу або некоректний формат даних. Це є критично важливим для створення надійних програм, які можуть самостійно обробляти помилки, не припиняючи виконання.Таким чином, у результаті виконання роботи було досягнуто поставленої мети: набуті знання та навички є базовими для подальшого вивчення програмування та розробки складніших, надійних програмних продуктів.