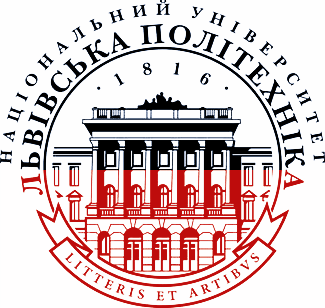
Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



*Звіт*

до лабораторної роботи №8

З дисципліни: «Кросплатформні засоби програмування»

На тему: «ФАЙЛИ ТА ВИКЛЮЧЕННЯ У PYTHON»

Виконала: ст. гр. КІ-305

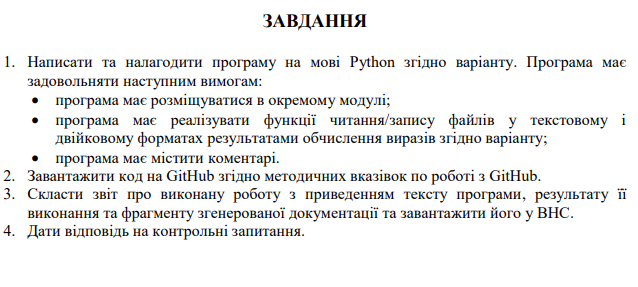
Романюк М. Р.

Прийняв: доцент кафедри ЕОМ

Іванов Ю. С.

Львів 2023

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



Мій варіант №22: y=tg(4x)/x

**Код програми:**

import os  
import struct  
import sys  
import math  
  
  
# Функція для запису результату у текстовий файл  
def write\_result\_txt(filename, result):  
 # Відкриваємо файл для запису  
 with open(filename, 'w') as file:  
 # Записуємо результат як текстовий рядок  
 file.write(str(result))  
  
  
# Функція для читання результату з текстового файлу  
def read\_result\_txt(filename):  
 result = 0.0  
 try:  
 if os.path.exists(filename):  
 # Відкриваємо файл для читання  
 with open(filename, 'r') as file:  
 # Зчитуємо текстовий рядок та перетворюємо його в число з плаваючою точкою  
 result = float(file.read())  
 else:  
 raise FileNotFoundError(f"Файл {filename} не знайдено.")  
 except FileNotFoundError as e:  
 print(e)  
 return result  
  
  
# Функція для запису результату у двійковий файл  
def write\_result\_bin(filename, result):  
 # Відкриваємо файл для запису в бінарному режимі  
 with open(filename, 'wb') as file:  
 # Записуємо результат у форматі двійкового числа з подвійною точністю (double)  
 file.write(struct.pack('d', result))  
  
  
# Функція для читання результату з двійкового файлу  
def read\_result\_bin(filename):  
 result = 0.0  
 try:  
 if os.path.exists(filename):  
 # Відкриваємо файл для читання в бінарному режимі  
 with open(filename, 'rb') as file:  
 # Зчитуємо дані і розпаковуємо їх як double  
 result = struct.unpack('d', file.read())[0]  
 else:  
 raise FileNotFoundError(f"Файл {filename} не знайдено.")  
 except FileNotFoundError as e:  
 print(e)  
 return result  
  
  
# Функція для обчислення значення y = tg(4x) / x  
def calculate\_y(x):  
 try:  
 # Обчислюємо y згідно варіанту  
 y = math.tan(4 \* x) / x  
 return y  
 except ZeroDivisionError:  
 print("Помилка: Ділення на нуль. x не може дорівнювати 0.")  
  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 data = float(input("Введіть значення x: "))  
  
 # Обчислення результату за заданим варіантом  
 result = calculate\_y(data)  
  
 print(f"Результат обчислення: {result}")  
  
 try:  
 # Запис результату у текстовий та двійковий файли  
 write\_result\_txt("text\_result.txt", result)  
 write\_result\_bin("binary\_result.dat", result)  
  
 # Читання результату з файлів та виведення на екран  
 print("Результат зчитано з бінарного файлу: {0}".format(read\_result\_bin("binary\_result.dat")))  
 print("Результат зчитано з текстового файлу: {0}".format(read\_result\_txt("text\_result.txt")))  
 except FileNotFoundError as e:  
 print(e)  
 sys.exit(1)

# Результати роботи програми:

# 

# Відповіді на контрольні запитання

1. За допомогою якої конструкції у мові Python обробляються виключні ситуації?

В мові Python використовується конструкція **try...except** для обробки виключних ситуацій.

2. Особливості роботи блоку except?

Блок **except** використовується для обробки винятків, які можуть виникнути у внутрішньому блоку **try**. Код у блоку **except** виконується тільки в тому випадку, якщо сталася відповідна виняткова ситуація, яка вказана в параметрі **except**. Можна вказати більше одного блоку **except** для обробки різних видів винятків.

3. Яка функція використовується для відкривання файлів у Python?

Для відкриття файлів у Python використовується функція **open()**.

4. Особливості використання функції open?

Функція **open()** приймає два обов'язкових аргументи: ім'я файлу та режим відкриття. Режим відкриття вказує, як саме файл буде відкритий (на читання, запис, додавання і т.д.).

5. В яких режимах можна відкрити файл?

Файл можна відкрити у різних режимах, таких як:

* **'r'**: Читання (за замовчуванням).
* **'w'**: Запис (створює новий файл або перезаписує існуючий).
* **'a'**: Додавання (додає дані до кінця файлу).
* **'b'**: Бінарний режим (для роботи з двійковими файлами).
* **'t'**: Текстовий режим (за замовчуванням).

6. Як здійснити читання і запис файлу?

Для читання файлу використовується функція **read()**, а для запису - **write()**.

7. Особливості функцій у мові Python?

Функції в мові Python є об'єктами, які можна передавати як аргументи, повертати з функцій та зберігати у змінних. Функції також можуть бути визначені всередині інших функцій (вкладені функції).

8. Для чого призначений оператор with?

Оператор **with** призначений для керування контекстом. Він використовується для створення "контексту", в якому визначено, як обробляти певні дії, і автоматично видаляє ресурси (наприклад, закриває файл) після завершення цього контексту.

9. Які вимоги ставляться до об’єктів, що передаються під контроль оператору with?

Об'єкти, які передаються під контроль оператору **with**, повинні мати методи **\_\_enter\_\_** та **\_\_exit\_\_**. Наприклад, для файлів об'єкти файлів відповідають цим вимогам.

10. Як поєднуються обробка виключних ситуацій і оператор with?

Оператор **with** добре поєднується з обробкою виключних ситуацій, оскільки він гарантує правильне завершення ресурсів після виходу з контексту, навіть якщо виникають виключення. Можна використовувати **with** для автоматичного закриття файлу, коли робите операції з файлами, і це полегшує обробку помилок при роботі з файлами.

**Висновок:** у ході виконання лабораторної роботи я оволоділа навичками використання засобів мови Python для роботи з файлами.