Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра «Електронних обчислювальних машин»

Звіт

з лабораторної роботи № 8

з дисципліни: «Кросплатформенні засоби програмування»

на тему: «Файли та виключення у python»

Виконав:

студент групи КІ-306

Ширий Б. І.

Прийняв:

доцент кафедри ЕОМ

Іванов Ю. С.

Львів – 2023

МЕТОДИЧНІ ВІДОМОСТІ РОБОТИ

МЕТА

Оволодіти навиками використання засобів мови Python для роботи з файлами.

ЗАВДАННЯ

№1

Написати та налагодити програму на мові Python згідно варіанту, а саме

$$y = \frac{1}{\cos 4x}.$$

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.

№2

Для розробленої програми згенерувати документацію

№3

Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.

№4

Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.

№5

Дати відповідь на контрольні запитання.

ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ

ВИХІДНИЙ КОД

Написав програму, що реалізує метод обчислення виразу $y = \frac{1}{\cos 4x}$, код якої наведено у лістингу 2.1. Відповідно до завдання лабораторної роботи створив клас, що реалізує методи читання/запису у текстовому і двійковому форматах результатів роботи обчислювального класу, та навів його у лістингу 2.2.

Лістинг 2.1. Код основної програми.

```
import tkinter as tk
from tkinter import ttk
from tkinter import messagebox
import math
import FileManager
# Підключення власного модулю FileManager

class ExpressionCalculatorGUI:
    def __init__(self, root):
        self.root = root
        root.title("Calculator")

        # Створення віджетів та розміщення їх на головному вікні
        self.x_label = ttk.Label(root, text="Enter the value of x:")
        self.x_label.grid(row=0, column=0, padx=5, pady=5)

        self.x_entry = ttk.Entry(root)
        self.x_entry.grid(row=0, column=1, padx=5, pady=5)

        self.file_name_label = ttk.Label(root, text="Enter the file name:")
        self.file_name_label.grid(row=1, column=0, padx=5, pady=5)

        self.file_name_entry = ttk.Entry(root)
        self.file_name_entry.grid(row=1, column=1, padx=5, pady=5)

        self.pi_checkbox = ttk.Checkbutton(root, text="π")
        # Чекбокс для вибору використання символу π
        self.pi_checkbox.grid(row=1, column=3, padx=5, pady=5)

        self.calculate_button = ttk.Button(root, text="Calculate",
                                           command=self.calculate_expression)
        # Кнопка для обчислення виразу
        self.calculate_button.grid(row=0, column=3, padx=5, pady=5)

        # Кнопки для взаємодії з файлами
        self.open_text_button = ttk.Button(root, text="Open Text",
                                           command=lambda: FileManager.FileManager
                                           .open_file(self.file_name_entry.get()
                                                      + ".txt"))
        self.open_text_button.grid(row=2, column=0, padx=5, pady=5)
```

```
        self.read_text_button = ttk.Button(root, text="Read Text",
                                           command=lambda: FileManager.FileManager
                                           .read_text(self.file_name_entry.get()
                                                      + ".txt"))
        self.read_text_button.grid(row=2, column=1, padx=5, pady=5)

        self.read_binary_button = ttk.Button(root, text="Read Binary",
                                             command=lambda: FileManager.FileManager
                                             .read_binary(self.file_name_entry.get()
                                                          + ".dat"))
        self.read_binary_button.grid(row=2, column=2, columnspan=2, padx=5, pady=5)

    def calculate_expression(self):
        x_value = self.x_entry.get()
        # Отримання значення x з введеного тексту
        file_name = self.file_name_entry.get()
        # Отримання імені файлу з введеного тексту
        use_pi = self.pi_checkbox.instate(['selected'])
        # Перевірка, чи вибрано символ π

        try:
            x = float(x_value)  # Конвертація введеного значення x у тип float
            result = 1 / (0 if (use_pi and (abs(4 * x) == 0.5 or (4 * x - 0.5) % 1 == 0))
                          else math.cos(4 * x))

            if math.isnan(result) or result == float('-inf') or result == float('inf'):
                raise ArithmeticError

            if not file_name:
                file_name = "result"
                # Якщо ім'я файлу не вказано, використовується "result"

            # Запис результатів обчислення в текстовий і бінарний файл
            FileManager.FileManager.write_text(file_name + ".txt", x, result)
            FileManager.FileManager.write_binary(file_name + ".dat", x, result)

            messagebox.showinfo("Success", f"Calculation result written to file '{file_name}'")
            # Вивід повідомлення про успішний запис

        except ValueError:
            messagebox.showerror("Error", "Invalid input format")
            # Обробка помилки невірного формату введення
        except ArithmeticError:
            messagebox.showerror("Error", "Division by zero: cos(4x) equals zero.")
            # Обробка помилки ділення на нуль

if __name__ == "__main__":
    root = tk.Tk()
    app = ExpressionCalculatorGUI(root)
    root.mainloop()
```

Лістинг 2.2. Клас "Файловий менеджер".

```
import os
import re
import struct
from tkinter import messagebox

class FileManager:

    # Метод для запису результату у текстовий файл
    @staticmethod
    def write_text(file_name, x, result):
        try:
            with open(file_name, 'w') as file:
                file.write(f'Значення y при x = {x} дорівнює {result}')
        except IOError as e:
            messagebox.showerror("Error", f'Помилка запису в текстовий файл: {str(e)}')

    # Метод для запису результату у бінарний файл
    @staticmethod
    def write_binary(file_name, x, result):
        try:
            with open(file_name, 'wb') as file:
                x_bytes = bytearray(struct.pack('d', x))
                result_bytes = bytearray(struct.pack('d', result))
                file.write(x_bytes)
                file.write(result_bytes)
        except IOError as e:
            messagebox.showerror("Error", f'Помилка запису в двійковий файл: {str(e)}')

    # Метод для читання результату з текстового файлу
    @staticmethod
    def read_text(file_name):
        if not os.path.exists(file_name):
            messagebox.showerror("Error", 'Помилка: Файл не існує.')

        try:
            with open(file_name, 'r') as file:
                data = file.read()
                matches = re.findall(r'[-+]?\d*\.\d+|\d+', data)
                if len(matches) >= 2:
                    x = float(matches[0])
                    result = float(matches[1])
                    messagebox.showinfo("Success", f'У текстовому файлі: y = {result}, а x = {x}')
                else:
                    messagebox.showerror("Error", 'Помилка: Немає достатньо чисел у файлі.')
        except (IOError, ValueError) as e:
            messagebox.showerror("Error", f'Помилка читання з текстового файлу: {str(e)}')

    # Метод для читання результату з бінарного файлу
    @staticmethod
    def read_binary(file_name):
        if not os.path.exists(file_name):
```

```
            messagebox.showerror("Error", 'Помилка: Файл не існує.')

        try:
            with open(file_name, 'rb') as file:
                x_bytes = file.read(8)
                result_bytes = file.read(8)
                x = struct.unpack('d', x_bytes)[0]
                result = struct.unpack('d', result_bytes)[0]
                messagebox.showinfo("Success", f'У бінарному файлі: y = {result}, а x = {x}')
        except (IOError, ValueError) as e:
            messagebox.showerror("Error", f'Помилка читання з двійкового файлу: {str(e)}')

    # Метод для відкриття файлу зі стандартною програмою для перегляду
    @staticmethod
    def open_file(file_name):
        if not os.path.exists(file_name):
            messagebox.showerror("Error", 'Помилка: Файл не існує.')

        try:
            os.system(f'start {file_name}')
            # Відкрити файл зі стандартною програмою для перегляду
        except Exception as e:
            messagebox.showerror("Error", f'Помилка відкриття файлу: {str(e)}')

        messagebox.showerror("Error", f'Файл {file_name} відкрито')
```

РЕЗУЛЬТАТИ ВИКОНАННЯ

Початковий вигляд програми наведено на рисунку 2.1.

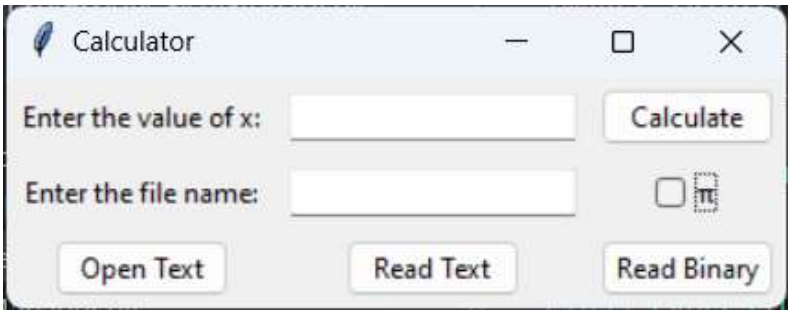

Рисунок 2.1. Початковий вигляд програми.

Якщо ми обрахуємо певне значення (рисунок 2.2), то через клас «Файловий менеджер» відбудеться запис у текстовий та бінарний файли.

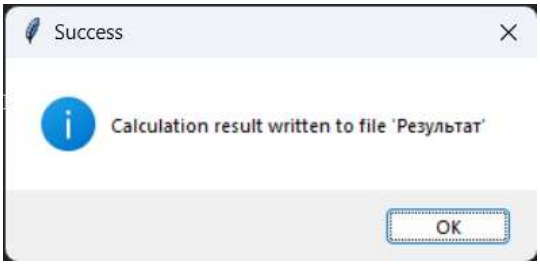

Рисунок 2.2. Запис у файли.

Відповідно, ми можемо:

- Прочитати бінарний файл - рисунок 2.3,
- Прочитати текстовий файл - рисунок 2.4,
- Відкрити текстовий файл - рисунки 2.5 та 2.6.

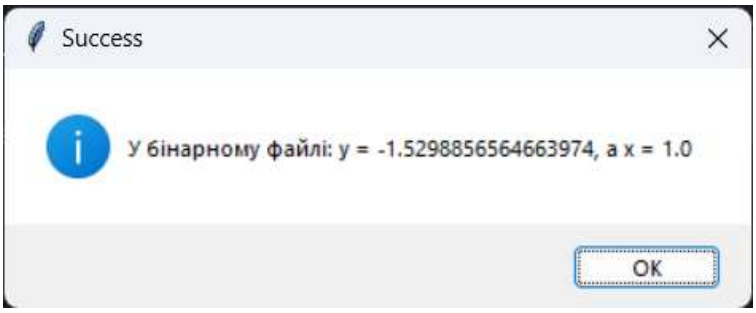

Рисунок 2.3. Читання з бінарного файлу.

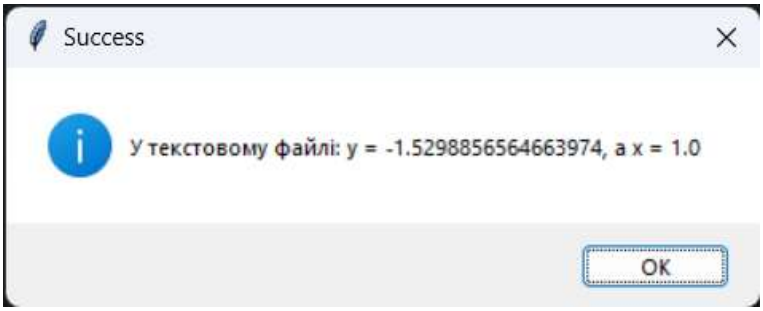

Рисунок 2.4. Читання з текстового файлу.

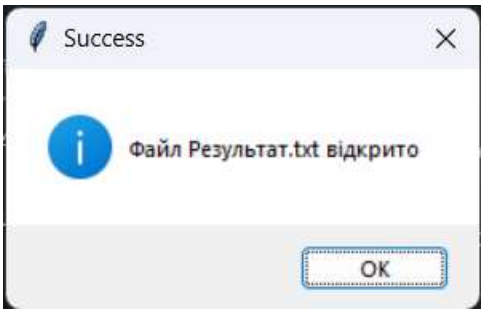

Рисунок 2.5. Відкриття текстового файлу.

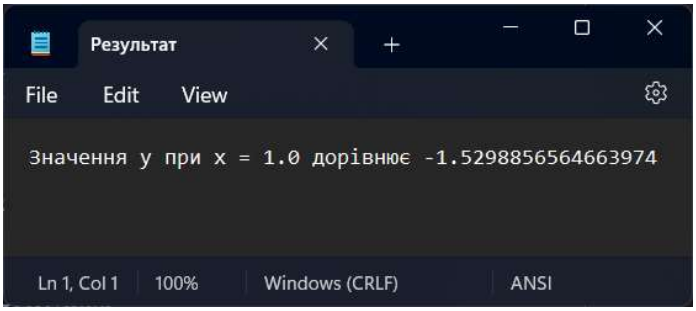

Рисунок 2.6. Відкритий текстовий файл.

Якщо будуть введені параметри, за якими рівняння не має розв’язку, то програма покаже помилку, як зображено на рисунку 2.7.

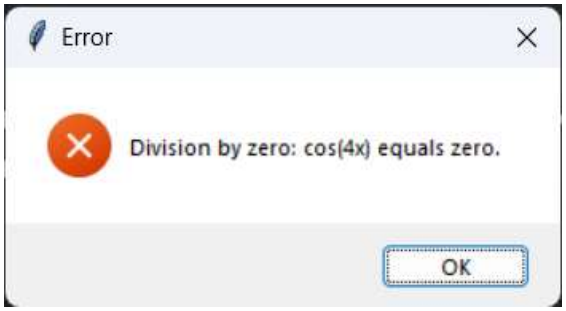

Рисунок 2.7. Ділення на нуль.

ВІДПОВІДІ НА КОНТРОЛЬНІ ПИТАННЯ

ЗА ДОПОМОГОЮ ЯКОЇ КОНСТРУКЦІЇ У МОВІ PYTHON ОБРОБЛЯЮТЬСЯ ВИКЛЮЧНІ СИТУАЦІЇ?

В мові Python виключні ситуації обробляються за допомогою конструкції "try-except."

ОСОБЛИВОСТІ РОБОТИ БЛОКУ EXCEPT?

- У блоку "except" вказується код, який виконується, якщо виникає виключна ситуація.
- Можна вказати один або декілька типів виключних ситуацій, які будуть оброблятися цим блоком.
- Блок "except" виконується лише, якщо відповідна виключна ситуація виникла в блоку "try."

ЯКА ФУНКЦІЯ ВИКОРИСТОВУЄТЬСЯ ДЛЯ ВІДКРИВАННЯ ФАЙЛІВ У PYTHON?

Для відкривання файлів у Python використовується функція "open."

ОСОБЛИВОСТІ ВИКОРИСТАННЯ ФУНКЦІЇ OPEN?

- Функція "open" приймає два аргументи: ім'я файлу і режим відкриття.
- Режими включають "r" (читання), "w" (запис), "a" (додавання), "rb" (читання в двійковому режимі), "wb" (запис в двійковому режимі) та інші.
- Функція "open" повертає об'єкт файлу, який використовується для подальшої роботи з файлом.

В ЯКИХ РЕЖИМАХ МОЖНА ВІДКРИТИ ФАЙЛ?

Файл можна відкрити в різних режимах, таких як "r" (читання), "w" (запис), "a" (додавання), "rb" (читання в двійковому режимі), "wb" (запис в двійковому режимі) і багато інших.

ЯК ЗДІЙСНИТИ ЧИТАННЯ І ЗАПИС ФАЙЛУ?

Для читання файлу використовується функція "read," а для запису - функція "write."

ОСОБЛИВОСТІ ФУНКЦІЙ У МОВІ PYTHON?

Особливості функцій у мові Python включають можливість передавати аргументи, повертати значення, та обробляти виключні ситуації за допомогою блоку "try-except."

ДЛЯ ЧОГО ПРИЗНАЧЕНЙИ ОПЕРАТОР WITH?

Оператор "with" призначений для створення контексту, в якому можна автоматично відкривати і закривати ресурси, такі як файли. Він гарантує, що ресурс буде коректно закритий після виходу з контексту.

ЯКІ ВИМОГИ СТАВЛЯТЬСЯ ДО ОБ’ЄКТІВ, ЩО ПЕРЕДАЮТЬСЯ ПІД КОНТРОЛЬ ОПЕРАТОРУ WITH?

До об'єктів, які передаються під контроль оператору "with," ставляться вимоги щодо наявності методів "enter" і "exit," які визначають, як відкривати і закривати ресурси.

ЯК ПОЄДНУЮТЬСЯ ОБРОБКА ВИКЛЮЧНИХ СИТУАЦІЙ І ОПЕРАТОР WITH?

Обробка виключних ситуацій може бути поєднана з оператором "with," якщо виключна ситуація виникає в контексті "with," то метод "exit" буде викликаний автоматично для закриття ресурсів.

ВИСНОВОК

Завдання лабораторної роботи виконано, а саме у мові Python були реалізовані функції читання і запису файлів у різних форматах, використовуючи конструкції "try-except" для обробки виключних ситуацій. Код був організований в окремому модулі та містить коментарі для кращого розуміння.