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 – 2022

# 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

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 , 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.*

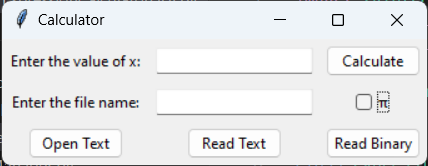
|  |
| --- |
| import tkinter as tk  from tkinter import ttk  from tkinter import messagebox  import math  import FileManager  # Підключення власного модулю FileManager  class ExpressionCalculatorGUI:      def \_\_init\_\_(self, root):          self.root = root          root.title("Calculator")          # 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: у = {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: у = {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') |

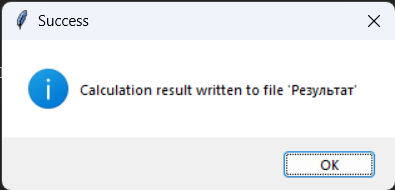
## Результати виконання

Початковий вигляд програми наведено на рисунку 2.1.



*Рисунок 2.1. Початковий вигляд програми.*

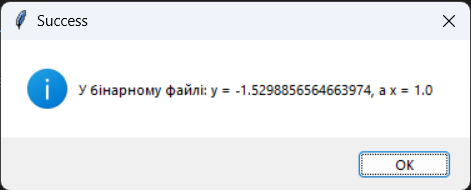
Якщо ми обрахуємо певне значення (рисунок 2.2), то через клас «Файловий менеджер» відбудеться запис у текстовий та бінарний файли.



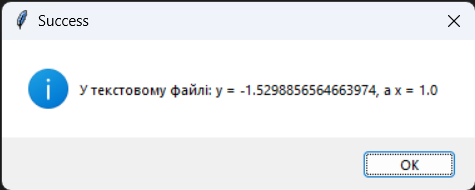
*Рисунок 2.2. Запис у файли.*

Відповідно, ми можемо:

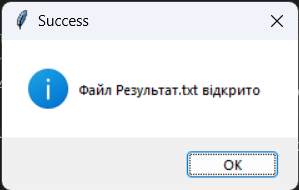
* Прочитати бінарний файл - рисунок 2.3,
* Прочитати текстовий файл - рисунок 2.4,
* Відкрити текстовий файл - рисунки 2.5 та 2.6.



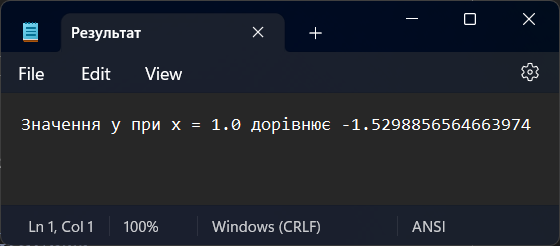
*Рисунок 2.3. Читання з бінарного файлу.*



*Рисунок 2.4. Читання з текстового файлу.*

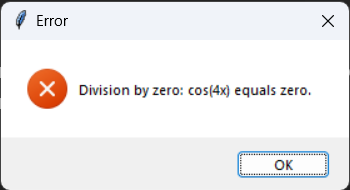


*Рисунок 2.5. Відкриття текстового файлу.*



*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.