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

Чаус Б.В.

**Прийняв:**

доцент кафедри ЕОМ

Іванов Ю. С.

Львів – 2023

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

**Завдання (варіант № 24)**

24. y=sin(x-9)/(x-cos(2x))

1. 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.

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

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

4. Дати відповідь на контрольні запитання.**Вихідний код програми**

Файл Lab8ChausKI306.py

import os

import math

class CalcException(Exception):

pass

class Equations:

def calculate(self, x):

y = 0

rad = x \* 3.14159 / 180.0

try:

denominator = x - math.cos(2 \* rad)

if denominator == 0:

raise ArithmeticError()

y = math.sin(rad - 9) / denominator

if math.isnan(y) or math.isinf(y):

raise ArithmeticError()

except ArithmeticError:

if x == 0:

raise CalcException("Exception reason: X = 0")

else:

raise CalcException("Exception reason: Division by zero")

return y

def main():

fName = "Result.txt"

out\_file\_bin = "Out\_binary.dat"

try:

eq = Equations()

x = int(input("Enter X: "))

try:

result = eq.calculate(x)

print(f"Result: {result}")

with open(fName, 'w') as fout:

fout.write(f"Result: {result}\n")

with open(out\_file\_bin, 'wb') as fout\_bin:

fout\_bin.write(f"Result {result}\n".encode())

except CalcException as ex:

print(ex)

except FileNotFoundError:

print("Exception reason: File not found")

# Reading from files

try:

with open(fName, 'r') as fin, open(out\_file\_bin, 'rb') as fin\_bin:

resultText = fin.readline()

binaryResult = fin\_bin.readline().decode()

print(f"Result from Result.txt: {resultText}", end="")

print(f"Result from Out\_binary.dat: {binaryResult}", end="")

except FileNotFoundError:

print("Exception reason: File not found")

except IOError:

print("Exception while reading the file")

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

main()

Файл Main.py

import math # Import the math module for mathematical operations

class CalcException(Exception):

def \_\_init\_\_(self, cause=None):

super().\_\_init\_\_(cause) # Initialize the CalcException with an optional cause message

class Equations:

def calculate(self, x):

y = 0 # Initialize variable y

rad = x \* 3.14159 / 180.0 # Convert the angle x to radians

try:

denominator = x - math.cos(2 \* rad) # Calculate the denominator of the expression

if denominator == 0:

raise ArithmeticError() # Raise an exception if the denominator is zero

y = math.sin(rad - 9) / denominator # Calculate the value of y

if math.isnan(y) or math.isinf(y):

raise ArithmeticError() # Raise an exception if y is NaN (Not a Number) or Infinite

except ArithmeticError:

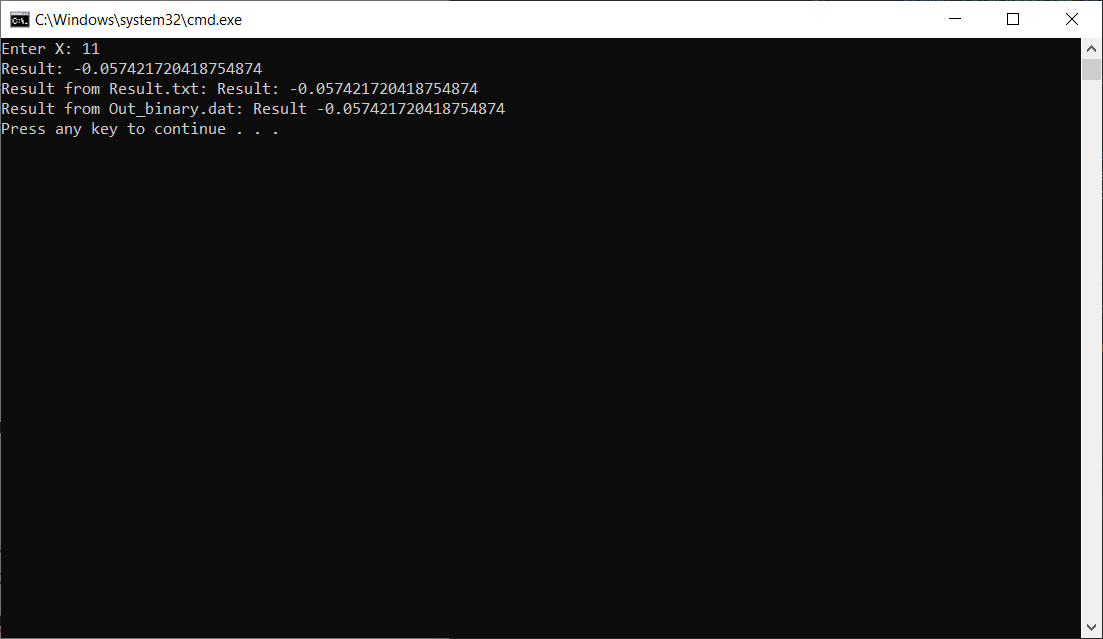
if x == 0:

raise CalcException("Exception reason: X = 0") # Raise a CalcException with a specific message for x = 0

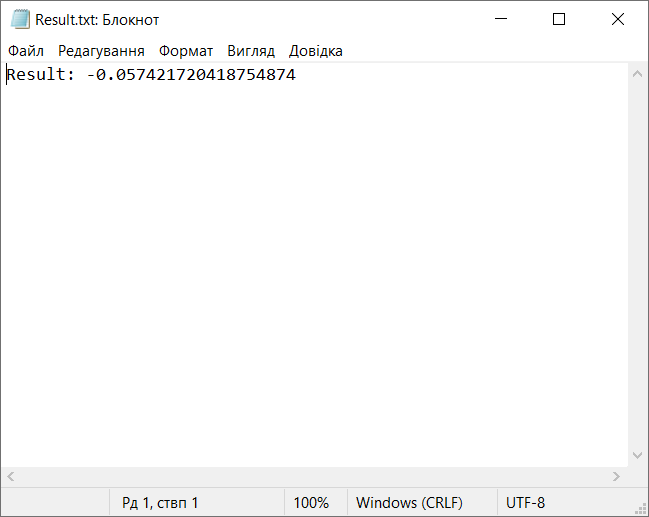
else:

raise CalcException("Exception reason: Division by zero") # Raise a CalcException with a specific message for division by zero

return y # Return the calculated value of y

**Результат виконання програми**

**Результат виконання програми записаний в txt файл**



**Відповіді на контрольні запитання**

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

2. Блок `except` використовується для обробки виняткових ситуацій, які виникають під час виконання коду в блоку `try`.

3. Функція, яка використовується для відкриття файлів у Python, називається `open()`.

4. Функція `open()` приймає два аргументи: шлях до файлу і режим відкриття.

5. Файл можна відкрити в різних режимах, таких як "читання" (`'r'`), "запис" (`'w'`), "додавання" (`'a'`), "бінарний режим" (`'b'`) та інші.

6. Для читання файлу використовується метод `read()` або ітерація по файловому об'єкту. Для запису - метод `write()`.

7. Функції в мові Python можуть приймати аргументи, повертати значення, бути вкладеними, а також бути передані в якості аргументу іншій функції.

8. Оператор `with` використовується для автоматичного управління контекстом. Він забезпечує відкриття та закриття ресурсів в правильному порядку.

9. Об'єкти, що передаються під контроль оператору `with`, повинні мати методи `\_\_enter\_\_` та `\_\_exit\_\_`.

10. Обробка виключних ситуацій може бути вбудована в оператор `with`, щоб гарантувати коректне закриття ресурсів, навіть якщо виникає виняткова ситуація.

**Висновок**

Під час лабораторної роботи, я оволодів навиками використання засобів мови Python для роботи з файлами.