Міністерство освіти і науки України

Національний університет «Львівська політехніка»



Лабораторна робота №2

з навчальної дисципліни

«Спеціалізовані мови програмування»

на тему

«Основи побудови об’єктно-орієнтованих додатків на Python»

*Інститут комп’ютерних наук*

*та інформаційних технологій*

*Кафедри інформаційних*

*систем та мереж*

***Виконав:***

*Студент групи РІ-31*

*Олександр ЛУЧКЕВИЧ*

***Прийняла:***

*викладач*

*Cергій ЩЕРБАК*

*Львів – 2024*

**Мета роботи:** розробка консольного калькулятора в об’єктно орієнтованому стилі з використанням класів

**План роботи:**

*Завдання 1:* Створення класу Calculator

Створіть клас Calculator, який буде служити основою для додатка калькулятора.

*Завдання 2:* Ініціалізація калькулятора

Реалізуйте метод \_\_init\_\_ у класі Calculator для ініціалізації необхідних атрибутів або змінних.

*Завдання 3:* Введення користувача

Перемістіть функціональність введення користувача в метод у межах класу Calculator. Метод повинен приймати введення для двох чисел і оператора.

*Завдання 4:* Перевірка оператора

Реалізуйте метод у класі Calculator, щоб перевірити, чи введений оператор є дійсним (тобто одним із +, -, \*, /). Відобразіть повідомлення про помилку, якщо він не є дійсним.

*Завдання 5:* Обчислення

Створіть метод у класі Calculator, який виконує обчислення на основі введення користувача (наприклад, додавання, віднімання, множення, ділення).

*Завдання 6:* Обробка помилок

Реалізуйте обробку помилок у межах класу Calculator для обробки ділення на нуль або інших потенційних помилок. Відобразіть відповідні повідомлення про помилку.

*Завдання 7:* Повторення обчислень

Додайте метод до класу Calculator, щоб запитати користувача, чи він хоче виконати ще одне обчислення. Якщо так, дозвольте йому ввести нові числа і оператор. Якщо ні, вийдіть з програми.

*Завдання 8:* Десяткові числа

Модифікуйте клас Calculator для обробки десяткових чисел (плаваюча кома) для більш точних обчислень.

*Завдання 9:* Додаткові операції

Розширте клас Calculator, щоб підтримувати додаткові операції, такі як піднесення до степеня (^), квадратний корінь (√) та залишок від ділення (%).

*Завдання 10:* Інтерфейс, зрозумілий для користувача

Покращте інтерфейс користувача у межах класу Calculator, надавши чіткі запити, повідомлення та форматування виводу для зручності читання.

**Код програми:**

Файл lab\_calculator/main:

from functions import calculator, show\_history, log\_history

memory\_value = 0

decimal\_places = 1

def set\_decimal\_places():

global decimal\_places

while True:

try:

decimal\_places = int(input("How many decimal places do you want to display: "))

print(f"Decimal places set to: {decimal\_places}")

break

except ValueError:

print("Incorrect input! Enter an integer.")

def main():

global memory\_value, decimal\_places

while True:

print("\n--- Calculator ---")

print("1. New calculation")

print("2. Show history")

print("3. Set the number of decimal places")

print("4. Exit")

choice = input("Choose option (1-4): ")

if choice == '1':

result, first\_number, operator, second\_number = calculator(decimal\_places, memory\_value)

if result is not None:

choice\_memory = input('Would you like to store result in memory (MS), add to memory (M+), clear memory (MC), or skip? ').upper()

match choice\_memory:

case 'MS':

memory\_value = result

print(f"Stored {result} in memory.")

case 'M+':

memory\_value += result

print(f"Added {result} to memory. New memory value: {memory\_value}.")

case 'MC':

memory\_value = 0

print("Memory cleared.")

log\_history(first\_number, operator, second\_number, result)

print(f"Result: {result}")

if input("Do you want to view history? (y/n): ").strip().lower() == 'y':

print(show\_history())

if input('Do you want to make another calculation? (y/n): ').lower() != 'y':

break

elif choice == '2':

print(show\_history())

elif choice == '3':

set\_decimal\_places()

elif choice == '4':

print("Exit.")

break

else:

print("Wrong choice! Try again.")

main()

Файл lab\_calculator/functions:

import math

def log\_history(first\_operand, operator, second\_operand, result):

with open('./lab\_calculator/source/history\_log.txt', 'a') as file:

file.write(f"{first\_operand} {operator} {second\_operand} = {result}\n")

def show\_history():

try:

with open('./lab\_calculator/source/history\_log.txt', 'r') as file:

return file.read()

except FileNotFoundError:

return "History is empty."

def input\_number(prompt, memory\_value):

while True:

user\_input = input(prompt).upper()

if user\_input == 'MR':

print(memory\_value)

return memory\_value

try:

return float(user\_input)

except ValueError:

print("Incorrect input! Please enter a valid number or 'MR'.")

def calculator(decimal\_places, memory\_value):

first\_number = input\_number('Input first operand (or MR for memory recall): ', memory\_value)

operator = input("Enter the operator (+, -, \*, /, ^, sq, %): ")

while operator not in ['+', '-', '\*', '/', '^', 'sq', '%']:

print("Invalid operator. Available operators: +, -, \*, /, ^, sq, %.")

operator = input("Enter operator (+, -, \*, /, ^, sq, %): ")

second\_number = input\_number('Input second operand (or MR for memory recall): ', memory\_value)

result = None

if operator == '+':

result = round(first\_number + second\_number, decimal\_places)

elif operator == '-':

result = round(first\_number - second\_number, decimal\_places)

elif operator == '\*':

result = round(first\_number \* second\_number, decimal\_places)

elif operator == '/':

if second\_number != 0:

result = round(first\_number / second\_number, decimal\_places)

else:

print("Error: division by zero!")

elif operator == '^':

result = round(first\_number \*\* second\_number, decimal\_places)

elif operator == 'sq':

result = round(first\_number \*\* (1 / second\_number), decimal\_places)

elif operator == '%':

if second\_number != 0:

result = round(first\_number % second\_number, decimal\_places)

else:

print("Error: division by zero!")

print(f"{first\_number} {operator} {second\_number} = {result}")

return result, first\_number, operator, second\_number

**Висновок.**

Виконавши ці завдання, я перетворив консольний калькулятор у об'єктно-орієнтований калькулятор, використовуючи класи в Python. Цей проект допоміг мені вивчити концепції об'єктно-орієнтованого програмування та організацію, зберігаючи функціональність і інтерфейс користувача калькулятора.