Міністерство освіти і науки України
Національний університет «Львівська політехніка»

Звіт про виконання лабораторної роботи №2
з дисципліни «Спеціалізовані мови програмування»
на тему «Основи побудови об’єктно-орієнтованих додатків на Python»

Виконав:
студент групи РІ-32
Гусак Віктор
Прийняв:
Щербак С.С.

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.

Хід роботи

calculator_functions.py:

```
import math

def get_user_input():
    try:
        num1 = float(input("Введіть перше число: "))
        operator = input("Введіть оператор (+, -, *, /, ^, %,
sqrt): ")
        if operator != 'sqrt':
            num2 = float(input("Введіть друге число: "))
        else:
            num2 = None
        return num1, operator, num2
    except ValueError:
        print("Помилка! Введіть правильне число.")
        return get_user_input()

def is_valid_operator(operator):
    if operator in ['+', '-', '*', '/', '^', '%', 'sqrt']:
        return True
    else:
        print("Помилка! Невірний оператор.")
        return False

def perform_calculation(num1, operator, num2=None):
    try:
        if operator == '+':
            return num1 + num2
        elif operator == '-':
            return num1 - num2
        elif operator == '*':
            return num1 * num2
        elif operator == '/':
```

```
            if num2 == 0:
                return "Помилка! Ділення на нуль."
            return num1 / num2
        elif operator == '^':
            return num1 ** num2
        elif operator == '%':
            return num1 % num2
        elif operator == 'sqrt':
            return math.sqrt(num1)
    except Exception as e:
        return f"Помилка обчислення: {str(e)}"
```

calculator.py:

```
import sys
import os
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(
__file__))))

from classes.base_calculator import BaseCalculator

class Calculator(BaseCalculator):
    def run(self):
        while True:
            num1, operator, num2 = self.get_input()

            if not self.is_valid_operator(operator):
                print("Невірний оператор. Спробуйте ще раз.")
                continue

            result = self.calculate(num1, operator, num2)
            self.save_to_memory(result)
            self.add_to_history(num1, operator, num2, result)

            print(f"Результат: {result}")
            self.show_history()

            another = input("Виконати ще одне обчислення? (y/n):
").lower()
            if another != 'y':
                print("Дякую за використання калькулятора!")
                break
```

base_calculator.py:

```
import sys
```

```
import os
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(
__file__))))

from functions.calculator_functions import get_user_input,
is_valid_operator, perform_calculation

class BaseCalculator:
    def __init__(self):
        self._memory = None
        self._history = []

    def get_input(self):
        return get_user_input()

    def is_valid_operator(self, operator):
        return is_valid_operator(operator)

    def calculate(self, num1, operator, num2):
        return perform_calculation(num1, operator, num2)

    def save_to_memory(self, result):
        self._memory = result

    def add_to_history(self, num1, operator, num2, result):
        self._history.append(f"{num1} {operator} {num2} =
{result}")

    def show_history(self):
        print("\nІсторія обчислень:")
        for record in self._history:
            print(record)
```

calculator_runner.py:

```
import sys
import os
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(
__file__))))

from classes.calculator import Calculator

calc = Calculator()
calc.run()
```

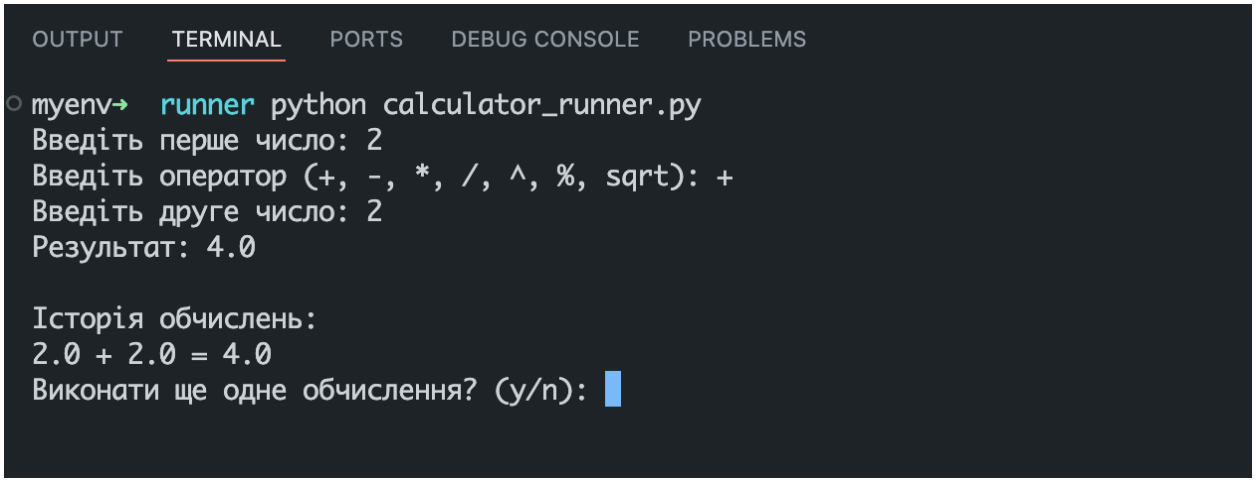

Рис. 1. Приклад роботи програми

Висновки: Виконавши ці завдання, було перетворено консольний калькулятор у об'єктно-орієнтований калькулятор, використовуючи класи в Python.