МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра інформаційних систем та мереж

Лабораторна робота №2

з курсу

СПЕЦІАЛІЗОВАНІ МОВИ ПРОГРАМУВАННЯ

Виконав студент

групи ІТ-21сп

**Гузар А.І.**

Прийняв

**Щербак С.С.**

Львів - 2023

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

**План роботи**

**Завдання 1:** Створення класу Calculator.

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

**Завдання 2:** Ініціалізація калькулятора.

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

**Завдання 3:** Введення користувача.

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

**Завдання 4:** Перевірка оператора.

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

**Завдання 5:** Обчислення.

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

**Завдання 6:** Обробка помилок.

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

**Завдання 7:** Повторення обчислень.

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

**Завдання 8:** Десяткові числа.

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

**Завдання 9:** Додаткові операції.

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

**Завдання 10:** Інтерфейс, зрозумілий для користувача.

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

Код виконаних завдань представлено нижче.

import math

#Task 1: Creating a class.

class Calculator:

# Task 2: Calculator Initialization

def \_\_init\_\_(self):

self.result = None

# Task 3: User Input

def get\_user\_input(self):

try:

# Task 8: Decimal Numbers (float)

self.num1 = float(input("Enter the first number: "))

# Task 9: Additional Operations

self.operator = input("Enter an operator (+, -, \*, /, sqrt, ^, %): ")

except ValueError:

print("Error: Input must be a number.")

return False

return True

# Task 4: Operator Validation

def is\_valid\_operator(self):

valid\_operators = ('+', '-', '\*', '/', 'sqrt', '^', '%')

if self.operator in valid\_operators:

return True

else:

print("Error: Entered operator is invalid. Valid operators are: +, -, \*, /, sqrt, ^, %.")

return False

# Task 5: Calculation

def calculate(self):

if self.is\_valid\_operator():

if self.operator == '+':

self.num2 = float(input("Enter the second number: "))

self.result = self.num1 + self.num2

elif self.operator == '-':

self.num2 = float(input("Enter the second number: "))

self.result = self.num1 - self.num2

elif self.operator == '\*':

self.num2 = float(input("Enter the second number: "))

self.result = self.num1 \* self.num2

elif self.operator == '/':

self.num2 = float(input("Enter the second number: "))

# Task 6.1: Error Handling

if self.num2 == 0:

print("Error: Division by zero is not possible.")

else:

self.result = self.num1 / self.num2

elif self.operator == '^':

self.num2 = float(input("Enter the exponent: "))

self.result = self.num1 \*\* self.num2

elif self.operator == 'sqrt':

# Task 6.2: Error Handling

if self.num1 < 0:

print("Error: Square root of a negative number is not possible.")

else:

self.result = math.sqrt(self.num1)

elif self.operator == '%':

self.num2 = float(input("Enter the percentage number: "))

self.result = (self.num1 \* self.num2) / 100

return True

return False

# Task 7: Repeating Calculations

# Task 10: User-Friendly Interface

def run\_calculator(self):

print("Perform calculations!")

while True:

if self.get\_user\_input():

if self.calculate():

print("Result: {:.2f}".format(self.result))

another\_calculation = input("Perform another operation? (Yes/No): ")

if another\_calculation.lower() != "yes":

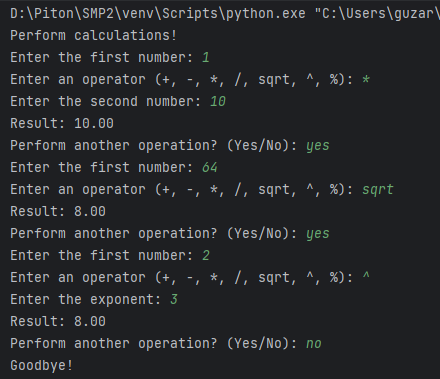
print("Goodbye!")

break

calc = Calculator()

calc.run\_calculator()

На рисунку 1 зображено результат виконання програми.



*Рис.1 Робота калькулятора*

**Висновок:** під час виконання лабораторної роботи я навчився перетворювати консольний калькулятор у об'єктно-орієнтований калькулятор, використовуючи класи в Python. Цей проєкт допоміг мені вивчити концепції об'єктно-орієнтованого програмування та організацію, зберігаючи функціональність і інтерфейс користувача калькулятора.