МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ



Лабораторна робота №2

з дисципліни “Спеціалізовані мови програмування”

на тему

“Основи побудови об’єктно-орієнтованих додатків на Python.”

Виконав:

студент групи РІ-32

Царук В’ячеслав

Прийняв:

к.т.н.,

доц. кафедри ІСМ

Сергій ЩЕРБАК

Львів-2023

**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.

Код програми:

class Calculator:

pass

Завдання 2: Ініціалізація калькулятора

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

Код програми:

class Calculator:

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

self.x = 0

self.y = 0

self.operator = ""

Завдання 3: Введення користувача

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

Код програми:

def user\_input(self):

try:

self.x = float(input("Write your first number: "))

self.y = float(input("Write your second number: "))

except:

print("Invalid number")

exit()

self.operator = input("Choose the operator (+, -, \*, /): ")

while not self.validate\_operator():

print("Invalid operator")

self.operator = input("Choose again (+, -, \*, /): ")

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.

Код програми:

def user\_input(self):

try:

self.x = float(input("Write your first number: "))

self.y = float(input("Write your second number: "))

except:

print("Invalid number")

exit()

self.operator = input("Choose the operator (+, -, \*, /): ")

while not self.validate\_operator():

print("Invalid operator")

self.operator = input("Choose again (+, -, \*, /): ")

def validate\_operator(self):

return self.operator in ['+', '-', '\*', '/']

Завдання 5: Обчислення

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

Код програми:

def calculation(self):

if self.operator == '+':

return self.x + self.y

if self.operator == '-':

return self.x - self.y

if self.operator == '\*':

return self.x \* self.y

if self.operator == '/':

return self.x / self.y

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.

Код програми:

class Calculator:

def calculation(self):

if self.operator == '+':

return self.x + self.y

if self.operator == '-':

return self.x - self.y

if self.operator == '\*':

return self.x \* self.y

if self.operator == '/':

try:

return self.x / self.y

except ZeroDivisionError:

return "You can't divide by 0!"

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.

Код програми:

class AdvancedCalculator(Calculator):

def running\_calculation(self):

print("Welcome to Advanced Calculator!")

while True:

self.user\_input()

result = self.calculation()

if self.operator == 'sqrt':

print(f"The square root of {self.x} = {result}")

else:

print(f"Result of {self.x} {self.operator} {self.y} = {result}")

answer = input("Would you like to do another calculation? (y/n): ")

while answer.lower() not in ["y", "n"]:

print("Invalid input. Please enter 'y' for yes or 'n' for no.")

answer = input("Would you like to do another calculation? (y/n): ")

if answer.lower() == "n":

print("Thank you for using the calculator. Goodbye!")

break

Завдання 8: Десяткові числа

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

Код програми:

def user\_input(self):

try:

self.x = float(input("Write your first number: "))

self.y = float(input("Write your second number: "))

except:

print("Invalid number")

exit()

Завдання 9: Додаткові операції

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

Код програми:

class AdvancedCalculator(Calculator):

...

def calculation(self):

if self.operator in ['^', 'sqrt', '%']:

if self.operator == '^':

return self.x \*\* self.y

if self.operator == 'sqrt':

if self.x < 0:

return "Cannot compute the square root of a negative number!"

return math.sqrt(self.x)

if self.operator == '%':

return self.x % self.y

else:

return super().calculation()

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:

#lab 2

import math

class Calculator:

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

self.x = 0

self.y = 0

self.operator = ""

def user\_input(self):

try:

self.x = float(input("Write your first number: "))

self.y = float(input("Please enter the second number: "))

except ValueError:

print("Error: Invalid number input. Please try again.")

self.user\_input()

return

self.operator = input("Choose the operator (+, -, \*, /): ")

while not self.validate\_operator():

print("Invalid operator")

self.operator = input("Choose again (+, -, \*, /): ")

def validate\_operator(self):

return self.operator in ['+', '-', '\*', '/']

def calculation(self):

if self.operator == '+':

return self.x + self.y

if self.operator == '-':

return self.x - self.y

if self.operator == '\*':

return self.x \* self.y

if self.operator == '/':

try:

return self.x / self.y

except ZeroDivisionError:

return "You can't divide by 0!"

class AdvancedCalculator(Calculator):

def user\_input(self):

try:

self.x = float(input("Write your first number: "))

self.y = float(input("Please enter the second number: "))

except ValueError:

print("Error: Invalid number input. Please try again.")

self.user\_input()

return

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

while not self.validate\_operator():

print("Invalid operator")

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

def validate\_operator(self):

return super().validate\_operator() or self.operator in ['^', 'sqrt', '%']

def calculation(self):

if self.operator in ['^', 'sqrt', '%']:

if self.operator == '^':

return self.x \*\* self.y

if self.operator == 'sqrt':

if self.x < 0:

return "Cannot compute the square root of a negative number!"

return math.sqrt(self.x)

if self.operator == '%':

return self.x % self.y

else:

return super().calculation()

def running\_calculation(self):

print("Welcome to Advanced Calculator!")

while True:

self.user\_input()

result = self.calculation()

if self.operator == 'sqrt':

print(f"The square root of {self.x} = {result}")

else:

print(f"Result of {self.x} {self.operator} {self.y} = {result}")

answer = input("Would you like to do another calculation? (y/n): ")

while answer.lower() not in ["y", "n"]:

print("Invalid input. Please enter 'y' for yes or 'n' for no.")

answer = input("Would you like to do another calculation? (y/n): ")

if answer.lower() == "n":

print("Thank you for using the calculator. Goodbye!")

break

calc = AdvancedCalculator()

calc.running\_calculation()

**GitHub-репозиторій:**

**Висновок:** Зробивши завдання цієї лабораторної роботи, я перетворив консольний калькулятор у об'єктно-орієнтований калькулятор, використовуючи класи в Python. Також ця лабораторна робота допомогла вивчити концепції об'єктно-орієнтованого програмування та організацію, зберігаючи функціональність і інтерфейс користувача калькулятора.