МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра інформаційних систем та мереж



Лабораторна робота №2

з дисципліни «Спеціалізовані мови програмування»

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

Виконав студент

групи РІ-32:

Фігура Т.Я

Прийняв:

Щербак С.С

Львів - 2024

Мета: розробити консольний калькулятор в об’єктно орієнтованому стилі з використанням класів

Завдання 1: Створення класу Calculator

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

class Calculator:  
 def \_\_init\_\_(self, decimal\_places=2):  
 self.memory = 0.0  
 self.decimal\_places = decimal\_places  
 self.history = []  
  
 def get\_user\_input(self):  
 *"""Отримує введення користувача для операції або команд."""*  
first\_number = input("Введіть перше число або 'm' для використання збереженого результату: ")  
 if first\_number == 'm':  
 first\_number = self.memory  
 else:  
 first\_number = validate\_numbers(first\_number)  
  
 operator = input("Введіть оператор (+, -, \*, /, ^, √, %): ")  
 validate\_operator(operator)  
  
 second\_number = None  
 if operator != '√': *# Квадратний корінь не потребує другого числа*  
second\_number = input("Введіть друге число або 'm' для використання збереженого результату: ")  
 if second\_number == 'm':  
 second\_number = self.memory  
 else:  
 second\_number = validate\_numbers(second\_number)  
  
 return first\_number, operator, second\_number

Завдання 2: Ініціалізація калькулятора

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

def \_\_init\_\_(self, decimal\_places=2):  
 self.memory = 0.0  
 self.decimal\_places = decimal\_places  
 self.history = []

Завдання 3: Введення користувача

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

def get\_user\_input(self):  
 *"""Отримує введення користувача для операції або команд."""*  
first\_number = input("Введіть перше число або 'm' для використання збереженого результату: ")  
 if first\_number == 'm':  
 first\_number = self.memory  
 else:  
 first\_number = validate\_numbers(first\_number)  
  
 operator = input("Введіть оператор (+, -, \*, /, ^, √, %): ")  
 validate\_operator(operator)  
  
 second\_number = None  
 if operator != '√': *# Квадратний корінь не потребує другого числа*  
second\_number = input("Введіть друге число або 'm' для використання збереженого результату: ")  
 if second\_number == 'm':  
 second\_number = self.memory  
 else:  
 second\_number = validate\_numbers(second\_number)  
  
 return first\_number, operator, second\_number

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.

def validate\_operator(operator):  
 if operator not in ['+', '-', '\*', '/', '%', '^', '√']:  
 raise ValueError("Invalid operator. Please use +, -, \*, /, %, ^, or √.")  
 return operator

Завдання 5: Обчислення

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

def calculate(self):  
 while True:  
 try:  
 first\_number, operator, second\_number = self.get\_user\_input()  
  
 result = perform\_operation(first\_number, operator, second\_number)  
  
 self.memory = result  
  
 log\_calculation(first\_number, second\_number, operator, result)  
  
 print(f"Result: {round(result, self.decimal\_places)}")  
  
 if input("Хочете продовжити? (y/n): ").lower() != 'y':  
 break  
  
 except Exception as e:  
 print(f"Помилка: {e}")

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.

def divide(a, b):  
 if b == 0:  
 raise ValueError("Division by zero is impossible")  
 return a / b

def square\_root(a, \_):  
 if a < 0:  
 raise ValueError("You cannot calculate the square root of a negative number")  
 return math.sqrt(a)

if operator == '√' and num < 0:  
 raise ValueError("Cannot calculate the square root of a negative number.")  
return num

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.

if input("Хочете продовжити? (y/n): ").lower() != 'y':  
 break

Завдання 8: Десяткові числа

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

def set\_decimal\_places(self, places):  
 self.decimal\_places = places  
 print(f"Decimal places set to: {self.decimal\_places}")

Завдання 9: Додаткові операції

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

*# Словник вибору функції*  
operations = {  
 '+': add,  
 '-': subtract,  
 '\*': multiply,  
 '/': divide,  
 '^': power,  
 '√': square\_root,  
 '%': modulus,  
}

Завдання 10: Інтерфейс, зрозумілий для користувача

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

def main():  
 calc = Calculator()  
  
 while True:  
 print("\n1. Perform a calculation")  
 print("2. View calculation history")  
 print("3. Use memory")  
 print("4. Clear history")  
 print("5. Change decimal places")  
 print("6. Exit")  
  
  
 choice = input("Select an option: ")  
  
 action = options.get(choice)  
 if action:  
 action(calc) *# Передаємо калькулятор як аргумент, при потребі*  
else:  
 print("Invalid choice, try again.")

Посилання на GitHub: [***https://github.com/FihuraTaras/SPL\_Python\_Project.git***](https://github.com/FihuraTaras/SPL_Python_Project.git)

Висновок: 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.