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

Виконала:

ст. гр. ІТ-21сп

Кобленц Д.А

Прийняв:

Щербак С.С.

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.

class Calculator:

Завдання 2. Ініціалізація калькулятора

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

class Calculator:

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

self.history = []

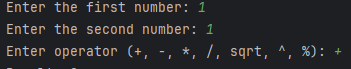
self.memory = None

self.settings = {"decimal\_places": 2, "use\_memory": True}

Завдання 3. Введення користувача

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

first\_input\_str = input("Enter the first number: ")  
self.first\_input = Decimal(first\_input\_str)  
second\_input\_str = input("Enter the second number: ")  
self.second\_input = Decimal(second\_input\_str)



*Рис.1 Виконання програми*

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 get\_operator(self) -> str:

while True:

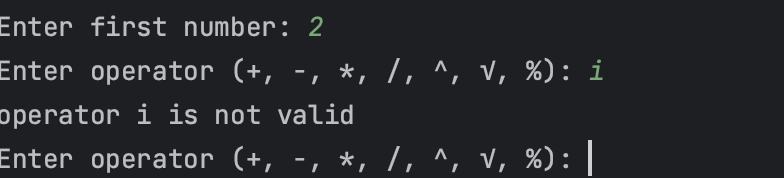
value = input("Enter operator (+, -, \*, /, ^, √, %): ")

if value not in ('+', '-', '\*', '/', '^', '√', '%'):

print(f"Operator {value} is not valid")

else:

return value



*Рис.2 Вивід роботи програми*

Завдання 5. Обчислення

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

def calculate(self, num1: float, operator: str, num2: float) -> float:

operations = {'+': num1 + num2, '-': num1 - num2, '\*': num1 \* num2,

'^': num1 \*\* num2}

if operator == '/':

if num2 == 0:

print("Error: you can't divide by zero")

return None

else:

return num1 / num2

elif operator == '%':

if num2 == 0:

print("Error: you can't modulo by zero")

return None

else:

return num1 % num2

elif operator == '√':

if num1 < 0:

print("Error: you can't get root of negative")

return None

else:

return math.sqrt(num1)

else:

return operations[operator]

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.

Рисунок 8

*Рис.3 Результат виконання програми*

Завдання 8. Десяткові числа

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

self.first\_input = Decimal(first\_input\_str)

self.second\_input = Decimal(second\_input\_str)

Рисунок 5

*Рис.4 Вивід роботи програми*

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.

import math

class Calculator:

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

self.history = []

self.memory = None

self.settings = {"decimal\_places": 2, "use\_memory": True}

def get\_number(self, message: str) -> float:

if self.memory is not None and self.settings["use\_memory"]:

print(f"Value in memory {self.memory}")

use\_from\_memory = input("Do you want to use value from memory? (yes/no)")

if use\_from\_memory.lower() == 'yes':

return self.memory

while True:

try:

return float(input(message))

except:

print("Incorrect number! Enter valid")

def get\_operator(self) -> str:

while True:

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

if value not in ('+', '-', '\*', '/', '^', '√', '%'):

print(f"Operator {value} is not valid")

else:

return value

def calculate(self, num1: float, operator: str, num2: float) -> float:

operations = {'+': num1 + num2, '-': num1 - num2, '\*': num1 \* num2,

'^': num1 \*\* num2}

if operator == '/':

if num2 == 0:

print("Error: you can't divide by zero")

return None

else:

return num1 / num2

elif operator == '%':

if num2 == 0:

print("Error: you can't modulo by zero")

return None

else:

return num1 % num2

elif operator == '√':

if num1 < 0:

print("Error: you can't get root of negative")

return None

else:

return math.sqrt(num1)

else:

return operations[operator]

def save(self, value: float) -> None:

self.memory = value

def start\_calculator(self) -> None:

while True:

first\_num = self.get\_number("Enter first number: ")

operation = self.get\_operator()

second\_num = self.get\_number("Enter second number: ")

result = self.calculate(first\_num, operation, second\_num)

if result is None:

continue

result = round(result, self.settings["decimal\_places"])

print(f"Result: {result}")

self.history.append(f"{first\_num} {operation} {second\_num} = {result}")

if self.settings["use\_memory"]:

save\_choice = input("Do you want to save result? (yes, no)")

if save\_choice == 'yes':

self.save(result)

print(f"Value {result} is saved")

else:

self.memory = None

self.call\_menu()

def call\_menu(self) -> None:

while True:

print("\nCalculator Menu:")

print("1. Continue.")

print("2. View history.")

print("3. Change settings.")

print("4. Quit.")

choice = input("Enter your choice: ")

if choice == "1":

break

elif choice == "2":

self.show\_history()

continue\_choice = input("Do you want to continue? (yes/no)")

if continue\_choice.lower() != 'yes':

return

elif choice == "3":

self.change\_settings()

continue\_choice = input("Do you want to continue? (yes/no)")

if continue\_choice.lower() != 'yes':

return

elif choice == "4":

return

else:

print("Enter valid option")

def change\_settings(self) -> None:

print("1. Change decimal places.")

print("2. Toggle memory function.")

choice = input("Enter your choice: ")

if choice == '1':

try:

places = int(input("Enter number of decimal places (0-10): "))

if 0 <= places <= 10:

self.settings["decimal\_places"] = places

print(f"Set decimal places to {places}.")

else:

print("Invalid number of decimal places.")

except ValueError:

print("Invalid input.")

elif choice == '2':

self.settings["use\_memory"] = not self.settings["use\_memory"]

status = "enabled" if self.settings["use\_memory"] else "disabled"

print(f"Memory function is now {status}.")

def show\_history(self) -> None:

print("History:")

for item in self.history:

print(item)

**Висновок:** я розробив консольний калькулятора в об’єктно орієнтованому стилі з використанням класів.