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 студентка

групи РІ-32

Копейка Х.А.

Прийняв:

Щербак С.С

Львів – 2024

Мета: Розробка консольного калькулятора в об’єктно-орієнтованому стилі з використанням класів.

Результати 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.

```
from Shared.AppSettings import decimal_places
from Shared.logs.logger import log_operation, log_history, show_history
from functions import calculate

class Calculator:
    def __init__(self):
        self.memory = None
        self.decimal_places = decimal_places
```

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.

```
def get_input(self):
        try:
            num1 = float(input("Введіть перше число: "))
            operator = input("Введіть оператор (+, -, *, /, ^, %, √): ")
            num2 = None
            if operator != '√':
                num2 = float(input("Введіть друге число: "))
            return num1, operator, num2
        except ValueError:
            print("Неправильний ввід. Спробуйте знову.")
            return self.get_input()

    def is_valid_operator(self, operator):
        return operator in ['+', '-', '*', '/', '^', '%', '√']
```

Завдання 5: Обчислення

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

```
class Calculator:
   def calculate(self, num1, operator, num2=None):
    match operator:
        case '+':
            result = num1 + num2
        case '-':
            result = num1 - num2
        case '*':
            result = num1 * num2
        case '/':
            if num2 == 0:
                raise ZeroDivisionError("Ділення на нуль!")
            result = num1 / num2
        case '^':
            result = num1 ** num2
        case '%':
            result = num1 % num2
        case '√':
            result = num1 ** 0.5
        case _:
            raise ValueError("Невірний оператор")
```

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 perform_calculation(self, num1, operator, num2):
        try:
            result = calculate(num1, operator, num2)
            return round(result, self.decimal_places)
        except ZeroDivisionError:
            print("Помилка: Ділення на нуль неможливе.")
        except ValueError as e:
            print(f"Помилка: {e}")
        except Exception as e:
            print(f"Виникла непередбачена помилка: {e}")
        return None
```

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.

```
def ask_to_continue(self):
```

```
        return input("Бажаєте виконати ще одне обчислення? (так/ні): ").lower() ==
'так'
```

Завдання 8: Десяткові числа

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

 # Використовуємо float для введення користувача, що дозволяє обробляти десяткові числа

```
num1 = float(input("Введіть перше число: "))
num2 = float(input("Введіть друге число: "))
```

Завдання 9: Додаткові операції

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

 # Піднесення до степеня вже реалізовано в calculate(), аналогічно квадратний корінь та залишок

Завдання 10: Інтерфейс, зрозумілий для користувача

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

```
def store_in_memory(self, result):
        self.memory = result
        print(f"Результат {result} збережений у пам'яті.")

    def recall_memory(self):
        if self.memory is not None:
            print(f"Збережене значення: {self.memory}")
            return self.memory
        else:
            print("Пам'ять порожня.")
            return None

def run(self):
        print(f"Результати відображатимуться з {self.decimal_places} десятковими
знаками.")
        while True:
            num1, operator, num2 = self.get_input()

            if not self.is_valid_operator(operator):
                print("Недійсний оператор. Спробуйте ще раз. Ви можете
використовувати тільки +, -, *, /, ^, %, √")
                continue
```

```
            result = self.perform_calculation(num1, operator, num2)
            if result is not None:
                print(f"Результат: {result}")
                self.store_in_memory(result)

                expression = f"{num1} {operator} {num2 if operator != '√' else ''}"
                log_operation(f"{expression} = {result}")
                log_history(expression, result)

            if input("Бажаєте переглянути історію розрахунків? (так/ні): ").lower()
== 'так':
                show_history()

            if not self.ask_to_continue():
                break
```

```
Результати відображатимуться з  2 з десятковими знаками.
Введіть перше число: 34
Введіть оператор (+, -, *, /, ^, %, √): -
Введіть друге число: 12.678
Результат: 21.32
Результат 21.32 збережений у пам'яті.
Бажаєте переглянути історію розрахунків? (так/ні): ні
Бажаєте виконати ще одне обчислення? (так/ні): так
Введіть перше число: 454
Введіть оператор (+, -, *, /, ^, %, √): =
Введіть друге число: 23.23432344
Недійсний оператор. Спробуйте ще раз. Ви можете використовувати тільки +, -, *, /, ^, %, √
Введіть перше число: 6768.64788
Введіть оператор (+, -, *, /, ^, %, √): /
Введіть друге число: 0
Помилка: ділення на нуль!
Бажаєте переглянути історію розрахунків? (так/ні): ні
Бажаєте виконати ще одне обчислення? (так/ні): так
Введіть перше число: -56
Введіть оператор (+, -, *, /, ^, %, √): √
Помилка: не можна знайти квадратний корінь з від'ємного числа!
Бажаєте переглянути історію розрахунків? (так/ні): ні
Бажаєте виконати ще одне обчислення? (так/ні): так
Введіть перше число: 894
Введіть оператор (+, -, *, /, ^, %, √): ^
Введіть друге число: 12.11
Результат: 5.50423988016225 66e+35
Результат 5.50423988016225 66e+35 збережений у пам'яті.
Бажаєте переглянути історію розрахунків? (так/ні): ні
Бажаєте виконати ще одне обчислення? (так/ні): так
Введіть перше число: 5565747
Введіть оператор (+, -, *, /, ^, %, √): %
Введіть друге число: 726
Результат: 231.0
Результат 231.0 збережений у пам'яті.
```

Рис 1. Робота програми

Посилання на GitHub: https://github.com/Khr178/Python_lab2

Висновок: На цій лабораторній роботі було перетворено консольний калькулятор у об'єктно-орієнтований калькулятор, використовуючи класи в Python. За допомогою цього проекту я вивчила концепції об'єктно-орієнтованого програмування та організацію, зберігаючи функціональність і інтерфейс користувача калькулятора.