Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра ІСМ

Звіт
про виконання лабораторної роботи № 2
«Основи побудови об’єктно-орієнтованих додатків на Python»
з дисципліни
«Спеціалізовані мови програмування»

Виконав:
Студент групи ІТ-32,
Вольвенко І. Р.

Прийняв:
Щербак С.С

Львів 2023

Мета роботи:

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

Завдання

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.

```
def __init__(self, decimal=2, max_history_size=10):
    self.memory = []
    self.history = []
    self.decimal = decimal
    self.max_history_size = max_history_size
    self.result = 0
    self.num1 = 0
    self.operator = '';
    self.num2 = 0
```

Завдання 3: Введення користувача

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

```
def calculate(self):
    self.num1 = float(input("Введіть перше число: "))
    self.operator = input("Введіть оператор (+, -, *, /, ^, √, %): ")

    if self.operator not in ['+', '-', '*', '/', '^', '√', '%']:
        raise ValueError("Недійсний оператор. Введіть один із +, -, *, /, ^, √, %")

    if self.operator != '√':
        self.num2 = float(input("Введіть друге число: "))
```

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.

```
if self.operator not in ['+', '-', '*', '/', '^', '√', '%']:
    raise ValueError("Недійсний оператор. Введіть один із +, -, *, /, ^, √, %")
```

Завдання 5: Обчислення

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

```
if self.operator == '+':
    result = self.num1 + self.num2
elif self.operator == '-':
    result = self.num1 - self.num2
elif self.operator == '*':
    result = self.num1 * self.num2
elif self.operator == '/':
    if self.num2 == 0:
        raise ZeroDivisionError("Ділення на нуль недопустимо.")
    result = self.num1 / self.num2
```

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.

```
elif self.operator == '/':
    if self.num2 == 0:
        raise ZeroDivisionError("Ділення на нуль недопустимо.")
    result = self.num1 / self.num2
```

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.

```
another_calculation = input("Бажаєте виконати ще одне обчислення? (y/n): ")
if another_calculation.lower() != 'y':
    break
```

Завдання 8: Десяткові числа

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

```
result = round(result, self.decimal)
print(f"Результат: {result}")
self.result = result
```

Завдання 9: Додаткові операції

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

```
elif self.operator == '^':
    result = self.num1 ** self.num2
elif self.operator == '√':
    result = self.num1 ** 0.5
elif self.operator == '%':
    result = self.num1 % self.num2
```

Завдання 10: Інтерфейс, зрозумілий для користувача

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

```
if __name__ == "__main__":
    decimal = int(input("Введіть скільки десяткових розрядів після коми показувати: "))
    max_history_size = int(input("Введіть скільки обчислень зберігати у історії: "))

    calculator = Calculator(decimal, max_history_size)
    calculator.run()
```

```python
def run(self):
    while True:
        try:
            self.show_menu()
            choice = input("Ваш вибір: ").lower()

            if choice == 'k':
                self.calculate()
            elif choice == 'h':
                self.display_history()
            elif choice == 'r':
                self.display_memory()
            else:
                self.calculate()

            self.save_result()
            self.update_history()
```

Висновки: Виконавши ці завдання, я 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.