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-21сп

Владислав ДМИТРЕНКО

Львів – 2024

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

**План роботи**

**Завдання 1:** Створіть клас Calculator, який буде служити основою для додатка калькулятора.

**Завдання 2:** Реалізуйте метод \_\_init\_\_ у класі Calculator для ініціалізації необхідних атрибутів або змінних.

**Завдання 3:** Перемістіть функціональність введення користувача в метод у межах класу Calculator. Метод повинен приймати введення для двох чисел і оператора.

**Завдання 4:** Реалізуйте метод у класі Calculator, щоб перевірити, чи введений оператор є дійсним (тобто одним із +, -, \*, /). Відобразіть повідомлення про помилку, якщо він не є дійсним.

**Завдання 5:** Створіть метод у класі Calculator, який виконує обчислення на основі введення користувача (наприклад, додавання, віднімання, множення, ділення).

**Завдання 6:** Реалізуйте обробку помилок у межах класу Calculator для обробки ділення на нуль або інших потенційних помилок. Відобразіть відповідні повідомлення про помилку.

**Завдання 7:** 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.

**Завдання 8:** Модифікуйте клас Calculator для обробки десяткових чисел (плаваюча кома) для більш точних обчислень.

**Завдання 9:** Розширте клас Calculator, щоб підтримувати додаткові операції, такі як піднесення до степеня (^), квадратний корінь (√) та залишок від ділення (%).

**Завдання 10:** Покращте інтерфейс користувача у межах класу Calculator, надавши чіткі запити, повідомлення та форматування виводу для зручності читання.

**Результати тестування:**

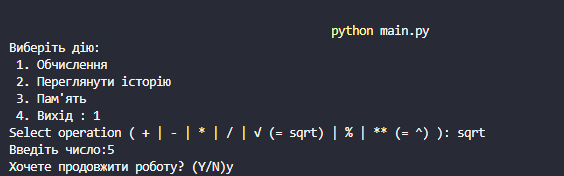


Рис. 1. Результат обрахунків

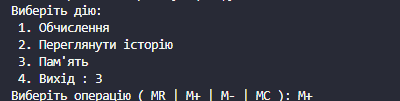


Рис. 2. Приклад зберігання результату в пам’яті

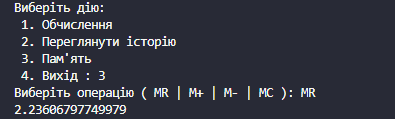
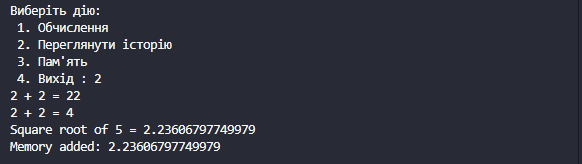


Рис. 3. Приклад зчитування з пам’яті

  
Рис. 3. Приклад роботи історії

Приклад тексту обробника помилок

def GetOperator():

while True:

operator = input("Select operation ( + | - | \* | / | √ (= sqrt) | % | \*\* (= ^) ): ").lower()

try:

if operator not in ["+", "-", "\*", "/", "√", "sqrt", "%", "\*\*"]:

raise ValueError("Invalid operator")

return operator

except ValueError as e:

print(f"Error: {e}. Please enter a valid operator.")

def GetNumber():

while True:

num = input("Введіть число:")

try:

if num == "":

raise ValueError("Field cannot be empty. Please enter a valid number.")

return num

except ValueError as e:

print(f"Error: {e}. Please enter a valid number.")

Приклад конвертора типу числа

def ConvertNumberType(nums):

nums = nums.replace(',', '.')

try:

number = float(nums)

if number.is\_integer():

return int(number)

return number

except ValueError:

raise ValueError(f"Invalid number: '{nums}'")

**Висновки:** В ході виконання даної лабораторної роботи було покращено консольний додаток “Калькулятор” використовуючи наслідування класів, та рефакторинг основної функції за для кращої читабельності та введення такої функції, як історія.