мІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

нАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «лЬВІВСЬКА ПОЛІТЕХНІКА»

ІНСТИТУТ КОМП’ЮТЕРНИХ НАУК ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

|  |
| --- |
| 2023 |



Звіт

Про виконання лабораторної роботи №2

З предмету «Спеціалізовані мови програмування»

Виконав

ст.гр ІТ-21

Капанайко А.Т

Прийняв:

доц.каф

Щербак С.С

**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.

**Завдання 2:** Ініціалізація калькулятора

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

**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.

**Завдання 5:** Обчислення

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

**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.

**Завдання 8:** Десяткові числа

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

**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.

**Код програми:**

import math

class Calculator:

    # Конструктор з атрибутами класу

    def \_\_init\_\_(self):

        self.memory = []  # Інкапсуляція, що створює список пам'яті в класі

        self.history = []  # Інкапсуляція, що створює історію обчислень

        self.decimals = 2  # Інкапсуляція, що визначає кількість знаків після коми

    # Ввід чисел

    def input\_numbers(self):

        self.value1 = float(input("Enter your first value: "))

        self.value2 = float(input("Enter your second value: "))

    # Вибір дії над числами

    def choose\_operator(self):

        while True:

            print("1. +")

            print("2. -")

            print("3. \*")

            print("4. /")

            print("5. ^")

            print("6. √")

            print("7. %")

            self.choose\_operator = input("Enter an arithmetic operator you prefer: ")

            # Якщо некоректно - ерор в консоль + рестарт

            if self.choose\_operator not in ["+", "-", "\*", "/", "^", "√", "%"]:

                print(

                    "Invalid operator has been chosen. Please, follow the list and try one more time!. Please, reboot the program."

                )

                break

            else:

                break

    # Калькуляція чисел і помилки в разі неправильного вибору одного з числа

    def calculation(self):

        if self.choose\_operator == "+":

            self.result = self.value1 + self.value2

        elif self.choose\_operator == "-":

            self.result = self.value1 - self.value2

        elif self.choose\_operator == "\*":

            self.result = self.value1 \* self.value2

        elif self.choose\_operator == "/":

            if self.value1 == 0 or self.value2 == 0:

                print("You can not divide by zero")

            else:

                self.result = self.value1 / self.value2

        elif self.choose\_operator == "^":

            self.result = self.value1\*\*self.value2

        elif self.choose\_operator == "√":

            print("Alert! the square root working only with the first number!")

            self.result = math.sqrt(self.value1)

        elif self.choose\_operator == "%":

            self.result = (self.value1 / self.value2) \* 100

    # Зміна числа після коми

    def change\_decimals(self):

        change\_decimals = input(

            f"Enter a number of decimals if you want to change the default parameter: "

        )

        if change\_decimals:

            self.decimals = int(change\_decimals)  # Інкапсуляція: змінна decimals змінюється в межах класу

            self.result = f"{self.result:.{self.decimals}f}"  # Поліморфізм: форматування результату залежно від decimals

            print(f"Result after changing: {self.result}")

    # Збереження в пам'ять

    def memory\_save(self):

        save\_into\_memory = input("Do you want to save the result in memory? (Y/N): ")

        if save\_into\_memory.lower() == "y":

            self.memory.append(self.result)  # Інкапсуляція: робимо зміни в пам'яті через метод класу

            self.history.append((self.value1, self.choose\_operator, self.value2, self.result))

     # Запит чи хоче користувач виконувати подальші калькуляції

    def more\_calculations(self):

        more\_calc = input("Do you want to make more calculations? (Y/N): ")

        return more\_calc.lower() == "y"

    # Запит на перегляд історії та вивід значень

    def view\_history(self):

        save\_into\_history = input("Do you want to view the history of your calculations? (Y/N): ")

        if save\_into\_history.lower() == "y":

            print("History of calculations:")

            for history\_item in self.history:

                value1, choose\_operator, value2, result = history\_item

                print(f"{value1} {choose\_operator} {value2} = {result}")

            if self.memory:

                print(f"Memory: {self.memory}")

    # Старт програми, а саме виконання всіх блоків-функцій описаних вище

    def run(self):

        print("Starting...")

        while True:

            self.input\_numbers()

            self.choose\_operator()

            self.calculation()

            self.change\_decimals()

            self.memory\_save()

            if not self.more\_calculations():

                print("Disposing...")

                break

        self.view\_history()

# Основна частина програми

if \_\_name\_\_ == "\_\_main\_\_":

    calculator = Calculator()

    calculator.run()

**Посилання на GitHub репозиторій:** [**https://github.com/Senichkaa/lab1-on-python**](https://github.com/Senichkaa/lab1-on-python)

**Висновок:** 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.