МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра інформаційних систем та мереж

Лабораторна робота №2

з дисципліни

«Спеціалізовані мови приграмування»

на тему

«Основи побудови об’єктно-орієнтованих додатків на Python»

Виконав:

ст. гр. РІ-32

Сергій БИХНЮК

Прийняв:

Сергій Щербак

Львів – 2024

**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:**

Settings

DECIMAL\_PLACES = 2 # Кількість десяткових розрядів за замовчуванням

from nis import match

class Calculator:

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

self.num1 = 0

self.num2 = 0

self.operator = ''

def get\_operator(self):

self.operator = input("Виберіть операцію (+, -, \*, /, ^, √, %): ").strip()

return self.operator

def get\_numbers(self):

# Введення двох чисел, якщо оператор не √ (корінь)

try:

self.num1 = float(input("Введіть перше число: "))

if self.operator != '√':

self.num2 = float(input("Введіть друге число: "))

else:

self.num2 = None # Друге число не потрібне для кореня

return self.num1, self.num2

except ValueError:

print("Помилка: введіть коректне число.")

return None

def calculate(self):

try:

# Обчислення на основі вибраного оператора

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 == '/':

result = self.num1 / self.num2 if self.num2 != 0 else float('inf')

elif self.operator == '^':

result = self.num1 \*\* self.num2

elif self.operator == '√':

result = self.num1 \*\* 0.5 # Корінь квадратний з першого числа

elif self.operator == '%':

result = self.num1 % self.num2

else:

print("Неправильний оператор!")

return None

# Округлення результату до 2 знаків після коми

return round(result, 2)

except ZeroDivisionError:

print("Помилка: ділення на нуль.")

return None

except ValueError as e:

print(f"Помилка обчислення: {e}")

return None

class Memory:

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

self.memory\_value = None

def save\_to\_memory(self, value):

self.memory\_value = value

print(f"Значення {value} збережено в пам'ять.")

def recall\_memory(self):

if self.memory\_value is not None:

print(f"Значення в пам'яті: {self.memory\_value}")

else:

print("Пам'ять порожня.")

def clear\_memory(self):

self.memory\_value = None

print("Пам'ять очищено.")

def get\_operator():

# Запитує у користувача оператор.

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

return operator

from lab2.Class.calculator import Calculator

from lab2.Interface.user\_settings import change\_decimal\_places

from lab2.Logs.history import History

from lab2.Class.memory import Memory

import lab2.App\_settings.settings as settings

calc = Calculator()

history = History()

memory = Memory()

def handle\_calculation():

while True: # Цикл для повторного виконання обчислень

operator = calc.get\_operator()

num1, num2 = calc.get\_numbers()

result = calc.calculate()

if result is not None:

print(f"Результат: {result}")

history.add\_record(f"{num1} {operator} {num2 if num2 is not None else ''}", result)

save\_memory = input("Зберегти результат в пам'ять? (так/ні): ").lower()

if save\_memory == 'так':

memory.save\_to\_memory(result)

# Запит на продовження

continue\_calculation = input("Хочете виконати ще одне обчислення? (так/ні): ").lower()

if continue\_calculation != 'так':

print("Дякуємо за використання програми! Вихід з програми.")

break # Вихід з циклу, щоб завершити програму

def handle\_settings():

print("\nНалаштування:")

print("1. Змінити кількість десяткових розрядів")

print("2. Вийти з налаштувань")

choice = input("Виберіть дію (1-2): ")

if choice == '1':

change\_decimal\_places()

elif choice == '2':

return

else:

print("Неправильний вибір, спробуйте ще раз.")

handle\_settings() # Виклик знову для повторного вибору

def handle\_memory():

print("\nПам'ять:")

print("1. Показати значення з пам'яті")

print("2. Очистити пам'ять")

print("3. Додати значення до пам'яті вручну")

print("4. Вийти з меню пам'яті")

memory\_choice = input("Виберіть дію (1-4): ")

if memory\_choice == '1':

memory.recall\_memory()

elif memory\_choice == '2':

memory.clear\_memory()

elif memory\_choice == '3':

try:

manual\_value = float(input("Введіть значення для збереження в пам'ять: "))

memory.save\_to\_memory(manual\_value)

except ValueError:

print("Невірне значення! Введіть число.")

elif memory\_choice == '4':

return

else:

print("Неправильний вибір!")

def get\_numbers(operator):

if operator == '√':

num1 = float(input("Введіть число для обчислення квадратного кореня: "))

return num1, None

else:

num1 = float(input("Введіть перше число: "))

num2 = float(input("Введіть друге число: "))

return num1, num2

def display\_main\_menu():

print("\nМеню:")

print("1. Обчислення")

print("2. Налаштування")

print("3. Використати пам'ять")

print("4. Вийти")

def get\_user\_choice():

return input("Виберіть дію (1-4): ")

def get\_numbers(self):

if self.operator == '√':

self.num1 = float(input("Введіть число для обчислення квадратного кореня: "))

self.num2 = None

else:

self.num1 = float(input("Введіть перше число: "))

self.num2 = float(input("Введіть друге число: "))

return self.num1, self.num2

import lab2.App\_settings.settings as settings

def change\_decimal\_places():

try:

decimal\_places = int(input("Введіть кількість десяткових розрядів для результату: "))

if decimal\_places >= 0:

settings.DECIMAL\_PLACES = decimal\_places

print(f"Кількість десяткових розрядів змінено на {settings.DECIMAL\_PLACES}.")

else:

print("Кількість десяткових розрядів повинна бути не від'ємною!")

except ValueError:

print("Введіть ціле число!")

def settings\_menu():

print("\nНалаштування:")

print("1. Змінити кількість десяткових розрядів")

print("2. Вийти з налаштувань")

choice = input("Виберіть дію (1-2): ")

if choice == '1':

change\_decimal\_places()

elif choice == '2':

return

else:

print("Неправильний вибір, спробуйте ще раз.")

settings\_menu()

class History:

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

self.records = []

def add\_record(self, expression, result):

self.records.append(f"{expression} = {result}")

def show\_history(self):

if self.records:

for record in self.records:

print(record)

else:

print("Історія порожня.")

def clear\_history(self):

self.records = []

print("Історія очищена.")

from lab2.Interface.user\_interface import display\_main\_menu, get\_user\_choice

from lab2.Interface.menu\_actions import handle\_calculation, handle\_settings, handle\_memory

def main2():

while True:

display\_main\_menu()

choice = get\_user\_choice()

if choice == '1':

handle\_calculation()

elif choice == '2':

handle\_settings()

elif choice == '3':

handle\_memory()

elif choice == '4':

print("Вихід з програми.")

break

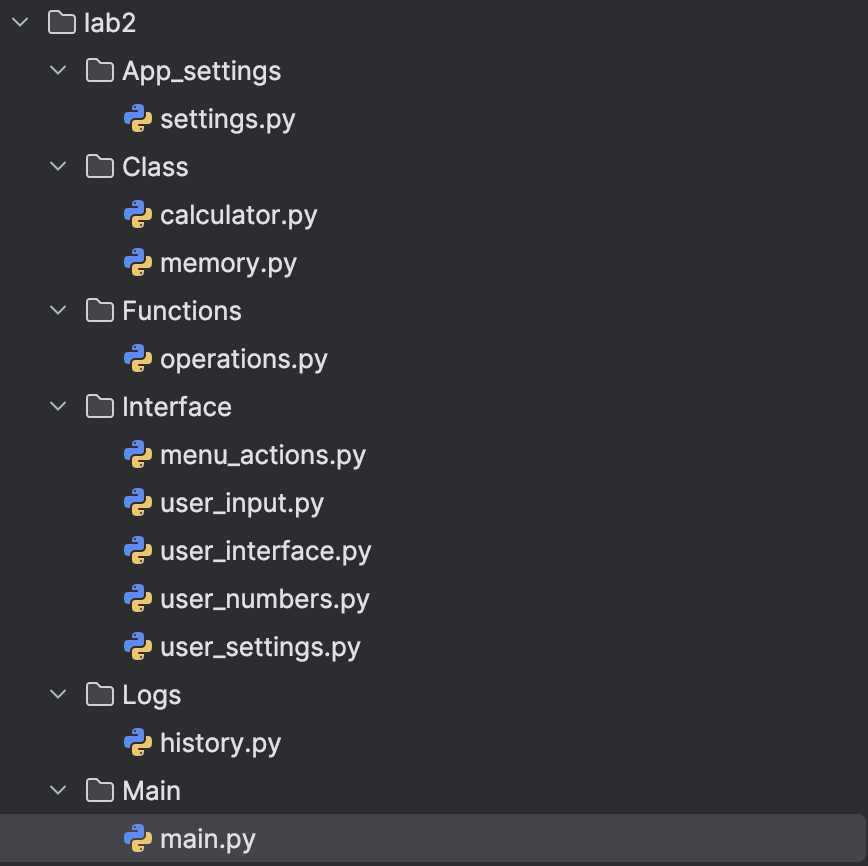
else:

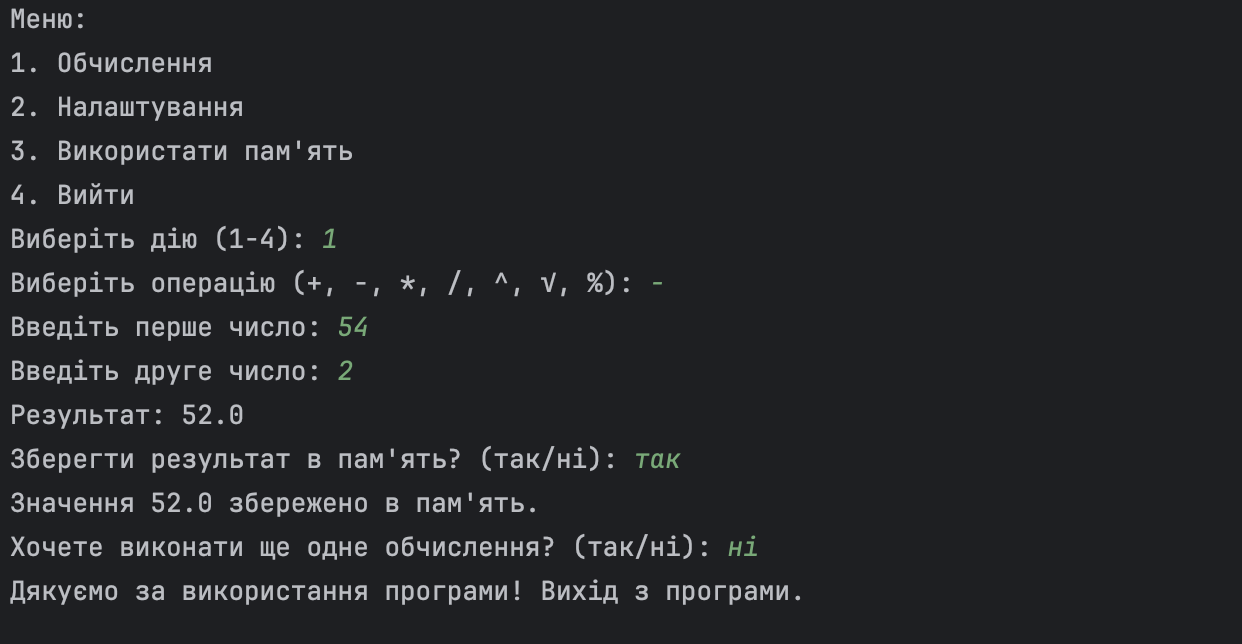
print("Невірний вибір, спробуйте знову.")

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

main2()

**Виконання:**

****



**Висновок:**

Розробив консольний калькулятора в об’єктно орієнтованому стилі з використанням класів