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:

ст. гр. РІ-32

Мельничук Ю. А.

Прийняв:

доц. каф. ІСМ

Щербак С. С.

Львів - 2024

**Мета**

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

**Хід виконання роботи**

**Завдання 1: Створення класу Calculator**

Створіть клас Calculator, який буде служити основою для додатка калькулятора.

Створено клас Calculator:

class Calculator:

pass

**Завдання 2: Ініціалізація калькулятора**

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

Додано до класу метод \_\_init\_\_.

class Calculator:

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

self.num1 = None

self.num2 = None

self.operator = None

**Завдання 3: Введення користувача**

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

Додано метод, що запитує в користувача оператор та два числа, й присвоює ці значення відповідним полям класу.

def set\_operation(self):

try:

self.num1 = int(input("Enter first number: "))

self.operator = input(

f"Enter operator ({', '.join(self.available\_operations)}): "

)

self.num2 = int(input("Enter second number: "))

except ValueError:

print("Wrong input")

return self.set\_operation()

**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.

До ініціалізатора класу додано перелік доступних операторів, які є у вигляді масиву всі операцій, які підтримує програма.

def \_\_init\_\_(self, available\_operations):

self.num1 = None

self.num2 = None

self.operator = None

self.available\_operations = available\_operations

А також сам метод перевірки:

def is\_setted\_operator\_valid(self):

if self.operator in self.available\_operations:

return True

else:

print(f"Wrong operator {operator}")

return False

**Завдання 5: Обчислення**

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

Додано поле результату яке при ініціалізації отримує значення None, але при роботі програми зберігатиме останній результат операцій.

self.result = None

А також метод, який виконує операції над полями:

def calculate(self):

if self.operator == "+":

self.result = self.num1 + self.num2

elif self.operator == "-":

self.result = self.num1 - self.num2

elif self.operator == "\*":

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

elif self.operator == "/":

self.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.

Додано клас для логування помилок. Він має два конструктори, перший приймає шлях до файлу логування, а також опціонально чи варто записувати помилки в консоль, а другий просто виводить помилки в консоль.

# imports above

# In creation, if file don't exists

# raise error

class Logger:

# Log to file and to console if needed

def \_\_init\_\_(self, file\_path, is\_write\_to\_console=True):

self.\_ensure\_file\_exists(file\_path)

self.file\_path = file\_path

self.is\_write\_to\_file = True

self.is\_write\_to\_console = is\_write\_to\_console

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

self.is\_write\_to\_console = True

self.is\_write\_to\_file = False

def log\_error(self, message, error\_level="ERROR"):

error\_message = f"{str(datetime.now())} - {message}"

if self.is\_write\_to\_console:

print(f"{error\_level}: {message}")

if self.is\_write\_to\_file:

self.\_write\_to\_file(error\_message)

def \_ensure\_file\_exists(file\_path):

try:

if exists(file\_path):

return

makedirs(dirname(file\_path), exist\_ok=True)

with open(file\_path, "w") as file:

return

except (OSError, PermissionError) as e:

raise f"Error occurred while creating the file or directories: {e}"

def \_write\_to\_file(self, message):

try:

with open(self.file\_path, "a", encoding="utf-8") as file:

file.write(f"{message}\n")

except Exception as e:

print(f"Problem writing to file: {e}")

До конструктора додано створення екземпляру класу логера, якому передається шлях до файлу логів, що є новим параметром конструктора. Також додано перевірку при ділені чи друге число не нуль та замість звичайного запису помилок в консоль, передаємо їх в логер:

if self.num2 == 0:

self.logger.log\_error("Division by zero")

**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.

Додано метод меню, який має нескінченний цикл та право вибору користувачем продовження використання або вихід.

def menu(self):

while True:

print("\n=== Console calculator ===")

print("1. Calculate")

print("0. Exit")

choice = input("Choose option (1 or 0): ")

match choice:

case "1":

self.set\_operation()

self.calculate()

print(f"{self.num1} {self.operator} {self.num2} = {self.result}")

case "0":

print("Exit.")

break

case \_:

print("Wrong input. Try again.")

**Завдання 8: Десяткові числа**

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

Змінено перетворення введення користувача на числа з плаваючою комою.

self.num1 = float(input("Enter first number: "))

self.num2 = float(input("Enter second number: "))

**Завдання 9: Додаткові операції**

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

Оскільки серед нових операцій є квадратний корінь, який є операцією щодо одного числа, необхідно додати також масив одиничних операцій, змінити логіку перевірки правильності введених чисел тощо.

**Завдання 10: Інтерфейс, зрозумілий для користувача**

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

Зображення, що містить текст, знімок екрана, Шрифт, число

Автоматично згенерований опис

Рис. 1. Початкове меню програми

**Висновки**

Я створив простий консольний калькулятор використовуючи класи та ООП на Python, котрий також може перевіряти помилки, відображати та записувати їх у файли, реалізує функцію пам’яті. Я вивчив основний синтаксис Python та його концепції введення користувача, обробку помилок, умовні оператори, імпорт модулів, цикли тощо.