МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"

Кафедра інформаційних систем та мереж



Лабораторна робота №1

з дисципліни «Спеціалізовані мови програмування»

на тему «Введення в Python»

Виконав студент

групи РІ-32:

Фігура Т.Я

Прийняв:

Щербак С.С

Львів - 2024

**Мета:** створення консольної програми-калькулятора за допомогою основних синтаксичних конструкцій Python, з іншим завданням на заміну тестуванню та валідації.

**Хід роботи:**

*Завдання 1:* Введення користувача

Створіть Python-програму, яка приймає введення користувача для двох чисел і оператора (наприклад, +, -, \*, /).

def get\_user\_input():  
 *# Input two numbers and operator*  
num1 = input("Enter the first number (or 'm' to use memory): ")  
 num2 = input("Enter the second number: ")  
 operator = input("Enter the operator (+, -, \*, /, %, ^, √): ")

*Завдання 2:* Перевірка оператора

Перевірте чи введений оператор є дійсним (тобто одним із +, -, \*, /). Якщо ні, відобразіть повідомлення про помилку і попросіть користувача ввести дійсний оператор.

def validate\_operator(operator):  
 if operator not in ['+', '-', '\*', '/', '%', '^', '√']:  
 raise ValueError("Invalid operator. Please use +, -, \*, /, %, ^, or √.")  
 return operator

*Завдання 3:* Обчислення

Виконайте обчислення на основі введення користувача (наприклад, додавання, віднімання, множення, ділення) і відобразіть результат.

def calculate(self):  
 while True:  
 try:  
 num1, num2, operator = self.get\_user\_input()  
  
 *# If the user entered 'm', we use the value from memory*  
if num1 == 'm':  
 if self.memory is None:  
 print("Memory is empty.")  
 continue  
 num1 = self.memory  
  
 result = perform\_operation(num1, num2, operator)  
  
 *# We format the result by the number of decimal places*  
result = round(result, self.decimal\_places)  
  
 print(f"The result of {num1} {operator} {num2} is: {result}")  
 log\_calculation(num1, num2, operator, result)  
 self.history.append((num1, num2, operator, result))

*Завдання 4:* Повторення обчислень

Запитайте користувача, чи він хоче виконати ще одне обчислення. Якщо так, дозвольте йому ввести нові числа і оператор. Якщо ні, вийдіть з програми.

if input("Save result to memory? (y/n): ").lower() == 'y':  
 self.memory = result  
  
 if input("Do you want to perform another calculation? (y/n): ").lower() != 'y':  
 break  
except Exception as e:  
 print(f"Error: {e}")

*Завдання 5:* Обробка помилок

Реалізуйте обробку помилок для обробки ділення на нуль або інших потенційних помилок. Відобразіть відповідне повідомлення про помилку, якщо виникає помилка.

elif operator == '/':  
 if num2 == 0:  
 raise ZeroDivisionError("Cannot divide by zero.")  
 return num1 / num2

else:  
 raise ValueError(f"Unsupported operator: {operator}")

def validate\_numbers(num, operator=None):  
 try:  
 num = float(num)  
 if operator == '√' and num < 0:  
 raise ValueError("Cannot calculate the square root of a negative number.")  
 return num  
 except ValueError:  
 raise ValueError("Invalid number entered.")  
  
def validate\_operator(operator):  
 if operator not in ['+', '-', '\*', '/', '%', '^', '√']:  
 raise ValueError("Invalid operator. Please use +, -, \*, /, %, ^, or √.")  
 return operator

*Завдання 6:* Десяткові числа

Змініть калькулятор так, щоб він обробляв десяткові числа (плаваючу кому) для більш точних обчислень.

def perform\_operation(num1, num2, operator):  
 if operator == '+':  
 return num1 + num2  
 elif operator == '-':  
 return num1 - num2  
 elif operator == '\*':  
 return num1 \* num2  
 elif operator == '/':  
 if num2 == 0:  
 raise ZeroDivisionError("Cannot divide by zero.")  
 return num1 / num2  
 elif operator == '%':  
 return num1 % num2  
 elif operator == '^':  
 return num1 \*\* num2  
 elif operator == '√':  
 return math.sqrt(num1)  
 else:  
 raise ValueError(f"Unsupported operator: {operator}")

print(f"The result of {num1} {operator} {num2} is: {result}")  
log\_calculation(num1, num2, operator, result)  
self.history.append((num1, num2, operator, result))

*Завдання 7:* Додаткові операції

Додайте підтримку додаткових операцій, таких як піднесення до степеня (^), квадратний корінь (√) і залишок від ділення (%).

elif operator == '%':  
 return num1 % num2  
elif operator == '^':  
 return num1 \*\* num2  
elif operator == '√':  
 return math.sqrt(num1)  
else:  
 raise ValueError(f"Unsupported operator: {operator}")

*Завдання 8:* Функція пам'яті

Реалізуйте функцію пам'яті, яка дозволяє користувачам зберігати і відновлювати результати. Додайте можливості для зберігання та отримання значень з пам'яті.

while True:  
 try:  
 num1, num2, operator = self.get\_user\_input()  
  
 *# If the user entered 'm', we use the value from memory*  
if num1 == 'm':  
 if self.memory is None:  
 print("Memory is empty.")  
 continue  
 num1 = self.memory  
  
 result = perform\_operation(num1, num2, operator)

*Завдання 9:* Історія обчислень

Створіть журнал, який зберігає історію попередніх обчислень, включаючи вираз і результат. Дозвольте користувачам переглядати історію своїх обчислень.

def log\_calculation(num1, num2, operator, result):  
 with open("Logs/calculation\_history.txt", "a") as log\_file:  
 log\_file.write(f"{num1} {operator} {num2} = {result}\n")  
  
def show\_history():  
 try:  
 with open("Logs/calculation\_history.txt", "r") as log\_file:  
 history = log\_file.readlines()  
 for line in history:  
 print(line.strip())  
 except FileNotFoundError:  
 print("No calculation history found.")

*Завдання 10:* Налаштування користувача

Надайте користувачам можливість налаштувати поведінку калькулятора, таку як зміну кількості десяткових розрядів, які відображаються, або налаштування функцій пам'яті.

ef \_\_init\_\_(self, decimal\_places=2):  
 self.memory = None  
 self.history = []  
 self.decimal\_places = decimal\_places *# Number of decimal*

def change\_decimal\_places(self):  
 *# The user can change the number of decimal places*  
try:  
 new\_places = int(input("Enter the number of decimal places to display: "))  
 if new\_places >= 0:  
 self.decimal\_places = new\_places  
 print(f"Decimal places set to: {self.decimal\_places}")  
 else:  
 print("Decimal places must be 0 or higher.")  
 except ValueError:  
 print("Invalid input. Please enter an integer.")

***Посилання на GitHub:***[***https://github.com/FihuraTaras/SPL\_Python\_Project.git***](https://github.com/FihuraTaras/SPL_Python_Project.git)

**Висновок:** створив простий консольний калькулятор на Python, який може виконувати арифметичні операції, обробляти помилки та надавати користувачу зручний інтерфейс. Проект допоміг вивчити основний синтаксис Python і концепції, такі як введення користувача, умовні оператори, цикли та обробка помилок.