Міністерство освіти і науки України

Національний університет «Львівська політехніка»

лабораторної роботи №1

з дисципліни «Спеціалізовані мови програмування»

на тему «Введення в Python»

Виконав:

Коханець П. Р.

Перевірив:

Щербак С.С.

Львів 2024

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

Git: <https://github.com/Pivinter/-.git>

Завдання 1: Введення користувача

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

def add(self, x, y):

return x + y

def subtract(self, x, y):

return x - y

def multiply(self, x, y):

return x \* y

def divide(self, x, y):

if y == 0:

return "Error: Division by zero is undefined."

return x / y

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.

else:

print("Invalid input, please select a valid operation.")

Завдання 3: Обчислення

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

if choice == '1':

result = calc.add(num1, num2)

print(f"{num1} + {num2} = {format\_result(result)}")

log\_history(f"{num1} + {num2}", format\_result(result))

elif choice == '2':

result = calc.subtract(num1, num2)

print(f"{num1} - {num2} = {format\_result(result)}")

log\_history(f"{num1} - {num2}", format\_result(result))

elif choice == '3':

result = calc.multiply(num1, num2)

print(f"{num1} \* {num2} = {format\_result(result)}")

log\_history(f"{num1} \* {num2}", format\_result(result))

elif choice == '4':

result = calc.divide(num1, num2)

print(f"{num1} / {num2} = {format\_result(result)}")

log\_history(f"{num1} / {num2}", format\_result(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.

next\_calculation = input("Do you want to perform another calculation? (yes/no): ")

if next\_calculation.lower() != 'yes':

break

Завдання 5: Обробка помилок

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

def sqrt(self, x):

if x < 0:

return "Error: Square root of a negative number is undefined."

return math.sqrt(x)

def remainder(self, x, y):

if y == 0:

return "Error: Division by zero is undefined."

return x % y

def divide(self, x, y):

if y == 0:

return "Error: Division by zero is undefined."

return x / y

Завдання 6: Десяткові числа

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

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

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

Завдання 7: Додаткові операції

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

def exponentiate(self, x, y):

return x \*\* y

def sqrt(self, x):

if x < 0:

return "Error: Square root of a negative number is undefined."

return math.sqrt(x)

def remainder(self, x, y):

if y == 0:

return "Error: Division by zero is undefined."

return x % y

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.

def memory\_save(result):

global memory

memory = result

print(f"Result {result} saved to memory.")

def memory\_recall():

global memory

if memory is not None:

print(f"Memory recall: {memory}")

return memory

else:

print("Memory is empty.")

return None

def memory\_clear():

global memory

memory = None

print("Memory cleared.")

Завдання 9: Історія обчислень

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

def log\_history(expression, result):

history.append(f"{expression} = {result}")

def view\_history():

if history:

print("\n--- Calculation History ---")

for entry in history:

print(entry)

else:

print("No history available.")

def clear\_history():

history.clear()

print("Calculation history cleared.")

Завдання 10: Налаштування користувача

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

settings = {

"decimal\_places": 2, # Default number of decimal places

"auto\_memory\_save": False, # Automatically save results to memory

"auto\_memory\_clear": False # Automatically clear memory after session

}

def set\_decimal\_places():

try:

decimal\_places = int(input("Enter the number of decimal places to display (0-10): "))

if 0 <= decimal\_places <= 10:

settings["decimal\_places"] = decimal\_places

print(f"Decimal places set to {decimal\_places}.")

else:

print("Please enter a number between 0 and 10.")

except ValueError:

print("Invalid input. Please enter a number.")

def toggle\_auto\_memory\_save():

settings["auto\_memory\_save"] = not settings["auto\_memory\_save"]

status = "enabled" if settings["auto\_memory\_save"] else "disabled"

print(f"Auto Memory Save is now {status}.")

def toggle\_auto\_memory\_clear():

settings["auto\_memory\_clear"] = not settings["auto\_memory\_clear"]

status = "enabled" if settings["auto\_memory\_clear"] else "disabled"

print(f"Auto Memory Clear is now {status}.")

def format\_result(result):

return round(result, settings["decimal\_places"])

Висновок: Виконавши ці завдання, ви створите простий консольний калькулятор на Python, який може виконувати арифметичні операції, обробляти помилки та надавати користувачу зручний інтерфейс. Цей проект допоможе вам вивчити основний синтаксис Python і концепції, такі як введення користувача, умовні оператори, цикли та обробка помилок.