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:

ст. гр. РІ-32

Мельничук Ю. А.

Прийняв:

доцент кафедри ІСМ

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 виконання роботи**

Завдання 1: Введення користувача

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

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.

Завдання 3: Обчислення

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

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.

Завдання 5: Обробка помилок

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

Завдання 6: Десяткові числа

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

Завдання 7: Додаткові операції

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

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.

Завдання 9: Історія обчислень

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

Завдання 10: Налаштування користувача

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

**Виконання**

import tkinter as tk  
from tkinter import messagebox  
import math  
  
history = []  
memory = None  
decimal\_places = 2 # Значення за замовчуванням для кількості знаків після коми  
  
def calculate(num1, num2, operator):  
 try:  
 if operator == '+':  
 return num1 + num2  
 elif operator == '-':  
 return num1 - num2  
 elif operator == '\*':  
 return num1 \* num2  
 elif operator == '/':  
 return num1 / num2  
 elif operator == '^':  
 return num1 \*\* num2  
 elif operator == '%':  
 return num1 % num2  
 else:  
 return None  
 except ZeroDivisionError:  
 return "Помилка: Ділення на нуль!"  
  
def sqrt(num):  
 try:  
 return math.sqrt(num)  
 except ValueError:  
 return "Помилка: Неможливо взяти корінь з від'ємного числа!"  
  
def update\_history(entry):  
 history.append(entry)  
 history\_list.insert(tk.END, entry)  
  
def clear\_inputs():  
 entry\_num1.delete(0, tk.END)  
 entry\_num2.delete(0, tk.END)  
  
def perform\_calculation(operator):  
 global memory  
 try:  
 if operator == 'sqrt':  
 num = float(entry\_num1.get())  
 result = sqrt(num)  
 formatted\_result = round(result, decimal\_places)  
 update\_history(f"sqrt({num}) = {formatted\_result}")  
 result\_label.config(text=f"Результат: {formatted\_result}")  
 else:  
 num1 = float(entry\_num1.get())  
 num2 = float(entry\_num2.get())  
 result = calculate(num1, num2, operator)  
 formatted\_result = round(result, decimal\_places)  
 update\_history(f"{num1} {operator} {num2} = {formatted\_result}")  
 result\_label.config(text=f"Результат: {formatted\_result}")  
  
 # Збереження в пам'ять  
 if memory\_var.get():  
 memory = formatted\_result  
 memory\_label.config(text=f"Пам'ять: {memory}")  
  
 except ValueError:  
 messagebox.showerror("Помилка", "Введіть коректне число!")  
  
def clear\_history():  
 history\_list.delete(0, tk.END)  
 history.clear()  
  
# Функції пам'яті  
def memory\_clear():  
 global memory  
 memory = None  
 memory\_label.config(text="Пам'ять: Немає")  
  
def memory\_save():  
 global memory  
 memory = result\_label.cget("text").split(": ")[1] # Витягнути результат  
 memory\_label.config(text=f"Пам'ять: {memory}")  
  
def memory\_recall():  
 if memory is not None:  
 entry\_num1.delete(0, tk.END)  
 entry\_num1.insert(0, memory)  
 else:  
 messagebox.showinfo("Пам'ять", "Пам'ять порожня!")  
  
def memory\_add():  
 global memory  
  
 print(memory)  
 try:  
 if memory is None:  
 memory = float(result\_label.cget("text").split(": ")[1])  
 else:  
 memory += float(str(result\_label.cget("text").split(": ")[1]))  
 memory\_label.config(text=f"Пам'ять: {memory}")  
 except ValueError:  
 messagebox.showerror("Помилка", "Неможливо додати до пам'яті. Невірне значення!")  
  
# Функція для зміни кількості десяткових знаків  
def set\_decimal\_places(value):  
 global decimal\_places  
 decimal\_places = int(value)  
 decimal\_label.config(text=f"Знаків після коми: {decimal\_places}")  
  
root = tk.Tk()  
root.title("Lab 1")  
  
label\_num1 = tk.Label(root, text="Перше число:")  
label\_num1.grid(row=0, column=0, padx=10, pady=5)  
entry\_num1 = tk.Entry(root)  
entry\_num1.grid(row=0, column=1, padx=10, pady=5)  
label\_num2 = tk.Label(root, text="Друге число (не потрібно для √):")  
label\_num2.grid(row=1, column=0, padx=10, pady=5)  
entry\_num2 = tk.Entry(root)  
entry\_num2.grid(row=1, column=1, padx=10, pady=5)  
  
def create\_operator\_button(text, operator):  
 return tk.Button(root, text=text, command=lambda: perform\_calculation(operator))  
  
button\_add = create\_operator\_button("+", "+")  
button\_add.grid(row=2, column=0, padx=10, pady=5)  
button\_subtract = create\_operator\_button("-", "-")  
button\_subtract.grid(row=2, column=1, padx=10, pady=5)  
button\_multiply = create\_operator\_button("\*", "\*")  
button\_multiply.grid(row=3, column=0, padx=10, pady=5)  
button\_divide = create\_operator\_button("/", "/")  
button\_divide.grid(row=3, column=1, padx=10, pady=5)  
button\_power = create\_operator\_button("^", "^")  
button\_power.grid(row=4, column=0, padx=10, pady=5)  
button\_modulus = create\_operator\_button("%", "%")  
button\_modulus.grid(row=4, column=1, padx=10, pady=5)  
button\_sqrt = create\_operator\_button("√", "sqrt")  
button\_sqrt.grid(row=5, column=0, columnspan=2, padx=10, pady=5)  
  
# Кнопки пам'яті  
button\_mc = tk.Button(root, text="MC", command=memory\_clear)  
button\_mc.grid(row=6, column=0, padx=10, pady=5)  
  
button\_ms = tk.Button(root, text="MS", command=memory\_save)  
button\_ms.grid(row=6, column=1, padx=10, pady=5)  
  
button\_mr = tk.Button(root, text="MR", command=memory\_recall)  
button\_mr.grid(row=7, column=0, padx=10, pady=5)  
  
button\_mplus = tk.Button(root, text="M+", command=memory\_add)  
button\_mplus.grid(row=7, column=1, padx=10, pady=5)  
  
  
memory\_var = tk.BooleanVar()  
memory\_checkbox = tk.Checkbutton(root, text="Зберегти результат у пам'ять", variable=memory\_var)  
memory\_checkbox.grid(row=8, column=0, columnspan=2, padx=10, pady=5)  
  
memory\_label = tk.Label(root, text="Пам'ять: Немає")  
memory\_label.grid(row=9, column=0, columnspan=2, padx=10, pady=5)  
  
  
result\_label = tk.Label(root, text="Результат:")  
result\_label.grid(row=10, column=0, columnspan=2, padx=10, pady=5)  
  
history\_label = tk.Label(root, text="Історія:")  
history\_label.grid(row=11, column=0, padx=10, pady=5)  
history\_list = tk.Listbox(root, height=10, width=40)  
history\_list.grid(row=12, column=0, columnspan=2, padx=10, pady=5)  
  
clear\_history\_button = tk.Button(root, text="Очистити історію", command=clear\_history)  
clear\_history\_button.grid(row=13, column=0, columnspan=2, padx=10, pady=5)  
  
# Повзунок для вибору кількості знаків після коми  
decimal\_scale = tk.Scale(root, from\_=0, to=10, orient=tk.HORIZONTAL, label="Знаки після коми", command=set\_decimal\_places)  
decimal\_scale.set(decimal\_places) # Встановити початкове значення  
decimal\_scale.grid(row=14, column=0, columnspan=2, padx=5, pady=5)  
  
decimal\_label = tk.Label(root, text=f"Знаків після коми: {decimal\_places}")  
decimal\_label.grid(row=15, column=0, columnspan=2, padx=10, pady=5)  
  
root.mainloop()

**Висновки**

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.