

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-21сп
Рак В.П.
Прийняв:
Щербак С.С.

Львів-2024

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

Завдання

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 роботи

Текст програмної реалізації:

```
import json
```

```
import math
```

```
# Локалізація через словник
```

```
locale = {
```

```
    "input_first": "Введіть перше число: ",
```

```
    "input_second": "Введіть друге число: ",
```

```
    "input_operator": "Введіть оператор (+, -, *, /, ^, %, √): ",
```

```
    "invalid_operator": "Неправильний оператор! Спробуйте ще раз.",
```

```
    "result": "Результат: ",
```

```
    "use_memory": "Використати збережене значення з пам'яті? (так/ні): ",
```

```
    "calc_again": "Виконати ще одне обчислення? (так/ні): ",
```

```
    "history": "Переглянути історію? (так/ні): ",
```

```
    "save_history": "Історія збережена у файл history.json.",
```

```
    "error_div_by_zero": "Помилка: ділення на нуль!",
```

```
    "error_negative_sqrt": "Помилка: корінь з від'ємного числа!",
```

```
    "error_invalid_input": "Помилка: введено некоректні дані!",
```

```
    "error_file": "Помилка доступу до файлу!",
```

```
    "error_unknown": "Несподівана помилка!"
```

```
}
```

```
# Читання і запис історії у файл
```

```
def save_history(history):
```

```
"""
```

Зберігає історію обчислень у файл history.json.

Аргументи:

history (list): Список рядків, що містять історію обчислень.

Викидає:

IOError: Якщо файл не можна відкрити для запису.

```
"""
```

```
try:
```

```
    with open("history.json", "w", encoding="utf-8") as f:
```

```
        json.dump(history, f, ensure_ascii=False, indent=4)
```

```
except IOError:
```

```
    print(locale["error_file"])
```

```
def load_history():
```

```
    """
```

Завантажує історію обчислень з файлу history.json.

Повертає:

list: Список рядків з історією обчислень.

```
"""
```

```
try:
```

```
    with open("history.json", "r", encoding="utf-8") as f:
```

```
        return json.load(f)
```

```
except FileNotFoundError:
```

```
    return [] # Повертаємо порожній список, якщо файл не знайдений
```

```
except json.JSONDecodeError:
```

```
    print(locale["error_file"])
```

```
    return []
```

```
# Введення користувача і перевірка оператора
```

```
def get_operator():
```

"""

Отримує оператор для обчислення від користувача і перевіряє його на коректність.

Повертає:

str: Оператор, вибраний користувачем (+, -, *, /, ^, %, √).

"""

while True:

try:

operator = input(locale["input_operator"])

if operator in ['+', '-', '*', '/', '^', '%', '√']:

return operator # Повертаємо правильний оператор

print(locale["invalid_operator"]) # Якщо оператор неправильний

except Exception:

print(locale["error_unknown"]) # Обробка будь-якої іншої помилки

Функція для виконання обчислень з використанням модуля math

def calculate(num1, num2, operator):

"""

Виконує обчислення двох чисел з заданим оператором.

Аргументи:

num1 (float): Перше число для обчислення.

num2 (float): Друге число для обчислення (для кореня num2 не використовується).

operator (str): Оператор обчислення.

Повертає:

float або str: Результат обчислення або повідомлення про помилку.

"""

try:

if operator == '+':

return num1 + num2

if operator == '-':

return num1 - num2

```

    if operator == '*':
        return num1 * num2
    if operator == '/':
        return num1 / num2 if num2 != 0 else locale["error_div_by_zero"]
    if operator == '^':
        return math.pow(num1, num2)
    if operator == '%':
        return num1 % num2
    if operator == '√':
        return math.sqrt(num1) if num1 >= 0 else locale["error_negative_sqrt"]
except Exception as e:
    print(f'Помилка при обчисленні: {e}')
    return None

```

Основна функція

```
def main():
```

```
    """
```

Основна функція калькулятора, яка керує процесом введення чисел, операцій, обчислень, збереженням історії та використанням пам'яті.

```
    """
```

```
memory = None
```

```
history = load_history() # Завантажуємо історію обчислень
```

Запит точності відображення з обробкою помилок

```
while True:
```

```
    try:
```

```
        precision = int(input("Введіть кількість десяткових розрядів для результату: "))
```

```
        break
```

```
    except ValueError:
```

```
        print(locale["error_invalid_input"]) # Обробка некоректного введення
```

```
while True:
```

```
    try:
```

```

# Використання пам'яті або запит першого числа
if memory and input(locale["use_memory"]).strip().lower() == 'так':
    num1 = memory
    print(f'Перше число: {num1} (з пам'яті)')
else:
    while True:
        try:
            num1 = float(input(locale["input_first"]))
            break
        except ValueError:
            print(locale["error_invalid_input"]) # Обробка некоректного введення

# Оператор і введення другого числа
operator = get_operator()

if operator != '√':
    while True:
        try:
            num2 = float(input(locale["input_second"]))
            break
        except ValueError:
            print(locale["error_invalid_input"]) # Обробка некоректного введення
    else:
        num2 = None # Для кореня друге число не потрібне

# Обчислення
result = calculate(num1, num2, operator)
if result is None:
    continue
if isinstance(result, float):
    result = round(result, precision) # Округлюємо результат до заданої точності

print(f'{locale["result"]} {result}')

```

```

memory = result # Зберігаємо результат у пам'яті

# Історія
history.append(f'{num1} {operator} {num2 if num2 is not None else ""} = {result}')
if input(locale["history"]).strip().lower() == 'так':
    print("\n".join(history)) # Вивести історію

# Повторення обчислень
if input(locale["calc_again"]).strip().lower() != 'так':
    save_history(history) # Зберігаємо історію в файл
    print(locale["save_history"])
    break

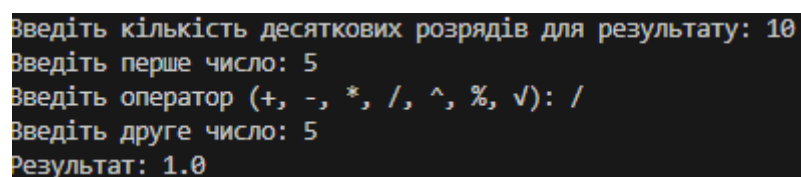
except ValueError:
    print(locale["error_invalid_input"]) # Обробка некоректного введення
except Exception as e:
    print(f'Несподівана помилка: {e}') # Обробка інших помилок

# Функція для виклику з runner.py
def run_lab1():
    """
    Запускає основну програму калькулятора.
    """
    main()

if __name__ == "__main__":
    run_lab1() # Для запуску лабораторної роботи 1

```

Результат роботи програми:



```

Введіть кількість десяткових розрядів для результату: 10
Введіть перше число: 5
Введіть оператор (+, -, *, /, ^, %, √): /
Введіть друге число: 5
Результат: 1.0

```

Рис. 1 – Результат роботи програми


```
Переглянути історію? (так/ні): так
5.0 - 5.0 = 0.0
125.0 - 5.0 = 120.0
5.0 - 5.0 = 0.0
10.0 * 10.0 = 100.0
10.0 / 5.0 = 2.0
5.0 + 5.0 = 10.0
2.0 + 3.0 = 5.0
9.0 + 1.0 = 10.0
1.0 + 2.0 = 3.0
2.0 + 3.0 = 5.0
5.0 / 5.0 = 1.0
Виконати ще одне обчислення? (так/ні): ☐
```

Рис. 2 – Перегляд історії калькулятора

Висновок: У ході виконання лабораторної роботи я створив простий консольний калькулятор на Python, який може виконувати арифметичні операції, обробляти помилки та надавати користувачу зручний інтерфейс.