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»

Виконала студентка

групи РІ-32

Копейка Х.А.

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.

Результати 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.

```
def get_input():
    try:
        num1 = float(input("Введіть перше число: "))
        operator = input("Введіть оператор (+, -, *, /, ^, %, √): ")
        num2 = None
        if operator != '√':
            num2 = float(input("Введіть друге число: "))
        return num1, operator, num2
    except ValueError:
        print("Неправильний ввід. Спробуйте знову.")
        return get_input()
```

Завдання 3: Обчислення

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

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 calculate(num1, operator, num2=None):
    match operator:
        case '+':
            return num1 + num2
        case '-':
            return num1 - num2
        case '*':
            return num1 * num2
        case '/':
            match num2:
                case 0:
```

```
                    raise ZeroDivisionError("Помилка: ділення на нуль!")
                case _:
                    return num1 / num2
        case '^':
            return num1 ** num2
        case '%':
            return num1 % num2
        case '√':
            match num1:
                case n if n < 0:
                    raise ValueError("Помилка: не можна знайти квадратний корінь з від'ємного числа!")
                case _:
                    return math.sqrt(num1)
        case _:
            raise ValueError("Неправильний оператор")

def is_valid_operator(operator):
    return operator in OPERATORS
```

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.

```
def ask_to_continue():
    return input("Бажаєте виконати ще одне обчислення? (так/ні): ").lower() == 'так'
```

Завдання 6: Десяткові числа

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

 # Використовуємо float для введення користувача, що дозволяє обробляти десяткові числа

```
num1 = float(input("Введіть перше число: "))
num2 = float(input("Введіть друге число: "))
```

Завдання 7: Додаткові операції
Додайте підтримку додаткових операцій, таких як піднесення до степеня (^), квадратний корінь (√) і залишок від ділення (%).

 # Піднесення до степеня вже реалізовано в calculate(), аналогічно квадратний корінь та залишок

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 store_in_memory(result):
    global memory
    memory = result
    print(f"Результат {result} збережений у пам'яті.")

def recall_memory():
    if memory is not None:
        print(f"Збережене значення: {memory}")
        return memory
    else:
        print("Пам'ять порожня.")
        return None
```

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_file = os.path.join(os.path.dirname(__file__), 'history.txt')
    with open(history_file, 'a') as file:
        file.write(f"{expression} = {result}\n")

def show_history():
    history_file = os.path.join(os.path.dirname(__file__), 'history.txt')
    with open(history_file, 'r') as file:
        history = file.read()
    print(history)
```

Завдання 10: Налаштування користувача

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

```
def calculator():
    print(f"Результати будуть відображатися з {decimal_places} десятковими
знаками.")
    while True:
        num1, operator, num2 = get_input()
```

Посилання на GitHub: https://github.com/Khr178/Python_lab1

Висновок: На цій лабораторній роботі було створено 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.