МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра інформаційних систем та мереж

Звіт

до лабораторної роботи №1

Введення в Python

Виконала:

ст. гр. PI-32

Довгошия А.А

Балів	Дата

Прийняв:

асис. каф. 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 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.

Main.py

```
from calculator import calc, format_result
from history import add_to_history, show_history
from settings import change_settings

extended = False
memory = 0
decimal_places = 2
```

```
def display_layout(extended):
    layout = (
        "MC  C   X   %   /   ^n\n"
        "MR  7   8   9   *   √\n"
        "MS  4   5   6   -   %\n"
        "M+  1   2   3   +\n"
        "ext 0   ."
    ) if extended else (
        "C   X   %   /\n"
        "7   8   9   *\n"
        "4   5   6   -\n"
        "1   2   3   +\n"
        "ext 0   ."
    )
    print("\n" + layout)


def get_number(prompt):
    while True:
        try:
            value = input(prompt)
            if value.lower() == 'ext':
                global extended
                extended = not extended
                display_layout(extended)
                continue
            return float(value)
        except ValueError:
            print('! not a number')


def main():
    global memory, extended, decimal_places
    while True:
        display_layout(extended)
        action = input('\n# enter "settings" to configure or "calculate" to perform a
calculation: ').strip().lower()

        if action == 'settings':
            decimal_places, memory = change_settings(decimal_places, memory)
            continue

        elif action == 'calculate':
```

```
            x = get_number('x: ')
            y = get_number('y: ')
            operation_prompt = '# operation (+, -, /, *, %, ^, √): ' if extended else '#
operation (+, -, /, *): '
            operation = input(operation_prompt)

            memory, result = calc(x, y, operation, memory, extended,
decimal_places)
            result = format_result(result, decimal_places)
            add_to_history(x, y, operation, result)
            print(f'# result: {result}')

            choice = input('\n# continue? (y/n), or type "history" to view history:
').strip().lower()
            if choice == 'n':
                print('# exiting...')
                break
            elif choice == 'history':
                show_history()

        else:
            print("! invalid option")


if __name__ == '__main__':
    main()
```

calculator.py

```
def calc(x, y, action, memory, extended, decimal_places):
    def memory_clear():
        return 0, '~ memory cleared'

    def memory_store(value):
        return value, f'~ memory saved: {format_result(value, decimal_places)}'

    def memory_add(value):
        updated_memory = memory + value
        return updated_memory, f'~ memory updated:
{format_result(updated_memory, decimal_places)}'

    actions = {
```

```
        'MC': memory_clear,
        'MR': lambda: (memory, format_result(memory, decimal_places)),
        'MS': lambda: memory_store(x),
        'M+': lambda: memory_add(x),
        '+': lambda: (memory, format_result(x + y, decimal_places)),
        '-': lambda: (memory, format_result(x - y, decimal_places)),
        '*': lambda: (memory, format_result(x * y, decimal_places)),
        '/': lambda: (memory, 'error: divide by zero' if y == 0 else format_result(x /
y, decimal_places)),
        '%': lambda: (memory, 'error: divide by zero' if y == 0 else format_result(x
% y, decimal_places)),
        '^': lambda: (memory, format_result(x ** y, decimal_places)),
        '√': lambda: (memory, 'error: root of zero' if y == 0 else format_result(x **
(1 / y), decimal_places))
    }

    if not extended and action in {'^', '√', '%', 'MC', 'MR', 'MS', 'M+'}:
        return memory, '! error: operation is not allowed in non-extended mode'

    return actions.get(action, lambda: (memory, 'unknown action'))()

def format_result(result, decimal_places):
    return f'{result:.{decimal_places}f}' if isinstance(result, (int, float)) else
result
```

history.py

```
history = []

def add_to_history(x, y, action, result):
    history.append(f'{x} {action} {y} = {result}')

def show_history():
    if history:
        print("\nHISTORY:")
        for entry in history:
            print(f"\t {entry}")
        print("\n")
    else:
        print('# no history available.')
```

settings.py

```
def change_settings(decimal_places, memory):
    while True:
        print("\n\tSETTINGS")
        print("1. change decimal")
        print("2. reset memory")
        print("3. return")
        choice = input("\tSelect an option: ").strip()

        if choice == '1':
            try:
                decimal_places = int(input("\n# enter the number of decimal:
").strip())
                if decimal_places < 0:
                    print("! decimal must be greater than 0")
                    decimal_places = 2
                print(f"~ decimal set to {decimal_places}\n")
            except ValueError:
                print("! invalid input")

        elif choice == '2':
            memory = 0
            print("~ memory has been reset.")
        elif choice == '3':
            break
        else:
            print("! invalid input")

    return decimal_places, memory
```

main.py

```
from calculator import calc, format_result
from history import add_to_history, show_history
from settings import change_settings

extended = False
memory = 0
decimal_places = 2

def display_layout(extended):
```

```
    layout = (
        "MC  C   X   %   /   ^n\n"
        "MR  7   8   9   *   √\n"
        "MS  4   5   6   -   %\n"
        "M+  1   2   3   +\n"
        "ext 0   ."
    ) if extended else (
        "C   X   %   /\n"
        "7   8   9   *\n"
        "4   5   6   -\n"
        "1   2   3   +\n"
        "ext 0   ."
    )
    print("\n" + layout)


def get_number(prompt):
    while True:
        try:
            value = input(prompt)
            if value.lower() == 'ext':
                global extended
                extended = not extended
                display_layout(extended)
                continue
            return float(value)
        except ValueError:
            print('! not a number')


def main():
    global memory, extended, decimal_places
    while True:
        display_layout(extended)
        action = input('\n# enter "settings" to configure or "calculate" to perform a
calculation: ').strip().lower()

        if action == 'settings':
            decimal_places, memory = change_settings(decimal_places, memory)
            continue

        elif action == 'calculate':
            x = get_number('x: ')
```

```
            y = get_number('y: ')
            operation_prompt = '# operation (+, -, /, *, %, ^, √): ' if extended else '# operation (+, -, /, *): '
            operation = input(operation_prompt)

            memory, result = calc(x, y, operation, memory, extended, decimal_places)
            result = format_result(result, decimal_places)
            add_to_history(x, y, operation, result)
            print(f'# result: {result}')

            choice = input('\n# continue? (y/n), or type "history" to view history: ').strip().lower()
            if choice == 'n':
                print('# exiting...')
                break
            elif choice == 'history':
                show_history()

        else:
            print("! invalid option")


if __name__ == '__main__':
    main()
```

Рис.1. Результат програми.

```
C   X   %   /
7   8   9   *
4   5   6   -
1   2   3   +
ext 0   .

# enter "settings" to configure or "calculate" to perform a calculation: calculate
x: 2
y: 3
# operation (+, -, /, *): +
# result: 5.00000

# continue? (y/n), or type "history" to view history: y

C   X   %   /
7   8   9   *
4   5   6   -
1   2   3   +
ext 0   .

# enter "settings" to configure or "calculate" to perform a calculation: ext
! invalid option
```

Рис.2. Результат програми.

```
C   X   %   /
7   8   9   *
4   5   6   -
1   2   3   +
ext 0   .

# enter "settings" to configure or "calculate" to perform a calculation: calculate
x: ext

MC  C   X   %   /   ^n
MR  7   8   9   *   √
MS  4   5   6   -   %
M+  1   2   3   +
ext 0   .
x: 7
y: 2
# operation (+, -, /, *, %, ^, √): ^
# result: 49.00000

# continue? (y/n), or type "history" to view history: history

HISTORY:
     2.0 + 3.0 = 5.00000
     7.0 ^ 2.0 = 49.00000
```

Рис.3. Результат програми

Висновок: на цій лабораторній роботі я вивчила створення консольної програми-калькулятора за допомогою основних синтаксичних конструкцій Python, з іншим завданням на заміну тестуванню та валідації