

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. ІТ-32
Шоха А.А.
Прийняв:
Щербак С.С.

Львів-2023

Мета роботи: 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, +, -, *, /).

Фрагмент коду програми, що виконує дане завдання:

```
num1 = input("Введіть перше число: ")
num2 = input("Введіть друге число: ")
operator = input("Введіть оператор (+, -, *, /, ^, √, %): ")
```

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.

Фрагмент коду програми, що виконує дане завдання:

```
operators = ['+', '-', '*', '/', '^', '√', '%']
while operator not in operators:
    print("Помилка: невідомий оператор")
    operator = input("Введіть один з наступних операторів: +, -, *, /, ^, √, %")
```

Завдання 3: Обчислення

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

Фрагмент коду програми, що виконує дане завдання:

```
def add(a,b):
    return a+b

def sub(a,b):
    return a-b

def mul(a,b):
    return a*b

def div(a,b):
    return a/b
```

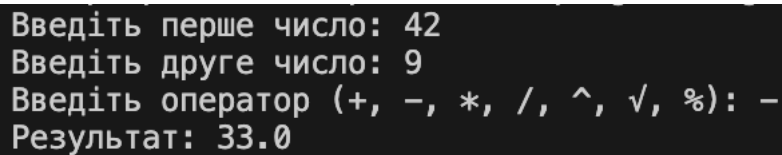
```
def exp(a,b):
    return a**b

def root(a,b):
    return pow(a,1./b)

def mod(a,b):
    return a%b

operations = {'+':add, '-':sub, '*':mul, '/':div, '^':exp, '√':root, '%':mod}
result = round(operations[operator](float(num1), float(num2)), rounding_value)
print(f'Результат: {result}\n')
```

Результат виконання операцій наведено на рис 1:



```
Введіть перше число: 42
Введіть друге число: 9
Введіть оператор (+, -, *, /, ^, √, %): /
Результат: 33.0
```

Рис. 1. Результат обчислень

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.

Функція, що виконує дане завдання:

```
def ask_if_repeat():
    global repeat
    query = input("Повторити? (yes/no/menu): ")

    if query.lower() == 'menu':
        call_menu()
    elif query.lower() == 'yes':
        return
    else:
        repeat = False
        save_history()
        display_history()
```

Завдання 5: Обробка помилок

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

Фрагмент коду програми, що виконує дане завдання:

```

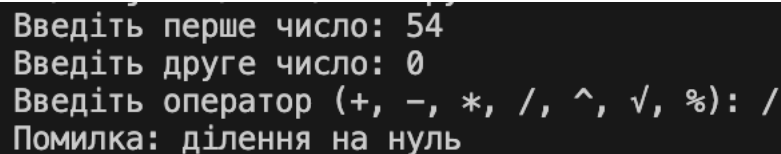
try:
#main code
except ZeroDivisionError:
    print("Помилка: ділення на нуль")
    result = "undefined"
    history.append((num1, num2, operator, result))
    ask_if_repeat()

except ValueError as e:
    print(f"Помилка: введено не число або його введено неправильно: {str(e)}")
    ask_if_repeat()

except Exception as e:
    print(f"Помилка: {str(e)}")

```

Прклад обробки помилки зображений на рис. 2:



```

Введіть перше число: 54
Введіть друге число: 0
Введіть оператор (+, -, *, /, ^, √, %): /
Помилка: ділення на нуль

```

Рис. 2. Результат спроби ділення на 0

Завдання 6: Десяткові числа

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

Фрагмент коду програми, що виконує дане завдання:

```
result = round(operations[operator](float(num1), float(num2)), rounding_value)
```

Завдання 7: Додаткові операції

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

Реалізовано в завд. 3

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.

Фрагмент коду програми, що виконує дане завдання:

```
history = []
```

```

history.append((num1, num2, operator, result))

def display_history():
    if history:
        print("\nІсторія обчислень за сесію:")
        for i, expression in enumerate(history, start=1):
            num1, num2, operator, result = expression
            print(f"{i}) {num1} {operator} {num2} = {result}")
    else:
        print("\nІсторія обчислень пуста")

```

Завдання 9: Історія обчислень

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

Фрагмент коду програми, що виконує дане завдання:

```

def save_history():
    with open('history.txt', 'a') as file:
        for i, expression in enumerate(history, start=1):
            num1, num2, operator, result = expression
            file.write(f"{i}) {num1} {operator} {num2} = {result}\n")
            file.write("\n")

with open('history.txt', 'r') as file:
    print(file.read())

```

Завдання 10: Налаштування користувача

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

Функції, що виконують дане завдання:

```

def show_options():
    print("[0] --- Продовжити обчислення")
    print("[1] --- Налаштувати кількість знаків після коми")
    print("[2] --- Показати історію обчислень за дану сесію")
    print("[3] --- Очистити пам'ять за дану сесію")
    print("[4] --- Показати історію обчислень за всі сесії")
    print("[5] --- Вихід")

def call_menu():
    show_options()
    choice = input("Виберіть операцію: ")
    if choice == '0':
        return

```

```

elif choice == '1':
    global rounding_value
    rounding_value = int(input("Введіть кількість знаків після коми: "))
    print(f"Кількість знаків після коми: {rounding_value}")
    call_menu()
elif choice == '2':
    display_history()
    call_menu()
elif choice == '3':
    global history
    history = []
    print("Історія обчислень очищена")
    call_menu()
elif choice == '4':
    print("Історія обчислень за всі сесії:")
    with open('history.txt', 'r') as file:
        print(file.read())
    display_history()
    call_menu()
elif choice == '5':
    save_history()
    global repeat
    repeat = False
    display_history()
else:
    print("Помилка: невідома операція")
    call_menu()

```

Результат виконання програми зображений на рисунку:

```

Повторити? (yes/no/menu): menu
[0] --- Продовжити обчислення
[1] --- Налаштувати кількість знаків після коми
[2] --- Показати історію обчислень за дану сесію
[3] --- Очистити пам'ять за дану сесію
[4] --- Показати історію обчислень за всі сесії
[5] --- Вихід
Виберіть операцію: 4
Історія обчислень за всі сесії:
1) 54 + 32 = 86.0
2) 23 / 0 = undefined
3) 9 √ 3 = 2.080083823051904

1) 3 % 5 = 3.0
2) 4 ^ 6 = 4096.0

```

Рис. 3. Меню програми

Висновок: виконавши ці завдання, я створив простий консольний калькулятор на Python, який може виконувати арифметичні операції, обробляти

помилки та надавати користувачу зручний інтерфейс. Цей проект допоміг мені вивчити основний синтаксис Python і концепції, такі як введення користувача, умовні оператори, цикли та обробка помилок.

Посилання на github-репозиторій з проектом: <https://github.com/DonQuiCode/Specialised-programming-languages>