

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Інститут комп'ютерних наук та інформаційних технологій

Кафедра інформаційних систем та мереж



Лабораторна робота №1
з дисципліни: «Спеціалізовані мови програмування»
на тему: «Введення в Python»

Виконала:
студентка групи ІТ-32
Моляща Ю.А.
Прийняв:
Щербак С.С.

Львів - 2023

Лабораторна робота №1

«Введення в Python»

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

Завдання на лабораторну роботу.

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.

Хід роботи.

Код програми:

```
HISTORY_FILE = 'source/lab1/history.txt'
VALID_OPERATORS = ['+', '-', '*', '/', '^', '√', '%']
# Default values
decimal_places = 2
number_of_calculations = None
# Input Handling Functions
def get_input():
    while True:
        operator = input("Enter an operator (+, -, *, /, ^, √, %): ")
        if check_operator(operator):
            break
    # Special case for square root and power operators
    if operator == '^':
        first_number = get_numeric_input("Enter number: ")
        second_number = get_numeric_input("Enter power: ")
    elif operator == '√':
        first_number = get_numeric_input("Enter number: ")
        second_number = None
    # For all other operators
    else:
        first_number = get_numeric_input("Enter first number: ")
        second_number = get_numeric_input("Enter second number: ")
    return first_number, second_number, operator
def check_operator(operator):
    if operator not in VALID_OPERATORS:
        print("Invalid operator. Please try again.")
        return False
    else:
        return True

# For handling non-numeric input
def get_numeric_input(prompt):
    while True:
        try:
            value = float(input(prompt))
            return value
        except ValueError:
            print("Invalid input. Please enter a numeric value.")

# Calculation Functions
def addition(first_number, second_number):
    return first_number + second_number
def subtraction(first_number, second_number):
    return first_number - second_number
def multiplication(first_number, second_number):
    return first_number * second_number
```

```

def division(first_number, second_number):
    if second_number == 0:
        print("Division by zero is not allowed. Enter values again.")
        first_number, second_number, operator = get_input()
        return division(first_number, second_number)
    else:
        return first_number / second_number
def power(first_number, second_number):
    return first_number ** second_number
def square_root(first_number):
    if first_number < 0:
        print("Square root of negative number is not allowed. Enter value again.")
        first_number, second_number, operator = get_input()
        return square_root(first_number)
    return first_number ** 0.5
def remainder(first_number, second_number):
    if second_number == 0:
        print("Division by zero is not allowed. Enter values again.")
        first_number, second_number, operator = get_input()
        return remainder(first_number, second_number)
    return first_number % second_number
def calculate(first_number, second_number, operator):
    if operator == '+':
        return addition(first_number, second_number)
    elif operator == '-':
        return subtraction(first_number, second_number)
    elif operator == '*':
        return multiplication(first_number, second_number)
    elif operator == '/':
        return division(first_number, second_number)
    elif operator == '^':
        return power(first_number, second_number)
    elif operator == '√':
        return square_root(first_number)
    elif operator == '%':
        return remainder(first_number, second_number)
    else:
        return "Invalid operator. Please try again."
# Format Result Function
def format_result(result):
    return f"{result:.{decimal_places}f}"
# Result Formatting Function
def format_dict(first_number, second_number, operator, formatted_result):
    # Create empty dictionary
    result_dict = {}

    # Format operation for special cases
    if operator == '√':

```

```

        result_dict['operation'] = f'{operator} {first_number}'
    else:
        result_dict['operation'] = f'{first_number} {operator} {second_number}'

    # Format result
    result_dict['result'] = formatted_result

    return result_dict
# Calculate the result and create a dictionary
def calculate_option():
    # Get input
    first_number, second_number, operator = get_input()
    # Calculate result
    result = calculate(first_number, second_number, operator)
    # Format result to decimal places
    formatted_result = format_result(result)
    # Create dictionary
    result_dict = format_dict(first_number, second_number, operator,
formatted_result)
    # Print result
    print("Result:", formatted_result)

    return result_dict
# Settings Functions
def settings_option():
    while True:
        print("\nOptions:")
        print("1. Display calculation history")
        print("2. Clear history")
        print("3. Set memory functions")
        print("4. Set decimal places")
        print("5. Back")

        choice = input("Enter your choice (1/2/3/4/5): ")

        if choice == '1':
            display_history()
        elif choice == '2':
            clear_history()
        elif choice == '3':
            set_memory_functions()
        elif choice == '4':
            set_decimal_places()
        elif choice == '5':
            return
        else:
            print("Invalid choice. Please select a valid option (1/2/3/4/5).")
# History Handling Functions
def save_history(history):

```

```

with open(HISTORY_FILE, "w") as file:
    for entry in history:
        file.write(f"{entry['operation']} = {entry['result']}\n")

def load_history():
    history = []
    try:
        with open(HISTORY_FILE, "r") as file:
            for line in file:
                operation, result = line.strip().split(" = ")
                result_dict = {'operation': operation, 'result': float(result)}
                history.append(result_dict)
    except FileNotFoundError:
        pass
    return history

def display_history():
    history = load_history()
    if len(history) == 0:
        print("History is empty")
    else:
        with open(HISTORY_FILE, "r") as file:
            for line in file:
                print(line.strip())

def clear_history():
    with open(HISTORY_FILE, "w") as file:
        file.write("")
    print("History cleared")

# Decimal places option
def set_decimal_places():
    # Use global variable to change value outside of function
    global decimal_places

    while True:
        try:
            decimal_places = int(input("Enter decimal places: "))
            if decimal_places >= 0:
                print(f"Decimal places set to {decimal_places}.")
                break
            else:
                print("Please enter a non-negative integer.")
        except ValueError:
            print("Invalid input. Please enter a valid integer.")

# Memory functions option
def set_memory_functions():
    print("1. Auto-delete history")
    print("2. Back")
    choice = input("Enter your choice: ")

```

```

    if choice == '1':
        set_auto_delete_history()
    elif choice == '2':
        return
    else:
        print("Invalid choice. Please try again.")

# Auto-delete history option
def set_auto_delete_history():
    # Use global variable to change value outside of function
    global number_of_calculations

    while True:
        print("1. On")
        print("2. Off")
        print("3. Back")
        choice = input("Enter your choice (1/2): ")
        if choice == '1':
            while True:
                try:
                    number_of_calculations = int(input("Enter the number of
calculations to store: "))
                    print("Auto-delete history is on.")
                    break
                except ValueError:
                    print("Invalid input. Please enter a valid number.")
                    break
            elif choice == '2':
                number_of_calculations = None
                print("Auto-delete history is off.")
                break
            elif choice == '3':
                return
            else:
                print("Invalid choice. Please select a valid option (1/2).")

def main():
    # Use global variable to change value outside of function
    global number_of_calculations

    while True:
        print("\nOptions:")
        print("1. Perform calculation")
        print("2. Settings")
        print("3. Quit")
        choice = input("Enter your choice (1/2/3): ")
        if choice == '1':
            history = load_history()

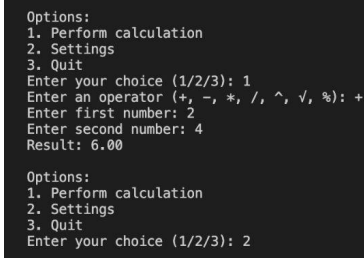
```

```

    result_dict = calculate_option()
    if number_of_calculations is not None:
        if len(history) >= number_of_calculations:
            history.pop(0)
        history.append(result_dict)
        save_history(history)
    elif choice == '2':
        settings_option()
    elif choice == '3':
        print("Exiting the calculator. Goodbye!")
        break
    else:
        print("Invalid choice. Please select a valid option (1/2/3).")
if __name__ == "__main__":
    main()

```

Приклад виконання зображено на рисунках 1-3.



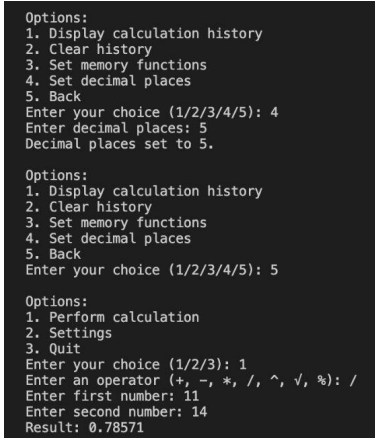
```

Options:
1. Perform calculation
2. Settings
3. Quit
Enter your choice (1/2/3): 1
Enter an operator (+, -, *, /, ^, √, %): +
Enter first number: 2
Enter second number: 4
Result: 6.00

Options:
1. Perform calculation
2. Settings
3. Quit
Enter your choice (1/2/3): 2

```

Рис. 1: Приклад виконання операції додавання



```

Options:
1. Display calculation history
2. Clear history
3. Set memory functions
4. Set decimal places
5. Back
Enter your choice (1/2/3/4/5): 4
Enter decimal places: 5
Decimal places set to 5.

Options:
1. Display calculation history
2. Clear history
3. Set memory functions
4. Set decimal places
5. Back
Enter your choice (1/2/3/4/5): 5

Options:
1. Perform calculation
2. Settings
3. Quit
Enter your choice (1/2/3): 1
Enter an operator (+, -, *, /, ^, √, %): /
Enter first number: 11
Enter second number: 14
Result: 0.78571

```

Рис. 2: Приклад використання функції зміни кількості десяткових розрядів


```
Options:
1. Perform calculation
2. Settings
3. Quit
Enter your choice (1/2/3): 2

Options:
1. Display calculation history
2. Clear history
3. Set memory functions
4. Set decimal places
5. Back
Enter your choice (1/2/3/4/5): 1
2.0 + 4.0 = 6.0
11.0 / 14.0 = 0.78571

Options:
1. Display calculation history
2. Clear history
3. Set memory functions
4. Set decimal places
5. Back
Enter your choice (1/2/3/4/5): 2
History cleared

Options:
1. Display calculation history
2. Clear history
3. Set memory functions
4. Set decimal places
5. Back
Enter your choice (1/2/3/4/5): 5

Options:
1. Perform calculation
2. Settings
3. Quit
Enter your choice (1/2/3): 3
Exiting the calculator. Goodbye!
(base) bulkobubulko@bulkos-MacBook-Air specialized-programming-languages %
```

Рис. 3: Приклад використання функцій для роботи з пам'яттю

Висновок: під час виконання лабораторної роботи було створено консольну програму-калькулятор за допомогою основних синтаксичних конструкцій Python, з іншим завданням на заміну тестуванню та валідації.