Lab 3: Strings, I/O, and Conditional Control Flow

This lab focuses on the practical application of **Strings, Input/Output, and Conditional Logic**—the backbone of any meaningful program.

This lab is designed to be completed in a .py file for the I/O and Control Flow sections, but you should still test the String Tasks directly in the **Python console** for immediate feedback.

Section 1: String Mastery (Indexing, Slicing, and Immutability)

Task 1: Indexing and String Length

Define the string s = "abcde" in your console. Predict the output for each command, then run it to check.

Command	Predicted Output	Actual Output	Explanation
len(s)			
s[0]			
s[4]			
s[-1]			
s[-3]			
s[5]			

Task 2: Basic Slicing

Using the string s = "abcdefgh", predict the result of these common slicing operations.

Command	Predicted Output	Actual Output	Explanation (What are the start/stop/step values?)
s[3:6]			
s[:3]			

Command	Predicted Output	Actual Output	Explanation (What are the start/stop/step values?)
s[4:]			
s[:]			

Task 3: Advanced Slicing (Step Value)

Still using s = "abcdefgh", focus on the step value. Remember: A negative step value reverses the direction!

Command	Predicted Output	Actual Output
s[3:6:2]		
s[::-1]		
s[4:1:-1]		
s[6:3]		

Task 4: Immutability Demonstration

Strings are **immutable**—they cannot be changed after creation. Run the following code in the console and observe the results.

1. **Attempt an illegal change:**

   Python

   ```
   my_string = "Car"
   my_string[0] = 'B'
   ```

   - What kind of error message do you get? Why?

2. **Create a new string (Legal):**

   Python

   ```
   my_string = "Car"
   new_string = 'B' + my_string[1:]
   print(new_string)
   ```

   - Explain the difference: How did Python handle the first attempt versus the second?

Section 2: Input, Output, and Type Casting

Task 5: Printing with Commas vs. Concatenation

Use the variables below and predict the difference in output between using a comma (,) and the concatenation operator (+) in your print statements.

Python

```
a = "The"
b = 3
c = "Musketeers"
```

1. print(a, b, c)
    - What separates the items?
2. print(a + str(b) + c)
    - Why must you use str(b) here?

Task 6: The Input Type Trap

This task shows why you **must** cast the result of input() when expecting a number. Write and run this code in a .py file.

1. **Run 1: String Repetition**

   Python

   ```
   num1 = input("Type a number: ")
   print("5 * num1 results in:", 5 * num1)
   ```

    - If the user enters 3, what is the output? Why?

2. **Run 2: Integer Multiplication (Correct)**

   Python

   ```
   num2 = int(input("Type a number: "))
   print("5 * num2 results in:", 5 * num2)
   ```

    - If the user enters 3, what is the output? Why is this different from Run 1?

Task 7: I/O Coding Challenge (The Verb Program)

Write a short Python program in a file that does the following:

1. Use input() to ask the user to **"Enter a verb (e.g., jump, code, run): "** and save it to a variable called verb.
2. Print the sentence: "I can [verb] better than you!"
3. On the next line, print the verb repeated **5 times**, separated by spaces (you will need to use both multiplication and concatenation).

Section 3: Branching and Decision Making

For these tasks, write complete, well-indented Python programs that take input from the user.

Task 8: Basic if Statement (Positive Number Check)

Write a program that:

1. Asks the user for an integer and converts it to a number variable, n.
2. Uses **only a single if statement** to check if n is greater than 0.
3. If the condition is True, print: "[n] is a positive number."
4. If the condition is False (zero or negative), the program should print nothing else.

Task 9: if-else (Even or Odd)

Write a program that checks if an input number is even or odd.

1. Ask the user for an integer, n.
2. Use the **modulo operator (%)** to check if the remainder when dividing n by 2 is 0.
3. Use an **if-else structure** to print:
    - "[n] is an even number."
    - **OR**
    - "[n] is an odd number."

Task 10: if-elif-else (Trinary Check)

Write a program that determines if a number is positive, negative, or zero.

1. Ask the user for an integer, n.
2. Use an **if-elif-else structure** to print one of three possible messages:
    - "[n] is positive."
    - "[n] is negative."
    - "[n] is zero."

Task 11: Comparing Two Numbers

Write a program that takes two separate integer inputs from the user, x and y, and tells the user which relationship is true.

1. Use an **if-elif-else structure** to print one of three possibilities:
    - "x is greater than y."
    - "y is greater than x."

- "x and y are equal."

Task 12: Coding Challenge (The Simple Guessing Game)

Write a program that plays a simple number guessing game against the user.

1. Set a **secret number** in a variable (e.g., secret = 42).
2. Ask the user for a single integer guess, user_guess.
3. Use an **if-elif-else structure** to compare user_guess to secret and print one of these outcomes:
    - If too low: "Your guess is too low!"
    - If too high: "Your guess is too high!"
    - If correct: "You guessed it! The secret was [secret]."