

ASSIGNMENT-1.5

Batch:24

Ht.No:2303A51963

Task 1: AI-Generated Logic Without Modularization (String Reversal Without Functions)

❖ Scenario

You are developing a basic text-processing utility for a messaging application.

❖ Task Description

Use GitHub Copilot to generate a Python program that:

- Reverses a given string
- Accepts user input
- Implements the logic directly in the main code
- Does not use any user-defined functions

❖ Expected Output

- Correct reversed string
- Screenshots showing Copilot-generated code suggestions
- Sample inputs and outputs

Prompt : write a python program a string reversal without function

```
# Task 1: String reversal without functions
text = input("Enter a string for Task 1: ")
reversed_text = ""
for i in range(len(text) - 1, -1, -1):
    reversed_text += text[i]
print("Task 1 - Reversed string:", reversed_text)
```

```
e/Desktop/AI LAB/lab-1.1.py"
Enter a string for Task 1: hello
Task 1 - Reversed string: olleh
```

Task 2: Efficiency & Logic Optimization (Readability Improvement)

❖ Scenario

The code will be reviewed by other developers.

❖ Task Description

Examine the Copilot-generated code from Task 1 and improve it by:

- Removing unnecessary variables
- Simplifying loop or indexing logic
- Improving readability

- Use Copilot prompts like:

- “Simplify this string reversal code” ▪

“Improve readability and efficiency” Hint:

Prompt Copilot with phrases like

“optimize this code”, “simplify logic”, or “make it more readable”

❖ Expected Output

- Original and optimized code versions
- Explanation of how the improvements reduce time complexity

Prompt: In the above code string reversal without function add optimized string reversal(readability & efficiency)

```
# Task 2: Optimized string reversal (readability & efficiency)
text = input("\nEnter a string for Task 2: ")
print("Task 2 - Reversed string:", text[::-1])
```

```
Enter a string for Task 2: python
Task 2 - Reversed string: nohtyp
```

Task 3: Modular Design Using AI Assistance (String Reversal Using Functions)

❖ Scenario

The string reversal logic is needed in multiple parts of an application.

❖ Task Description

Use GitHub Copilot to generate a function-based Python program that:

- Uses a user-defined function to reverse a string
- Returns the reversed string
- Includes meaningful comments (AI-assisted)

❖ Expected Output

- Correct function-based implementation
- Screenshots documenting Copilot's function generation
- Sample test cases and outputs

Prompt: in the above it should include modular design using functions

```
# Task 3: Modular design using functions
def reverse_string(text):
    """
    Reverse the given string.
    Args:
        text (str): The string to reverse.
    Returns:
        str: The reversed string.
    """
    return text[::-1]
user_input = input("\nEnter a string for Task 3: ")
print("Task 3 - Reversed string:", reverse_string(user_input))
```

Enter a string for Task 3: apple
Task 3 - Reversed string: elppa

Task 4: Comparative Analysis – Procedural vs Modular Approach (With vs Without Functions)

❖ Scenario

You are asked to justify design choices during a code review.

❖ Task Description

Compare the Copilot-generated programs:

- Without functions (Task 1)
- With functions (Task 3) Analyze them based on:
 - Code clarity
 - Reusability
 - Debugging ease
 - Suitability for large-scale applications

❖ Expected Output

Comparison table or short analytical report

Prompt: in the above it should include comparative analysis-procedural approach

```
# Task 4: Comparative Analysis - Procedural vs Modular Approach
# --- Procedural Approach (Without Functions) ---
text = input("Enter a string for procedural approach: ")
reversed_text = ""
for i in range(len(text) - 1, -1, -1):
    reversed_text += text[i]
print("Procedural result:", reversed_text)

# --- Modular Approach (With Functions) ---
def reverse_string(text):
    """
    Reverse the given string.
    Args:
        text (str): The string to reverse.
    Returns:
        str: The reversed string.
    """
    return text[::-1]

user_input = input("\nEnter a string for modular approach: ")
print("Modular result:", reverse_string(user_input))
```

```
Enter a string for procedural approach: watch
Procedural result: hctaw
```

```
Enter a string for modular approach: bucket
Modular result: tekcuB
```

Task 5: AI-Generated Iterative vs Recursive Fibonacci Approaches (Different Algorithmic Approaches to String Reversal)

❖ Scenario

Your mentor wants to evaluate how AI handles alternative logic paths.

❖ Task Description

Prompt GitHub Copilot to generate:

- A loop-based string reversal approach
- A built-in / slicing-based string reversal approach

❖ Expected Output

- Two correct implementations
- Comparison discussing:
 - Execution flow
 - Time complexity
 - Performance for large inputs
 - When each approach is appropriate

Prompt: in the above it should include itractive vs slicing approaches

```
#Task 5: Iterative vs Slicing approaches
# Iterative (loop-based)
text = input("\nEnter a string for Task 5 (Iterative): ")
reversed_text = ""
for char in text:
    reversed_text = char + reversed_text
print("Task 5 (Iterative) - Reversed string:", reversed_text)

# Slicing (built-in)
text = input("\nEnter a string for Task 5 (Slicing): ")
print("Task 5 (Slicing) - Reversed string:", text[::-1])
```

Enter a string for Task 5 (Iterative): bat
Task 5 (Iterative) - Reversed string: tab

Enter a string for Task 5 (Iterative): bat

Enter a string for Task 5 (Iterative): bat

Enter a string for Task 5 (Iterative): bat
Task 5 (Iterative) - Reversed string: tab

Enter a string for Task 5 (Slicing): ball
Task 5 (Slicing) - Reversed string: llab