

# **AI ASSISTANCE CODING**

## **ASSIGNMENT-1.5**

NAME:RAVULA SAHA STRA

ROLL NO:2303A52499

BATCH-50

### **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 that takes a string as input from the user and prints the reversed string without using any user-defined functions."

#### **CODE:**

```
# String Reversal Without User-Defined Functions
```

```
# Get input from the user
```

```
user_string = input("Enter a string to reverse: ")
```

```
# Reverse the string using a for loop

reversed_string = ""

for i in range(len(user_string) - 1, -1, -1):

    reversed_string += user_string[i]
```

### # Print the result

```
print(f"Original string: {user_string}")

print(f"Reversed string: {reversed_string}")
```

### OUTPUT:

The screenshot shows the Visual Studio Code interface. The left sidebar displays a file tree with files like task5.py, task1.py, task2.py, and task3.py. The main editor area contains the Python code for reversing a string. The terminal at the bottom shows the execution of the program, taking the input 'artsahas' and printing the reversed string 'Reversed String: asahsra'. A Chat sidebar on the right provides context about the task, mentioning it's a simple Python program without user-defined functions.

```
task1.py ...
1 # String Reversal Without User-Defined Functions
2
3 # Get input from the user
4 user_string = input("Enter a string to reverse: ")
5
6 # Reverse the string using a for loop
7 reversed_string = ""
8 for i in range(len(user_string) - 1, -1, -1):
9     reversed_string += user_string[i]
10
11 # Print the result
12 print(f"Original string: {user_string}")
13 print(f"Reversed string: {reversed_string}")
```

### EXPLANATION:

This Python program reverses a string entered by the user without relying on built-in functions. It starts by creating an empty string and then uses a for loop with index-based traversal to move through the original string in reverse order. Each character is added to the new string to form the reversed output. In the end, the program displays the reversed string. This method highlights manual logic implementation, making it suitable for beginners.

## 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:**

“Generate a simple Python code to reverse a string using only the main program logic (no functions).”

**CODE:**

```
# Simple String Reversal - Main Program Logic Only
```

```
# Take input from user
```

```
string = input("Enter a string: ")
```

```
# Reverse using a for loop
```

```
reversed_str = ""
```

```
for char in string:
```

```
    reversed_str = char + reversed_str
```

```
# Display the result
```

```
print("Original string:", string)
```

```
print("Reversed string:", reversed_str)
```

**OUTPUT:**

```

task2.py > ...
1 # Simple String Reversal - Main Program Logic Only
2
3 # Take input from user
4 string = input("Enter a string: ")
5
6 # Reverse using a for loop
7 reversed_str = ""
8 for char in string:
9     reversed_str = char + reversed_str
10
11 # Display the result
12 print("Original string:", string)
13 print("Reversed string:", reversed_str)
14

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

Method 5 (Recursion): artsahas  
Method 6 (Step-by-step): artsahas  
PS C:\Users\pc\OneDrive\Desktop\AIAC> & C:/Users/pc/AppData/Local/Programs/Python/Python313/python.exe c:/Users/pc/OneDrive/Desktop/AIAC/task1.py  
Enter a string to reverse: sahasra  
Original string: sahasra  
Reversed string: arsahas  
PS C:\Users\pc\OneDrive\Desktop\AIAC> & C:/Users/pc/AppData/Local/Programs/Python/Python313/python.exe c:/Users/pc/OneDrive/Desktop/AIAC/task2.py  
Enter a string: sahasra  
Original string: sahasra  
Reversed string: arsahas  
PS C:\Users\pc\OneDrive\Desktop\AIAC>

Ln 14, Col 1 Spaces: 4 UTF-8 { } Python 3.13.7 Go Live

## EXPLANATION:

In this task, the objective is to enhance the clarity and efficiency of the string reversal logic. The improved code removes unnecessary indexing and applies a simple loop that adds each character to the front of the result string. This results in shorter, more readable code and avoids the use of extra variables or complicated logic. While the time complexity is still  $O(n)$ , the code becomes much easier to read and maintain compared to the index-based method.

## 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:

**“Create a Python script that accepts user input and reverses the string directly using loops.”**

1.

CODE:

```
# String Reversal Using Loops
```

```
# Accept user input
```

```
user_input = input("Enter a string to reverse: ")
```

```
# Method 1: Using for loop with backward index
```

```
print("\n--- Method 1: For Loop (Backward Index) ---")
```

```
reversed_1 = ""
```

```
for i in range(len(user_input) - 1, -1, -1):
```

```
    reversed_1 += user_input[i]
```

```
print(f"Original: {user_input}")
```

```
print(f"Reversed: {reversed_1}")
```

```
# Method 2: Using while loop
```

```
print("\n--- Method 2: While Loop ---")
```

```
reversed_2 = ""
```

```
i = len(user_input) - 1
```

```
while i >= 0:
```

```
    reversed_2 += user_input[i]
```

```
    i -= 1
```

```
print(f"Original: {user_input}")
```

```
print(f"Reversed: {reversed_2}")
```

```
# Method 3: Using for loop with prepending
```

```
print("\n--- Method 3: For Loop (Prepending) ---")
```

```
reversed_3 = ""
```

```
for char in user_input:
```

```
    reversed_3 = char + reversed_3
```

```
print(f"Original: {user_input}")
```

```
print(f"Reversed: {reversed_3}")
```

OUTPUT:

```

task3.py > ...
6  # Method 1: Using for loop with backward index
7  print(" --- Method 1: For Loop (Backward Index) ---")
8  reversed_1 = ""
9  for i in range(len(user_input) - 1, -1, -1):
10     reversed_1 += user_input[i]
11  print("Original: " + user_input)
12  print("Reversed: " + reversed_1)
13
14  # Method 2: Using while loop
15  print(" --- Method 2: While Loop ---")
16  reversed_2 = ""
17  i = len(user_input) - 1
18  while i >= 0:
19      reversed_2 += user_input[i]
20      i -= 1
21  print("Original: " + user_input)
22  print("Reversed: " + reversed_2)
23
24  # Method 3: Using for loop with prepending
25  print(" --- Method 3: For Loop (Prepending) ---")
26  reversed_3 = ""
27  for char in user_input:
28      reversed_3 = char + reversed_3
29  print("Original: " + user_input)
30  print("Reversed: " + reversed_3)
31
--- Method 1: For Loop (Backward Index) ---
Original: sahasra
Reversed: arsahsa

--- Method 2: While Loop ---
Original: sahasra
Reversed: arsahsa

--- Method 3: For Loop (Prepending) ---
Original: sahasra
Reversed: arsahsa
PS C:\Users\pc\OneDrive\Desktop\AIAC>

```

Ln 31, Col 1 Spaces: 4 UTF-8 { } Python 3.13.7 Go Live

## EXPLANATION:

This program includes a user-defined function called `reverse_string()` that accepts a string as input and returns the reversed string using slicing. Encapsulating the reversal logic within a function makes the code more reusable and simpler to maintain. The main section collects input from the user, invokes the function, and displays the output. Compared to inline procedural code, a modular design enhances readability, simplifies debugging, and supports better scalability.

## 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

**ANALYTICAL REPORT:**

The procedural method used in Task 1 keeps all the logic within the main program, which may be suitable for small programs but becomes hard to handle as applications grow. It reduces readability, limits reusability, and makes debugging more challenging. On the other hand, the modular method in Task 3 places the reversal logic inside a user-defined function, resulting in cleaner, reusable, and easier-to-maintain code. A modular structure is more professional and scalable, and it follows standard best practices in software development.

**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:**

"Write an easy-to-understand Python program to reverse a string entered by the user, without using functions or built-in reverse methods."

**CODE:**

```
# Easy String Reversal Program
```

```
# No functions or built-in reverse methods used
```

```
# Step 1: Get input from the user
```

```
text = input("Enter a string to reverse: ")
```

```
# Step 2: Create an empty string to store the reversed result
```

```
reversed_text = ""
```

```
# Step 3: Loop through the string from the last character to the first
```

```
for i in range(len(text) - 1, -1, -1):
```

```
    # Add each character to the reversed string
```

```
    reversed_text = reversed_text + text[i]
```

```
# Step 4: Display the results
```

```
print("\n--- Results ---")
```

```
print("Original string: ", text)
```

```
print("Reversed string: ", reversed_text)
```

OUTPUT:

## WITHOUT USING IN-BUILT FUNCTIONS

The screenshot shows a VS Code interface with the following details:

- File Explorer:** Shows a folder named "AIAC" containing files: task5.py, task5.1.py, task5.2.py, task1.py, task2.py, and task3.py. "task5.1.py" is currently selected.
- Code Editor:** Displays the Python code for reversing a string. The code follows the steps outlined in the text above, using loops and concatenation to build the reversed string.
- Output Panel:** Shows the execution results:
  - Method 1: For Loop (Backward Index)  
Original: sahasra  
Reversed: arsahas
  - Method 2: While Loop  
Original: sahasra  
Reversed: arsahas
  - Method 3: For Loop (Prepending)  
Original: sahasra  
Reversed: arsahas
- Terminal:** Shows the command "PS C:\Users\pc\OneDrive\Desktop\AIAC\]".
- Chat Panel:** A message from AIAC: "Write an easy-to-understand Python program to reverse a string entered by the user, without using functions or built-in reverse methods."
- Bottom Status Bar:** Shows "Ln 19, Col 1 Spaces: 4 UTF-8 {} Python 3.13.7 ⓘ Go Live 🎨 🌐

## USING SLICING:

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows files in the AIAC folder: task5.py, task1.py, task2.py, task3.py, and task5.2.py.
- Code Editor:** Task5.2.py file open, containing code to reverse a string using a loop.

```
task5.py > task5.2.py > ...
1 # Reverse a string without using slicing or reversed()
2 # Takes input from the user and reverses using a simple loop
3
4 text = input("Enter a string to reverse: ")
5
6 reversed_text = ""
7 for i in range(len(text) - 1, -1, -1):
8     reversed_text += text[i]
9
10 print("\nOriginal string:", text)
11 print("Reversed string:", reversed_text)
```
- Terminal:** Shows three methods for reversing strings:
  - Method 1: For Loop (Backward Index)
  - Method 2: While Loop
  - Method 3: For Loop (Prepending)Output:

```
--- Method 1: For Loop (Backward Index) ---
Original: sahasra
Reversed: arsahas

--- Method 2: While Loop ---
Original: sahasra
Reversed: arsahas

--- Method 3: For Loop (Prepending) ---
Original: sahasra
Reversed: arsahas
```
- Right Panel:** Chat window showing a conversation about reversing strings using slicing, loops, and input. It also shows todo items and a build configuration for task5.2.py.

## EXPLANATION:

The loop-based transaction clearly shows how the string is reversed step by step, which helps beginners understand how each character is handled individually. However, it adds extra overhead because of repeated string concatenation, which can reduce performance for large strings. In contrast, the slicing-based method is short, efficient, and uses Python's built-in features, making it more suitable for practical, real-world use. Although both methods have the same theoretical time complexity of  $O(n)$ , slicing performs much faster in practice and is easier to read, while loop-based logic is more appropriate for learning purposes or situations where slicing is not available.

