

ASSIGNMENT - 01

Syed Althaf

2303A52496

Batch-50

TASK 1:

Question:

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

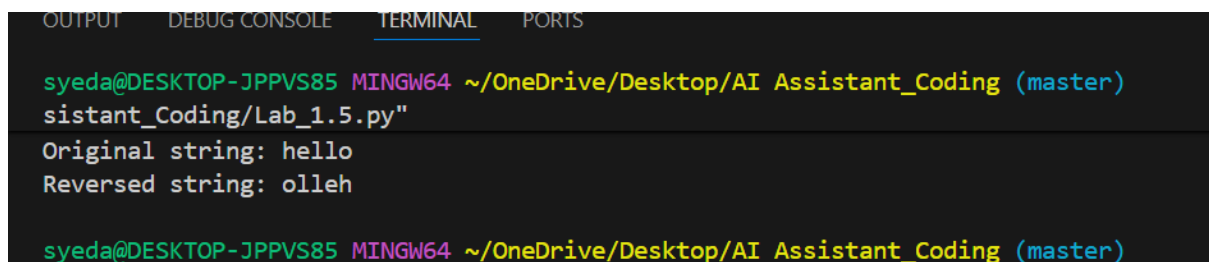
Prompt:

generate a python code for reverse of string without functions

Generated Code:

```
# Reverse a string without using built-in functions
string = input("Enter a string: ")
reversed_string = ""
for i in range(len(string) - 1, -1, -1):
    reversed_string += string[i]
print("Original string:", string)
print("Reversed string:", reversed_string)
```

OUTPUT:



```
OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

syeda@DESKTOP-JPPVS85 MINGW64 ~/OneDrive/Desktop/AI Assistant_Coding (master)
sistant_Coding/Lab_1.5.py"
Original string: hello
Reversed string: olleh

syeda@DESKTOP-JPPVS85 MINGW64 ~/OneDrive/Desktop/AI Assistant_Coding (master)
```

Explanation:

In this program it stores the string given by the user during runtime of code

It builds a new empty string

for loop is used to iterate from last character to first character

Each loop adds the current character to the new string reversing the order

Finally it prints the original string and reversed string

TASK-02:

Question:

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”

Prompt:

Optimize this code by removing unnecessary variables, simplifying the logic, and improving overall readability

Generated Code:

```
string = input("Enter a string: ")
reversed_string = string[::-1]
print(reversed_string)
```

Output:

```
syeda@DESKTOP-JPPVS85 MINGW64 ~/OneDrive/Desktop/AI Assistant_Coding (master)
$ C:/Users/syeda/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/syeda/OneDrive/Desktop/AI Assistant_Coding/Lab_1.5.py"
Original string: hello
Reversed string: olleh

syeda@DESKTOP-JPPVS85 MINGW64 ~/OneDrive/Desktop/AI Assistant_Coding (master)
$
```

Explanation:

First line of the code reads the string from the user during runtime and stores it in a variable

`reversed_string = string[::-1]` makes a new word from string but backwards.

`[::-1]` means take the characters from end to start.

Example: if string is "cat", `reversed_string` becomes "tac".

Task-03:

Create a Python function to reverse a string, return the result, add meaningful comments, and show sample test cases.

Question:

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)

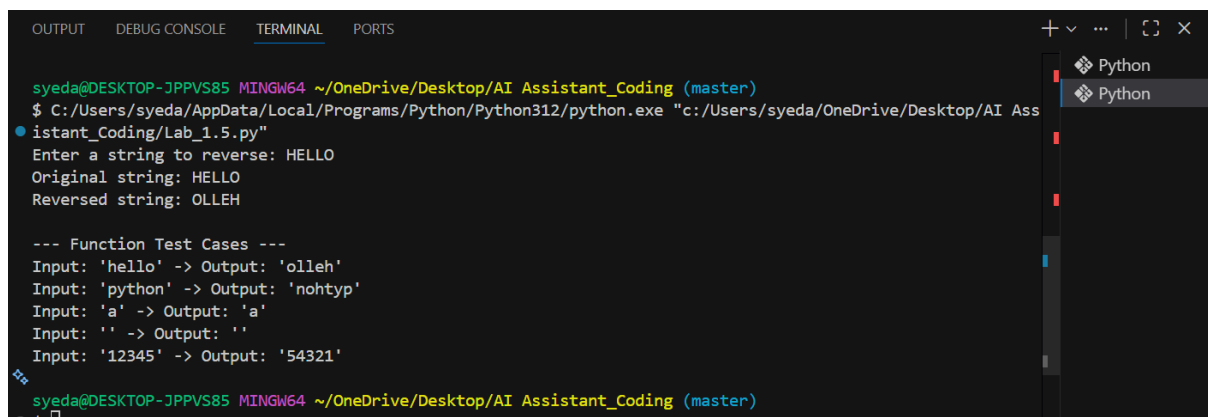
Prompt:

Create a Python function to reverse a string, return the result, add meaningful comments, and show sample test cases.

Generated Code:

```
def reverse_string(s):  
    """  
  
    Reverse a string using slicing.  
    Args:  
    s (str): The string to reverse  
    Returns:  
    str: The reversed string  
    """  
  
    return s[::-1]  
# Test cases  
print("\n--- Function Test Cases ---")  
test_cases = ["hello", "python", "a", "", "12345"]  
for test in test_cases:  
    result = reverse_string(test)  
    print(f"Input: '{test}' -> Output: '{result}'")
```

Output:



```
syeda@DESKTOP-JPPVS85 MINGW64 ~/OneDrive/Desktop/AI Assistant_Coding (master)  
$ C:/Users/syeda/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/syeda/OneDrive/Desktop/AI Assistant_Coding/Lab_1.5.py"  
Enter a string to reverse: HELLO  
Original string: HELLO  
Reversed string: OLLEH  
  
--- Function Test Cases ---  
Input: 'hello' -> Output: 'olleh'  
Input: 'python' -> Output: 'nohtyp'  
Input: 'a' -> Output: 'a'  
Input: '' -> Output: ''  
Input: '12345' -> Output: '54321'
```

Explanation:

The function `reverse_string(s)` takes a word or sentence `s`. `return s[::-1]` makes a new string with characters in reverse order. The test loop runs several example inputs through `reverse_string`. Each print shows the input and its reversed output (e.g., "hello" -> "olleh").

Task-04:

Question:

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

Prompt:

Compare procedural and function-based string reversal programs based on clarity, reusability, debugging, and scalability.

Generated Code:

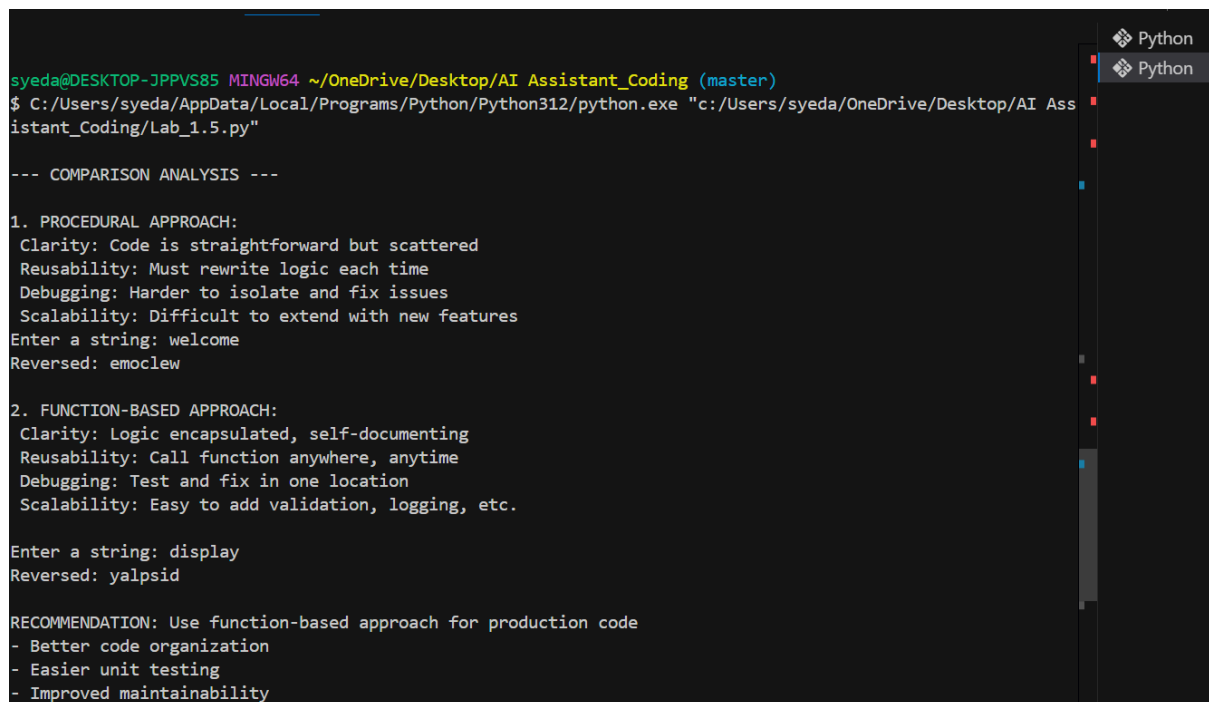
```
# Comparison: Procedural vs Function-based String Reversal
print("\n--- COMPARISON ANALYSIS ---\n")
# PROCEDURAL APPROACH
print("1. PROCEDURAL APPROACH:")
print(" Clarity: Code is straightforward but scattered")
print(" Reusability: Must rewrite logic each time")
print(" Debugging: Harder to isolate and fix issues")
print(" Scalability: Difficult to extend with new features")
string = input("Enter a string: ")
reversed_string = ""
for i in range(len(string) - 1, -1, -1):
    reversed_string += string[i]
```

```

print(f"Reversed: {reversed_string}\n")
# FUNCTION-BASED APPROACH
print("2. FUNCTION-BASED APPROACH:")print(" Clarity: Logic
encapsulated, self-documenting")
print(" Reusability: Call function anywhere, anytime")
print(" Debugging: Test and fix in one location")
print(" Scalability: Easy to add validation, logging, etc.\n")
def reverse_string(s):
    """Reverse a string using slicing."""
    return s[::-1]
string = input("Enter a string: ")
print(f"Reversed: {reverse_string(string)}\n")
# VERDICT
print("RECOMMENDATION: Use function-based approach for
production code")
print("- Better code organization")
print("- Easier unit testing")
print("- Improved maintainability")

```

OUTPUT :



```

syeda@DESKTOP-JPPVS85 MINGW64 ~/OneDrive/Desktop/AI Assistant_Coding (master)
$ C:/Users/syeda/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/syeda/OneDrive/Desktop/AI Assistant_Coding/Lab_1.5.py"

--- COMPARISON ANALYSIS ---

1. PROCEDURAL APPROACH:
Clarity: Code is straightforward but scattered
Reusability: Must rewrite logic each time
Debugging: Harder to isolate and fix issues
Scalability: Difficult to extend with new features
Enter a string: welcome
Reversed: emoclew

2. FUNCTION-BASED APPROACH:
Clarity: Logic encapsulated, self-documenting
Reusability: Call function anywhere, anytime
Debugging: Test and fix in one location
Scalability: Easy to add validation, logging, etc.

Enter a string: display
Reversed: yalpsid

RECOMMENDATION: Use function-based approach for production code
- Better code organization
- Easier unit testing
- Improved maintainability

```

TASK-05:

Question:

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

Prompt:

Write loop-based and slicing-based string reversal programs in Python.

Generated Code:

```
# BONUS: Side-by-side comparison of both methods
print("\n--- METHOD COMPARISON ---\n")
test_string = input("Enter a string for method comparison: ")
# Loop-based approach
reversed_loop = ""
for i in range(len(test_string) - 1, -1, -1):
    reversed_loop += test_string[i]
# Slicing-based approach
reversed_slice = test_string[::-1]
print(f"Original string: '{test_string}'")
print(f"Loop-based result: '{reversed_loop}'")
print(f"Slicing-based result: '{reversed_slice}'")
print(f"Both methods match: {reversed_loop == reversed_slice}")
```


OUTPUT :

```
syeda@DESKTOP-JPPVS85 MINGW64 ~/OneDrive/Desktop/AI Assistant_Coding (master)
$ C:/Users/syeda/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/syeda/OneDrive/Desktop/AI Assistant_Coding/Lab_1.5.py"

--- METHOD COMPARISON ---

Enter a string for method comparison: hello
Original string: 'hello'
Loop-based result: 'olleh'
Slicing-based result: 'olleh'
Both methods match: True

syeda@DESKTOP-JPPVS85 MINGW64 ~/OneDrive/Desktop/AI Assistant_Coding (master)
$
```

Explanation:

The code gets a string from input.

It reverses the string two ways: first using a loop then using slicing. Both results are printed side-by-side to show what each method produces.

The last line checks if both methods give the same answer—they always do, proving both approaches work.