

Lab Assignment -1

Name : M . Raja

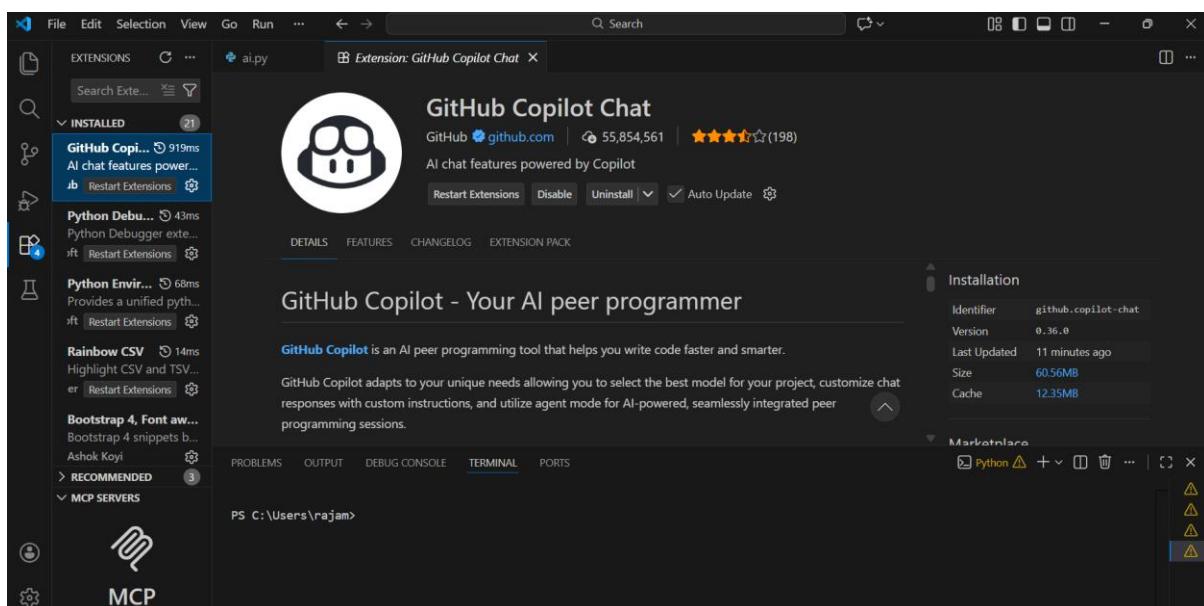
HT.No : 2303A52277

Batch : 36

Environment Setup – *GitHub Copilot and VS Code Integration + Understanding AI-assisted Coding Workflow*

Task 0

Install and configure GitHub Copilot in VS Code. Take screenshots of each step.



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

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

- Extensions View:** Shows the "string_reverse.py" file under "INSTALLED".
- Code Editor:** Displays the original Python code:

```
1 # Reverses a given string
2 user_input = input("Enter a string to reverse: ")
3 reversed_string = user_input[::-1]
4 print(f"Reversed string: {reversed_string}")
```
- Terminal:** Shows the output of running the script:

```
Enter a string to reverse: raja
Reversed string: ajar
PS C:\Users\rajam> []
```
- Bottom Bar:** Includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and PORTS.

Task 2: Efficiency & Logic Optimization (Readability Improvement)

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

- Extensions View:** Shows the "string_reverse.py" file under "INSTALLED".
- Code Editor:** Displays the optimized Python code:

```
1 # Task 1
2 # Reverses a given string
3 user_input = input("Enter a string to reverse: ")
4 reversed_string = user_input[::-1]
5 print(f"Reversed string: {reversed_string}")
6 #Task 2
7 ## Simplify this string reversal code and improve readability
8 def reverse_string(s):
9     return s[::-1]
10 user_input = input("Enter a string to reverse: ")
11 print(f"Reversed string: {reverse_string(user_input)})")
```
- Terminal:** Shows the output of running the script:

```
Enter a string to reverse: raja
Reversed string: ajar
PS C:\Users\rajam> & C:/Users/rajam/AppData/Local/Programs/Python/Python313/python.exe c:/Users/rajam/Downloads/string_reverse.py
Enter a string to reverse: raja
Reversed string: ajar
Enter a string to reverse: raja
Reversed string: ajar
PS C:\Users\rajam> []
```
- Bottom Bar:** Includes tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and PORTS.

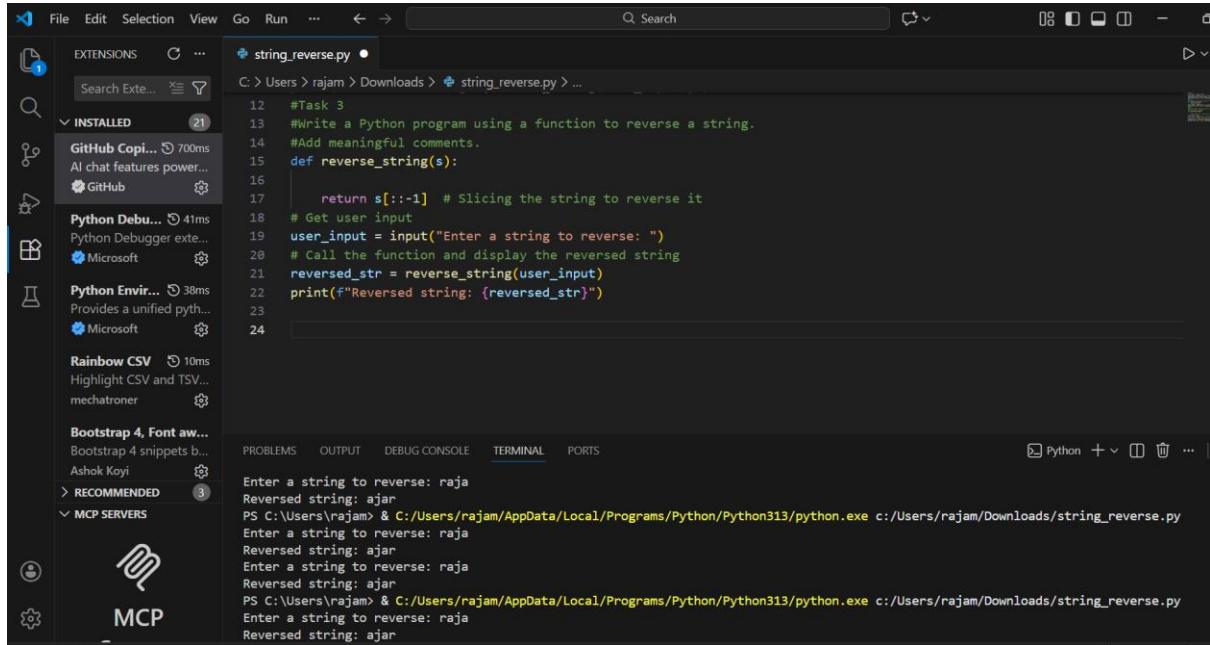
Explanation : In Task 2, the original string reversal code was optimized by removing unnecessary variables and simplifying the logic.

The loop and intermediate variable were eliminated, and Python slicing was used directly.

This improved readability and reduced code length.

Both versions have a time complexity of O(n), but the optimized version is more efficient in practice due to fewer operations.

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



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows the file `string_reverse.py` in the current workspace.
- Extensions sidebar:** Lists several extensions installed, including GitHub Copilot, Python Debugger, Python Environment, and Rainbow CSV.
- Terminal:** Displays the command-line output of running the script. It shows the user entering "raja" and the script outputting "Reversed string: ajar" twice, indicating it's reading from standard input.
- Code Editor:** Contains the Python code for reversing a string using a function.

```
12 #Task 3
13 #Write a Python program using a function to reverse a string.
14 #Add meaningful comments.
15 def reverse_string(s):
16     return s[::-1] # Slicing the string to reverse it
17
18 # Get user input
19 user_input = input("Enter a string to reverse: ")
20 # Call the function and display the reversed string
21 reversed_str = reverse_string(user_input)
22 print(f"Reversed string: {reversed_str}")
23
24
```

Explanation

- The function `reverse_string()` encapsulates the string reversal logic
- It uses Python slicing (`[::-1]`) for efficient reversal
- The function returns the reversed string to the caller
- This modular approach allows reuse of the same logic in multiple parts of an application
- Meaningful comments improve code readability and understanding

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

Comparison Table

| Criteria | Without Functions (Procedural) | With Functions (Modular) |
|------------------------|--|---|
| Code Clarity | Logic is mixed with input/output, making it less clear | Logic is separated into a function, improving clarity |
| Reusability | Code cannot be reused easily | Function can be reused in multiple parts of the application |
| Debugging Ease | Harder to debug due to lack of separation | Easier to debug and test individual functions |
| Maintainability | Changes must be made in multiple places | Changes can be made in one function |
| Scalability | Not suitable for large programs | Suitable for large-scale applications |

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

- A **loop-based** string reversal approach

The screenshot shows a code editor interface with a dark theme. At the top, there's a menu bar with options like 'File', 'Edit', 'Run', etc., followed by a search bar. Below the menu is a tab bar with two tabs: 'string_reverse.py' (which is active) and 'sat.trainin.py'. The main area displays the following Python code:

```
C: > Users > rajam > Downloads > string_reverse.py > ...
24 #Task 5
25 #Write a Python program to reverse a string using a loop.
26 def reverse_string_loop(s):
27     reversed_str = ""
28     for char in s:
29         reversed_str = char + reversed_str # Prepend each character
30     return reversed_str
31 # Get user input
32 user_input = input("Enter a string to reverse: ")
33 # Call the function and display the reversed string
34 reversed_str = reverse_string_loop(user_input)
35 print(f"Reversed string: {reversed_str}")
36
```

Below the code, there's a terminal window showing the execution of the script and its output. The terminal tab is labeled 'TERMINAL'.

```
+ FullyQualifiedErrorId : CommandNotFoundException
PS C:\Users\rajam> & C:/Users/rajam/AppData/Local/Programs/Python/Python313/python.exe c:/Users/rajam/Downloads/string_reverse.py
Enter a string to reverse: raja
Reversed string: ajar
Enter a string to reverse: raja
Reversed string: ajar
Enter a string to reverse: raja
Reversed string: ajar
```

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

This screenshot shows the same code editor interface as the previous one, but the code has been modified to demonstrate a built-in slicing approach for reversing strings.

```
C: > Users > rajam > Downloads > string_reverse.py > ...
25 #Write a Python program to reverse a string using a loop.
26 def reverse_string_loop(s):
27     reversed_str = ""
28     for char in s:
29         reversed_str = char + reversed_str # Prepend each character
30     return reversed_str
31 # Get user input
32 user_input = input("Enter a string to reverse: ")
33 # Call the function and display the reversed string
34 reversed_str = reverse_string_loop(user_input)
35 print(f"Reversed string: {reversed_str}")
36 #Task 5
37 #Write a Python program to reverse a string using slicing.
38 def reverse_string_slicing(s):
39     return s[::-1]
40 user_input = input("Enter a string to reverse: ")
41 print(f"Reversed string: {reverse_string_slicing(user_input)})
```

The terminal window below shows the execution of the script and its output. The terminal tab is labeled 'TERMINAL'.

```
+ FullyQualifiedErrorId : CommandNotFoundException
PS C:\Users\rajam> & C:/Users/rajam/AppData/Local/Programs/Python/Python313/python.exe
Enter a string to reverse: raja
Reversed string: ajar
Enter a string to reverse: raja
Reversed string: ajar
Enter a string to reverse: raja
Reversed string: ajar
Enter a string to reverse: raja
```

2. Time Complexity

- **Loop-Based Approach:**

Time complexity is **O(n)**, where n is the length of the string.

- **Built-in Approach:**

Time complexity is also **O(n)**, as the string must be traversed internally.

Conclusion

Although both approaches have the same time complexity, the built-in slicing method is more efficient and readable, making it the preferred choice for practical applications.