Lab Assignment -1

Name : M . Raja

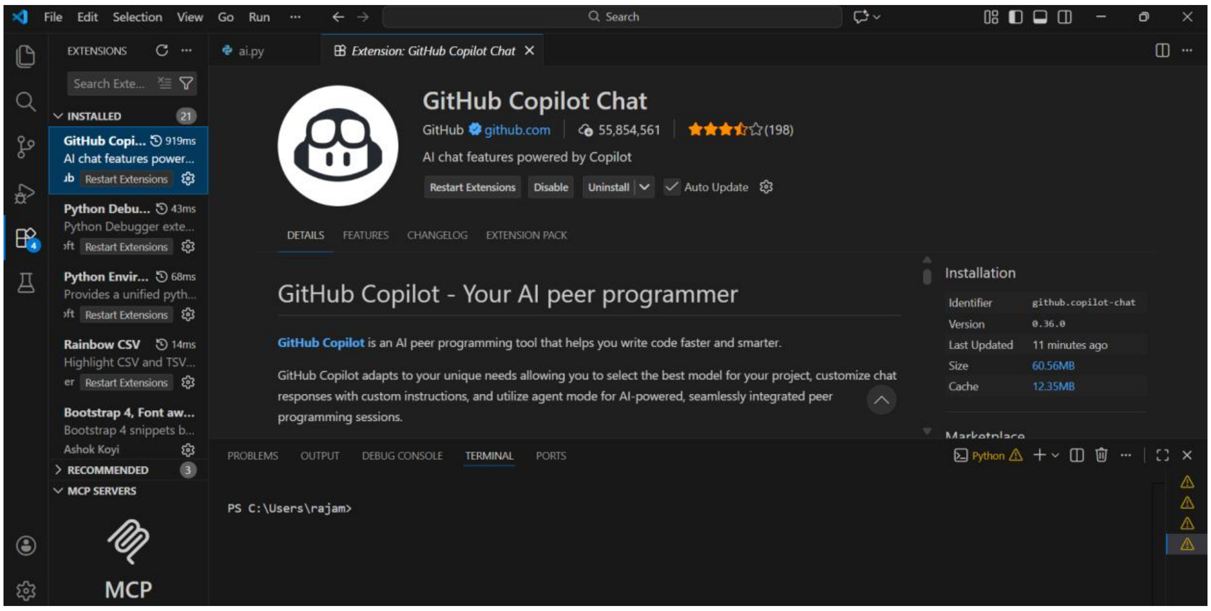
HT.No : 2303A52277

Batch : 36

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

<u>Task 0</u>

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



GitHub Copilot Chat

GitHub github.com | 55,854,561 | ★★★★☆(198)

AI chat features powered by Copilot

GitHub Copilot - Your AI peer programmer

GitHub Copilot is an AI peer programming tool that helps you write code faster and smarter.

GitHub Copilot adapts to your unique needs allowing you to select the best model for your project, customize chat responses with custom instructions, and utilize agent mode for AI-powered, seamlessly integrated peer programming sessions.

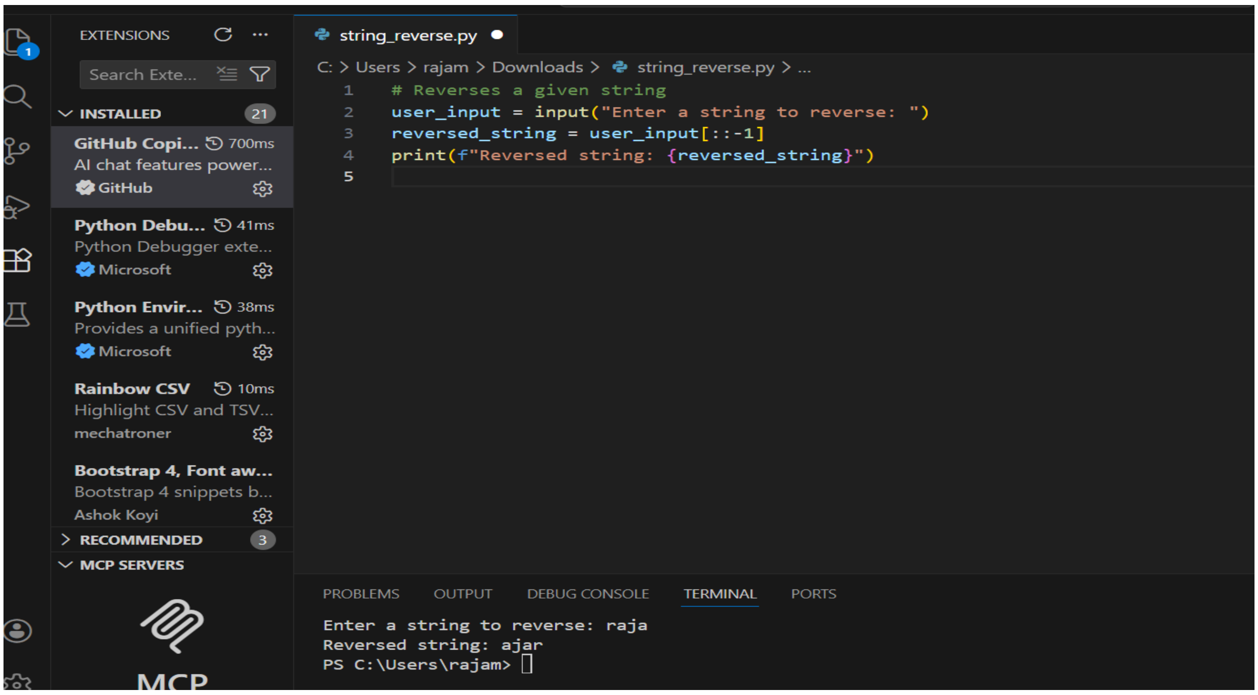
Installation

Identifier	github.copilot-chat
Version	0.36.0
Last Updated	11 minutes ago
Size	60.56MB
Cache	12.35MB

Marketplace

```
PS C:\Users\rajam>
```

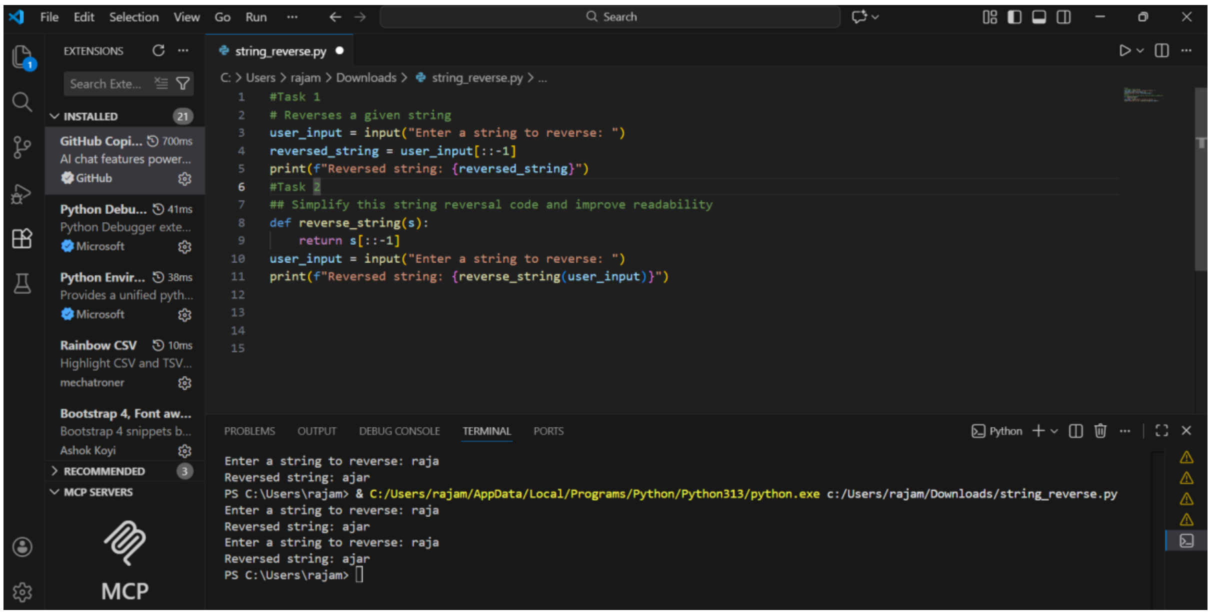
<u>Task 1:</u> AI-Generated Logic Without Modularization (String Reversal Without Functions)



```python
# Reverses a given string
user_input = input("Enter a string to reverse: ")
reversed_string = user_input[::-1]
print(f"Reversed string: {reversed_string}")
```

```
Enter a string to reverse: raja
Reversed string: ajar
PS C:\Users\rajam>
```

Task 2: Efficiency & Logic Optimization (Readability Improvement)

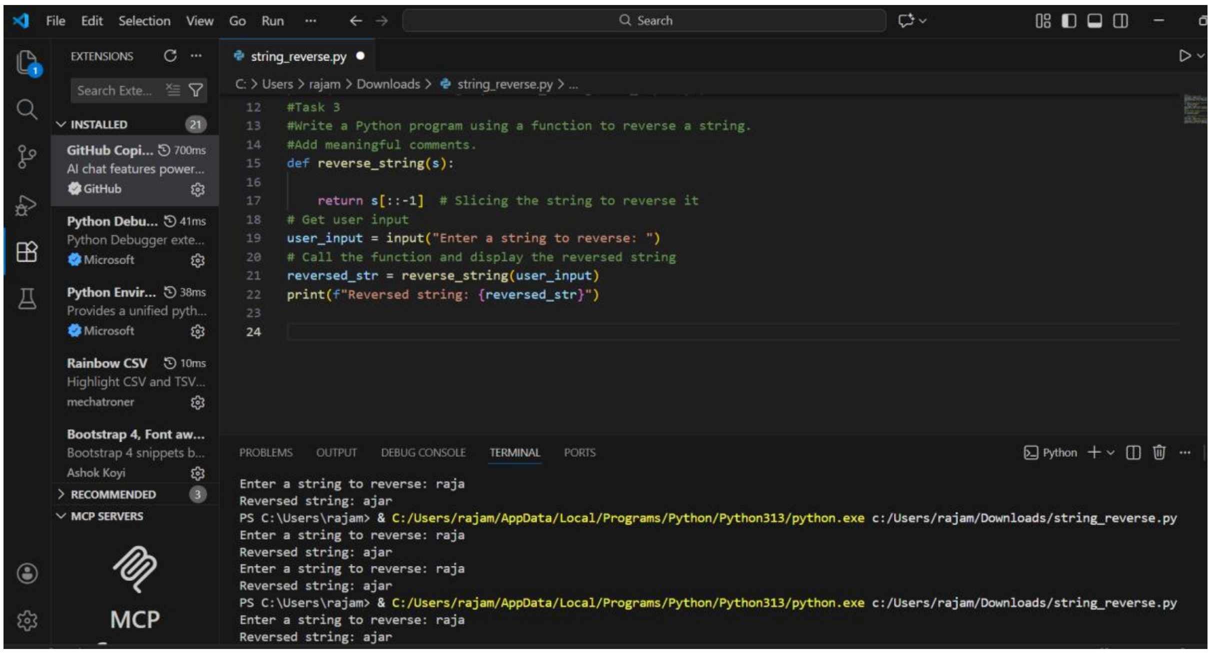


```python
#Task 1
# Reverses a given string
user_input = input("Enter a string to reverse: ")
reversed_string = user_input[::-1]
print(f"Reversed string: {reversed_string}")
#Task 2
## Simplify this string reversal code and improve readability
def reverse_string(s):
    return s[::-1]
user_input = input("Enter a string to reverse: ")
print(f"Reversed string: {reverse_string(user_input)}")
```

```
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>
```

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.

<u>Task 3:</u> Modular Design Using AI Assistance (String Reversal Using Functions)



```python
#Task 3
#Write a Python program using a function to reverse a string.
#Add meaningful comments.
def reverse_string(s):

    return s[::-1]  # Slicing the string to reverse it
# Get user input
user_input = input("Enter a string to reverse: ")
# Call the function and display the reversed string
reversed_str = reverse_string(user_input)
print(f"Reversed string: {reversed_str}")

```

```
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> & 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
```

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

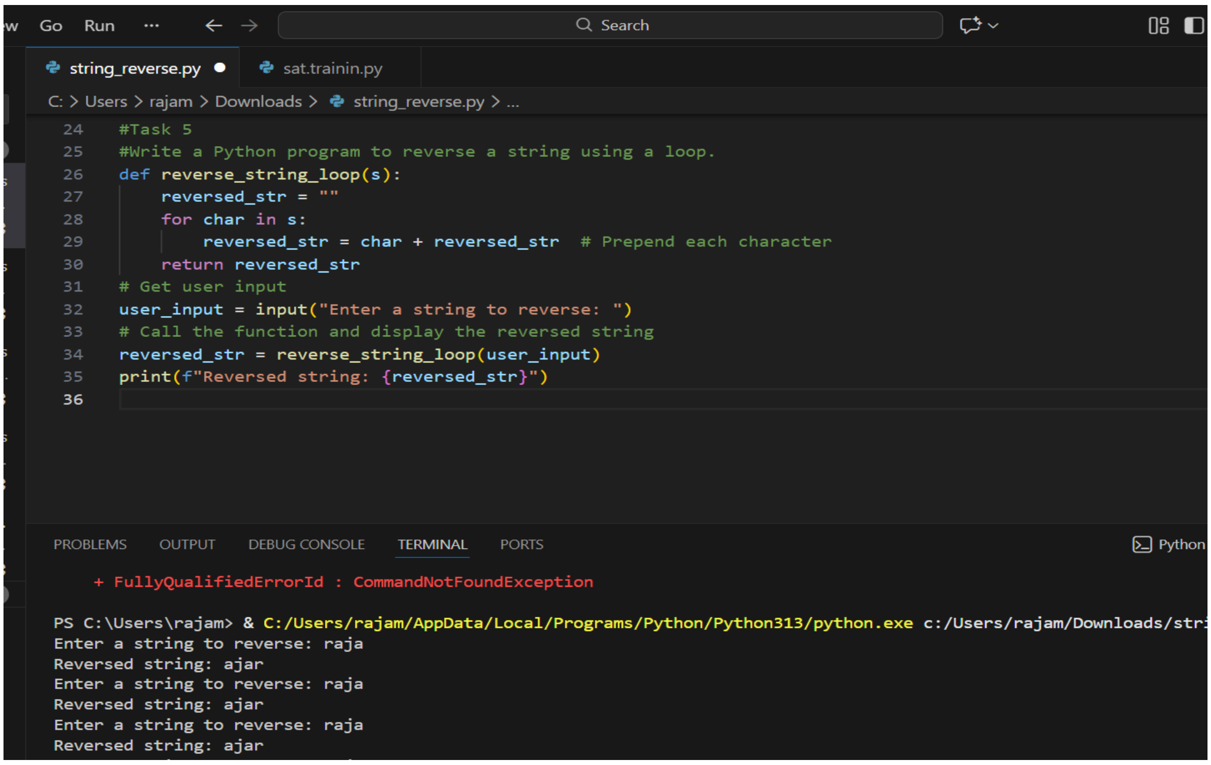
<u>Task 4:</u> Comparative Analysis – Procedural vs Modular Approach (With vs Without Functions)

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

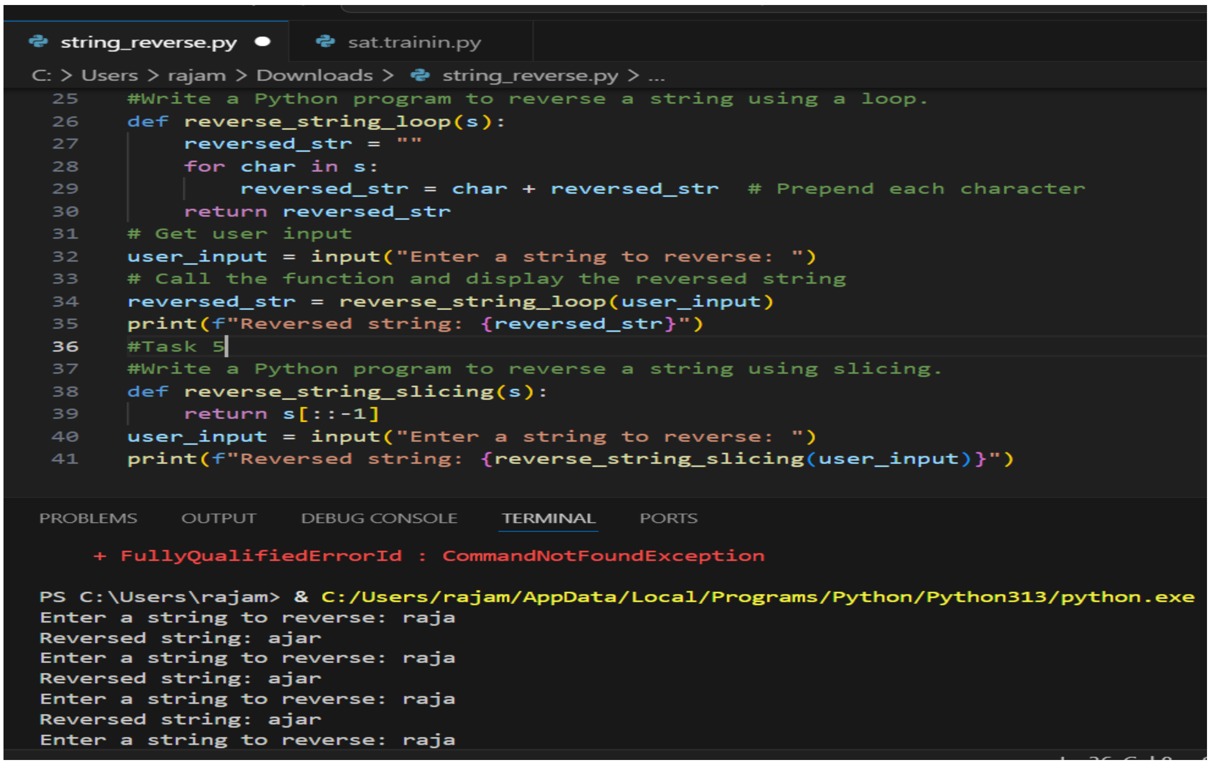


```python
#Task 5
#Write a Python program to reverse a string using a loop.
def reverse_string_loop(s):
    reversed_str = ""
    for char in s:
        reversed_str = char + reversed_str  # Prepend each character
    return reversed_str
# Get user input
user_input = input("Enter a string to reverse: ")
# Call the function and display the reversed string
reversed_str = reverse_string_loop(user_input)
print(f"Reversed string: {reversed_str}")
```

```
    + FullyQualifiedErrorId : CommandNotFoundException

PS C:\Users\rajam> & C:/Users/rajam/AppData/Local/Programs/Python/Python313/python.exe c:/Users/rajam/Downloads/str
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



```python
#Write a Python program to reverse a string using a loop.
def reverse_string_loop(s):
    reversed_str = ""
    for char in s:
        reversed_str = char + reversed_str  # Prepend each character
    return reversed_str
# Get user input
user_input = input("Enter a string to reverse: ")
# Call the function and display the reversed string
reversed_str = reverse_string_loop(user_input)
print(f"Reversed string: {reversed_str}")
#Task 5
#Write a Python program to reverse a string using slicing.
def reverse_string_slicing(s):
    return s[::-1]
user_input = input("Enter a string to reverse: ")
print(f"Reversed string: {reverse_string_slicing(user_input)}")
```

```
    + 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.