AI ASSISTED CODING

Name:O.Sai Keerthana

Ht.No: 2303A52397

Batch:36

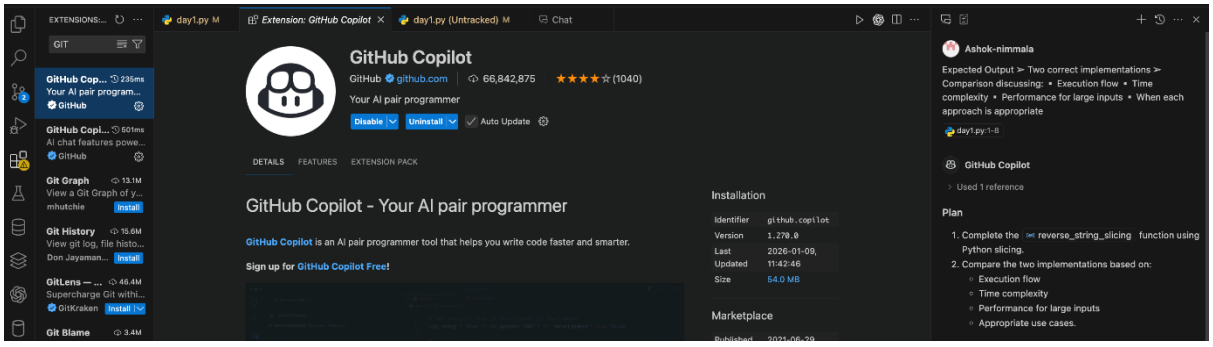
Assignment-1.5

Task-0:

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

Expected Output:

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



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

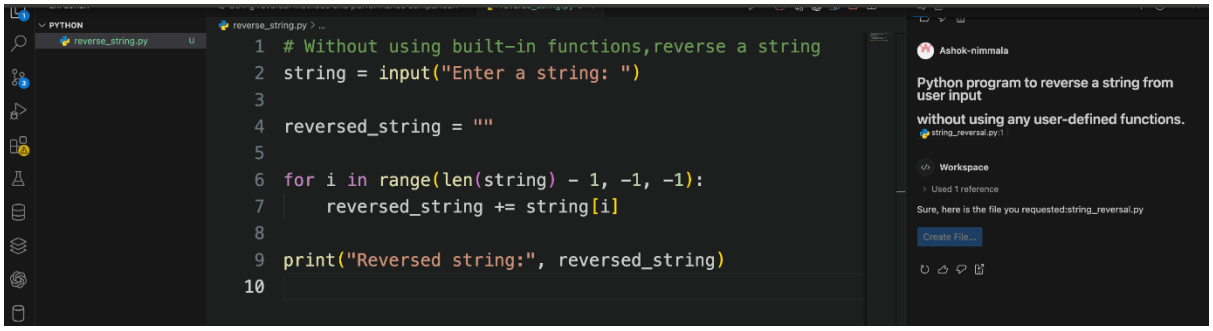
Expected Output

➢ Correct reversed string

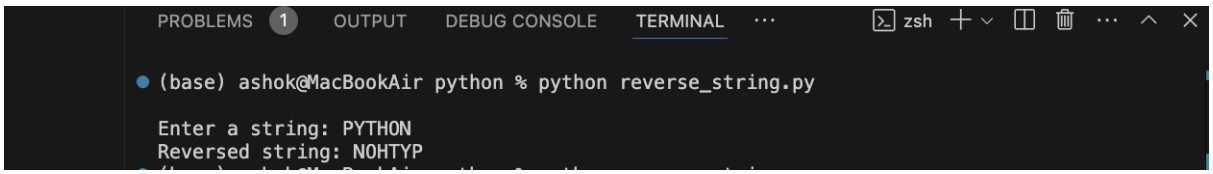
➢ Screenshots showing Copilot-generated code suggestions

➢ Sample inputs and outputs

Code



Output:



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

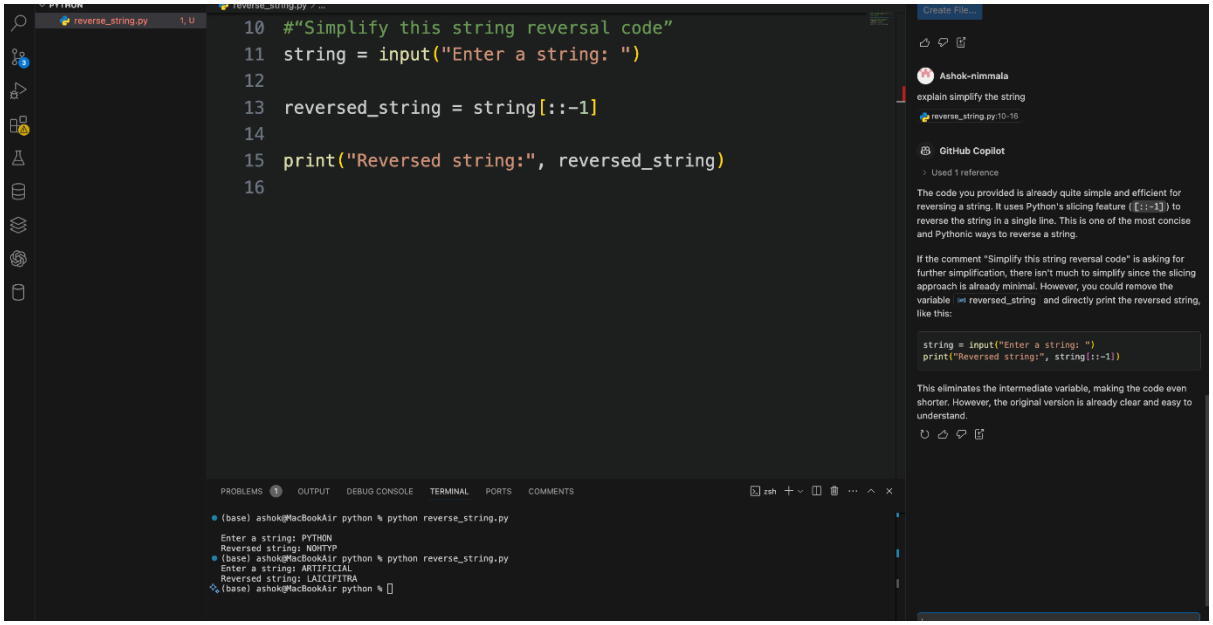
“optimize this code”, “simplify logic”, or “make it more readable”

Expected Output

➢ Original and optimized code versions

➢ Explanation of how the improvements reduce time complexity

Code & Output:



Justification:

By prompting Copilot with

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

the code was reduced from a loop-based O(n²) approach to an efficient O(n) solution using slicing

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)

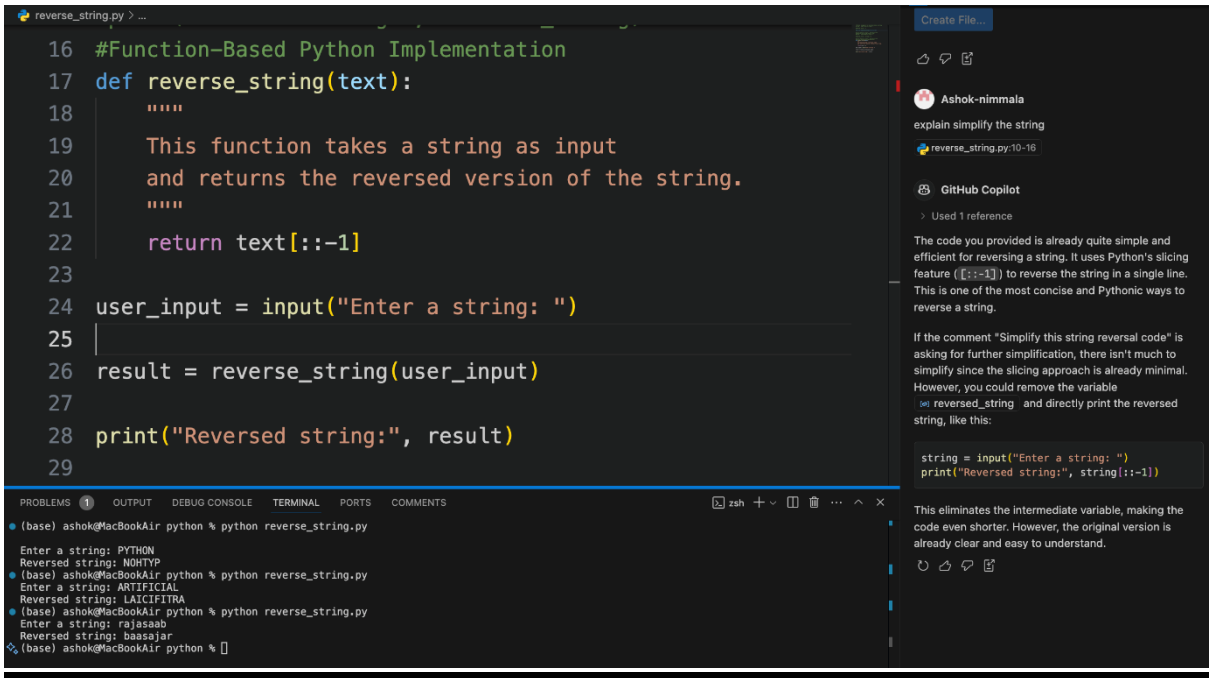
Expected Output

➢ Correct function-based implementation

➢ Screenshots documenting Copilot's function generation

➢ Sample test cases and outputs

Output:



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

Report:

In the procedural approach (Task 1), the string reversal logic is implemented directly in the main code. While this method is simple and suitable for very small programs, it leads to poor reusability and reduced readability when the same logic is required multiple times. Debugging also becomes challenging as the application grows.

In contrast, the modular approach (Task 3) uses a user-defined function to encapsulate the string reversal logic. This significantly improves code clarity, maintainability, and scalability. The function-based design allows the logic to be reused across multiple modules, making it ideal for large-scale and real-world applications. Additionally, modular code aligns with best software engineering practices and simplifies debugging and testing.

Conclusion:

During a code review, the modular (function-based) approach should be preferred over the procedural approach because it:

Enhances readability

Promotes reusability

Simplifies debugging

Supports scalable application design

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:

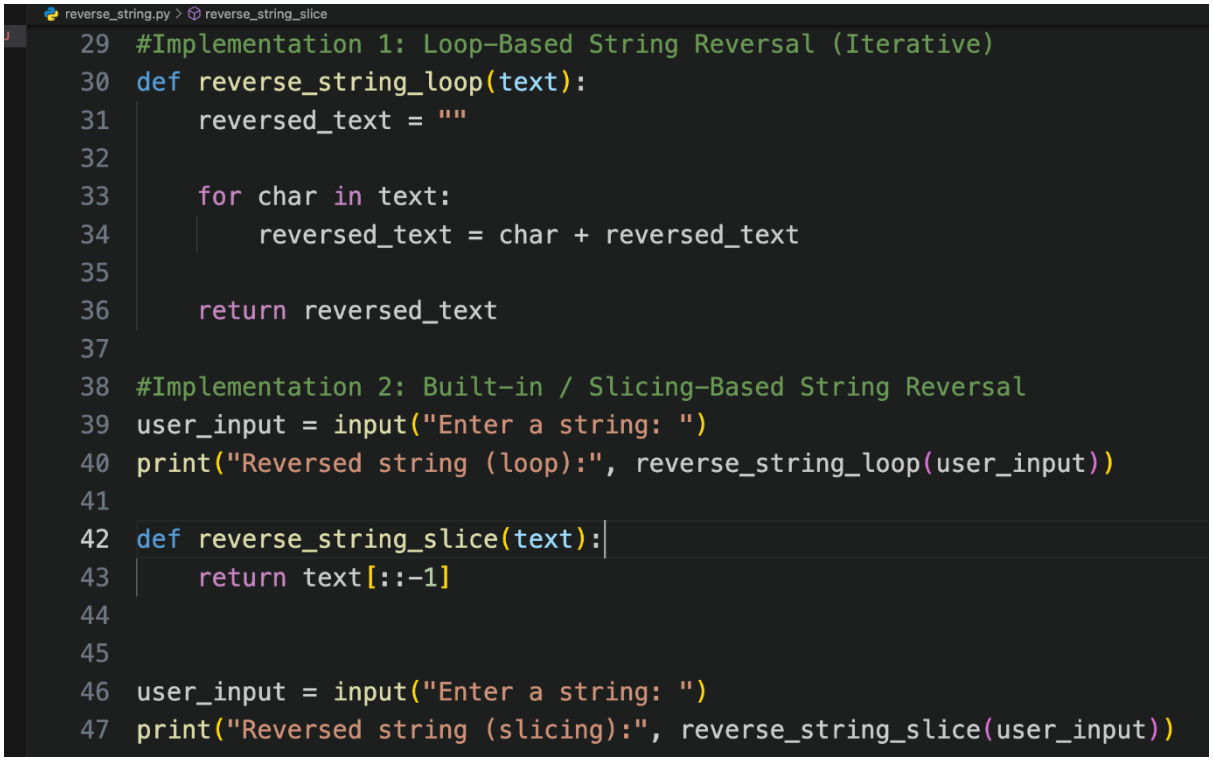
▪ Execution flow

▪ Time complexity

▪ Performance for large inputs

▪ When each approach is appropriate

Code:



Comparison:

Execution Flow

Loop-Based: Reverses the string character by character using a loop.

Slicing-Based: Uses Python's built-in slicing to reverse the string in one step.

Time Complexity

Loop-Based: O(n²) due to repeated string concatenation.

Slicing-Based: O(n) because it processes the string in a single pass.

Performance for Large Inputs

Loop-Based: Slower and inefficient for large strings.

Slicing-Based: Fast and efficient even for large inputs.

When Each Approach is Appropriate

Loop-Based: Suitable for learning basic logic and algorithm understanding.

Slicing-Based: Best for real-world applications and production code.

Conclusion:

Both approaches work correctly, but the slicing-based method is preferred due to better performance, simplicity, and scalability.