AI ASSISTED CODING

2303A52315

BATCH 45

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

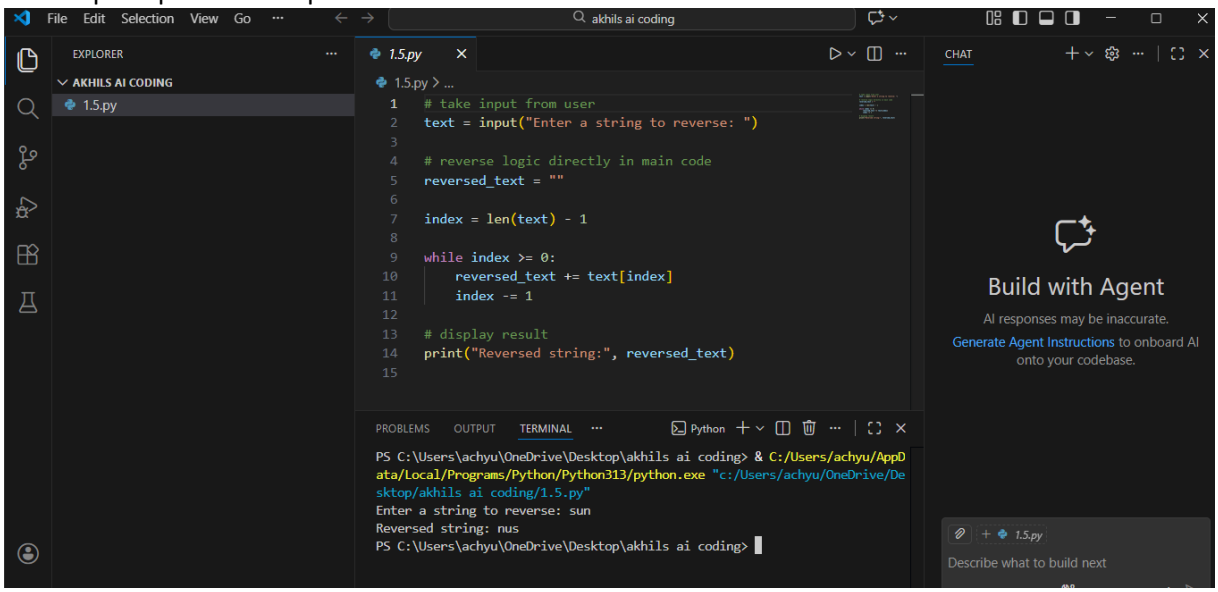
Does not use any user-defined functions

❖ Expected Output

➢ Correct reversed string

➢ Screenshots showing Copilot-generated code suggestions

➢ Sample inputs and outputs



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:

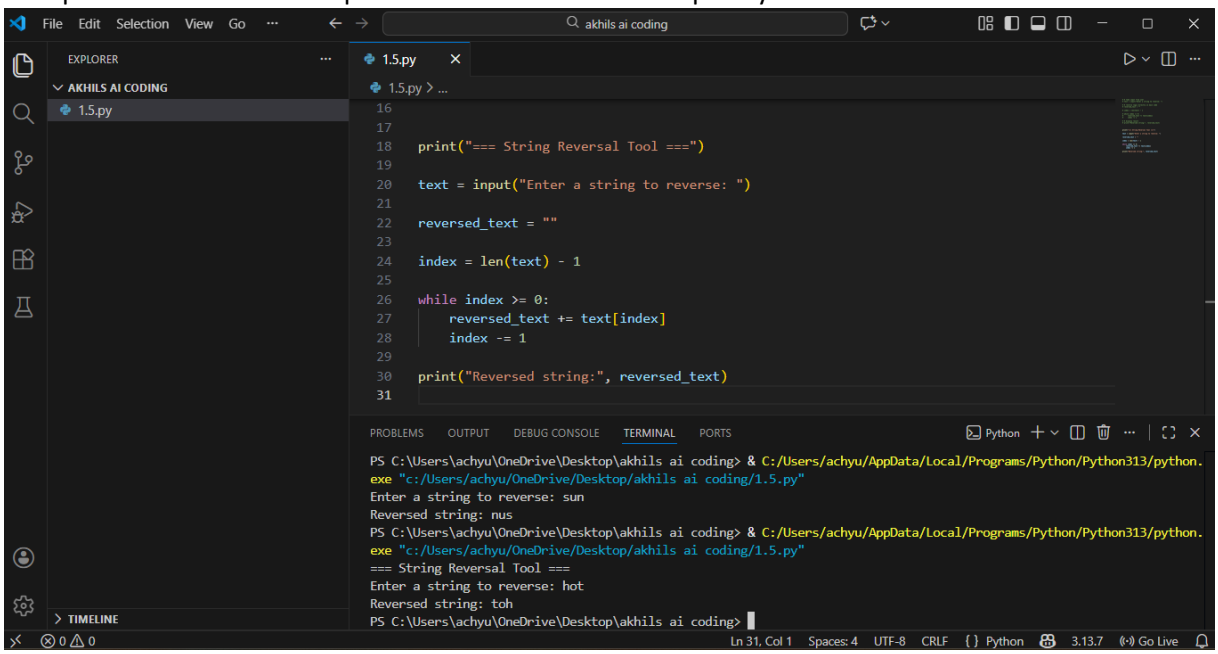
Prompt Copilot with phrases like

“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



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

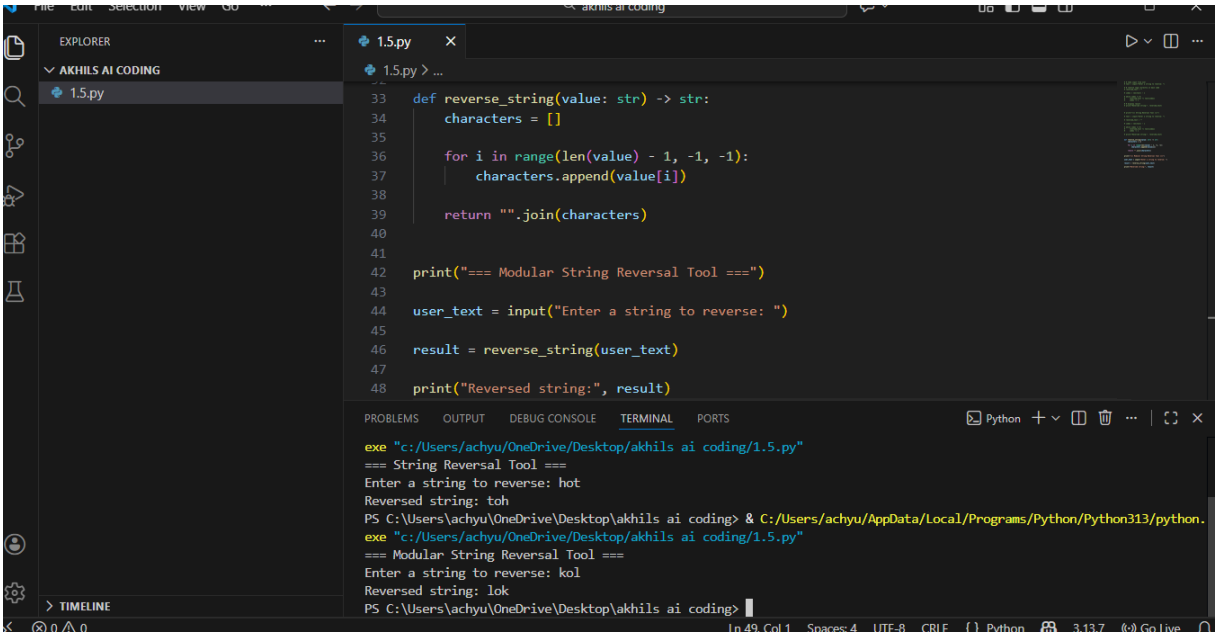
Includes meaningful comments (AI-assisted)

Expected Output

Correct function-based implementation

Screenshots documenting Copilot's function generation

Sample test cases and outputs



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

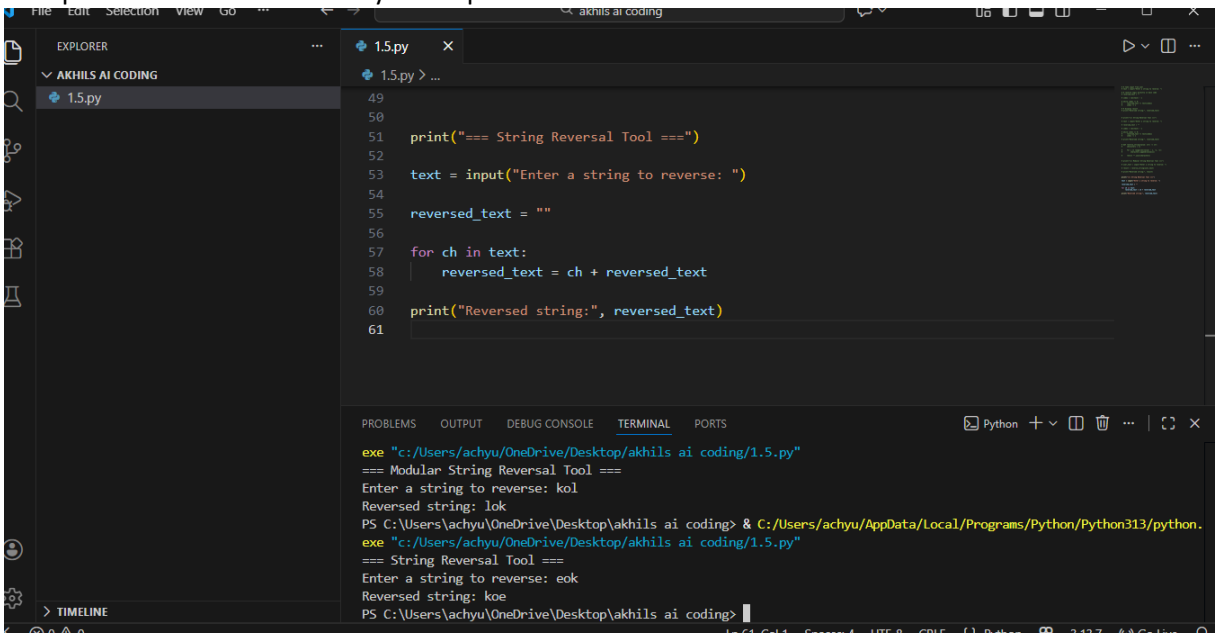
➢ Reusability

➢ Debugging ease

➢ Suitability for large-scale applications

❖ Expected Output

Comparison table or short analytical report



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

Note: Report should be submitted as a word document for all tasks in a

single document with prompts, comments & code explanation, and output

and if required, screenshots