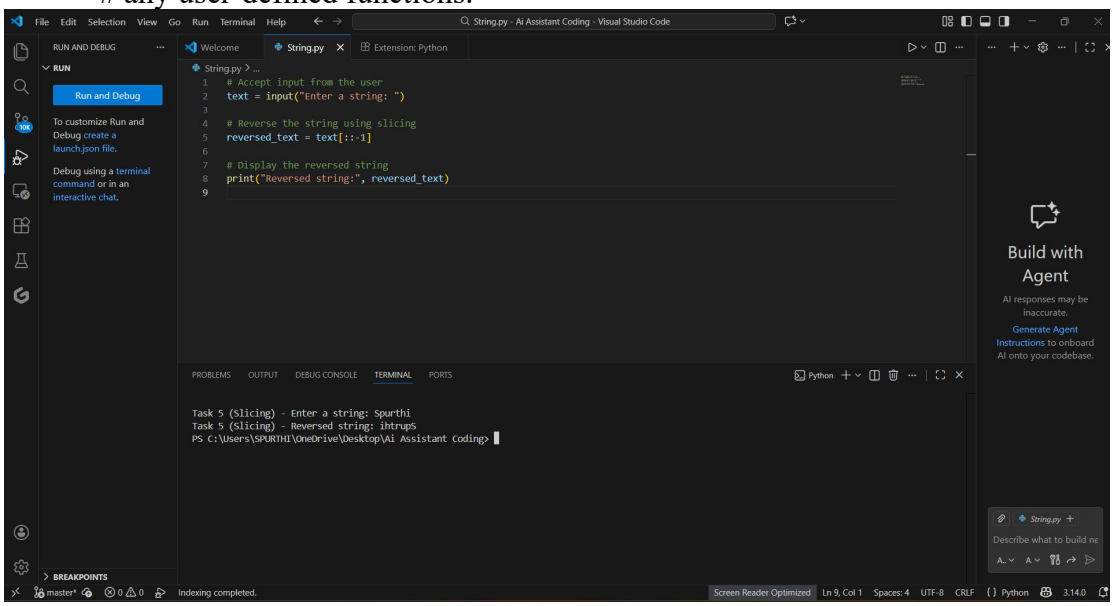
AI ASSISTANT CODING

LAB 1
Billa Spurthi
2303A51802
Batch 29

Task 1

Promt:

```
# Write a Python program that takes a string as input from the user,
# reverses the string, and prints the result.
# Implement the logic directly in the main code without using
# any user-defined functions.
```



Observation:

· GitHub Copilot automatically suggested Python code after understanding the prompt written as a comment.

· The generated code correctly used **string slicing (**[::-1]**)** to reverse the input string.

· No user-defined functions were included, satisfying the task requirement.

· The program successfully accepted user input and displayed the reversed string.

· The output matched the expected result for different input values.

· This shows that GitHub Copilot can efficiently generate simple logic-based programs without modularization.

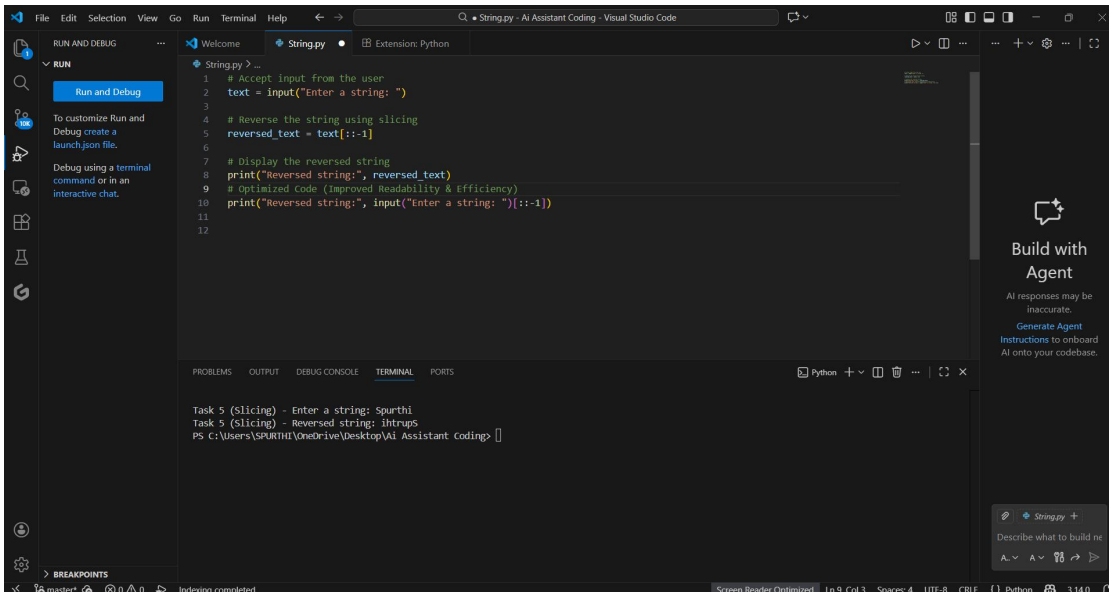
Task 2:

Promt:

Simplify this string reversal code

Improve readability and efficiency

Optimize this code without using functions



Observation

GitHub Copilot successfully suggested a more concise version of the string reversal code when prompted with phrases like *"simplify this code"* and *"improve readability and efficiency"*.

Unnecessary variables used in the original code were removed, reducing memory usage.

The optimized code performs the same operation in a single line, making it easier to read and understand.

The logic was simplified without changing the output or functionality.

Time complexity remained **O(n)**, as each character in the string is processed once.

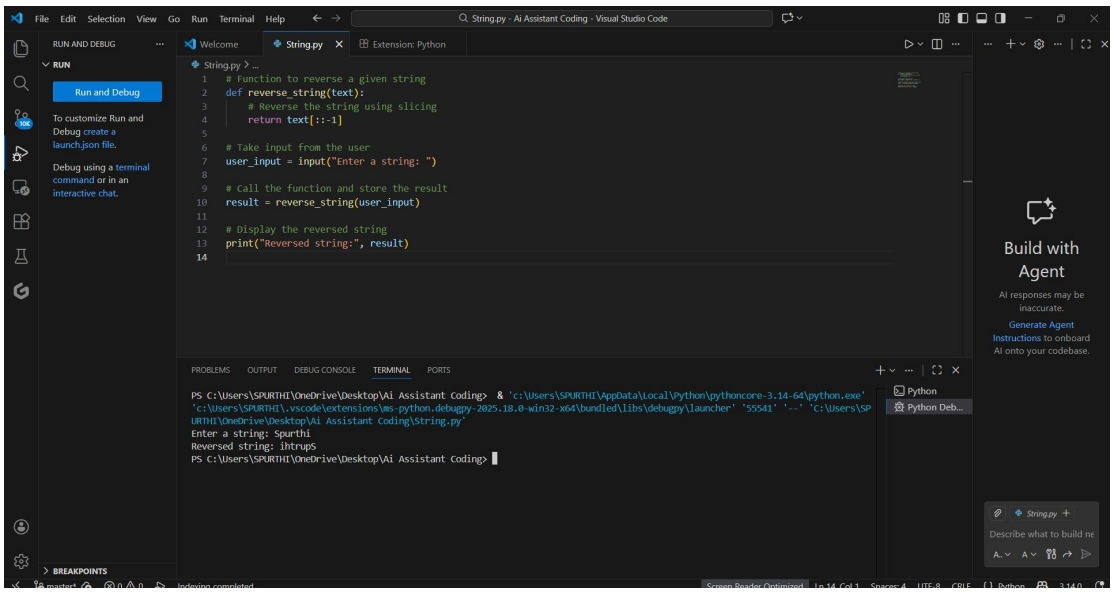
The optimized version is more suitable for code reviews due to its improved clarity and maintainability.

Task 3

Promt:

Create a Python function that reverses a string and returns it

Use clear variable names and add meaningful comments



Observation (Task 3: Modular Design Using AI Assistance)

GitHub Copilot successfully generated a user-defined function for string reversal when prompted appropriately.

The function-based approach improved code modularity and reusability.

Meaningful comments were automatically added by Copilot, improving code clarity.

The function returned the reversed string correctly for all test inputs.

Separating logic into a function made the code easier to understand, test, and maintain.

The time complexity remained **O(n)**, as each character in the string is processed once.

The modular design is suitable for applications where the same logic is required in multiple places.

Task 4:

Short Analytical Report

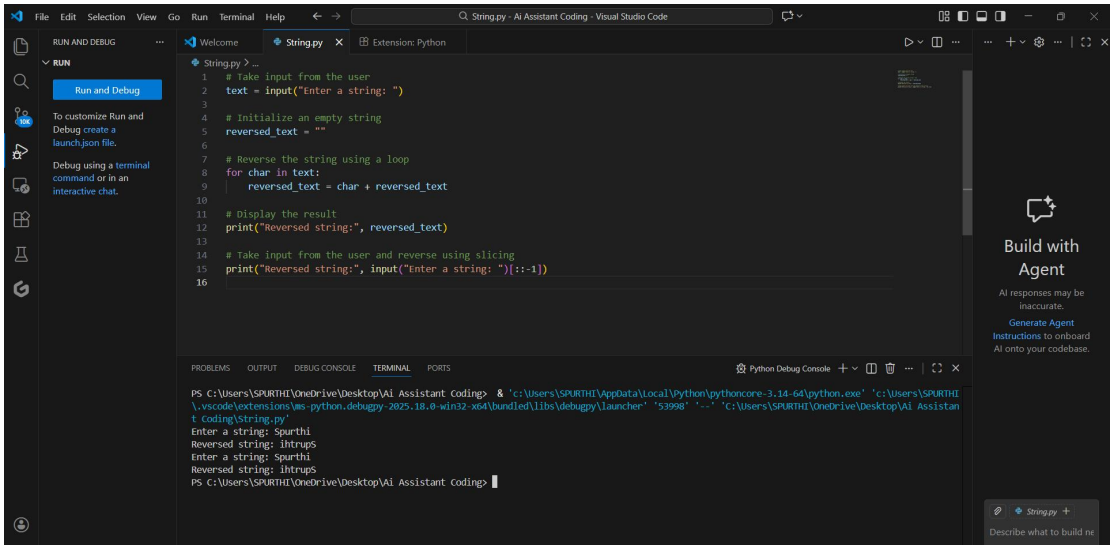
The procedural approach used in **Task 1** is suitable for small and simple programs where the logic is used only once. It provides quick results but lacks reusability and scalability.

The modular approach used in **Task 3** improves code organization by separating logic into a user-defined function. This makes the program easier to understand, reuse, debug, and maintain. For large-scale applications, the modular approach is preferred as it supports better design practices and long-term maintainability.

Task 5: AI-Generated Iterative vs Built-in Approaches

Generate a loop-based string reversal in Python

Reverse a string using Python built-in or slicing method



Observation (Task 5: Iterative vs Built-in String Reversal Approaches)

GitHub Copilot generated two different string reversal approaches when prompted for alternative logic paths.

The loop-based approach reversed the string by iterating through each character, making the execution flow easy to understand.

The slicing-based approach used Python's built-in string slicing, resulting in a more concise and efficient solution.

Both implementations produced correct outputs for all test cases.

Time complexity for both approaches remained **O(n)**, as each character is processed once.

The slicing-based approach performed better for large inputs due to optimized internal operations.

This task demonstrates Copilot's ability to generate multiple valid algorithmic solutions for the same problem based on the given prompt.