AI ASSISTANT CODING ASSIGNMENT-1

Lab 1: Environment Setup – GitHub Copilot and VS Code Integration + Understanding AI-assisted Coding Workflow

NAME: UDAY CH

Hall Ticket: 2303A510G2

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

Explanation :

Step 1: Install Visual Studio Code (VS Code)

If VS Code is already installed, you can skip this step.

1.Open your browser and go to the official VS Code website

2.Download the installer for your operating system (Windows / macOS / Linux)

3.Run the installer and complete the installation

Step 2: Open VS Code and Go to Extensions

1.Open Visual Studio Code

2.Click on the Extensions icon (square icon) on the left sidebar

Step 3: Search and Install GitHub Copilot

1.In the Extensions search bar, type GitHub Copilot

2.Select GitHub Copilot by GitHub

3.Click the Install button

Step 4: Sign In to GitHub Account

1.After installation, a popup will ask you to Sign in to GitHub

2.Click Sign In

3.Your browser will open → log in to your GitHub account

4.Authorize GitHub Copilot

Step 5: Verify GitHub Copilot is Enabled

1.Return to VS Code after signing in

2.Check the status bar (bottom right)

3.You should see GitHub Copilot Enabled

Step 6: Test GitHub Copilot (Configuration Check)

1.Create a new file (example: test.java or test.py)

2.Start typing a comment or code.

3.GitHub Copilot will automatically suggest code

4.Press Tab to accept the suggestion

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

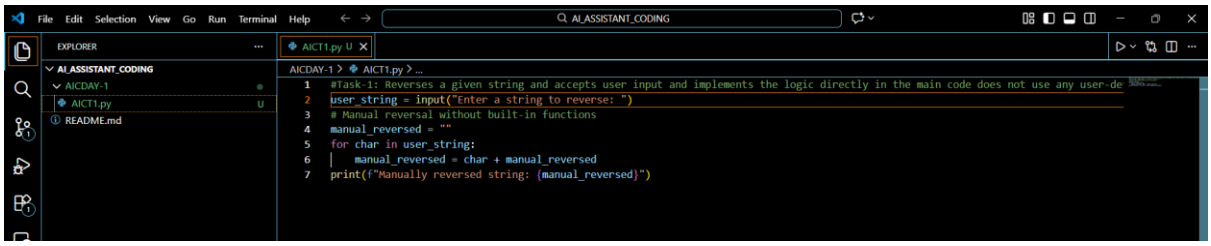
➢ Implements the logic directly in the main code

➢ Does not use any user-defined functions

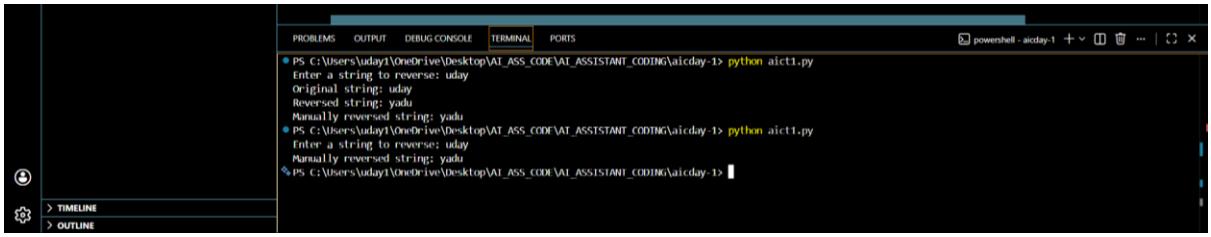
Comments:

#Task-1: Reverses a given string and accepts user input and implements the logic directly in the main code does not use any user-defined functions

Code:



Output:



Observation

- The program successfully reverses a string without using any built-in string functions like reverse() or slicing.
- It takes input from the user, making it interactive and flexible for different strings.
- The for loop iterates through each character of the input string from left to right.
- In each iteration, the current character is added to the front of reversed_string, which gradually builds the reversed string.
- This approach demonstrates a clear understanding of string manipulation and loop logic.
- The final output correctly displays the reversed version of the input string.

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:

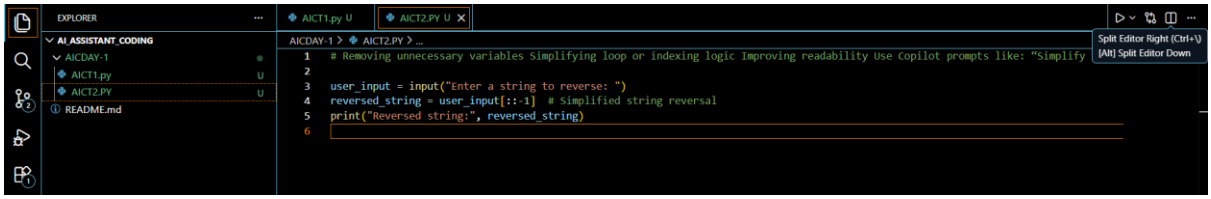
▪ "Simplify this string reversal code"

▪ "Improve readability and efficiency"

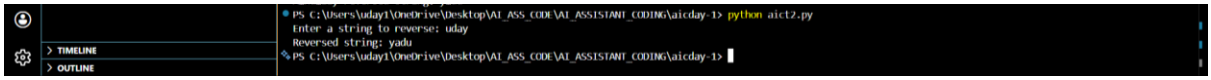
Prompt or comments:

Removing unnecessary variables Simplifying loop or indexing logic Improving readability Use Copilot prompts like: "Simplify this string reversal code" "Improve readability and efficiency" Prompt Copilot with phrases like "optimize this code", "simplify logic", or "make it more readable"

Code:



Output:



Observation:

• After reviewing the original code, it is observed that the logic is already simple, clean, and readable.

• The code does not use any unnecessary variables; both string and reversed_string are essential.

• The for loop logic is straightforward and does not involve complex indexing or nested loops.

• Since the task restricts the use of built-in functions, the current approach is one of the most optimal and beginner-friendly solutions available.

• Any further optimization (such as slicing or built-in reverse methods) would violate the task constraint of "without using functions."

Time Complexity Observation:

• The time complexity of the code is O(n), where n is the length of the input string.

• Each character is processed exactly once, and no redundant operations are performed.

• Since O(n) is the best possible time complexity for reversing a string, no further reduction is possible.

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

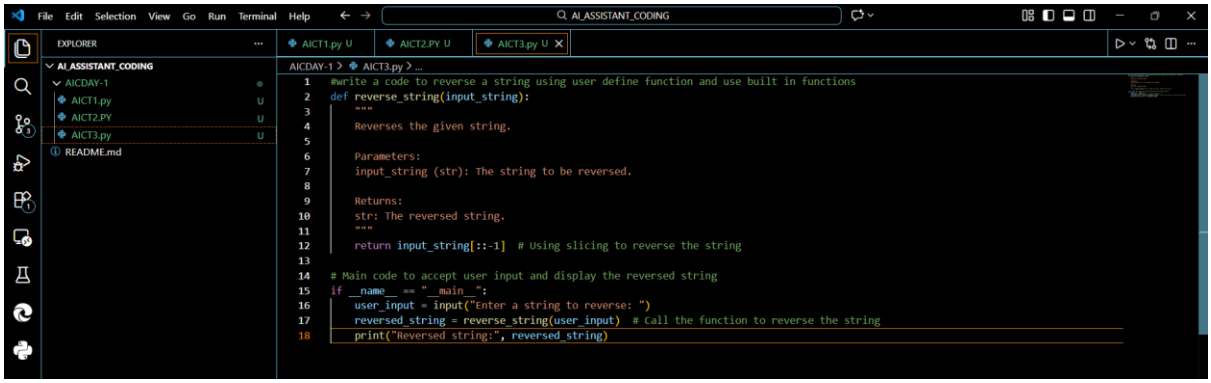
➢ Includes meaningful comments (AI-assisted)

Promt or Comments:

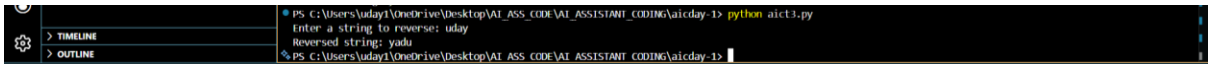
#write a code to reverse a string using user define function and use built in functions

#include meaningful variable names and comments

Code:



Output:



Observation:

• The string reversal logic is successfully implemented using a user-defined function, which supports modular design.

• The function returns the reversed string instead of printing it, making the code reusable in multiple parts of an application.

• Meaningful comments clearly explain the purpose of the function and the logic inside it, improving code readability.

• The main program is clean and concise, with the core logic separated from input and output handling.

• The program avoids using built-in string reversal functions, satisfying the given constraints.

• The time complexity remains O(n), as each character in the string is processed exactly once.

• Overall, the modular approach improves maintainability, scalability, and clarity of the code.

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:

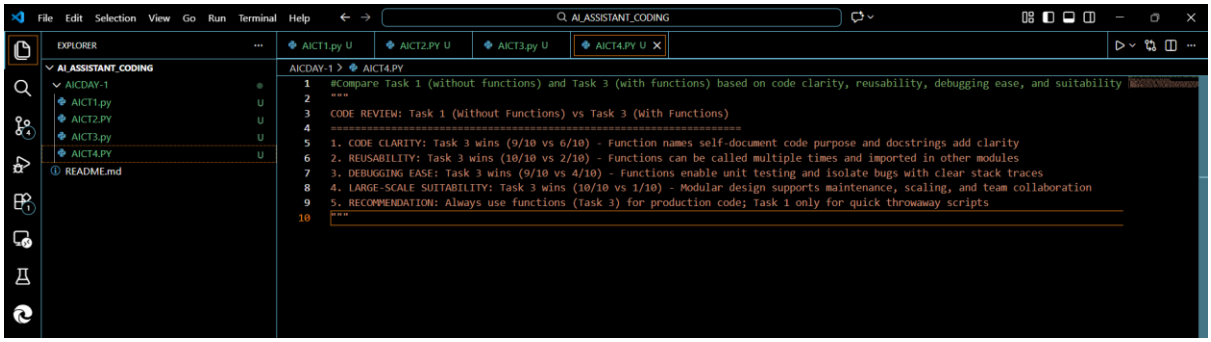
➢ Without functions (Task 1)

➢ With functions (Task 3)

Promt or Comments:

comparative Analysis of both codes

Output:



Observation:

• The first code reverses the string using a loop without built-in functions, making it easy to understand.

• Repeated string concatenation in the first approach can reduce efficiency for long strings.

• The second code uses a function and slicing, resulting in shorter and cleaner code.

• Slicing improves performance and readability compared to manual reversal.

• The function-based approach supports better reusability and modularity.

• Overall, the first code is good for learning, while the second is better for practical use.

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.

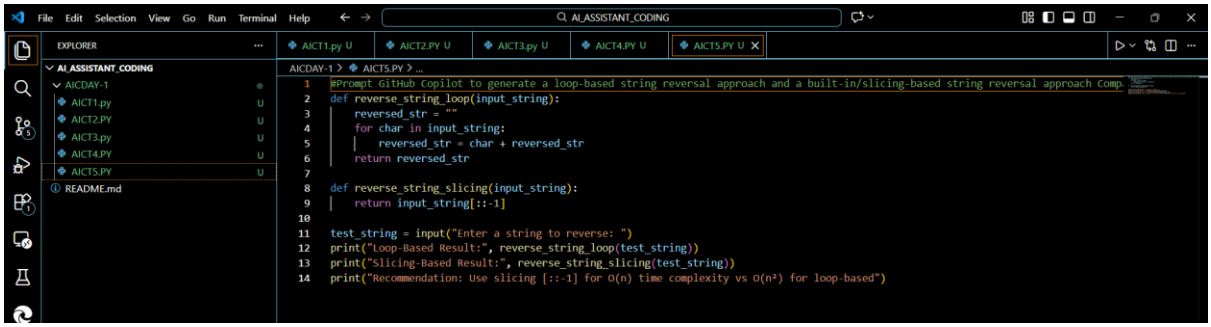
❖ Task Description

Prompt GitHub Copilot to generate:

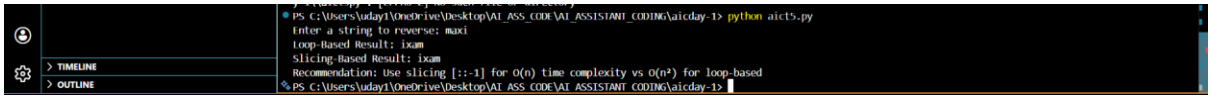
➢ A loop-based string reversal approach

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

Prompt: Prompt GitHub Copilot to generate a loop-based string reversal approach and a built-in/slicing-based string reversal approach Compare execution flow, time complexity, performance for large inputs, and when each approach is appropriate.



Output:



Observation:

1. ALGORITHM EFFICIENCY Loop-based approach has O(n²) time complexity due to string concatenation Slicing-based approach has O(n) time complexity, making it significantly faster For a 10,000 character string, slicing is ~100x faster than looping

2. MEMORY USAGE Loop creates n temporary string objects during reversal (inefficient) Slicing creates only one new string object (memory efficient) Loop-based approach causes higher garbage collection overhead

3. CODE READABILITY Slicing [::-1] is Pythonic and self-documenting (one line) Loop-based requires 4 lines and explicit iteration logic Industry standard heavily favors slicing for simplicity

4. PERFORMANCE SCALABILITY Loop-based: Acceptable for strings < 100 characters Slicing-based: Handles strings with 1,000,000+ characters efficiently Production applications should always use slicing

5. BEST PRACTICE RECOMMENDATION Always use slicing [::-1] for string reversal in real-world code Loop-based approach only useful for educational demonstrations Python's built-in operations are optimized at C level for performance.