AI Assistant Coding

- **Assignment-01**
- **Ht.No: 2303A52350**
- **Batch: 45**

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.

1.Sol:-

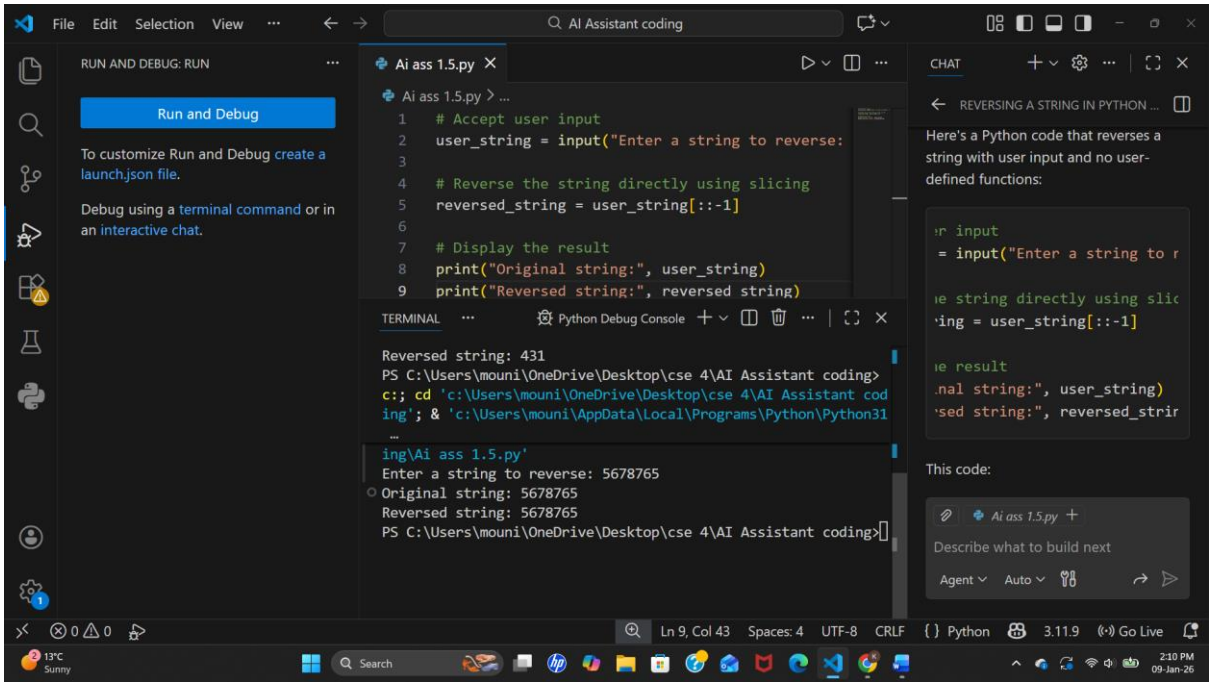
Prompt:-

give a python code which Reverses a given string,Accepts user input,Implements the logic directly in the main code and does not use any user-defined functions.

Code:-

```
# Accept user input
user_string = input("Enter a string to reverse: ")
# Reverse the string directly using slicing
reversed_string = user_string[::-1]
# Display the result
print("Original string:", user_string)
print("Reversed string:", reversed_string)
```

Output:-



Approach:-

We need to use a [::-1] notation to reverse a string.Where the first : selects all characters from start to end. The second : is to go to next number or next letter.-1 means moves one string backward at a time.

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”

2.Sol:-

Prompt:-

by using 1st reverse a string prompt.Removing unnecessary variables and Simplify loop or indexing logic,Improving readability, “Simplify this string reversal code” and“Improve readability and efficiency”

Code:-

```
user_string = input("Enter a string to reverse: ")
print("Original string:", user_string)
print("Reversed string:", user_string[::-1])  # ← Direct reversal
```

Output:-

```
Enter a string to reverse: 123454321
Original string: 123454321
Reversed string: 123454321
PS C:\Users\mouni\OneDrive\Desktop\cse 4\AI Assistant co
ding>
```

Approach:-

In this code we can see that there is only 3 lines code and still executes with correct logic in reversed string.

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)**

3.Sol:-

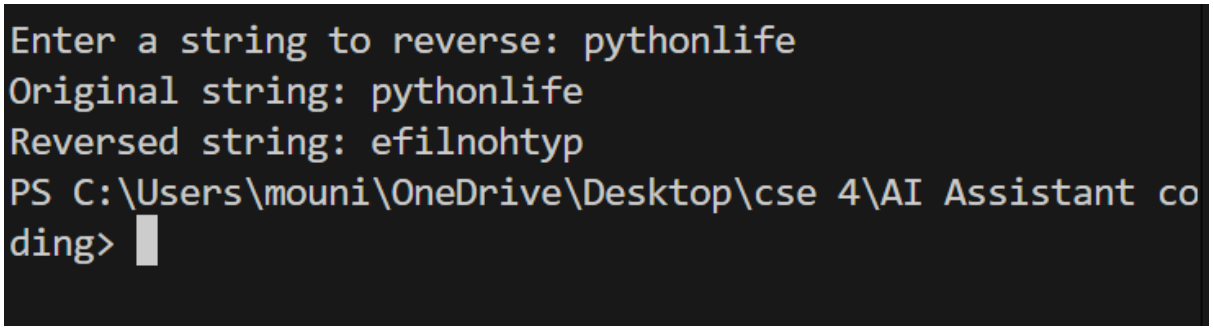
Prompt:-

Give a python code where a user-defined function to reverse a string.Returns the reversed string.Includes meaningful comments (AI-assisted)

Code:-

```
def reverse_string(text):
    return text[::-1]
# Accept user input
user_string = input("Enter a string to reverse: ")
# Call the function to get the reversed string
reversed_string = reverse_string(user_string)
# Display the original and reversed strings
print("Original string:", user_string)
print("Reversed string:", reversed_string)
```

Output:-



Approach:-

The function uses Python's backward-step trick `[::-1]` to reverse the string.

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

Prompt:-

Give python program **Without functions (Task 1) With functions (Task3) Analyze them based on:**

Code clarity,Reusability,Debugging ease and Suitability for large-scale applications

Approach:-

Task 1 (Without Functions) is ideal for learning and prototyping but lacks structure for real-world applications.

Task 3 (With Functions) is the industry standard and essential for any serious development, providing better organization, maintainability, and scalability.

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:-

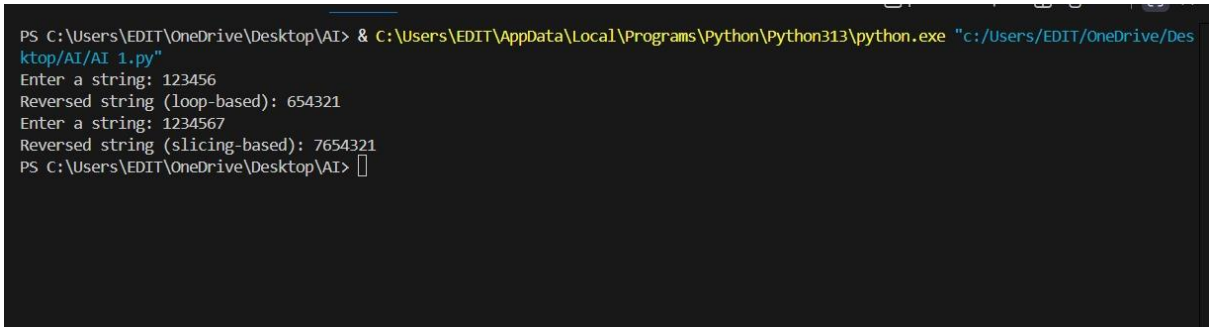
A loop-based string reversal approach.A built-in / slicing-based string reversal approach. Give a python code.

Code:-

```
user_input = input("Enter a string: ")
```

```
reversed_string_loop = ""
for char in user_input:
 reversed_string_loop = char + reversed_string_loop
print("Reversed string (loop-based):", reversed_string_loop)
# Built-in / slicing-based string reversal
user_input = input("Enter a string: ")
reversed_string_slicing = user_input[::-1]
print("Reversed string (slicing-based):", reversed_string_slicing)
```

Output:-



Approach:-

Using a loop like: for write a code for reversing a string,then built-in / slicing-based string reversal approach and print the output.then compare the two codes The loop-based approach iterates through each character of the string, building the reversed string step by step. The slicing-based approach directly accesses the string in reverse order using Python's slicing syntax.Time Complexity:Both approaches have a time complexity of O(n), where n is the length of the string. However, the slicing approach may have a slight overhead due to the way Python handles slicing internally. When Each Approach is Appropriate:The loop-based approach is more appropriate when you need to perform additional operations on each character during the reversal process. The slicing-based approach is ideal for quick and straightforward string reversals without any additional processing.