

Assignment - 1

Name: C.Vaishnav Reddy

Roll Number: 2303A52203

Batch - 45

AI Assisted Coding

09-01-2026

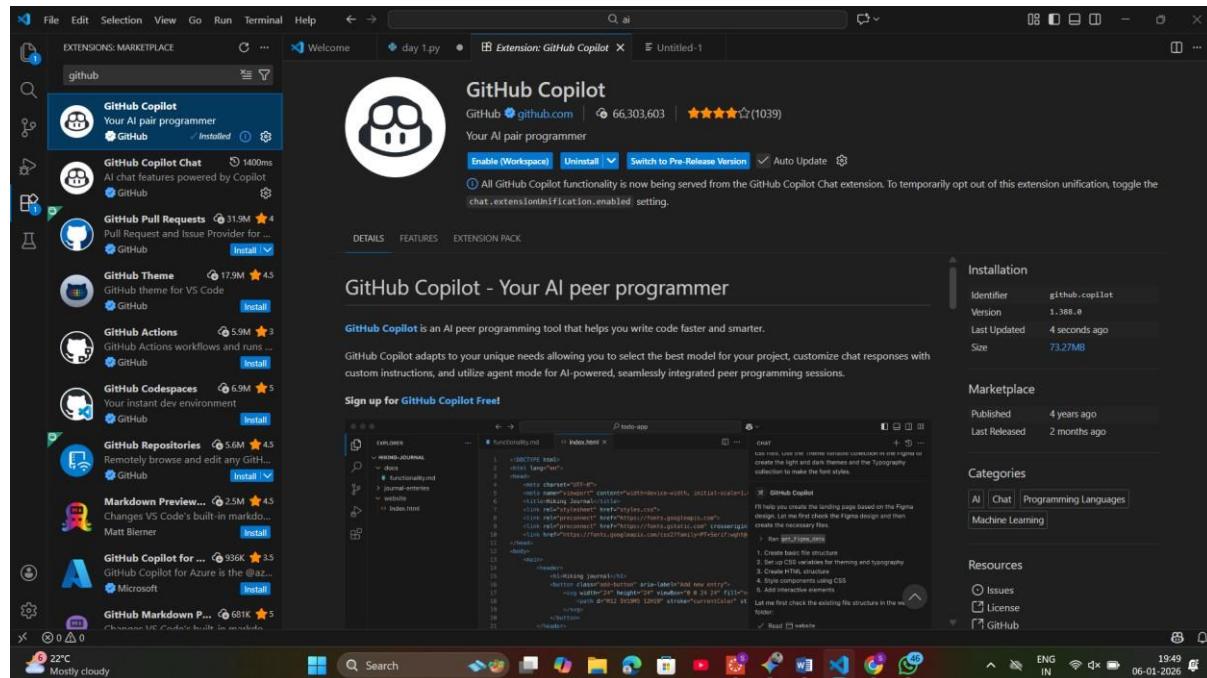
Task 0: Environment Setup:-

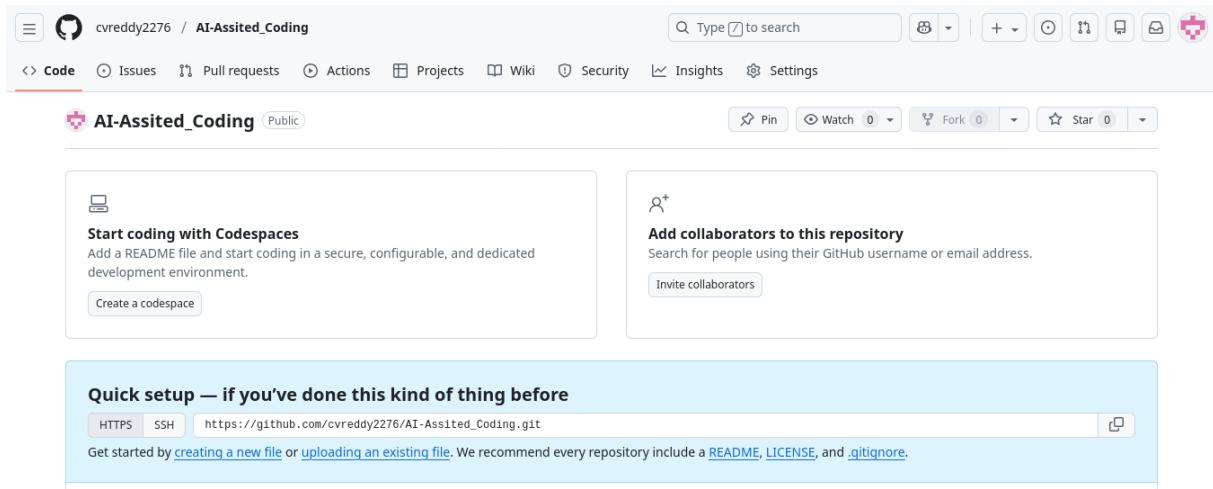
Task 0

- 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: Non-Modular Logic (Factorial):-

: AI-Generated Logic Without Modularization (String Reversal Without Functions)

❖ Scenario

You are developing a basic text-processing utility for a messaging application.

❖ Task Description

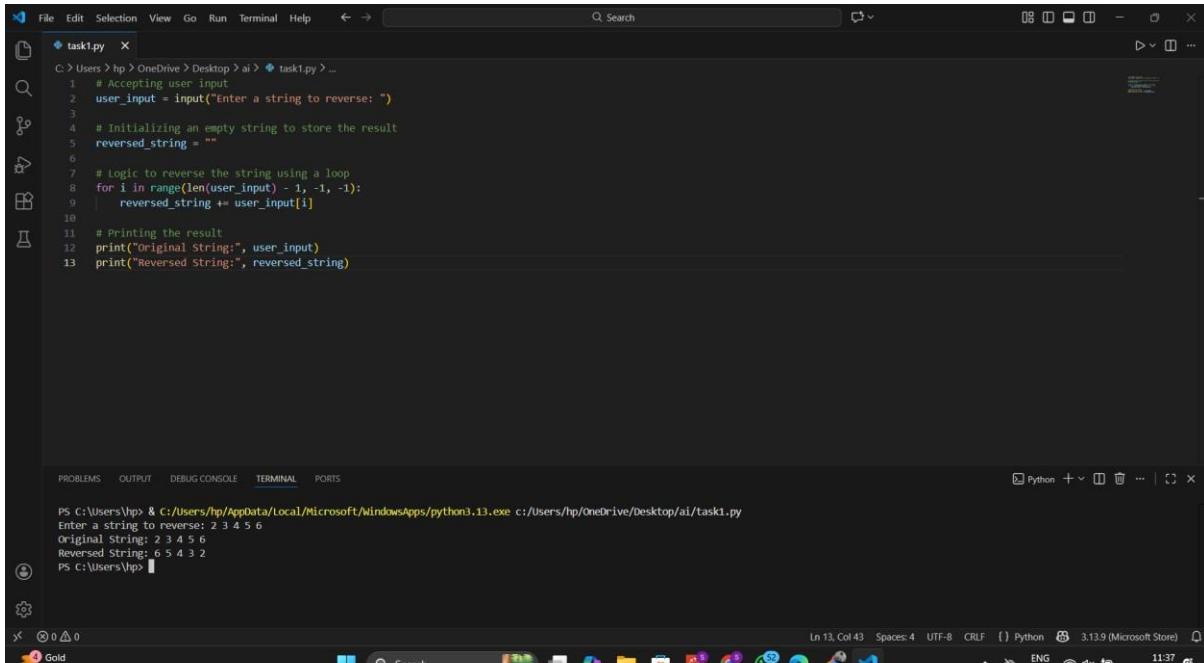
Use GitHub Copilot to generate a Python program that:

- 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



The screenshot shows the VS Code interface. The top part displays the Python file `task1.py` with the following code:

```
C:\> Users > hp > OneDrive > Desktop > ai > task1.py > ...
1 # Accepting user input
2 user_input = input("Enter a string to reverse: ")
3
4 # Initializing an empty string to store the result
5 reversed_string = ""
6
7 # Logic to reverse the string using a loop
8 for i in range(len(user_input) - 1, -1, -1):
9     reversed_string += user_input[i]
10
11 # Printing the result
12 print("Original String:", user_input)
13 print("Reversed String:", reversed_string)
```

The bottom part shows the terminal window with the following output:

```
PS C:\Users\hp> & C:/Users/hp/AppData/Local/Microsoft/WindowsApps/python3.13.exe c:/Users/hp/OneDrive/Desktop/ai/task1.py
Enter a string to reverse: 2 3 4 5 6
Original String: 2 3 4 5 6
Reversed String: 6 5 4 3 2
PS C:\Users\hp>
```



The screenshot shows a dark-themed terminal window with the same code and output as the VS Code terminal.

Task 2: AI Code Optimization:-

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**

The screenshot shows a code editor with a dark theme. The file 'task1.py' is open, containing the following Python code:

```
C:\Users\hp>OneDrive\Desktop>ai> task1.py > ...
1 user_input = input("Enter a string: ")
2
3 # Using Python's slicing for maximum efficiency
4 reversed_string = user_input[::-1]
5
6 print(f"Reversed: {reversed_string}")
```

Below the code editor is a terminal window with the following output:

```
PS C:\Users\hp\OneDrive\Desktop\ai> & 'c:\Users\hp\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\hp\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\lib\site-packages\debugpy\launcher' '50075' '--' 'c:\Users\hp\OneDrive\Desktop\ai\task1.py'
Enter a string: 40 50 60 70
Reversed: 70 60 50 40
PS C:\Users\hp\OneDrive\Desktop\ai>
```

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

```
task1.py •
C:\Users\hp\OneDrive\Desktop\ai> task1.py > ...
1 def reverse_string_functional(text):
2     """
3         Reverses the input string and returns it.
4     """
5     reversed_text = ""
6     for char in text:
7         |     reversed_text = char + reversed_text
8     return reversed_text
9
10 # Testing the function
11 input_str = input("Enter text: ")
12 result = reverse_string_functional(input_str)
13 print(f"Result: {result}")
```

```
on.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher`'56968`'---' c:\Users\hp\OneDrive\Desktop\ai\task1.py'
Enter text: Harshini
Result: inhrshinH
PS C:\Users\hp\OneDrive\Desktop\ai>
```

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

| Feature | Procedural (Without Functions) | Modular (With Functions) |
|---------------------|--|--|
| Code Clarity | Easy for tiny scripts; messy for large ones. | Very high; logic is isolated and named. |
| Reusability | Must copy-paste code to use it again. | Can be called anywhere in the app. |
| Debugging | Harder to isolate where an error occurs. | Easy to unit test the specific function. |
| Scalability | Not suitable for large applications. | Essential for professional development. |

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.

The screenshot shows the VS Code interface with the Python file 'task1.py' open. The code defines two functions: 'reverse_iterative' which uses a for loop to build a reversed string character by character, and 'reverse_slicing' which uses Python's slicing feature to reverse the string. A test input is provided to demonstrate both functions.

```
C: > Users > hp > OneDrive > Desktop > ai > task1.py > ...
1 def reverse_iterative(input_string):
2     reversed_str = ""
3     for char in input_string:
4         reversed_str = char + reversed_str
5     return reversed_str
6
7 def reverse_slicing(input_string):
8     return input_string[::-1]
9
10 test_input = input("Enter a string: ")
11
12 print(reverse_iterative(test_input))
13 print(reverse_slicing(test_input))
```

The screenshot shows the VS Code terminal window displaying the execution of 'task1.py'. The user enters the string '12345' at the prompt, and the program outputs the reversed strings from both functions: '54321' and '54321' respectively.

```
s\debgpy\launcher' '50436' '--' 'c:\Users\hp\OneDrive\Desktop\ai\task1.py'
PS C:\Users\hp\OneDrive\Desktop\ai>
PS C:\Users\hp\OneDrive\Desktop\ai> c;; cd 'c:\Users\hp\OneDrive\Desktop\ai'; & 'c:\Users\hp\AppData\Local\Microsoft\WindowsApps\python3.13.exe' 'c:\Users\hp\.vscode\extensions\ms-python.on.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '5717' '--' 'c:\Users\hp\OneDrive\Desktop\ai\task1.py'
on.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '5717' '--' 'c:\Users\hp\OneDrive\Desktop\ai\task1.py'
Enter a string: 12345
54321
54321
PS C:\Users\hp\OneDrive\Desktop\ai>
```