

Assignment - 1

Name:A.Akshith Reddy

Roll Number: 2303A51177

Batch - 03

AI Assisted Coding

09-01-2026

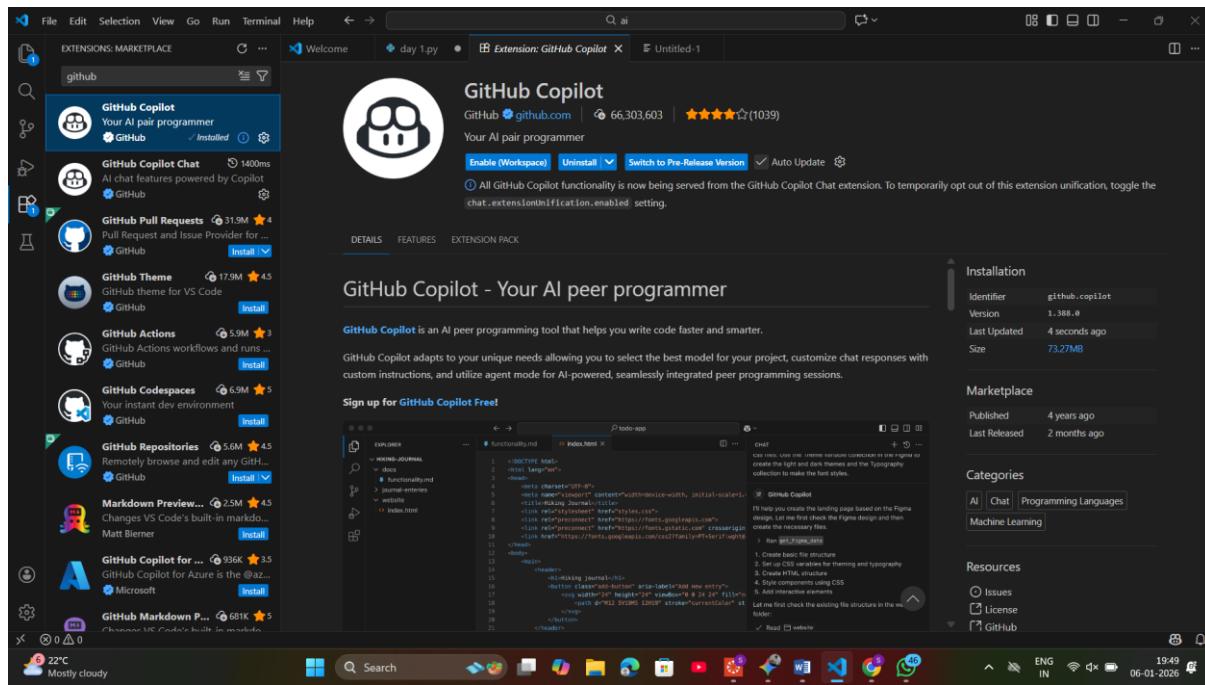
Task 0: Environment Setup:-

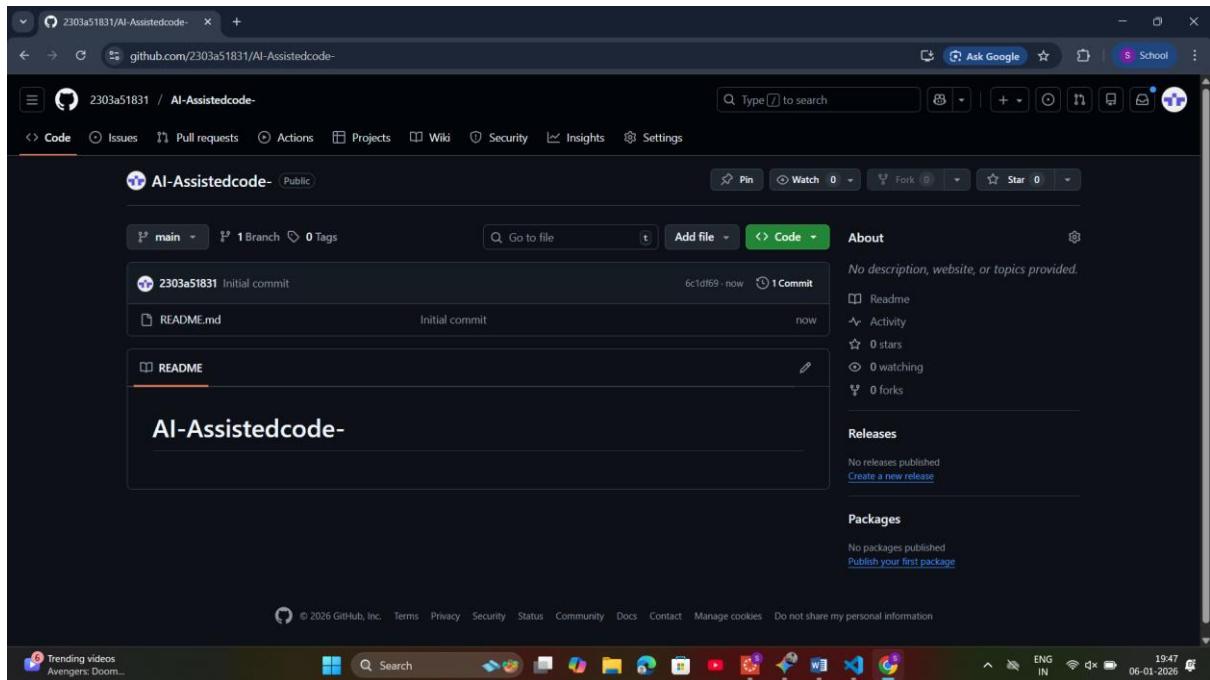
Task 0

- 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

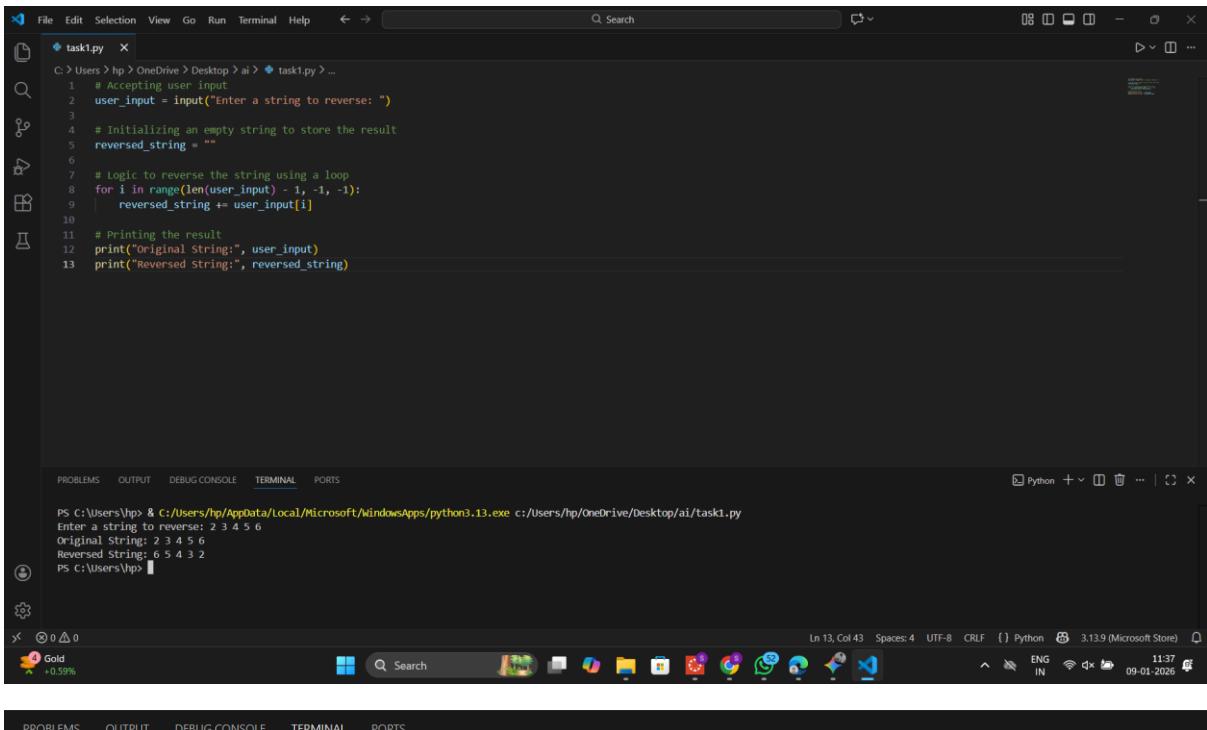
Use GitHub Copilot to generate a Python program that:

- Reverses a given string
- Accepts user input
- Implements the logic directly in the main code
- Does not use any user-defined functions

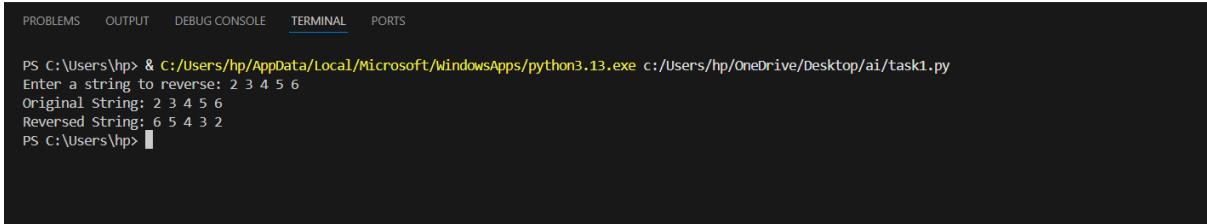
❖ Expected Output

- Correct reversed string
- Screenshots showing Copilot-generated code suggestions

➤ Sample inputs and outputs



The screenshot shows the Visual Studio Code interface. The top part displays the code for `task1.py` in a dark-themed editor. The code reads a string from the user, initializes an empty string for the result, and then uses a loop to build the reversed string character by character. The bottom part shows the terminal window with the command `python task1.py` run, followed by the original string "2 3 4 5 6" and its reversed version "6 5 4 3 2".



This is a dark-themed terminal window showing the same command and output as the one in VS Code, confirming the code's functionality.

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:

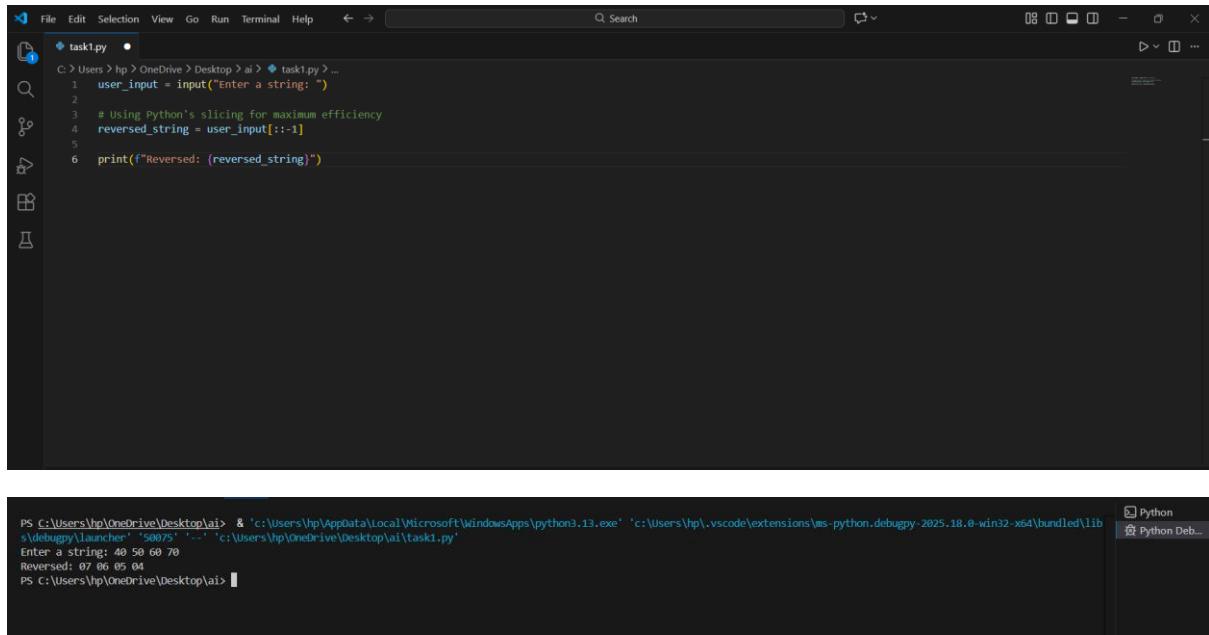
Prompt Copilot with phrases like

“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 the Visual Studio Code interface. On the left, the code editor displays a file named 'task1.py' with the following content:

```
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}")
```

On the right, a terminal window shows the output of running the script:

```
PS C:\Users\hp\OneDrive\Desktop\ai> & "c:\Users\hp\AppData\Local\Microsoft\WindowsApps\python3.13.exe" "c:\Users\hp\vscode\extensions\ms-python.python-2025.18.0-win32-x64\bundled\lib
s\debugpy\Launcher" "50075" "..." "c:\Users\hp\OneDrive\Desktop\ai\Task1.py"
Enter a string: 40 50 60 70
Reversed: 07 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

```
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}")
```

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
+
Enter text: Teju
Result: jueT
PS C:\Users\hp\OneDrive\Desktop\ai> ^
PS C:\Users\hp\OneDrive\Desktop\ai>
PS C:\Users\hp\OneDrive\Desktop\ai> 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' "59916" ...; c:\Users\hp\OneDrive\Desktop\ai\task1.py
Enter text: Akshith
Result: htihskA
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.**

A screenshot of the Visual Studio Code interface. The top bar shows the file menu (File, Edit, Selection, View, Go, Run, Terminal, Help), a search bar, and a Python Debugger status bar. The main area displays a Python script named `task1.py`. The code defines two functions: `reverse_iterative` which uses a loop to build a reversed string, and `reverse_slicing` which uses Python's slicing feature. It then prompts the user for input and prints both results.

```
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))
```

A screenshot of the Visual Studio Code interface focusing on the terminal tab. The terminal window shows the command `s\debugpy\launcher` being run to start a debugger, followed by the command to run the Python script `task1.py`. The script is executed in a Windows environment, and the user is prompted to enter a string. The user types "1 2 3 4 5" and the script outputs the reversed string "5 4 3 2 1". The terminal also indicates that indexing is completed.

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
+ v ... | ⌂ x
Python
Python Deb...
Python Deb...
TERMINAL
s\debugpy\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' '57517' ... 'c:\Users\hp\OneDrive\Desktop\ai\task1.py'
on.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher` '57517' ... 'c:\Users\hp\OneDrive\Desktop\ai\task1.py'
Enter a string: 1 2 3 4 5
5 4 3 2 1
PS C:\Users\hp\OneDrive\Desktop\ai> [Delta] 0 ⚡ Indexing completed.
```