

## AI ASSISTED CODING

Name: P.Swaran Raj  
Batch: 37

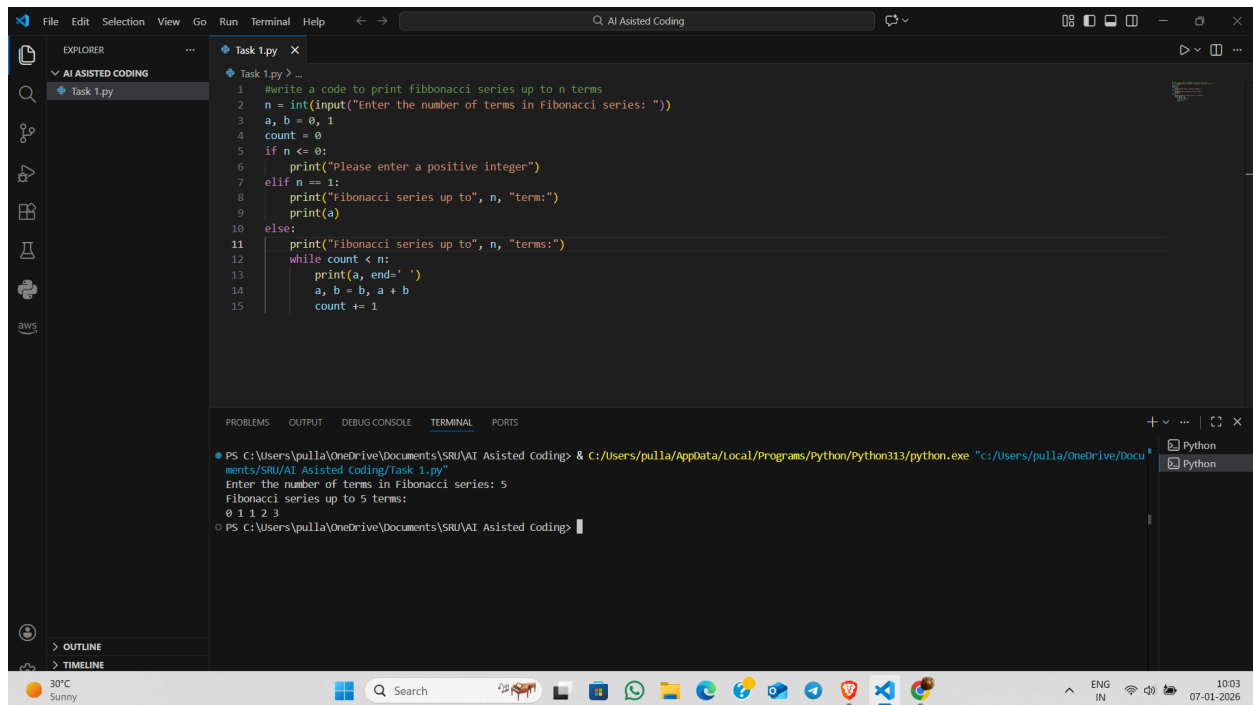
Roll.No: 2303A52247  
Assignment-1

### Task 1- AI-Generated Logic Without Modularization (Fibonacci Sequence Without Functions)

Prompt used:

“#write a code to print fibonacci series up to n terms”

“#write a code to print fibonacci series up to n terms without using functions”

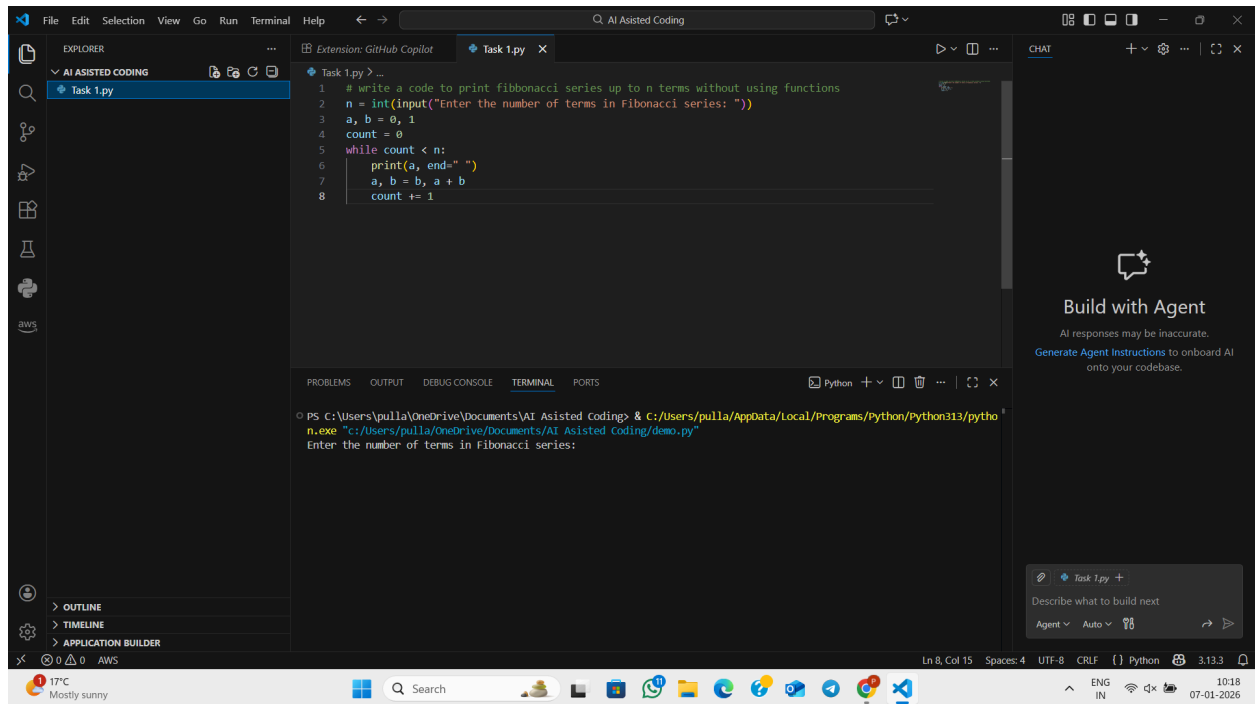


```
File Edit Selection View Go Run Terminal Help
AI Assisted Coding

EXPLORER
AI ASSISTED CODING
Task 1.py

Task 1.py
1 #write a code to print fibonacci series up to n terms
2 n = int(input("Enter the number of terms in Fibonacci series: "))
3 a, b = 0, 1
4 count = 0
5 if n <= 0:
6     print("Please enter a positive integer")
7 elif n == 1:
8     print("Fibonacci series up to", n, "term:")
9     print(a)
10 else:
11     print("Fibonacci series up to", n, "terms:")
12     while count < n:
13         print(a, end=' ')
14         a, b = b, a + b
15         count += 1

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\pulla\OneDrive\Documents\SRU\AI Assisted Coding> & C:/Users/pulla/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/pulla/OneDrive/Docu
ments\SRU\AI Assisted Coding/Task 1.py"
Enter the number of terms in Fibonacci series: 5
Fibonacci series up to 5 terms:
0 1 1 2 3
PS C:\Users\pulla\OneDrive\Documents\SRU\AI Assisted Coding>
```



## Explanation:

GitHub Copilot generated a simple loop-based Fibonacci program using inline comments as prompt, The logic is written directly in the main code without any function definition.

## Output:

For input 5, the output displayed is: 0 1 1 2 3

## Observation:

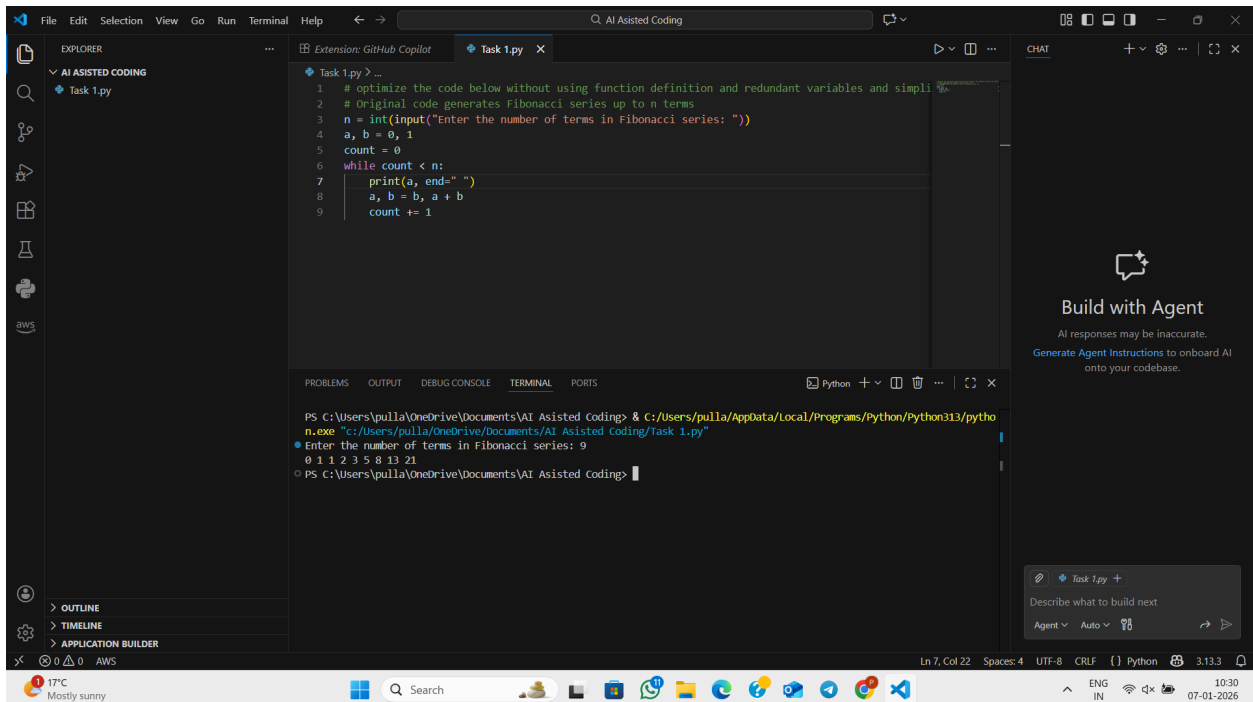
Copilot correctly understood the prompt and produced working procedural code.

However, the code was less reusable and slightly lengthy.

## Task 2: AI Code Optimization & Cleanup (Improving Efficiency)

**Prompt used: “ #Optimize this Fibonacci code by simplifying logic and removing redundant variables.”**

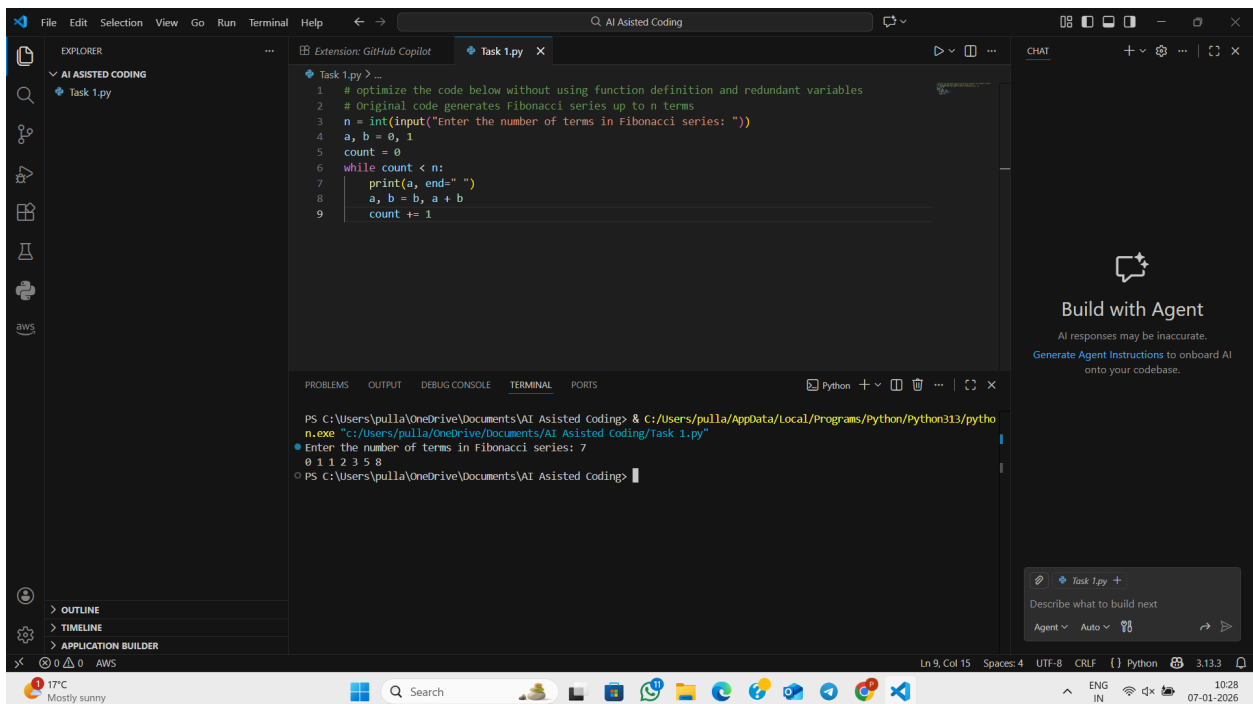
## “#Original code generates Fibonacci series up to n terms”



This screenshot shows the original Python code in a VS Code editor. The code is a script that prompts the user to enter the number of terms in a Fibonacci series and then prints the series. The code is as follows:

```
1 # optimize the code below without using function definition and redundant variables and simplify it
2 # Original code generates Fibonacci series up to n terms
3 n = int(input("Enter the number of terms in Fibonacci series: "))
4 a, b = 0, 1
5 count = 0
6 while count < n:
7     print(a, end=" ")
8     a, b = b, a + b
9     count += 1
```

The terminal output shows the user entering 9, and the program prints the first 9 terms of the Fibonacci series: 0 1 1 2 3 5 8 13 21.



This screenshot shows the optimized Python code in a VS Code editor. The code is a script that prompts the user to enter the number of terms in a Fibonacci series and then prints the series. The code is as follows:

```
1 # optimize the code below without using function definition and redundant variables
2 # Original code generates Fibonacci series up to n terms
3 n = int(input("Enter the number of terms in Fibonacci series: "))
4 a, b = 0, 1
5 count = 0
6 while count < n:
7     print(a, end=" ")
8     a, b = b, a + b
9     count += 1
```

The terminal output shows the user entering 7, and the program prints the first 7 terms of the Fibonacci series: 0 1 1 2 3 5 8.

## Explanation:

Copilot reduced unnecessary variables and simplified the loop structure. The optimized version is shorter, cleaner, and easier to read.

## Output:

For input 7, the output displayed is: 0 1 1 2 3 5 8

## Observation:

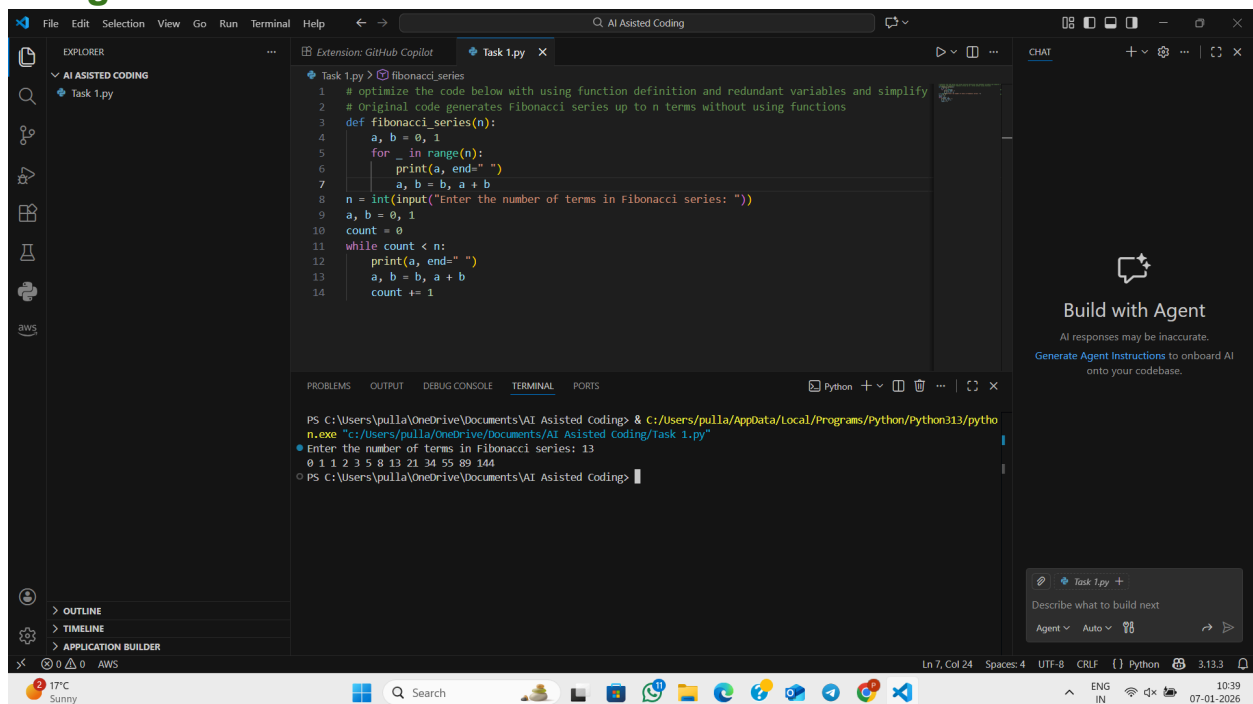
Optimization improved readability and performance.

The code became more concise while producing the same correct output.

## Task 3: Modular Design Using AI Assistance (Fibonacci Using Functions)

**Prompt used: “#Optimize this Fibonacci code by simplifying logic and removing redundant variables.”**

**“#Original code generates Fibonacci series up to n terms without using functions”**



The screenshot shows the Visual Studio Code editor with a file named `Task 1.py`. The code defines a function `fibonacci_series(n)` that prints the Fibonacci series up to `n` terms. The function uses a `for` loop and updates variables `a` and `b` to generate the series. The main program prompts the user to enter the number of terms and then calls the `fibonacci_series` function.

```
1 # optimize the code below with using function definition and redundant variables and simplify
2 # Original code generates Fibonacci series up to n terms without using functions
3 def fibonacci_series(n):
4     a, b = 0, 1
5     for _ in range(n):
6         print(a, end=" ")
7         a, b = b, a + b
8 n = int(input("Enter the number of terms in Fibonacci series: "))
9 a, b = 0, 1
10 count = 0
11 while count < n:
12     print(a, end=" ")
13     a, b = b, a + b
14     count += 1
```

The terminal output shows the execution of the script. The user enters 13, and the program outputs the Fibonacci series: 0 1 1 2 3 5 8 13 21 34 55 89 144.

```
PS C:\Users\pulla\OneDrive\Documents\AI Asisted Coding> & C:\Users\pulla\AppData\Local\Programs\Python\Python313\python.exe "c:\Users\pulla\OneDrive\Documents\AI Asisted coding\Task 1.py"
Enter the number of terms in Fibonacci series: 13
0 1 1 2 3 5 8 13 21 34 55 89 144
PS C:\Users\pulla\OneDrive\Documents\AI Asisted Coding>
```

## Explanation:

Copilot generated a user-defined function to encapsulate Fibonacci logic.

The function is called from the main program for better modularity.

**Output:**

For input 13, the output displayed is:

0 1 1 2 3 5 8 13 21 34 55 89 144

**Observation:**

Function-based approach improved code clarity and reusability.

This approach is more suitable for larger applications.

**Conclusion:** GitHub Copilot helped in quickly generating, optimizing, and understanding Fibonacci programs in VS Code, while manual review ensured correctness and better coding practices.