

## Assignment-1.3

Name: Y. Poojitha

Hall ticket No: 2303A51499

Batch: 08

Lab 1:

Task 1: AI-Generated Logic Without Modularization Fibonacci Sequence Without Functions

Copilot Prompt Used

Write a Python program to print Fibonacci sequence up to n terms

Take input from the user

Do not use any user-defined functions

The screenshot shows the VS Code interface with the following details:

- File Explorer:** Shows 'OPEN EDITORS' with 'Welcome' and '19\_01.py' selected.
- Code Editor:** Displays the Python code for generating a Fibonacci sequence:

```
1 # Program to print Fibonacci sequence up to n terms
2
3 n = int(input("Enter the number of terms: "))
4
5 # Initialize first two terms
6 a, b = 0, 1
7
8 print(f"Fibonacci sequence up to {n} terms:")
9
10 # Generate and print Fibonacci sequence
11 for i in range(n):
12     print(a, end=" ")
13     c = a + b
14     a = b
15     b = c
16
17 print() # New line at the end
18
19
```
- Right Panel:** A tooltip titled 'FIBONACCI SEQUENCE PROGRAM IN PY...' provides the following information:
  - Takes user input for the number of terms using `int(input())`
  - Uses no user-defined functions - only built-in Python constructs
  - Generates Fibonacci sequence using a simple loop where each term is the sum of the previous two
  - Prints the sequence with proper formatting
- Terminal:** Shows the command to run the script and the resulting output:

```
PS D:\AI_assistant_coding> & 'c:\Users\yaraV\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\yaraV\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '51494' '--' 'd:\AI_assistant_coding\19_01.py'
Enter the number of terms: 5
Fibonacci sequence up to 5 terms:
0 1 1 2 3 5 8
```

Result:

The terminal window shows the following output:

```
PS D:\AI_assistant_coding> & 'c:\Users\yaraV\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\yaraV\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '51494' '--' 'd:\AI_assistant_coding\19_01.py'
Enter the number of terms: 5
Fibonacci sequence up to 5 terms:
0 1 1 2 3 5 8
```

Task 2: AI Code Optimization & Cleanup

Copilot Prompt Used

Optimize this Fibonacci code

Simplify logic and variable usage

The screenshot shows the VS Code interface with the AI Assistant Coding extension active. The code editor has a file named `19_01.py` open, containing a simple Fibonacci sequence generator. The AI Assistant panel on the right suggests optimizing the code by eliminating intermediate variables and simplifying the loop variable. The terminal below shows the output of running the script with input `5`, producing the sequence `0 1 1 2 3`.

```

1 n = int(input("Enter the number of terms: "))
2
3 a, b = 0, 1
4
5 for _ in range(n):
6     print(a, end=" ")
7     a, b = b, a + b
8
9 print()
10
11

```

**Result:**

The screenshot shows the Python Debug Console in VS Code. It displays the command `python D:/AI_assistant_coding/19_01.py` being run, followed by the user input `Enter the number of terms: 5`. The console then outputs the Fibonacci sequence `0 1 1 2 3`.

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

Copilot Prompt Used

Create a function to generate Fibonacci sequence up to n

Add meaningful comments

Function-Based Code

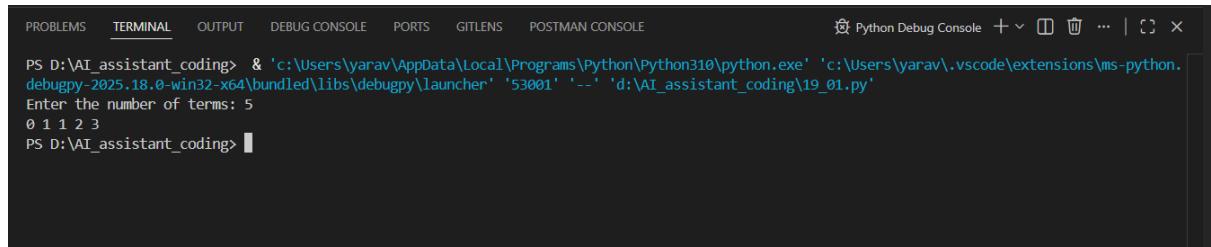
The screenshot shows the VS Code interface with the AI Assistant Coding extension active. The code editor has a file named `19_01.py` open, which has been refactored into a function-based approach. The code includes detailed comments explaining the logic, parameters, and structure. The AI Assistant panel on the right lists key improvements such as refactoring to a function-based approach, adding meaningful comments, and ensuring proper structure. The terminal below shows the output of running the script with input `5`, producing the sequence `0 1 1 2 3`.

```

1 def generate_fibonacci(n):
2     """
3         Generate and print the Fibonacci sequence up to n terms.
4     """
5     Args:
6         n (int): Number of terms to generate
7
8     # Initialize the first two Fibonacci numbers
9     a, b = 0, 1
10
11    # Generate Fibonacci sequence for n iterations
12    for _ in range(n):
13        # Print current number without newline to keep sequence on one line
14        print(a, end=" ")
15
16        # Update a and b using tuple unpacking for simultaneous assignment
17        # a becomes b, and b becomes the sum of previous a and b
18        a, b = b, a + b
19
20    # Print newline after sequence completes
21    print()
22
23
24 # Main program
25 if __name__ == "__main__":
26     # Get number of terms from user
27     n = int(input("Enter the number of terms: "))
28
29     # Validate input
30     if n <= 0:
31         print("Please enter a positive number")
32     else:
33         # Generate and display the Fibonacci sequence
34         generate_fibonacci(n)
35
36

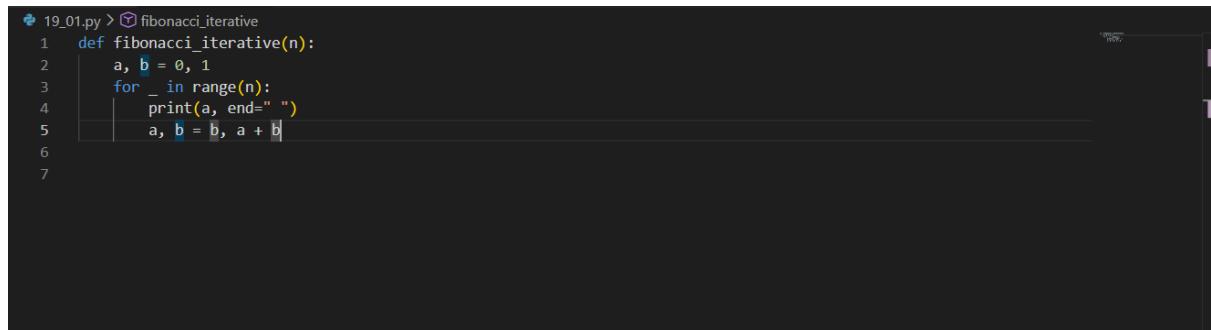
```

## Result:



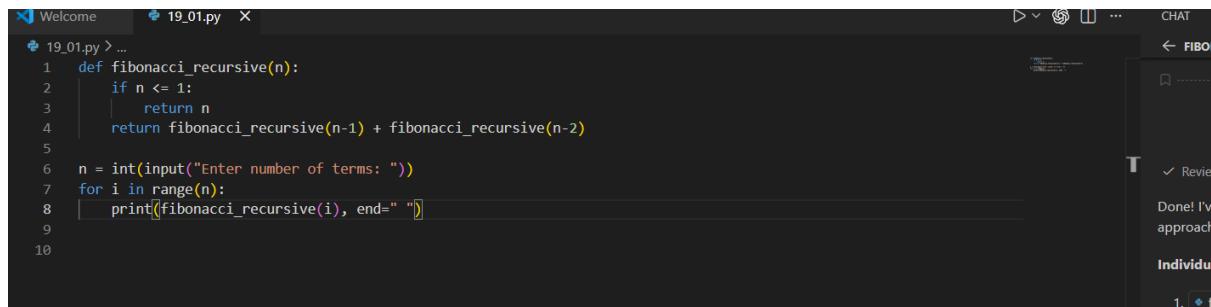
```
PS D:\AI_assistant_coding> & 'c:\Users\yaraV\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\yaraV\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '53001' '--' 'd:\AI_assistant_coding\19_01.py'
Enter the number of terms: 5
0 1 1 2 3
PS D:\AI_assistant_coding>
```

## Task 5: Iterative vs Recursive Fibonacci



```
19_01.py > fibonacci_iterative
1 def fibonacci_iterative(n):
2     a, b = 0, 1
3     for _ in range(n):
4         print(a, end=" ")
5         a, b = b, a + b
6
7
```

## Recursive Fibonacci



```
Welcome 19_01.py ...
19_01.py > ...
1 def fibonacci_recursive(n):
2     if n <= 1:
3         return n
4     return fibonacci_recursive(n-1) + fibonacci_recursive(n-2)
5
6 n = int(input("Enter number of terms: "))
7 for i in range(n):
8     print(fibonacci_recursive(i), end=" ")
9
10
```

## Conclusion

This lab demonstrated how GitHub Copilot supports AI-assisted coding by generating, optimizing, and refactoring Python programs in Visual Studio Code. It showed that Copilot can improve coding speed and help explore different programming approaches, but human judgment is still essential to ensure correctness, efficiency, and code quality. Overall, the assignment highlighted the effective use of AI as a supportive tool rather than a replacement for good programming practices.