

Assignment-1.3

Name: S.Vyshnavi

Hall ticket No: 2303A51920

Batch: 07

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

```
# Program to print Fibonacci sequence up to n terms
n = int(input("Enter the number of terms: "))
# Initialize first two terms
a, b = 0, 1
print(f"Fibonacci sequence up to {n} terms:")
# Generate and print Fibonacci sequence
for i in range(n):
    print(a, end=" ")
    c = a + b
    a = b
    b = c
print() # New line at the end
```

The right side of the screen shows an AI-generated explanatory sidebar titled "FIBONACCI SEQUENCE PROGRAM IN PY...". It includes bullet points about user input, no user-defined functions, generating the sequence, and printing it. Below that is a "How it works:" section with more detailed steps. At the bottom, there's an "Example output" section showing the program's output for 7 terms: 0 1 1 2 3 5 8.

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' '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
PS D:\AI_assistant_coding>
```

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. A tooltip suggests optimizing the code by simplifying logic and variable usage. It lists two improvements: eliminating intermediate variables and simplifying loop variables.

```

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 Python Debug Console output shows the execution of the script. It prompts for the number of terms and prints the sequence 0 1 1 2 3.

```

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' '51176' '--' 'd:\AI_assistant_coding\19_01.py'
Enter the number of terms: 5
0 1 1 2 3
PS D:\AI_assistant_coding>

```

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 refactored code using a function-based approach. The code includes a detailed docstring and inline comments explaining the logic. A tooltip provides key improvements: encapsulating logic in a function, adding meaningful comments, performing input validation, and using proper structure.

```

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:

PROBLEMS TERMINAL OUTPUT DEBUG CONSOLE PORTS GITLENS POSTMAN CONSOLE

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 X
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.