

## Assignment-1.3

**Name:** Y. Poojitha

**Hall ticket No:** 2303A51499

**Batch:** 08

**Lab 1:**

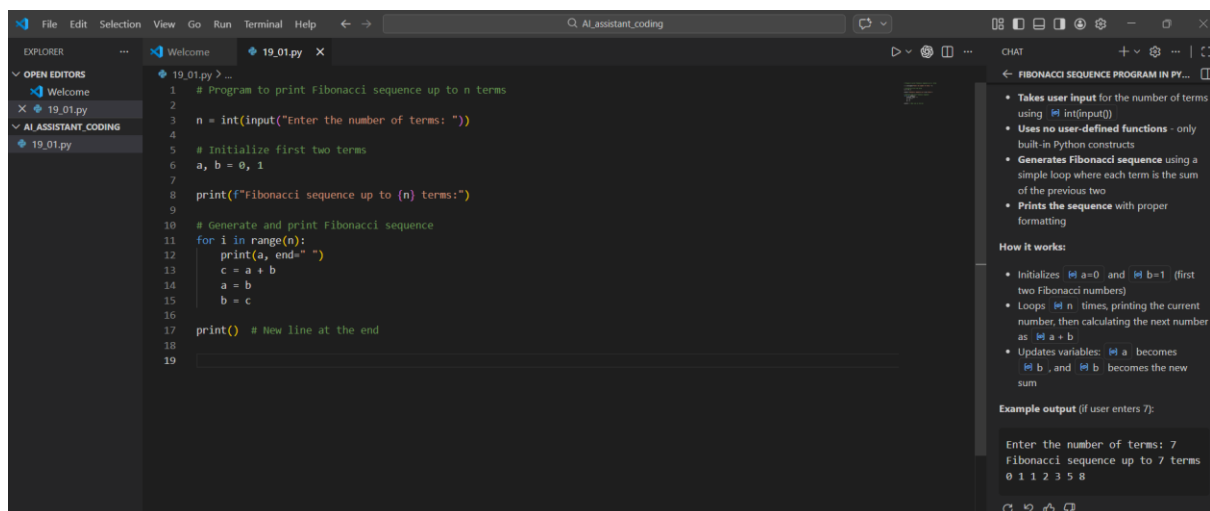
**Task 1:** AI-Generated Logic Without Modularization Fibonacci Sequence Without Functions

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 editor with a file named `19_01.py`. The code is as follows:

```
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
```

The chat window on the right provides the following explanation:

- 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

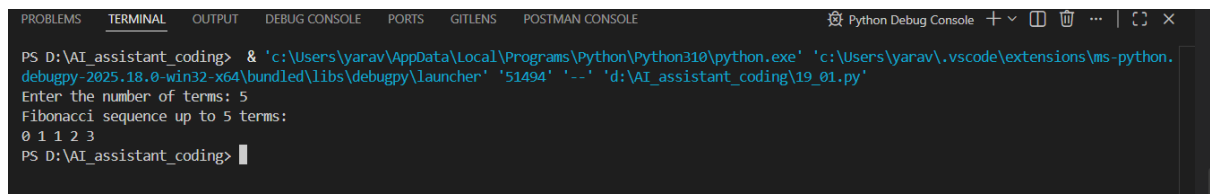
**How it works:**

- Initializes `a=0` and `b=1` (first two Fibonacci numbers)
- Loops `n` times, printing the current number, then calculating the next number as `a + b`
- Updates variables: `a` becomes `b`, and `b` becomes the new sum

**Example output (if user enters 7):**

```
Enter the number of terms: 7
Fibonacci sequence up to 7 terms
0 1 1 2 3 5 8
```

**Result:**



The terminal window shows the following output:

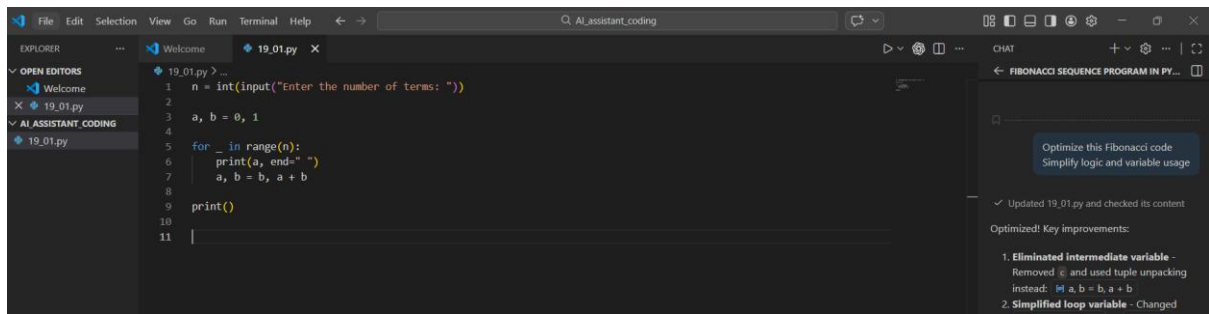
```
PS D:\AI_assistant_coding> & 'c:\Users\yarav\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\yarav\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher' '51494' '--' 'd:\AI_assistant_coding\19_01.py'
Enter the number of terms: 5
Fibonacci sequence up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



```
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
```

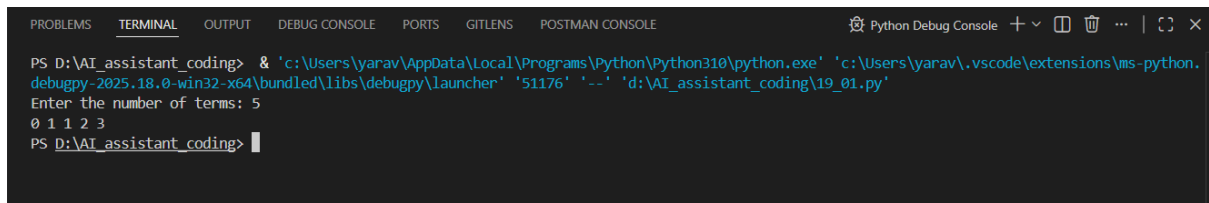
Optimize this Fibonacci code  
Simplify logic and variable usage

✓ Updated 19\_01.py and checked its content

Optimized! Key improvements:

1. **Eliminated intermediate variable** - Removed `c` and used tuple unpacking instead: `a, b = b, a + b`
2. **Simplified loop variable** - Changed

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

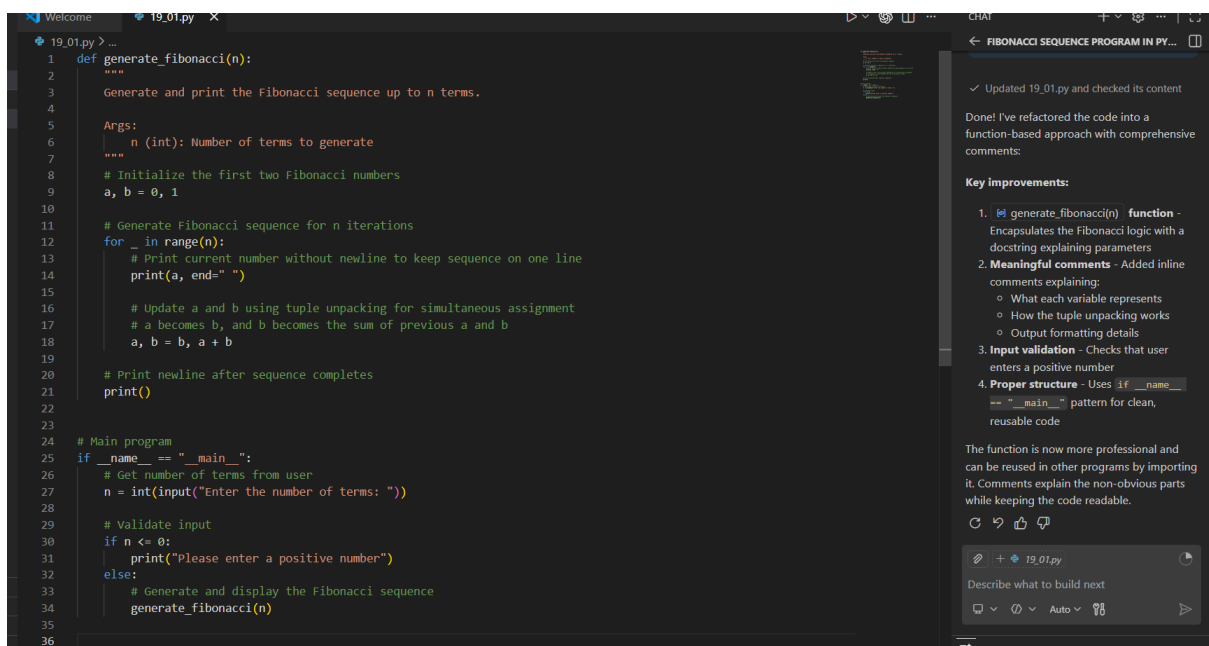
### Task 3: Modular Design Using AI Assistance Fibonacci Using Functions

#### Copilot Prompt Used

Create a function to generate Fibonacci sequence up to n

Add meaningful comments

Function-Based Code



```
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 # Main program
24 if __name__ == "__main__":
25     # Get number of terms from user
26     n = int(input("Enter the number of terms: "))
27
28     # Validate input
29     if n <= 0:
30         print("Please enter a positive number")
31     else:
32         # Generate and display the Fibonacci sequence
33         generate_fibonacci(n)
34
35
36
```

Updated 19\_01.py and checked its content

Done! I've refactored the code into a function-based approach with comprehensive comments.

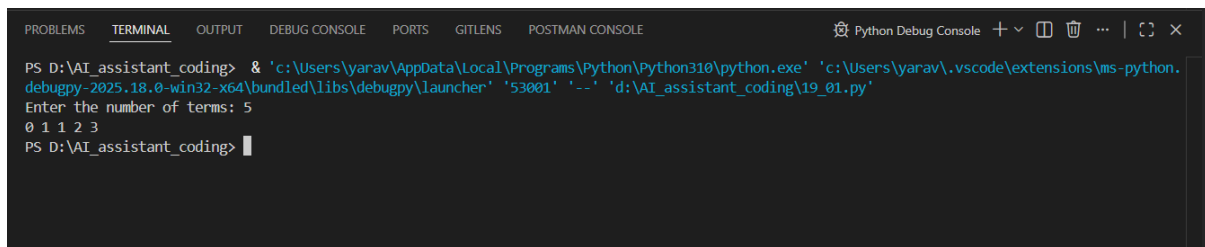
**Key improvements:**

1. **generate\_fibonacci(n) function** - Encapsulates the Fibonacci logic with a docstring explaining parameters
2. **Meaningful comments** - Added inline comments explaining:
  - What each variable represents
  - How the tuple unpacking works
  - Output formatting details
3. **Input validation** - Checks that user enters a positive number
4. **Proper structure** - Uses `if __name__ == "__main__":` pattern for clean, reusable code

The function is now more professional and can be reused in other programs by importing it. Comments explain the non-obvious parts while keeping the code readable.

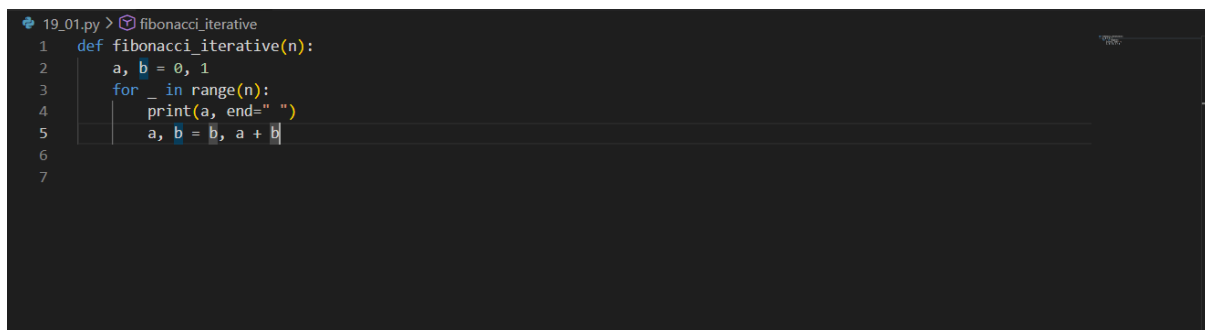
Describe what to build next

Result:



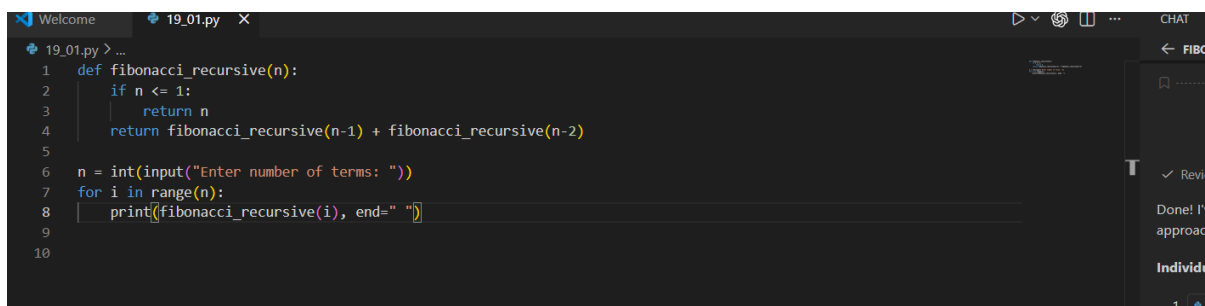
```
PS D:\AI_assistant_coding> & 'c:\Users\yarav\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\yarav\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundle\libs\debugpy\launcher' '53001' '--' 'd:\AI_assistant_coding\19_01.py'
Enter the number of terms: 5
0 1 1 2 3
PS D:\AI_assistant_coding>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

## 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



```
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.