Assignment-1.3

Name: B.Shravya

Hall ticket No: 2303A51492

Batch: 08

Lab 1:

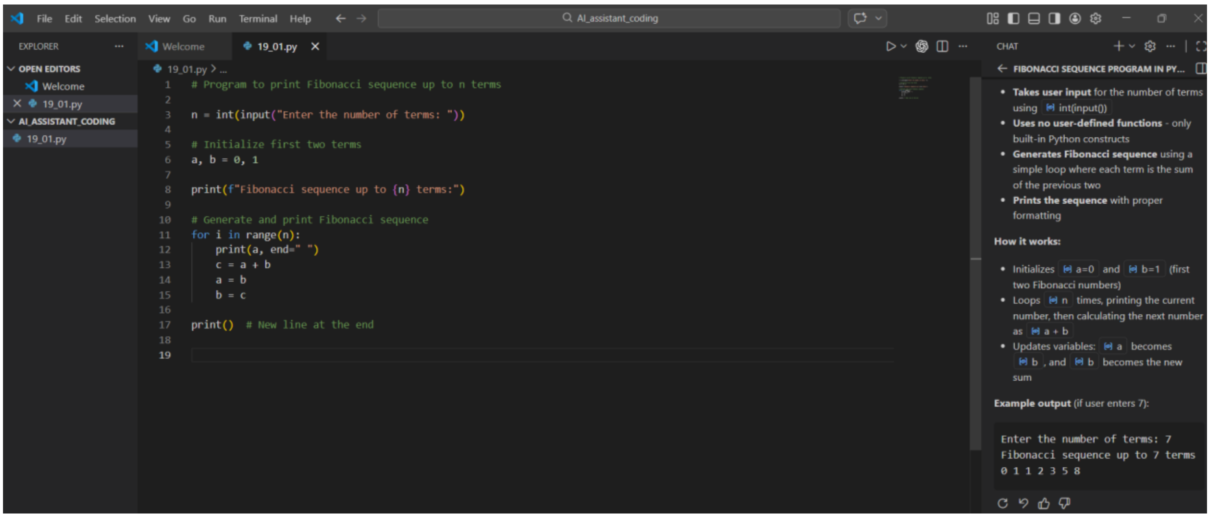
Task 1: AI-Generated Logic Without Modularization Fibonacci Sequence Without Functions

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



```python
# 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

```

- **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

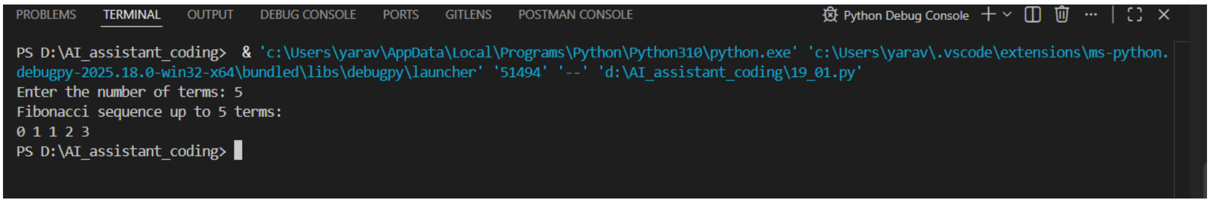
How it works:

- 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:



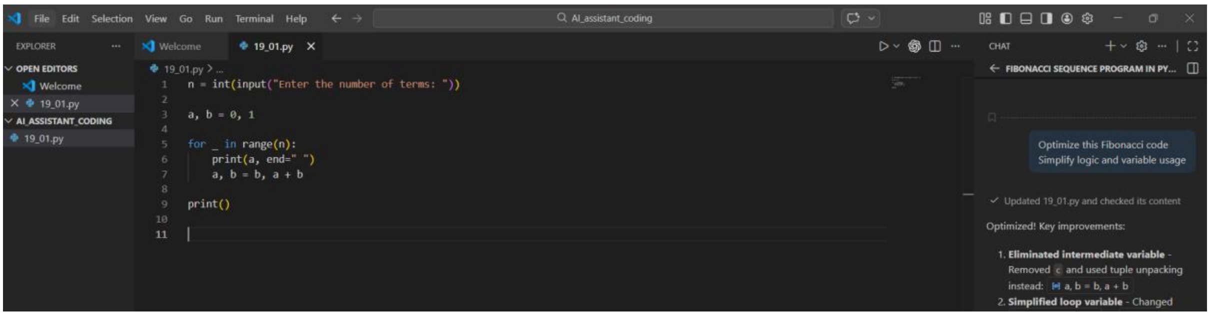
```
PS D:\AI_assistant_coding>  & 'c:\Users\yarav\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\yarav\.vscode\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '51494' '--' 'd:\AI_assistant_coding\19_01.py'
Enter the number of terms: 5
Fibonacci sequence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



```python
n = int(input("Enter the number of terms: "))

a, b = 0, 1

for _ in range(n):
    print(a, end=" ")
    a, b = b, a + b

print()

```

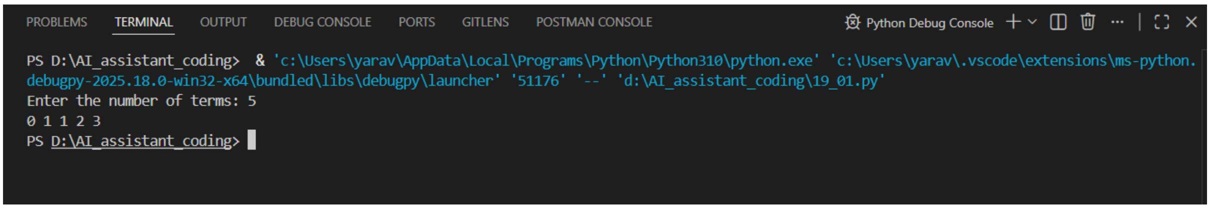
Optimize this Fibonacci code
Simplify logic and variable usage

✓ 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> 
```

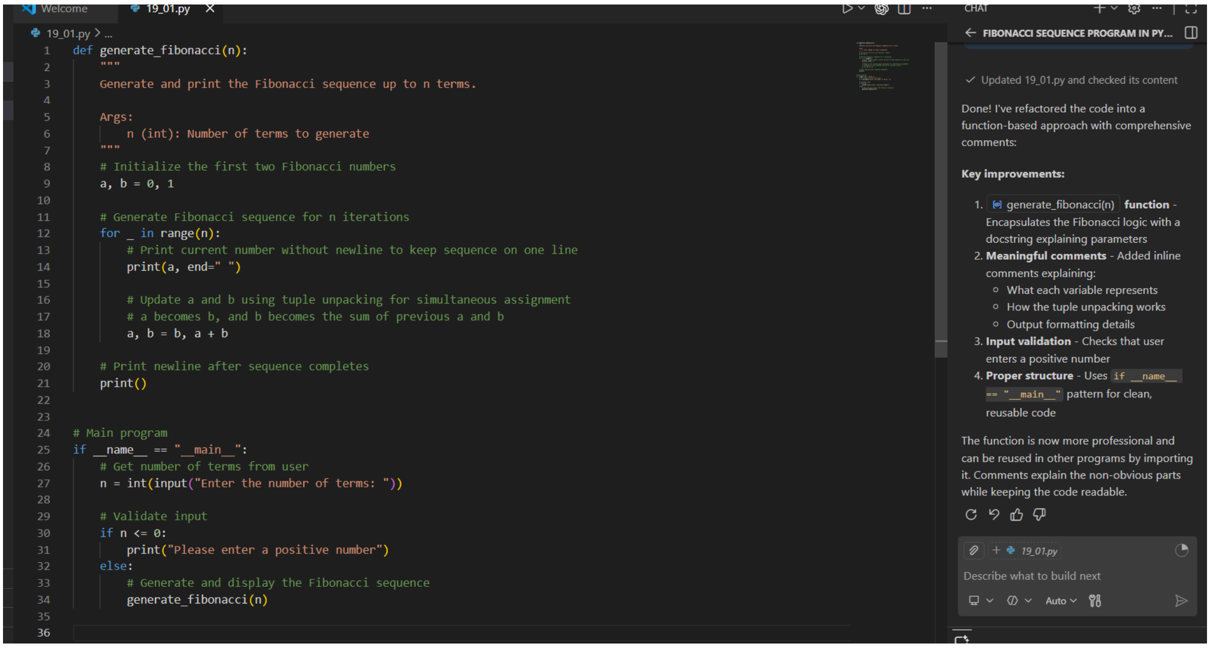
Task 3: Modular Design Using AI Assistance Fibonacci Using Functions

Copilot Prompt Used

Create a function to generate Fibonacci sequence up to n

Add meaningful comments

Function-Based Code



```python
def generate_fibonacci(n):
    """
    Generate and print the Fibonacci sequence up to n terms.
    
    Args:
        n (int): Number of terms to generate
    """
    # Initialize the first two Fibonacci numbers
    a, b = 0, 1
    
    # Generate Fibonacci sequence for n iterations
    for _ in range(n):
        # Print current number without newline to keep sequence on one line
        print(a, end=" ")
        
        # Update a and b using tuple unpacking for simultaneous assignment
        # a becomes b, and b becomes the sum of previous a and b
        a, b = b, a + b
    
    # Print newline after sequence completes
    print()


# Main program
if __name__ == "__main__":
    # Get number of terms from user
    n = int(input("Enter the number of terms: "))
    
    # Validate input
    if n <= 0:
        print("Please enter a positive number")
    else:
        # Generate and display the Fibonacci sequence
        generate_fibonacci(n)

```

✓ Updated 19_01.py and checked its content

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

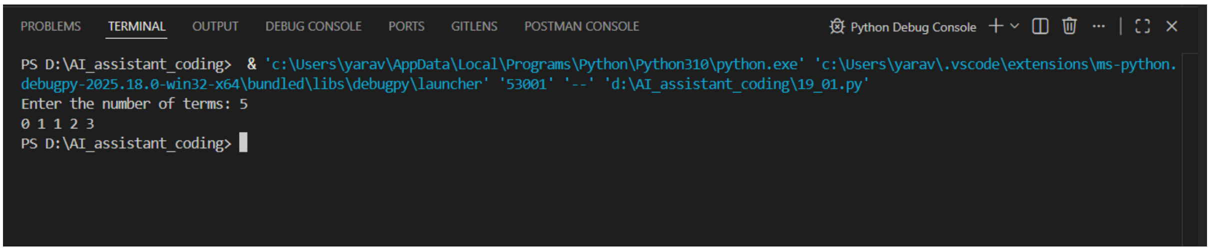
Key improvements:

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.

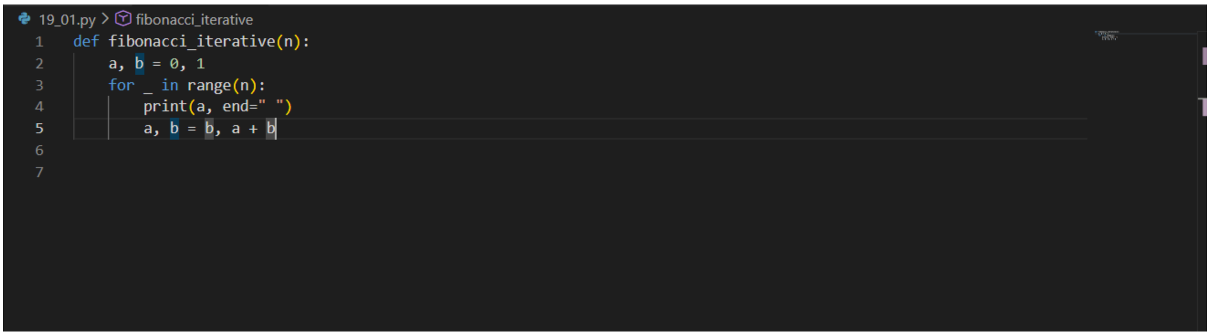
Welcome
19_01.py
CHAT
FIBONACCI SEQUENCE PROGRAM IN PY...
19_01.py
Describe what to build next
Auto

Result:



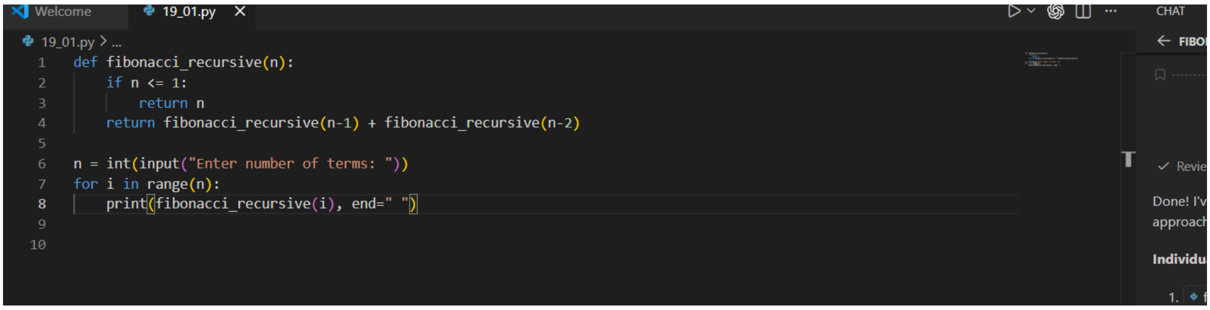
```
PS D:\AI_assistant_coding>  & 'c:\Users\yarav\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\yarav\.vscode\extensions\ms-python.
debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '53001' '--' 'd:\AI_assistant_coding\19_01.py'
Enter the number of terms: 5
0 1 1 2 3
PS D:\AI_assistant_coding>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

Task 5: Iterative vs Recursive Fibonacci



```python
def fibonacci_iterative(n):
    a, b = 0, 1
    for _ in range(n):
        print(a, end=" ")
        a, b = b, a + b
```

Recursive Fibonacci



```python
def fibonacci_recursive(n):
    if n <= 1:
        return n
    return fibonacci_recursive(n-1) + fibonacci_recursive(n-2)

n = int(input("Enter number of terms: "))
for i in range(n):
    print(fibonacci_recursive(i), end=" ")
```

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.