AI ASSISTED CODING

ASSIGNMENT-1

B. Sri Laxmi Gayathri

2303a52033

Batch:38

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

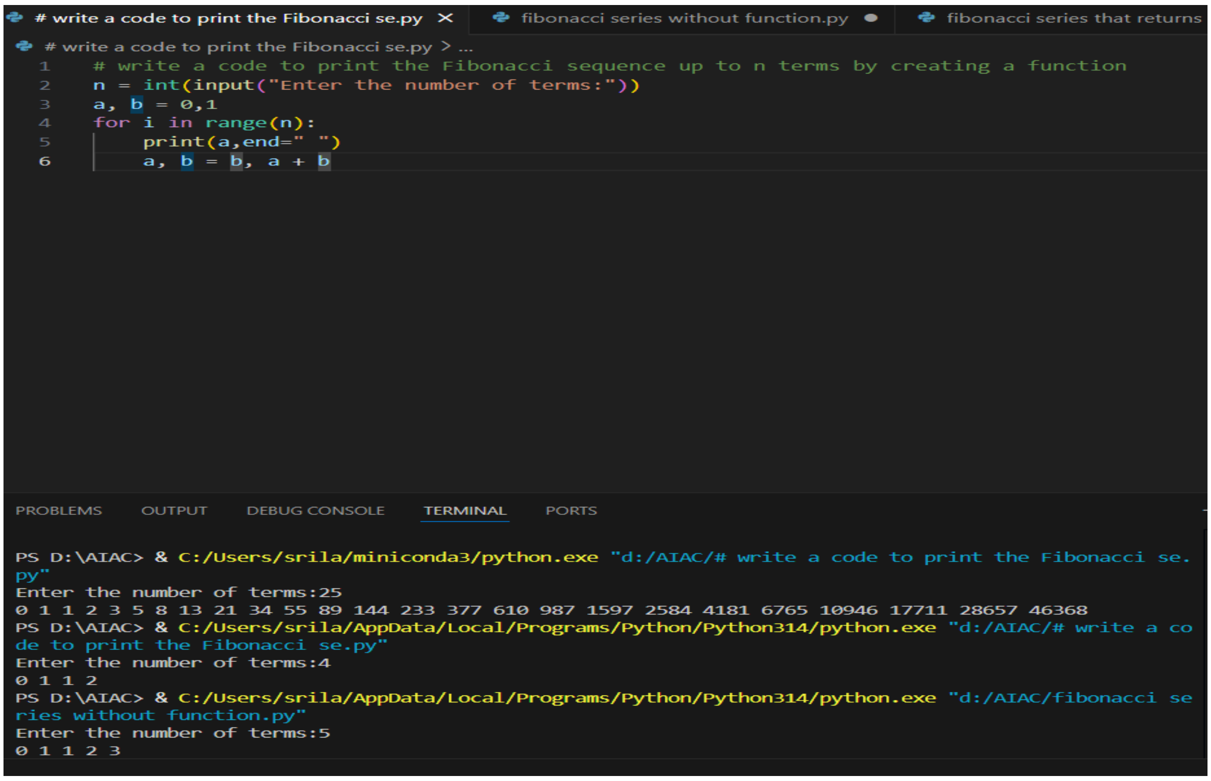
Use GitHub Copilot to generate a Python program that:

- ➢ Prints the Fibonacci sequence up to *n* terms
- ➢ Accepts user input for *n*
- ➢ Implements the logic directly in the main code ➢ Does not use any user- defined functions

Prompt:

write a code to print fibonacci series upto n without using a function

Code & Output:


# write a code to print the Fibonacci se.py
fibonacci series without function.py
fibonacci series that returns
# write a code to print the Fibonacci se.py > ...
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS

```python
# write a code to print the Fibonacci sequence up to n terms by creating a function
n = int(input("Enter the number of terms:"))
a, b = 0,1
for i in range(n):
    print(a,end=" ")
    a, b = b, a + b
```

```
PS D:\AIAC> & C:/Users/srila/miniconda3/python.exe "d:/AIAC/# write a code to print the Fibonacci se.
py"
Enter the number of terms:25
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368
PS D:\AIAC> & C:/Users/srila/AppData/Local/Programs/Python/Python314/python.exe "d:/AIAC/# write a co
de to print the Fibonacci se.py"
Enter the number of terms:4
0 1 1 2
PS D:\AIAC> & C:/Users/srila/AppData/Local/Programs/Python/Python314/python.exe "d:/AIAC/fibonacci se
ries without function.py"
Enter the number of terms:5
0 1 1 2 3
```

Explanation:

This program prints the Fibonacci series.The user enters a number n. The program starts with 0 and 1. In each loop, it prints the current number and adds the previous two numbers to get the next one. This repeats n times and prints the Fibonacci series.

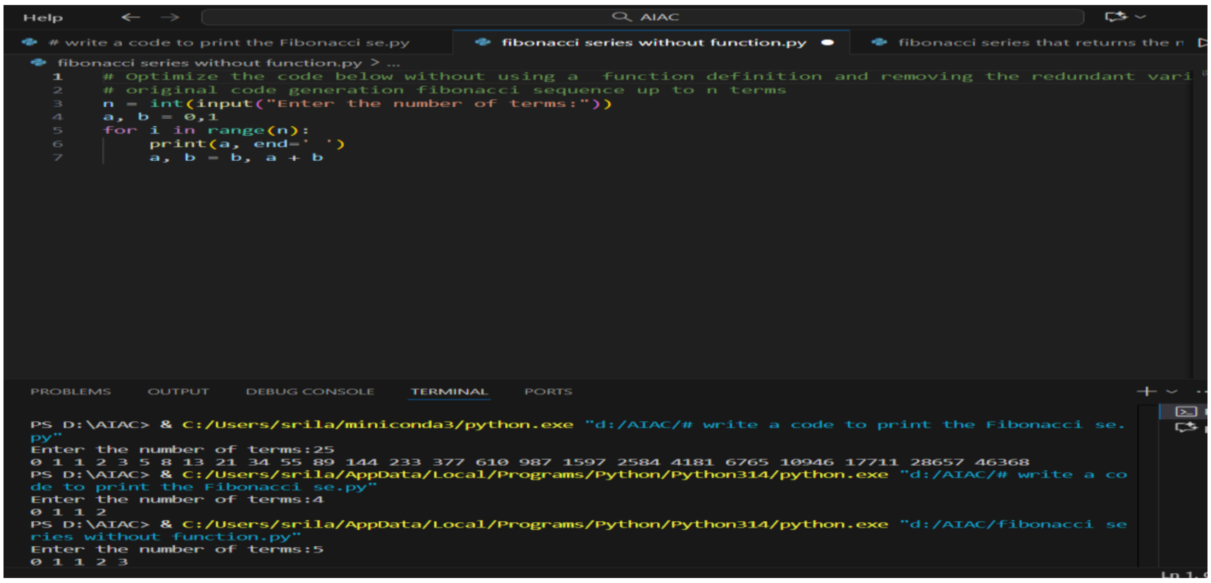
Task 2: AI Code Optimization & Cleanup (Improving Efficiency)

- **Examine the Copilot-generated code from Task 1 and improve it by:**
- **Removing redundant variables**
- **Simplifying loop logic**
- **Avoiding unnecessary computations ➢ Use Copilot prompts such as: ▪ "Optimize this Fibonacci code"**
- **"Simplify variable usage"**

Prompt:

Optimize the above code by removing unnecessary variables, simplifying the loop, and improving readability.

Code & Output:



```python
# Optimize the code below without using a  function definition and removing the redundant vari
# original code generation fibonacci sequence up to n terms
n = int(input("Enter the number of terms:"))
a, b = 0,1
for i in range(n):
    print(a, end=' ')
    a, b = b, a + b
```

```
PS D:\AIAC> & C:/Users/srila/miniconda3/python.exe "d:/AIAC/# write a code to print the Fibonacci se.
py"
Enter the number of terms:25
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368
PS D:\AIAC> & C:/Users/srila/AppData/Local/Programs/Python/Python314/python.exe "d:/AIAC/# write a co
de to print the Fibonacci se.py"
Enter the number of terms:4
0 1 1 2
PS D:\AIAC> & C:/Users/srila/AppData/Local/Programs/Python/Python314/python.exe "d:/AIAC/fibonacci se
ries without function.py"
Enter the number of terms:5
0 1 1 2 3
```

Help
AIAC
# write a code to print the Fibonacci se.py
fibonacci series without function.py
fibonacci series that returns the n
fibonacci series without function.py > ...
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS

Explanation:

This program prints the Fibonacci series up to a given number of terms.The user enters a number n. If the number is not valid, the program shows an error message. Otherwise, it starts with 0 and 1 and prints the Fibonacci numbers using a loop. In each step, the next number is found by adding the previous two numbers. The series is printed until n terms are displayed.

Task 3: Modular Design Using AI Assistance (Fibonacci Using Functions) Use GitHub Copilot to generate a function-based Python program that:

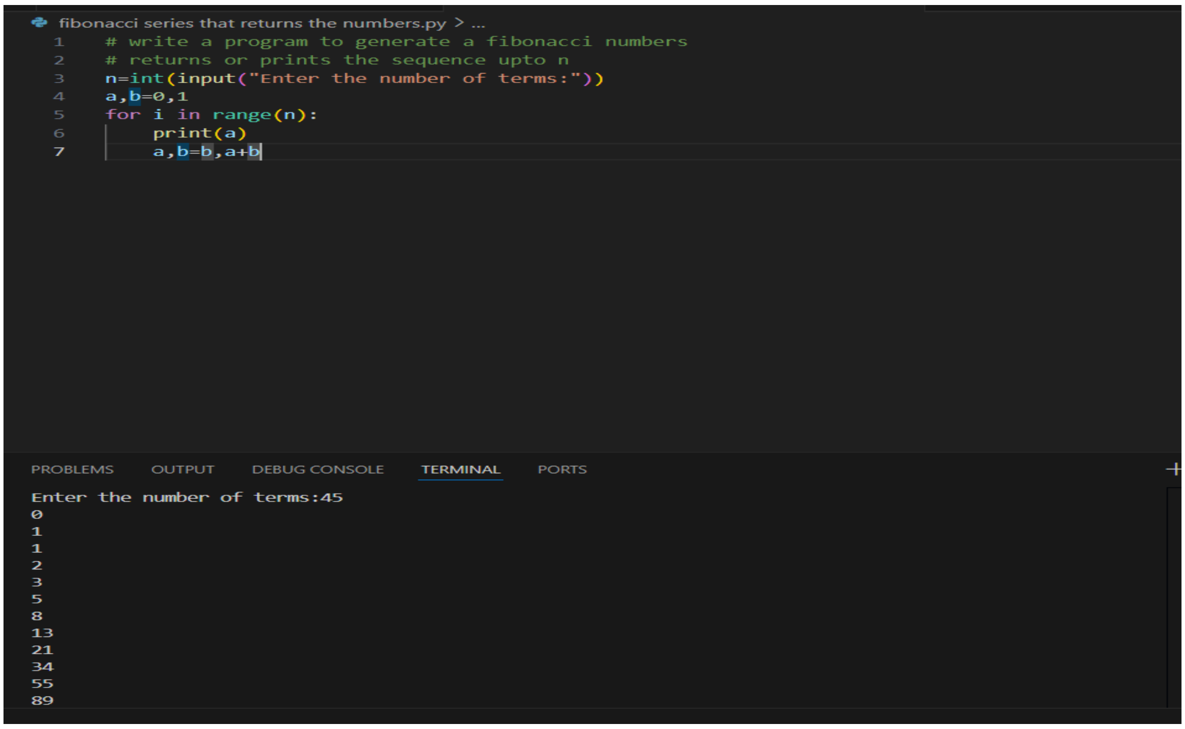
- **Uses a user-defined function to generate Fibonacci numbers**

- **Returns or prints the sequence up to n**
- **Includes meaningful comments (AI-assisted)**

Prompt:

Write a code for printing the Fibonacci series up to n terms using a function and include meaningful comments.

Code & Output:



```python
# write a program to generate a fibonacci numbers
# returns or prints the sequence upto n
n=int(input("Enter the number of terms:"))
a,b=0,1
for i in range(n):
    print(a)
    a,b=b,a+b
```

```
Enter the number of terms:45
0
1
1
2
3
5
8
13
21
34
55
89
```

Explanation:

This program prints the Fibonacci series using a function.

The user enters a number n. The function print_fibonacci checks whether the number is valid. If it is valid, the program starts with 0 and 1 and prints Fibonacci numbers by adding the previous two numbers each time. The series is printed up to n terms.

Task 4: Comparative Analysis – Procedural vs Modular Fibonacci Code

Compare the Copilot-generated Fibonacci programs:

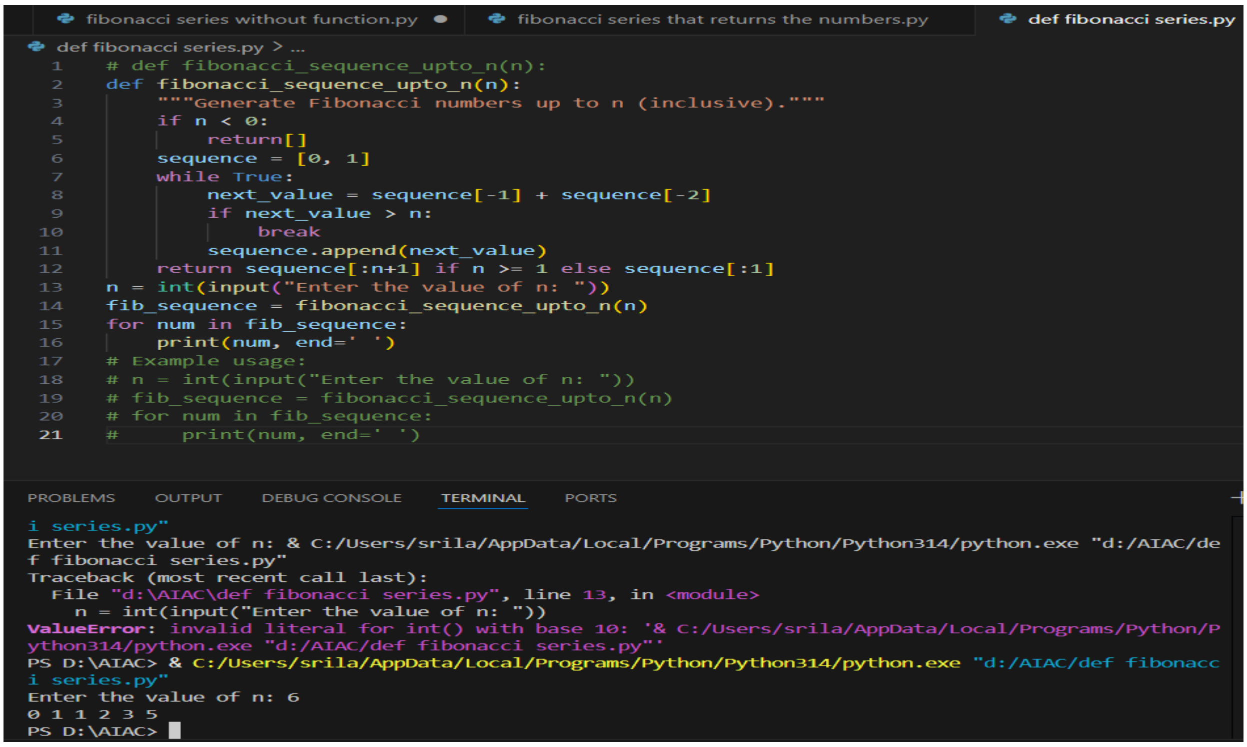
- **Without functions (Task 1)**
- **With functions (Task 3)**
- **Analyze them in terms of:**
  - **Code clarity**
  - **Reusability**
  - **Debugging ease**
  - **Suitability for larger systems**

Prompt:

Non-Modular: Write a code for printing a Fibonacci series up to n terms without using a function.

Modular: Write a code for printing the Fibonacci series up to n terms using a function and include meaningful comments.

Code & Output:


fibonacci series without function.py
fibonacci series that returns the numbers.py
def fibonacci series.py
def fibonacci series.py > ...

```python
# def fibonacci_sequence_upto_n(n):
def fibonacci_sequence_upto_n(n):
    """Generate Fibonacci numbers up to n (inclusive)."""
    if n < 0:
        return[]
    sequence = [0, 1]
    while True:
        next_value = sequence[-1] + sequence[-2]
        if next_value > n:
            break
        sequence.append(next_value)
    return sequence[:n+1] if n >= 1 else sequence[:1]
n = int(input("Enter the value of n: "))
fib_sequence = fibonacci_sequence_upto_n(n)
for num in fib_sequence:
    print(num, end=' ')
# Example usage:
# n = int(input("Enter the value of n: "))
# fib_sequence = fibonacci_sequence_upto_n(n)
# for num in fib_sequence:
#     print(num, end=' ')
```

```
i series.py"
Enter the value of n: & C:/Users/srila/AppData/Local/Programs/Python/Python314/python.exe "d:/AIAC/de
f fibonacci series.py"
Traceback (most recent call last):
  File "d:\AIAC\def fibonacci series.py", line 13, in <module>
    n = int(input("Enter the value of n: "))
ValueError: invalid literal for int() with base 10: '& C:/Users/srila/AppData/Local/Programs/Python/P
ython314/python.exe "d:/AIAC/def fibonacci series.py"'
PS D:\AIAC> & C:/Users/srila/AppData/Local/Programs/Python/Python314/python.exe "d:/AIAC/def fibonacc
i series.py"
Enter the value of n: 6
0 1 1 2 3 5
PS D:\AIAC>
```

Explanation:

In the **procedural approach**, the Fibonacci series code is written directly in the main program without using any function. The steps run one after another, and this method is simple but not reusable.In the **modular approach**, the Fibonacci logic is written inside a function. The main program only calls the function. This makes the code more organized, easy to understand, and reusable.

Task 5 : AI-Generated Iterative vs Recursive Fibonacci Approaches (Different Algorithmic Approaches for Fibonacci Series)

Prompt GitHub Copilot to generate:

An iterative Fibonacci implementation

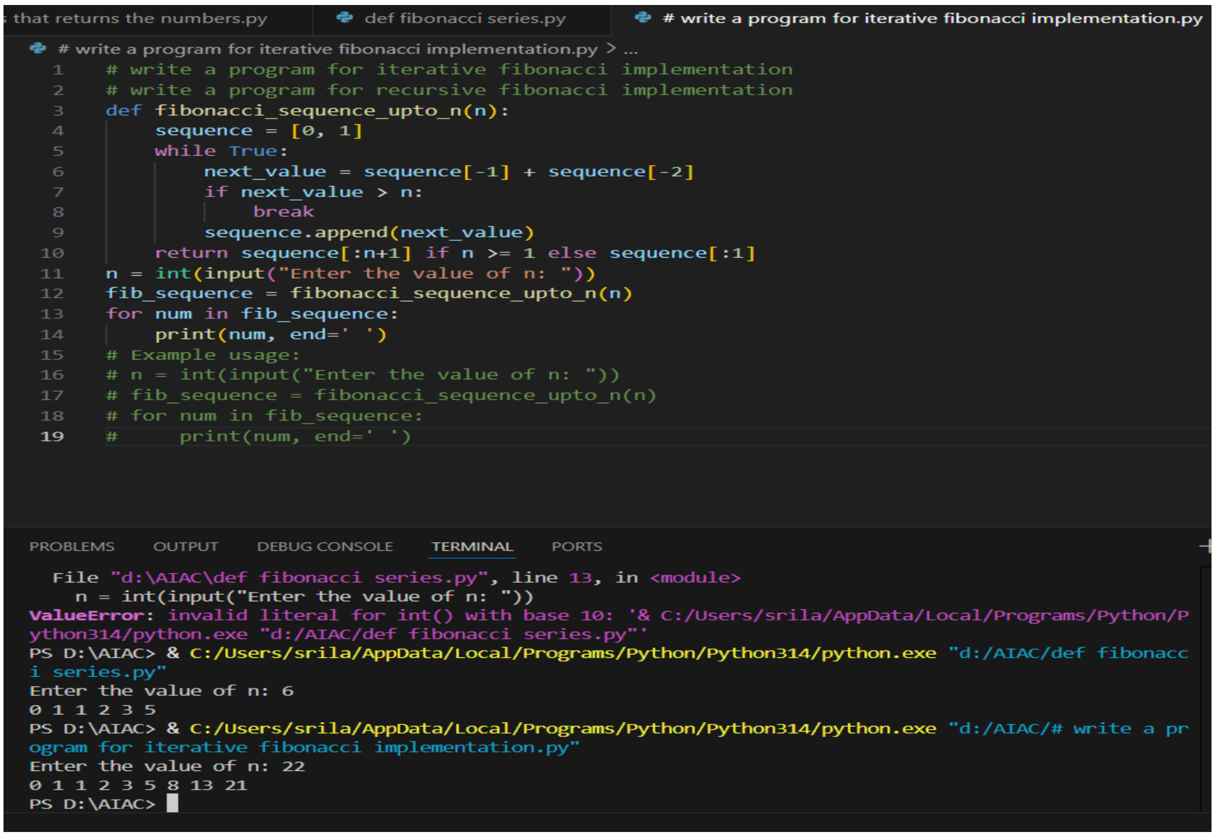
A recursive Fibonacci implementation

Prompt:

Write a code for printing a Fibonacci series up to n terms without using a function.

Write a code for printing the Fibonacci series up to n terms using recursion.

Code & Output :


that returns the numbers.py
def fibonacci series.py
# write a program for iterative fibonacci implementation.py
# write a program for iterative fibonacci implementation.py > ...
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS

```python
# write a program for iterative fibonacci implementation
# write a program for recursive fibonacci implementation
def fibonacci_sequence_upto_n(n):
    sequence = [0, 1]
    while True:
        next_value = sequence[-1] + sequence[-2]
        if next_value > n:
            break
        sequence.append(next_value)
    return sequence[:n+1] if n >= 1 else sequence[:1]
n = int(input("Enter the value of n: "))
fib_sequence = fibonacci_sequence_upto_n(n)
for num in fib_sequence:
    print(num, end=' ')
# Example usage:
# n = int(input("Enter the value of n: "))
# fib_sequence = fibonacci_sequence_upto_n(n)
# for num in fib_sequence:
#     print(num, end=' ')
```

```
  File "d:\AIAC\def fibonacci series.py", line 13, in <module>
    n = int(input("Enter the value of n: "))
ValueError: invalid literal for int() with base 10: '& C:/Users/srila/AppData/Local/Programs/Python/P
ython314/python.exe "d:/AIAC/def fibonacci series.py"'
PS D:\AIAC> & C:/Users/srila/AppData/Local/Programs/Python/Python314/python.exe "d:/AIAC/def fibonacc
i series.py"
Enter the value of n: 6
0 1 1 2 3 5
PS D:\AIAC> & C:/Users/srila/AppData/Local/Programs/Python/Python314/python.exe "d:/AIAC/# write a pr
ogram for iterative fibonacci implementation.py"
Enter the value of n: 22
0 1 1 2 3 5 8 13 21
PS D:\AIAC> 
```

Explanation:

This program prints the Fibonacci series in two ways .In the first part, the Fibonacci series is printed **without using a function**. The user enters the number of terms, and the program uses a loop to print the series by adding the previous two numbers each time.In the second part, the Fibonacci series is printed **using recursion**. A function is defined that prints one Fibonacci number and then calls itself to print the next one. This process continues until the required number of terms is printed. Both methods produce the same Fibonacci series but use different approaches.