AI ASSITANT CODING-1

Name: Siri Vempati

HT.NO:2303A51309

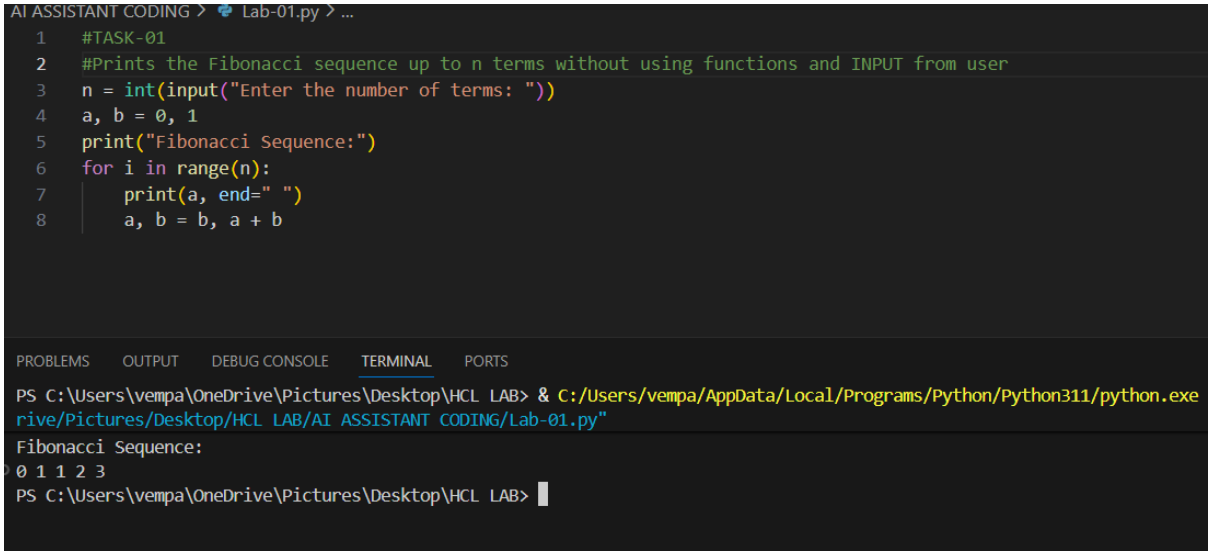
BATCH NO:05

TASK-01:

Prompt: Prints the Fibonacci sequence up to n terms without using functions and INPUT from user

Code:

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

a, b = 0, 1

print("Fibonacci Sequence:")

for i in range(n):

    print(a, end=" ")

    a, b = b, a + b
```



Explanation:

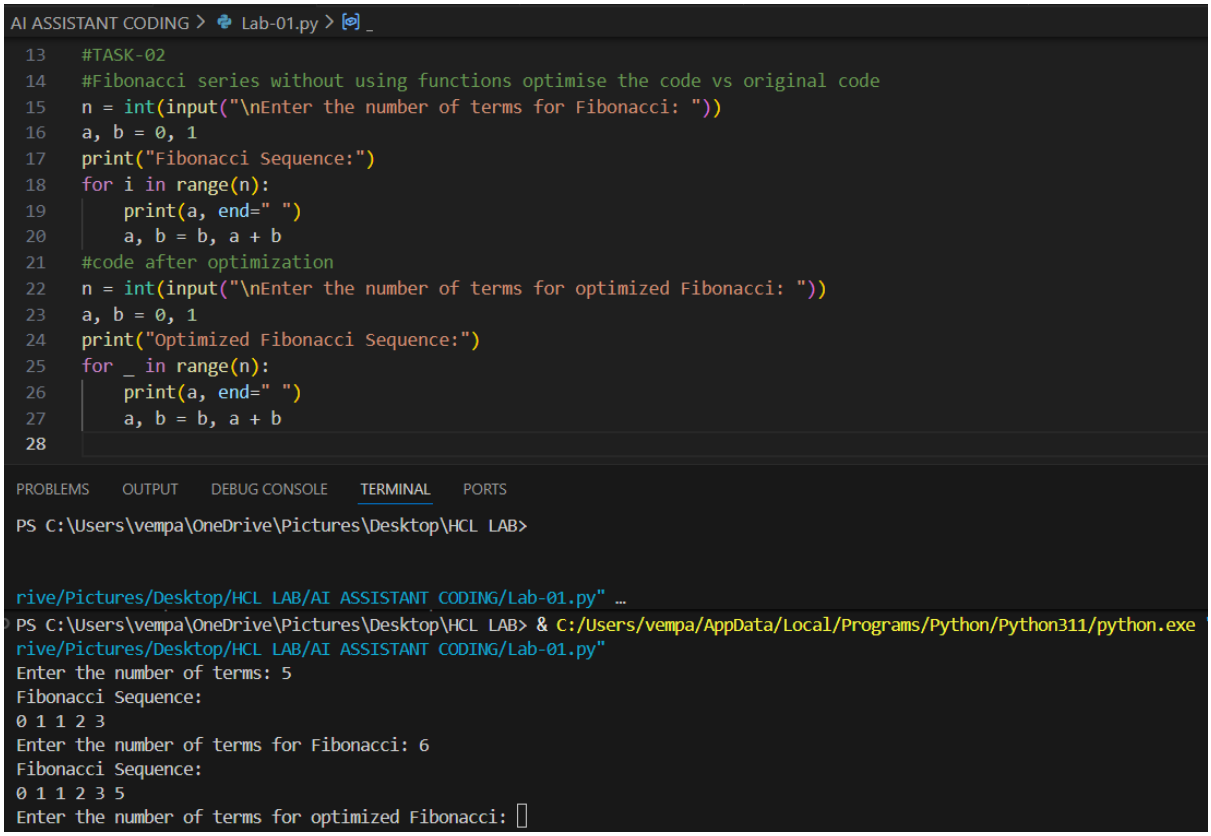
This code generates the Fibonacci sequence up to 'n' terms. It initializes the first two terms (0 and 1) and iteratively computes the next terms by updating the values of 'a' and 'b' in each iteration of the loop.

TASK-02

Prompt: Fibonacci series without using functions optimise the code vs original code

Code:

```
n = int(input("\nEnter the number of terms for Fibonacci: "))
a, b = 0, 1
print("Fibonacci Sequence:")
for i in range(n):
    print(a, end=" ")
    a, b = b, a + b
#code after optimization
n = int(input("\nEnter the number of terms for optimized Fibonacci: "))
a, b = 0, 1
print("Optimized Fibonacci Sequence:")
for _ in range(n):
    print(a, end=" ")
    a, b = b, a + b
```



Explanation:

The original code was already quite efficient for generating the Fibonacci sequence.

However, minor improvements were made by removing the loop variable 'i' which was unused, replacing it with an underscore '_' to indicate that it's intentionally ignored.

This enhances readability by signal that the loop variable is not needed.

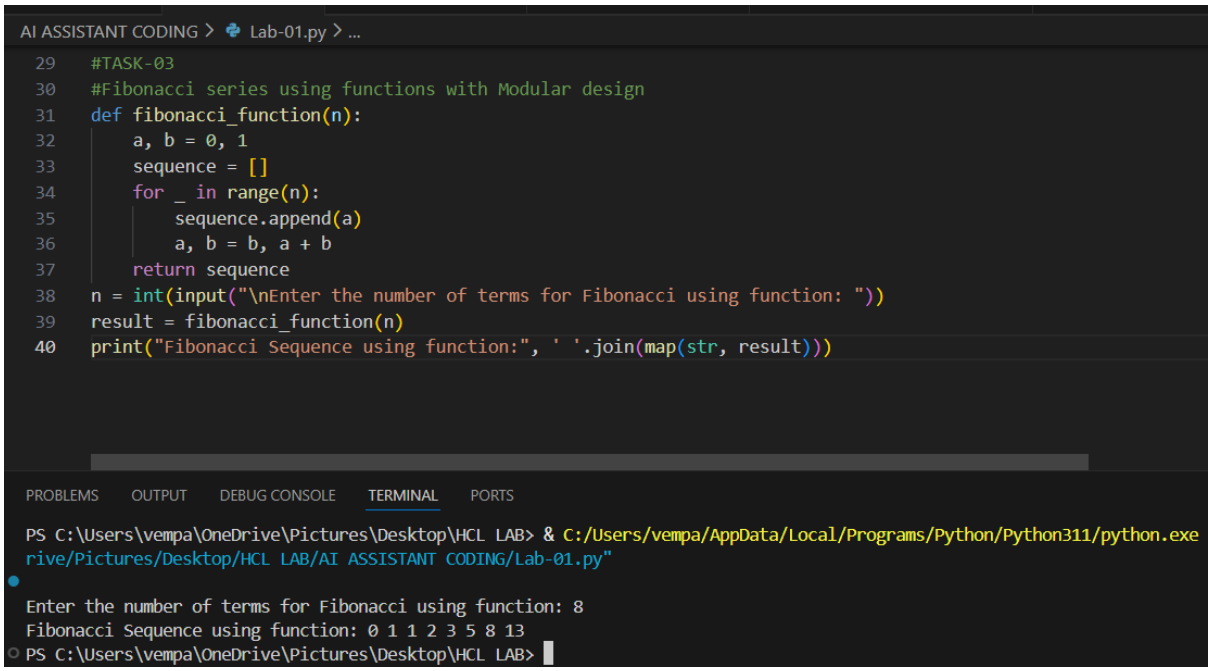
TASK-03:

Prompt: Fibonacci series using functions with Modular design

Code:

```
def fibonacci_function(n):
    a, b = 0, 1
    sequence = []
    for _ in range(n):
```

```
        sequence.append(a)
        a, b = b, a + b
    return sequence
n = int(input("\nEnter the number of terms for Fibonacci using function: "))
result = fibonacci_function(n)
print("Fibonacci Sequence using function:", ' '.join(map(str, result)))
```



Explanation:

This code defines a function 'fibonacci_function' that generates the Fibonacci sequence up to 'n' terms.

It uses a list to store the sequence and returns it after computation.

TASK-04

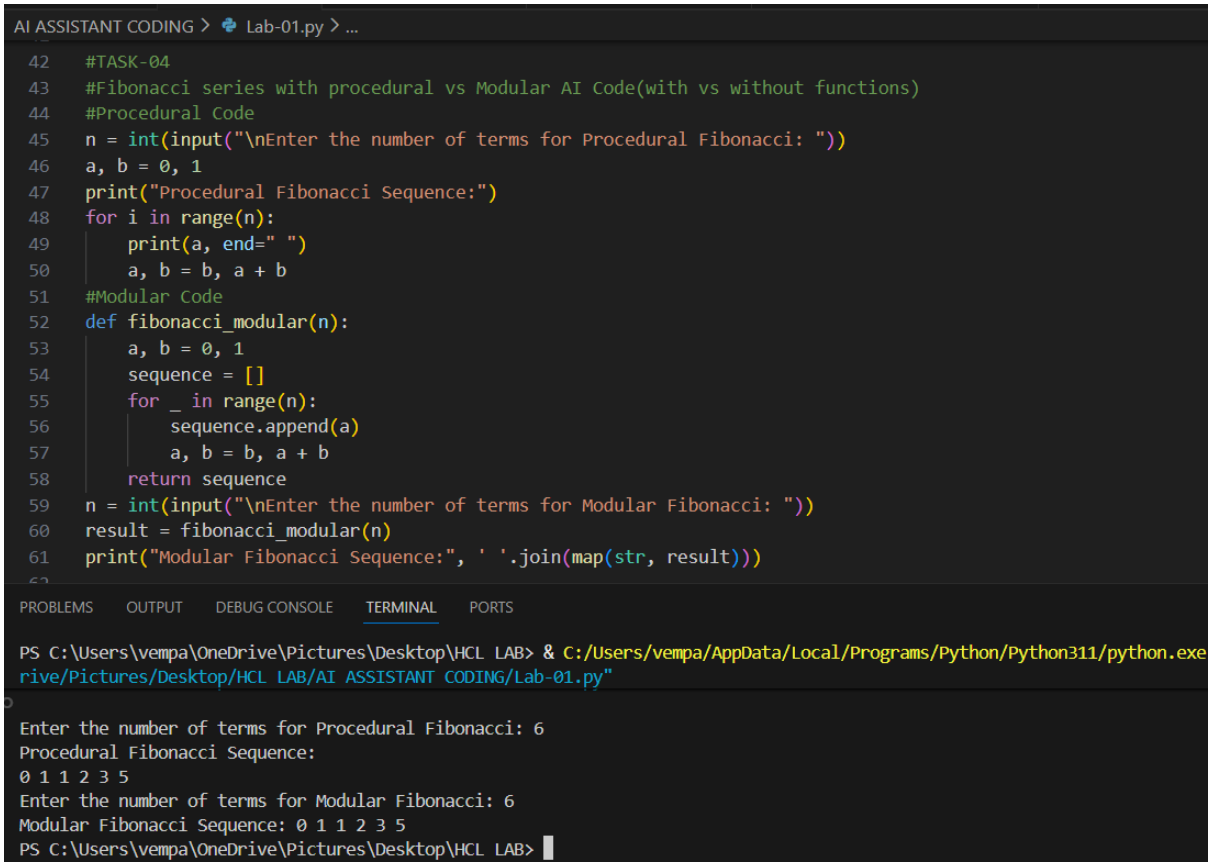
Prompt: Fibonacci series with procedural vs Modular AI Code(with vs without functions)

Code:

#Procedural Code

n = int(input("\nEnter the number of terms for Procedural Fibonacci: "))

```
a, b = 0, 1
print("Procedural Fibonacci Sequence:")
for i in range(n):
    print(a, end=" ")
    a, b = b, a + b
#Modular Code
def fibonacci_modular(n):
    a, b = 0, 1
    sequence = []
    for _ in range(n):
        sequence.append(a)
        a, b = b, a + b
    return sequence
n = int(input("\nEnter the number of terms for Modular Fibonacci: "))
result = fibonacci_modular(n)
print("Modular Fibonacci Sequence:", ' '.join(map(str, result)))
```



Explanation:

In the procedural code, the Fibonacci sequence is generated directly within the main body of the code, making it less reusable and harder to maintain.

In contrast, the modular code encapsulates the Fibonacci logic within a function, allowing for easier reuse and better organization.

The modular approach enhances readability and maintainability, as changes to the Fibonacci logic can be made within the function without affecting the main code.

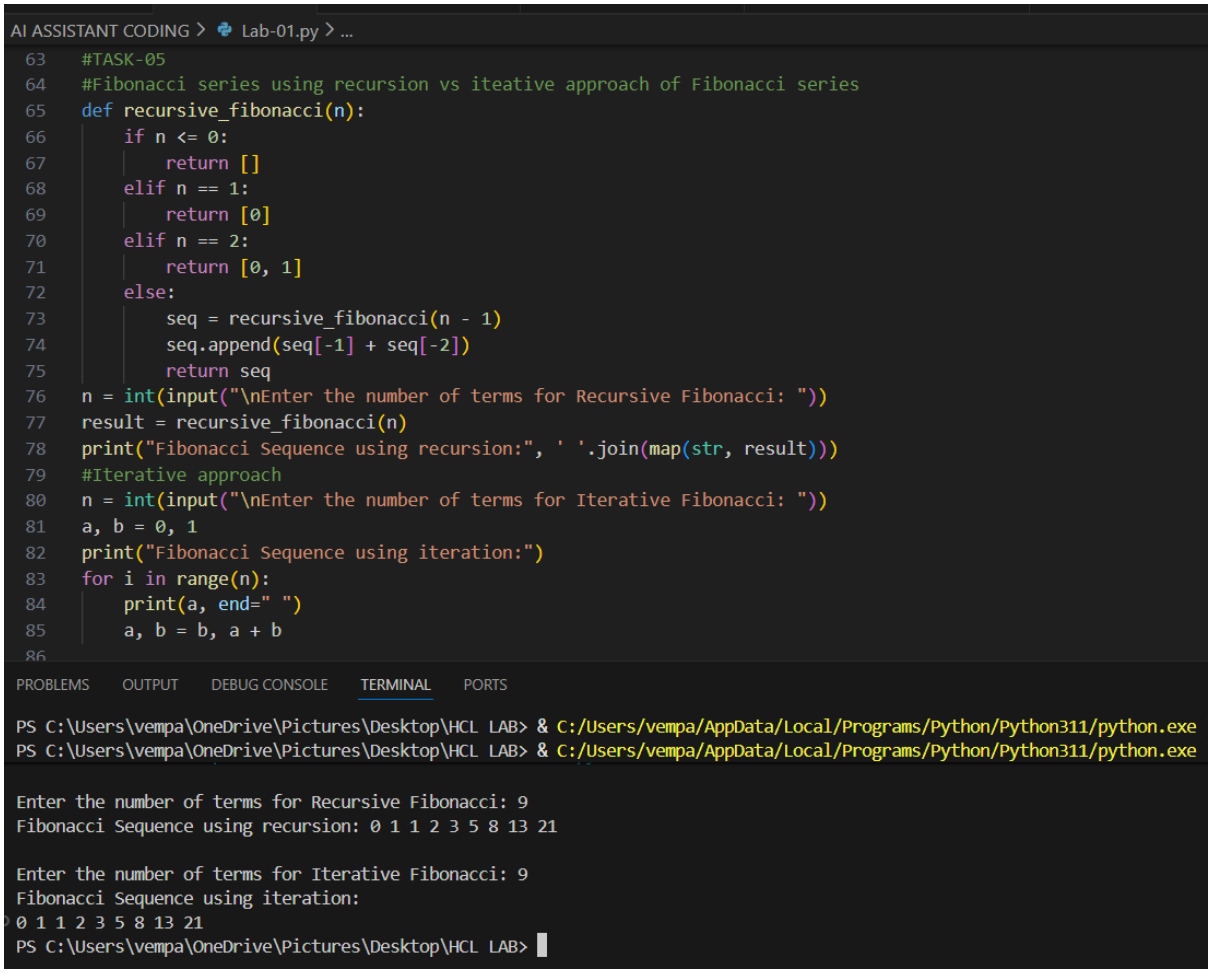
TASK-05:

Prompt: Fibonacci series using recursion vs iteative approach of Fibonacci series

Code:

def recursive_fibonacci(n):

```
    if n <= 0:
        return []
    elif n == 1:
        return [0]
    elif n == 2:
        return [0, 1]
    else:
        seq = recursive_fibonacci(n - 1)
        seq.append(seq[-1] + seq[-2])
        return seq
n = int(input("\nEnter the number of terms for Recursive Fibonacci: "))
result = recursive_fibonacci(n)
print("Fibonacci Sequence using recursion:", ' '.join(map(str, result)))
#Iterative approach
n = int(input("\nEnter the number of terms for Iterative Fibonacci: "))
a, b = 0, 1
print("Fibonacci Sequence using iteration:")
for i in range(n):
    print(a, end=" ")
    a, b = b, a + b
```



Explanation:

This code defines a recursive function to generate the Fibonacci sequence. The function builds the sequence by calling itself to get the sequence up to n-1 terms and then appending the next term.

The iterative approach is also provided for comparison, which uses a loop to generate the sequence.