

LAB-01

HTNO:2303A51279

BATCH NO:05

TASK-01

Prompt:

#Fibonacci series up to n terms it should accept user input

logic directly in main code and do not use any used defined functions

CODE:

```
n = int(input("Enter the number of terms in Fibonacci series: "))
a, b = 0, 1
count = 0
if n <= 0:
    print("Please enter a positive integer.")
elif n == 1:
    print("Fibonacci series up to", n, "term:")
    print(a)
else:
    print("Fibonacci series up to", n, "terms:")
    while count < n:
        print(a, end=' ')
        a, b = b, a + b
        count += 1
```

Output:

Enter the number of terms in Fibonacci series: 10

Fibonacci series up to 10 terms:

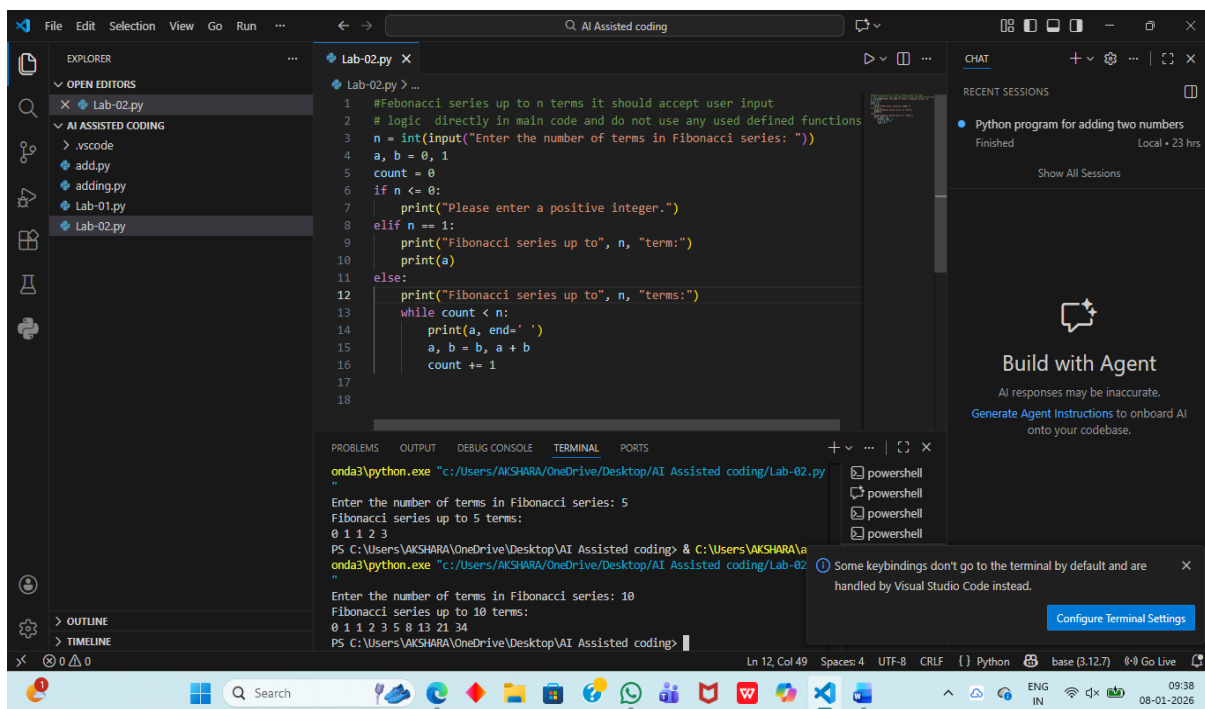
0 1 1 2 3 5 8 13 21 34

Explanation:

1. The program asks for n and starts the Fibonacci series with a = 0 and b = 1.

2. If n is 0 or negative, it shows an error; if n == 1, it prints only 0.
3. Otherwise, it runs a while loop, printing a each time and updating the values using a, b = b, a + b.
4. The loop stops after printing n numbers.

Implementation:



The screenshot shows a Visual Studio Code editor with a Python file named 'Lab-02.py'. The code implements a Fibonacci series generator. It prompts the user for the number of terms. If the input is 0 or negative, it prints an error message. If the input is 1, it prints the number 0. For other positive integers, it prints the Fibonacci series up to that number of terms. The terminal output shows the program being run twice: first for 5 terms, outputting '0 1 1 2 3', and then for 10 terms, outputting '0 1 1 2 3 5 8 13 21 34'. The interface also shows a sidebar with 'EXPLORER' and 'AI ASSISTED CODING' tabs, and a 'CHAT' panel on the right with a 'Build with Agent' section.

```
1 #Fibonacci series up to n terms it should accept user input
2 # logic directly in main code and do not use any used defined functions
3 n = int(input("Enter the number of terms in Fibonacci series: "))
4 a, b = 0, 1
5 count = 0
6 if n <= 0:
7     print("Please enter a positive integer.")
8 elif n == 1:
9     print("Fibonacci series up to", n, "term:")
10    print(a)
11 else:
12     print("Fibonacci series up to", n, "terms:")
13     while count < n:
14         print(a, end=' ')
15         a, b = b, a + b
16         count += 1
17
18
```

Terminal Output:

```
onda3\python.exe "c:/Users/AKSHARA/OneDrive/Desktop/AI Assisted coding/Lab-02.py"
Enter the number of terms in Fibonacci series: 5
Fibonacci series up to 5 terms:
0 1 1 2 3

PS C:\Users\AKSHARA\OneDrive\Desktop\AI Assisted coding> & C:\Users\AKSHARA\OneDrive\Desktop\AI Assisted coding\Lab-02.py
Enter the number of terms in Fibonacci series: 10
Fibonacci series up to 10 terms:
0 1 1 2 3 5 8 13 21 34

PS C:\Users\AKSHARA\OneDrive\Desktop\AI Assisted coding>
```

TASK-02:

Prompt:

- #Improve the task 1 by optimize this code,
- # simplify logic and reduce number of lines of code

CODE:

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

```
a, b = 0, 1
```

```
if n <= 0:
```

```
    print("Please enter a positive integer.")
```

```
else:
```

```
    print("Fibonacci series up to", n, "terms:")
```

```
    for _ in range(n):
```

```
print(a, end=' ')
```

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

Output:

Enter the number of terms in Fibonacci series: 5

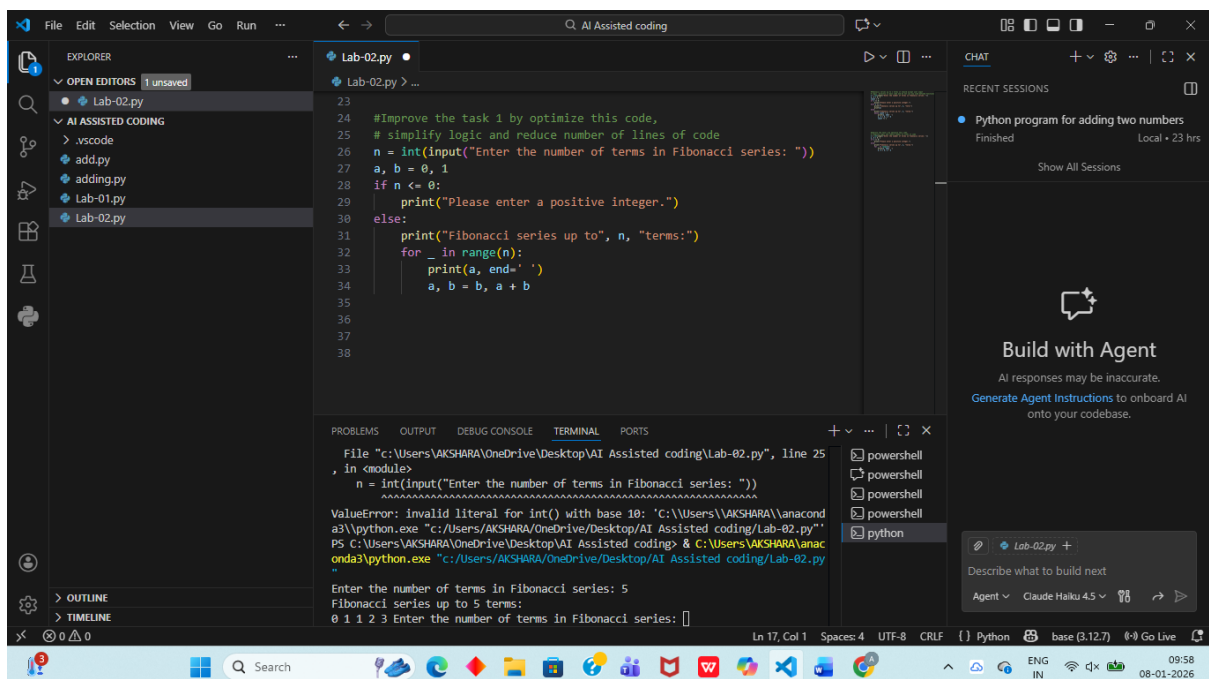
Fibonacci series up to 5 terms:

0 1 1 2 3 Enter the number of terms in Fibonacci series:

Explanation:

1. The program takes n from the user and starts with a = 0 and b = 1.
2. If n is not positive, it shows an error message.
3. Otherwise, a for loop runs n times, printing a each time.
4. Inside the loop, the next Fibonacci number is generated using a, b = b, a + b.

Implementation:



```
23
24 #Improve the task 1 by optimize this code,
25 # simplify logic and reduce number of lines of code
26 n = int(input("Enter the number of terms in Fibonacci series: "))
27 a, b = 0, 1
28 if n <= 0:
29     print("Please enter a positive integer.")
30 else:
31     print("Fibonacci series up to", n, "terms:")
32     for _ in range(n):
33         print(a, end=' ')
34         a, b = b, a + b
35
36
37
38
```

File "c:\Users\AKSHARA\OneDrive\Desktop\AI Assisted coding\Lab-02.py", line 25
, in <module>
n = int(input("Enter the number of terms in Fibonacci series: "))
~~~~~  
ValueError: Invalid literal for int() with base 10: 'c:\Users\AKSHARA\anaconda3\python.exe "c:\Users\AKSHARA\OneDrive\Desktop\AI Assisted coding\Lab-02.py" PS c:\Users\AKSHARA\OneDrive\Desktop\AI Assisted coding> & C:\Users\AKSHARA\anaconda3\python.exe "c:\Users\AKSHARA\OneDrive\Desktop\AI Assisted coding\Lab-02.py"

Enter the number of terms in Fibonacci series: 5  
Fibonacci series up to 5 terms:  
0 1 1 2 3 Enter the number of terms in Fibonacci series: []

## TASK-03

### Prompt:

#use user defined function to generate Fibonacci series up to n terms

#print the sequence with comments

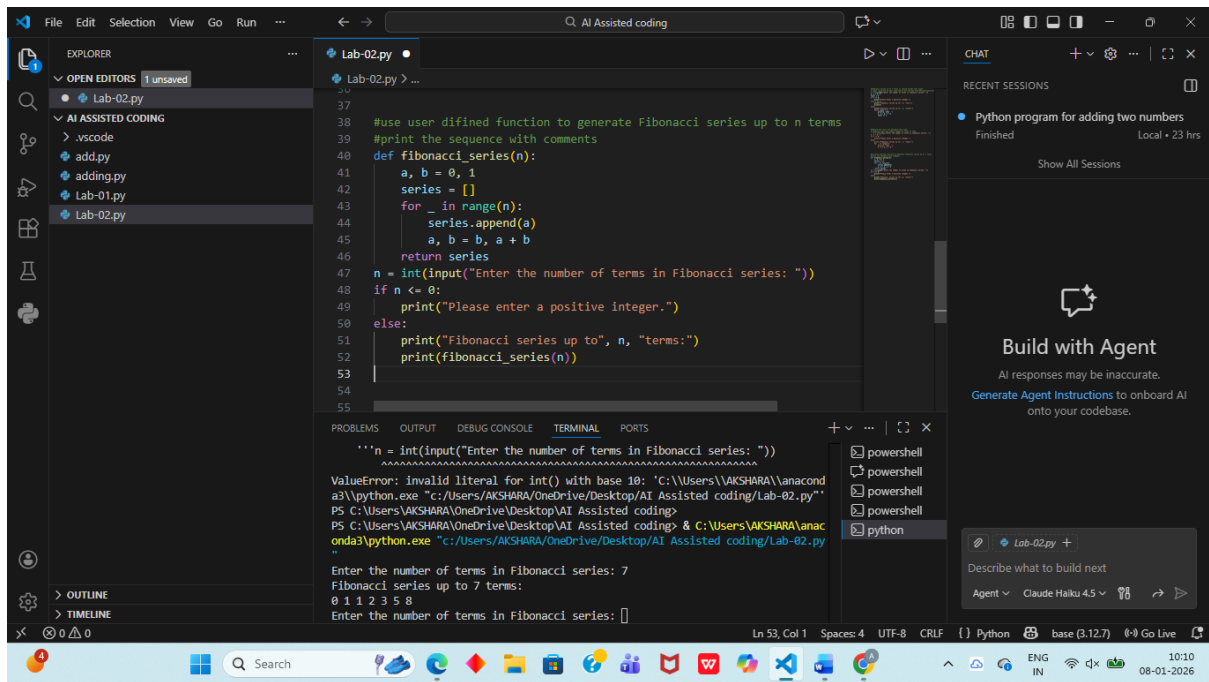
### Code:

```
def fibonacci_series(n):  
    a, b = 0, 1  
    series = []  
    for _ in range(n):  
        series.append(a)  
        a, b = b, a + b  
    return series  
  
n = int(input("Enter the number of terms in Fibonacci series: "))  
  
if n <= 0:  
    print("Please enter a positive integer.")  
else:  
    print("Fibonacci series up to", n, "terms:")  
    print(fibonacci_series(n))
```

**Explanation:**

1. The function `fibonacci_series(n)` starts with `a = 0` and `b = 1`, then uses a loop to build a list of Fibonacci numbers.
2. Each loop adds the current `a` to the list and updates values using `a, b = b, a + b`.
3. The function returns the full list of Fibonacci terms.
4. In the main program, the user enters `n`; if `n` is positive, the program prints the Fibonacci list — otherwise it shows an error message.

**Implementation:**



## TASK-04

### Prompt:

# Fibonacci series with Procedural vs Modular Fibonacci Code AI code with and without functions

# Procedural approach

### Code:

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

```
a, b = 0, 1
```

```
if n <= 0:
```

```
    print("Please enter a positive integer.")
```

```
else:
```

```
    print("Fibonacci series up to", n, "terms:")
```

```
    for _ in range(n):
```

```
        print(a, end=' ')
        a, b = b, a + b
```

```
    print()
```

# Modular approach

```
def fibonacci_series(n):
```

```

a, b = 0, 1
series = []
for _ in range(n):
    series.append(a)
    a, b = b, a + b
return series

n = int(input("Enter the number of terms in Modular Fibonacci series: "))
if n <= 0:
    print("Please enter a positive integer.")
else:
    print("Fibonacci series up to", n, "terms:")
    print(fibonacci_series(n))

```

### Output:

Enter the number of terms in Fibonacci series: 7

Fibonacci series up to 7 terms:

0 1 1 2 3 5 8

Enter the number of terms in Modular Fibonacci series: 7

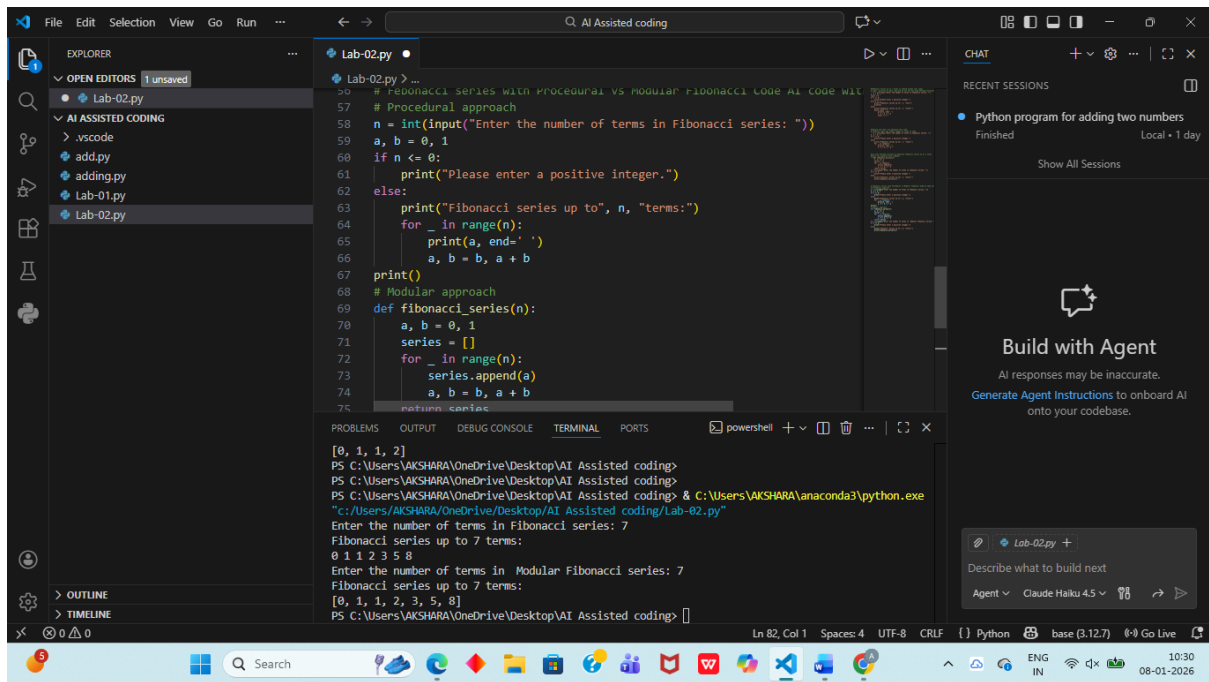
Fibonacci series up to 7 terms:

[0, 1, 1, 2, 3, 5, 8]

### Explanation:

- 1.First part prints Fibonacci numbers directly using a for loop: it starts with a = 0, b = 1, prints a each time, then updates with a, b = b, a + b.
- 2.It checks if n is positive before printing the series.
- 3.The second part uses a modular (function) approach: fibonacci\_series(n) builds the sequence in a list and returns it.
- 4.The program again asks for n, validates it, calls the function, and prints the returned list.

### Implementation:



## TASK-05

### Prompt:

#AI-Generated Iterative vs Recursive Fibonacci Approaches (Different

#Algorithmic Approaches for Fibonacci Series)

# Iterative approach and Recursive approach

### Code:

```
def fibonacci_iterative(n):
```

```
    a, b = 0, 1
```

```
    series = []
```

```
    for _ in range(n):
```

```
        series.append(a)
```

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

```
    return series
```

```
def fibonacci_recursive(n, a=0, b=1, series=None):
```

```
    if series is None:
```

```
        series = []
```

```
    if n == 0:
```

```
        return series
```

```

    series.append(a)

    return fibonacci_recursive(n - 1, b, a + b, series)

n = int(input("Enter the number of terms in Iterative Fibonacci series: "))

if n <= 0:

    print("Please enter a positive integer.")

else:

    print("Iterative Fibonacci series up to", n, "terms:")

    print(fibonacci_iterative(n))

n = int(input("Enter the number of terms in Recursive Fibonacci series: "))

if n <= 0:

    print("Please enter a positive integer.")

else:

    print("Recursive Fibonacci series up to", n, "terms:")

    print(fibonacci_recursive(n))

```

### Output:

Enter the number of terms in Iterative Fibonacci series: 6

Iterative Fibonacci series up to 6 terms:

[0, 1, 1, 2, 3, 5]

Enter the number of terms in Recursive Fibonacci series: 6

Recursive Fibonacci series up to 6 terms:

[0, 1, 1, 2, 3, 5]

### Explanation:

1. `fibonacci_iterative(n)` uses a loop: it starts with `a = 0`, `b = 1`, keeps appending `a` to a list, updates `a, b = b, a + b`, and finally returns the list.
2. `fibonacci_recursive(n, a, b, series)` builds the same sequence using recursion: it appends `a`, then calls itself with `n-1` until `n` becomes 0.
3. The program asks the user for `n` twice (once for iterative, once for recursive), checks that `n` is positive, and prints each result.



## Implementation:

The screenshot shows the VS Code editor with the file `Lab-02.py` open. The code implements two functions: `fibonacci_iterative(n)` and `fibonacci_recursive(n)`. The iterative function uses a loop to calculate the series, while the recursive function uses a recursive call. The code prompts the user to enter the number of terms for both series and prints the results.

```
84
85 #AI-Generated Iterative vs Recursive Fibonacci Approaches (Different
86 #Algorithmic Approaches for Fibonacci Series)
87 # Iterative approach and Recursive approach
88 def fibonacci_iterative(n):
89     a, b = 0, 1
90     series = []
91     for _ in range(n):
92         series.append(a)
93         a, b = b, a + b
94     return series
95 def fibonacci_recursive(n, a=0, b=1, series=None):
96     if series is None:
97         series = []
98         if n == 0:
99             return series
100        series.append(a)
101        return fibonacci_recursive(n - 1, b, a + b, series)
102 n = int(input("Enter the number of terms in Iterative Fibonacci series:
103 if n <= 0:
104     print("Please enter a positive integer.")
105 else:
106     print("Iterative Fibonacci series up to", n, "terms:")
107     print(fibonacci_iterative(n))
108 n = int(input("Enter the number of terms in Recursive Fibonacci series:
109 if n <= 0:
110     print("Please enter a positive integer.")
111 else:
112     print("Recursive Fibonacci series up to", n, "terms:")
113     print(fibonacci_recursive(n))
114
115
```

The screenshot shows the VS Code editor with the file `Lab-02.py` open. The code is the same as in the previous screenshot. The terminal window at the bottom shows the execution of the program. The user enters 6 for the number of terms in the iterative series, and the output is [0, 1, 1, 2, 3, 5]. The user then enters 6 for the number of terms in the recursive series, and the output is [0, 1, 1, 2, 3, 5].

```
ValueError: invalid literal for int() with base 10: 'C:\Users\AKSHARA\anaconda3\python.exe "c
:/Users/AKSHARA/OneDrive/Desktop/AI Assisted coding/Lab-02.py"'
PS C:\Users\AKSHARA\OneDrive\Desktop\AI Assisted coding> & C:\Users\AKSHARA\anaconda3\python.exe
"C:/Users/AKSHARA/OneDrive/Desktop/AI Assisted coding/Lab-02.py"
Enter the number of terms in Iterative Fibonacci series: 6
Iterative Fibonacci series up to 6 terms:
[0, 1, 1, 2, 3, 5]
Enter the number of terms in Recursive Fibonacci series: 6
Recursive Fibonacci series up to 6 terms:
[0, 1, 1, 2, 3, 5]
PS C:\Users\AKSHARA\OneDrive\Desktop\AI Assisted coding> & C:\Users\AKSHARA\anaconda3\python.exe
"C:/Users/AKSHARA/OneDrive/Desktop/AI Assisted coding/Lab-02.py"
```

