

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 user 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 the Visual Studio Code interface. In the Explorer sidebar, there are files like 'Lab-02.py', 'add.py', 'Lab-01.py', and 'Lab-02'. The 'Lab-02.py' file is open in the editor, displaying Python code for generating a Fibonacci series. The code includes logic to handle user input for the number of terms, check for non-positive integers, and print the series using a while loop. The terminal below shows the execution of the script and its output for 5 and 10 terms. The status bar at the bottom right indicates the date and time as 08-01-2026.

```

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

```

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:

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists files: Lab-02.py, .vscode, add.py, adding.py, Lab-01.py, and Lab-02.py (the active file). The code editor displays Python code for generating a Fibonacci series. The terminal at the bottom shows the execution of the program and its output. A sidebar on the right is titled 'Build with Agent' and includes a 'RECENT SESSIONS' section with a single entry: 'Python program for adding two numbers' (Finished, Local, 23 hrs ago).

```
File Edit Selection View Go Run ...
OPEN EDITORS 1 unsaved
EXPLORER
AI ASSISTED CODING
> .vscode
add.py
adding.py
Lab-01.py
Lab-02.py
Lab-02.py •
#Improve the task 1 by optimize this code,
# simplify logic and reduce number of lines of 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
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: []
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
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: "))
xxxxxxxxxxxxxx
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"
Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.
RECENT SESSIONS
Python program for adding two numbers
Finished Local 23 hrs
Show All Sessions
CHAT
+ × ⚙️ ...
RECENT SESSIONS
Python program for adding two numbers
Finished Local 23 hrs
Show All Sessions
Ln 17, Col 1 Spaces: 4 UTF-8 CRLF { Python 🐍 base (3.12.7) ⟲ Go Live 🔍
ENG IN 09:58 08-01-2026
```

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:

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files like .vscode, adding.py, Lab-01.py, and Lab-02.py.
- Editor:** Displays the code for `Lab-02.py`. The code defines a function `fibonacci_series(n)` to generate a Fibonacci sequence up to `n` terms. It includes error handling for non-positive integers and prints the sequence to the console.
- Terminal:** Shows the command line output of running the script, which asks for the number of terms and prints the sequence from 0 to 7.
- Bottom Status Bar:** Shows the date (08-01-2026), time (10:10), and language (ENG IN).
- AI Assistant Panel:** On the right, there's a "Build with Agent" section where the user can describe what to build next, with an AI agent named Claude Haku 4.5.

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:

```

File Edit Selection View Go Run ...
OPEN EDITORS 1 unsaved
Lab-02.py
EXPLORER
AI ASSISTED CODING
.vscode
adding.py
Lab-01.py
Lab-02.py
RECENT SESSIONS
Python program for adding two numbers
Finished Local 1 day
Show All Sessions
CHAT
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
[0, 1, 1, 2]
PS C:\Users\AKSHARA\OneDrive\Desktop\AI Assisted coding>
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: 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]
PS C:\Users\AKSHARA\OneDrive\Desktop\AI Assisted coding>

```

Build with Agent
AI responses may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.

Lab-02.py +
Describe what to build next
Agent Claude Haiku 4.5 ▾ 10:30 08-01-2026

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

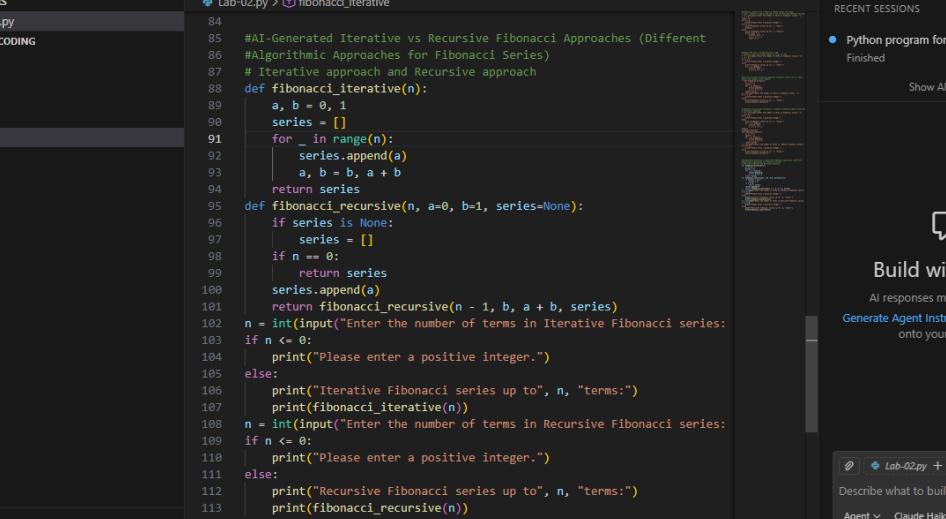
Recursive Fibonacci series up to 6 terms:

[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 Visual Studio Code interface with the following details:

- File Explorer:** Shows open files including `Lab-02.py`, `fibonacci_iterative`, `.vscode`, `add.py`, `adding.py`, `Lab-01.py`, and `Lab-02.py`.
- Code Editor:** Displays Python code for Fibonacci series generation.
- AI Assisted Coding:** A sidebar on the right provides assistance for the current file. It includes a "CHAT" section with a message from Claude Haiku 4.5 about building the agent, and a "RECENT SESSIONS" section showing a completed session for a Python program.
- Bottom Bar:** Includes icons for search, file operations, and extensions.

```
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))
```

The screenshot shows a Visual Studio Code (VS Code) interface with the following details:

- File Explorer:** Shows open files: Lab-02.py, fibonacci_iterative, .vscode, add.py, adding.py, Lab-01.py, and Lab-02.py (the active file).
- Code Editor:** Displays Python code for generating Fibonacci series iteratively and recursively. The iterative approach uses a loop to calculate terms from 0 to n, while the recursive approach uses a function that calls itself.
- Terminal:** Shows command-line output for both approaches. The iterative version takes user input for the number of terms (6) and prints the series [0, 1, 1, 2, 3, 5]. The recursive version also takes input (6) and prints the series [0, 1, 1, 2, 3, 5]. Both are run from the path C:\Users\AKSHARA\OneDrive\Desktop\AI Assisted coding\Lab-02.py.
- Status Bar:** Shows the current file as Lab-02.py, line 91, column 23, and other status information like Python 3.12.7 and Go Live.
- Bottom Bar:** Includes the Windows taskbar with various pinned icons like File Explorer, Edge, and File Explorer.

