

Assignment 01

Anyam Akshitha

2303A52453

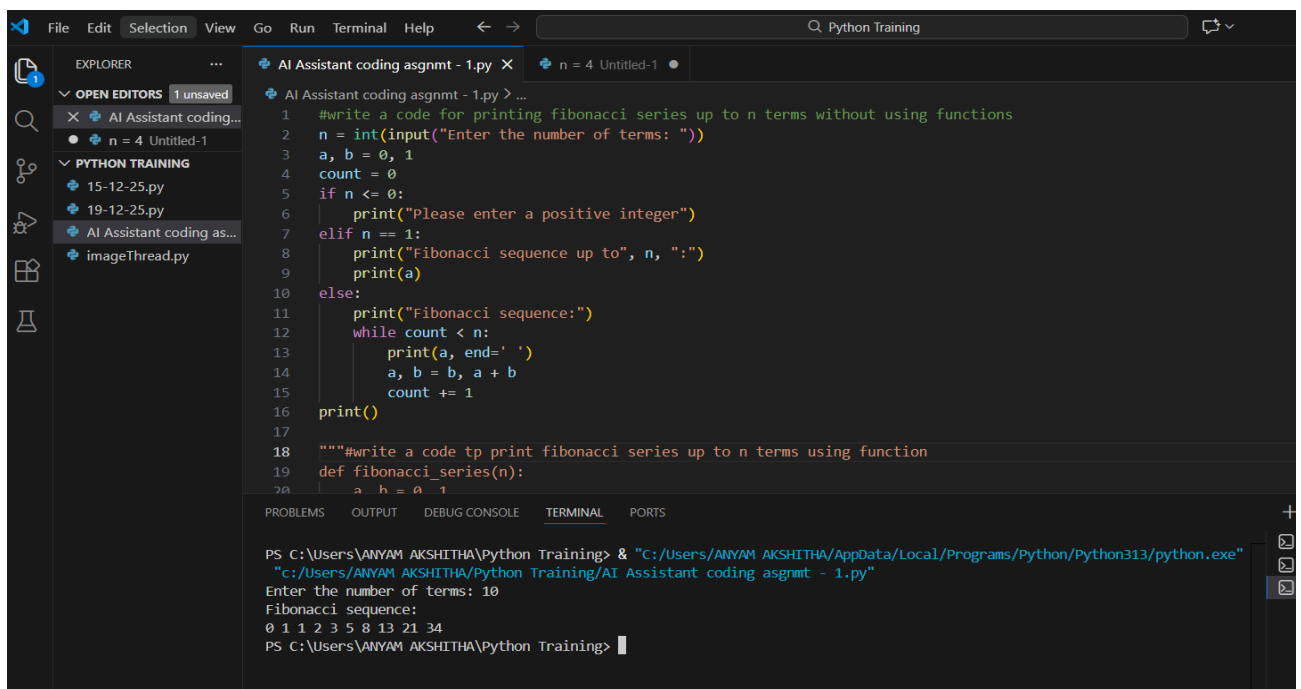
Batch – 42

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

PROMPT:

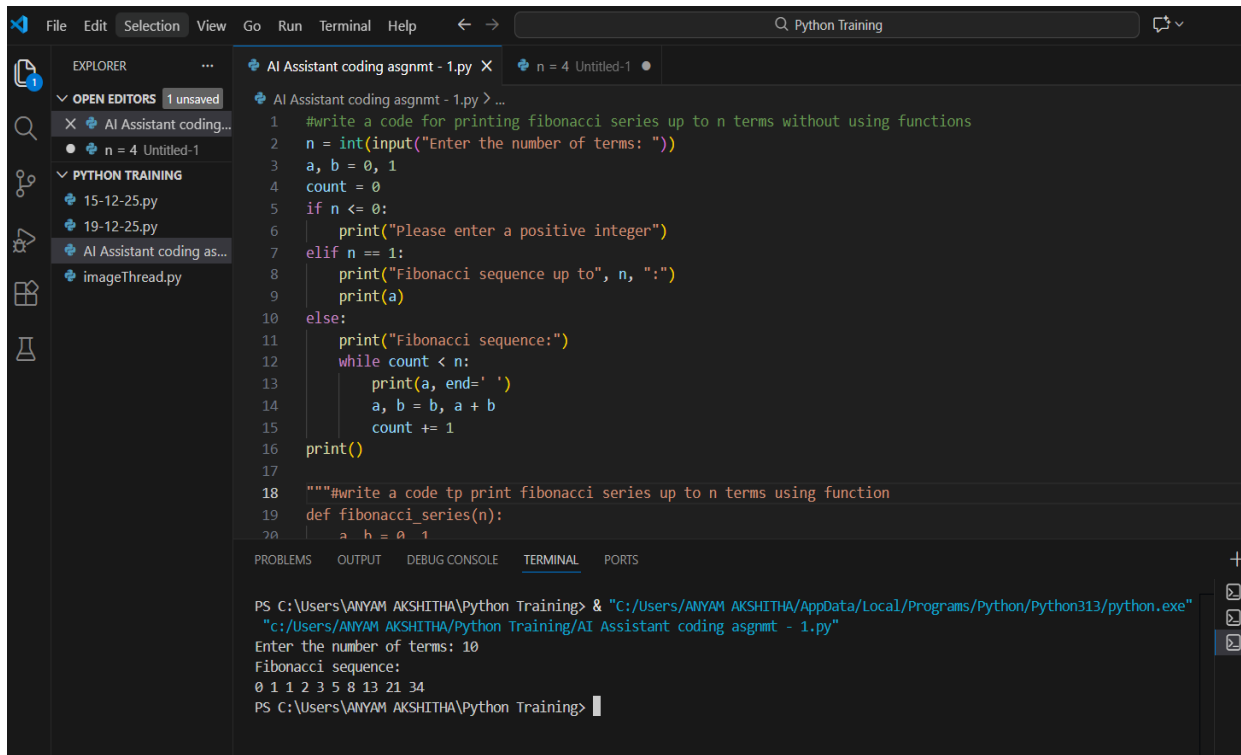
Write a code for printing fibonacci series up to n terms without using functions

CODE:



```
File Edit Selection View Go Run Terminal Help Python Training
EXPLORER
OPEN EDITORS 1 unsaved
AI Assistant coding...
n = 4 Untitled-1
PYTHON TRAINING
15-12-25.py
19-12-25.py
AI Assistant coding as...
imageThread.py
AI Assistant coding asgmt - 1.py X
n = 4 Untitled-1
AI Assistant coding asgmt - 1.py > ...
1 #write a code for printing fibonacci series up to n terms without using functions
2 n = int(input("Enter the number of terms: "))
3 a, b = 0, 1
4 count = 0
5 if n <= 0:
6     print("Please enter a positive integer")
7 elif n == 1:
8     print("Fibonacci sequence up to", n, ":")
9     print(a)
10 else:
11     print("Fibonacci sequence:")
12     while count < n:
13         print(a, end=' ')
14         a, b = b, a + b
15         count += 1
16 print()
17
18 """#write a code tp print fibonacci series up to n terms using function
19 def fibonacci_series(n):
20     a, b = 0, 1
PS C:\Users\ANYAM AKSHITHA\Python Training> & "C:/Users/ANYAM AKSHITHA/AppData/Local/Programs/Python/Python313/python.exe"
"C:/Users/ANYAM AKSHITHA/Python Training/AI Assistant coding asgmt - 1.py"
Enter the number of terms: 10
Fibonacci sequence:
0 1 1 2 3 5 8 13 21 34
PS C:\Users\ANYAM AKSHITHA\Python Training>
```

OUTPUT :



```
File Edit Selection View Go Run Terminal Help Python Training
EXPLORER
OPEN EDITORS 1 unsaved
AI Assistant coding asg...
n = 4 Untitled-1
PYTHON TRAINING
15-12-25.py
19-12-25.py
AI Assistant coding as...
imageThread.py

AI Assistant coding asgnt - 1.py > ...
1 #write a code for printing fibonacci series up to n terms without using functions
2 n = int(input("Enter the number of terms: "))
3 a, b = 0, 1
4 count = 0
5 if n <= 0:
6     print("Please enter a positive integer")
7 elif n == 1:
8     print("Fibonacci sequence up to", n, ":")
9     print(a)
10 else:
11     print("Fibonacci sequence:")
12     while count < n:
13         print(a, end=' ')
14         a, b = b, a + b
15         count += 1
16 print()
17
18 """#write a code tp print fibonacci series up to n terms using function
19 def fibonacci_series(n):
20     a, b = 0, 1

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\Users\ANYAM AKSHITHA\Python Training> & "C:/Users/ANYAM AKSHITHA/AppData/Local/Programs/Python/Python313/python.exe"
"C:/Users/ANYAM AKSHITHA/Python Training/AI Assistant coding asgnt - 1.py"
Enter the number of terms: 10
Fibonacci sequence:
0 1 1 2 3 5 8 13 21 34
PS C:\Users\ANYAM AKSHITHA\Python Training>
```

JUSTIFICATION:

By completing this task , we have printed the required fibonocci series up to n terms without modularization. In this code we are printing the series with only one for loop.

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

PROMPT:

Optimized version of Fibonacci series up to n terms without using functions

CODE:

The screenshot shows the VS Code interface with a Python file named 'AI Assistant coding assignmt - 1.py'. The code is an optimized version of a Fibonacci series generator. The terminal output shows the program running and printing the Fibonacci sequence for n=10.

```
1 #Optimized version of Fibonacci series up to n terms without using functions
2
3 n = int(input("Enter the number of terms: "))
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 sequence up to", n, ":")
10    print(a)
11 else:
12     print("Fibonacci sequence:")
13     while count < n:
14         print(a, end=' ')
15         a, b = b, a + b
16         count += 1
17 print()
18
19 """Write a code to print fibonacci series up to n terms using function
20 def fibonacci(n):
```

```
PS C:\Users\ANVAM AKSHITHA\Python Training> "C:/Users/ANVAM AKSHITHA/AppData/Local/Programs/Python/Python313/python.exe"
"C:/Users/ANVAM AKSHITHA/Python Training/AI Assistant coding assignmt - 1.py"
Enter the number of terms: 10
Fibonacci sequence:
0 1 1 2 3 5 8 13 21 34
PS C:\Users\ANVAM AKSHITHA\Python Training> "C:/Users/ANVAM AKSHITHA/AppData/Local/Programs/Python/Python313/python.exe"
"C:/Users/ANVAM AKSHITHA/Python Training/AI Assistant coding assignmt - 1.py"
Enter the number of terms: 10
Fibonacci sequence:
0 1 1 2 3 5 8 13 21 34
PS C:\Users\ANVAM AKSHITHA\Python Training>
```

OUTPUT :

This screenshot is identical to the one above, showing the same Python code and terminal output for the Fibonacci sequence generator.

JUSTIFICATION:

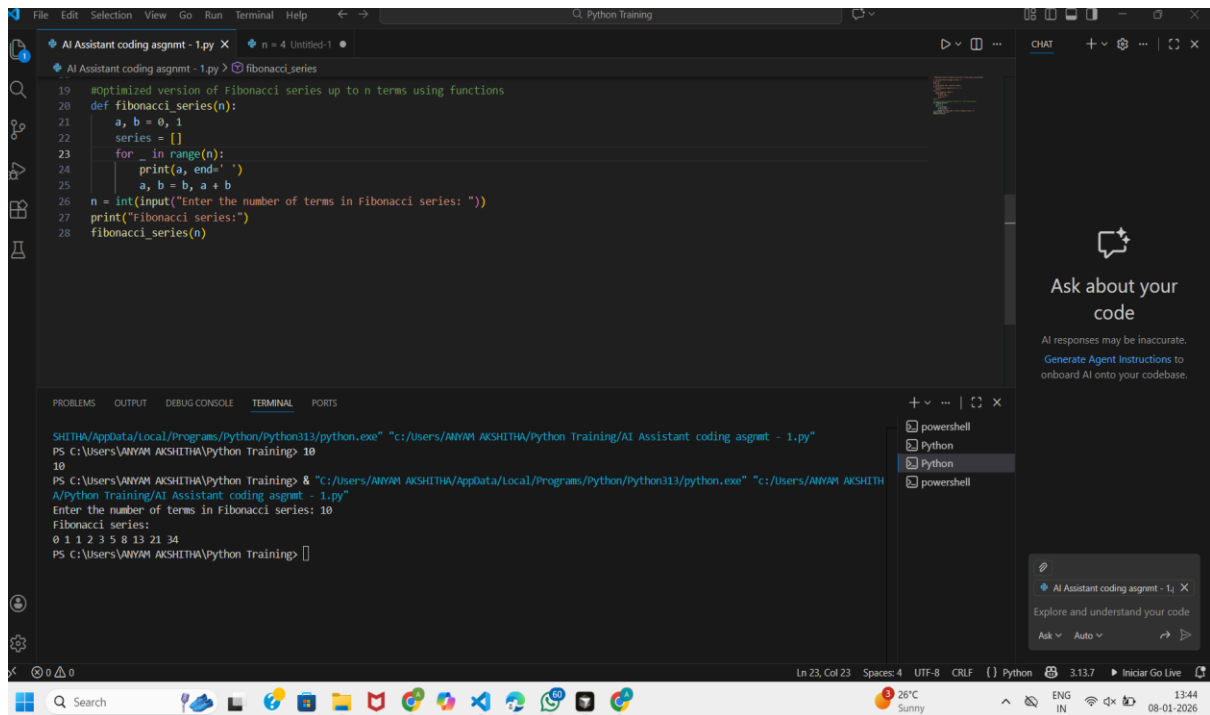
By completing this task , we have got the optimized version of the previous code , as we can see the code which we have got the in task 1 is already optimized one. Optimization of the code is very much important, as it reduces the time complexity and space complexity of the code, for the code which has more time complexity takes more time to run(slower) and which has more space complexity will utilize maximum space. So for completing our task efficiently , we need to Optimize our code.

Task 3: Modular Design Using AI Assistance (Fibonacci Using Functions)

PROMPT:

Optimized version of Fibonacci series up to n terms using functions

CODE:



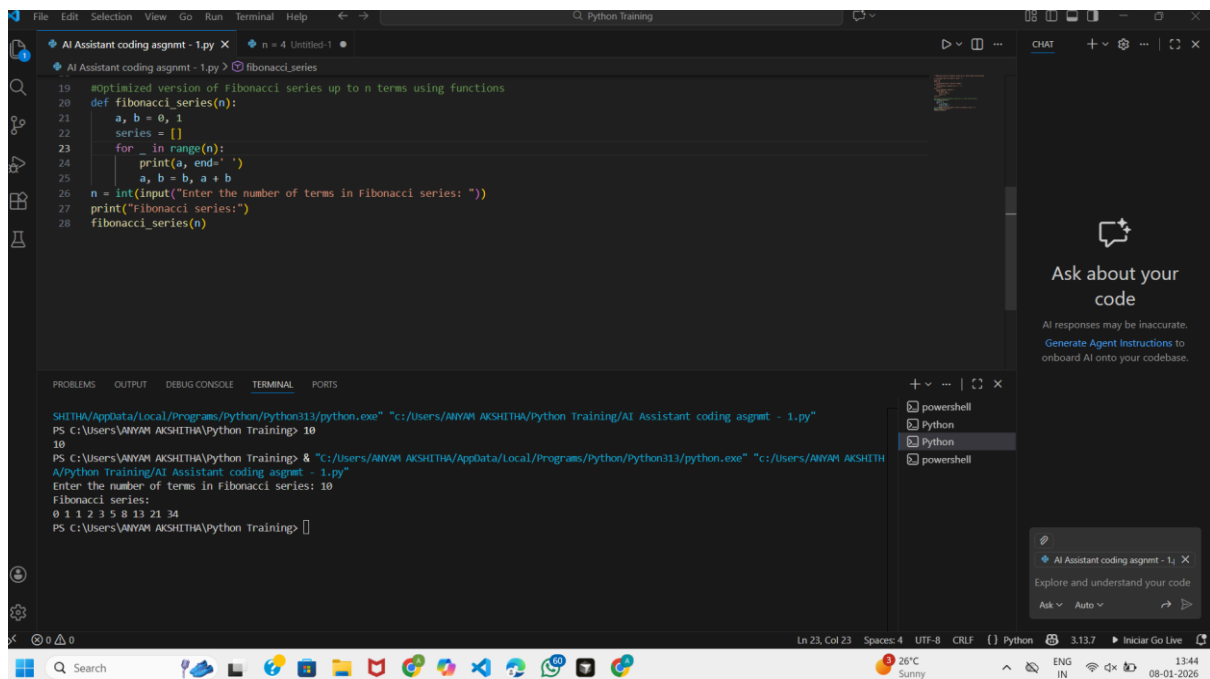
The screenshot shows a Visual Studio Code editor with a Python file named `fibonacci_series.py`. The code defines a function `fibonacci_series(n)` that prints the first `n` terms of the Fibonacci series. The terminal output shows the script being executed, and the user entering `10` for the number of terms. The output of the script is the Fibonacci series up to the 10th term: `0 1 1 2 3 5 8 13 21 34`.

```
19 #optimized version of Fibonacci series up to n terms using functions
20 def fibonacci_series(n):
21     a, b = 0, 1
22     series = []
23     for _ in range(n):
24         print(a, end=' ')
25         a, b = b, a + b
26 n = int(input("Enter the number of terms in Fibonacci series: "))
27 print("Fibonacci series:")
28 fibonacci_series(n)
```

Terminal Output:

```
SHITHA/AppData/Local/Programs/Python/Python313/python.exe "c:/Users/ANYAM AKSHITHA/Python Training/AI Assistant coding asgmt - 1.py"
PS C:\Users\ANYAM AKSHITHA\Python Training> 10
10
PS C:\Users\ANYAM AKSHITHA\Python Training> & "c:/Users/ANYAM AKSHITHA/AppData/Local/Programs/Python/Python313/python.exe" "c:/Users/ANYAM AKSHITHA/Python Training/AI Assistant coding asgmt - 1.py"
Enter the number of terms in Fibonacci series: 10
Fibonacci series:
0 1 1 2 3 5 8 13 21 34
PS C:\Users\ANYAM AKSHITHA\Python Training>
```

OUTPUT:



This screenshot is identical to the one above, showing the same Python code and terminal execution results for the Fibonacci series.

JUSTIFICATION:

By completing this task, we are able to print the Fibonacci series up to n numbers using functions by using the above code. At first we are calling the user-defined function using function calling statement and then our function runs. In our function the main for loop runs and prints the output.

Task 4: Comparative Analysis – Procedural vs Modular Fibonacci Code

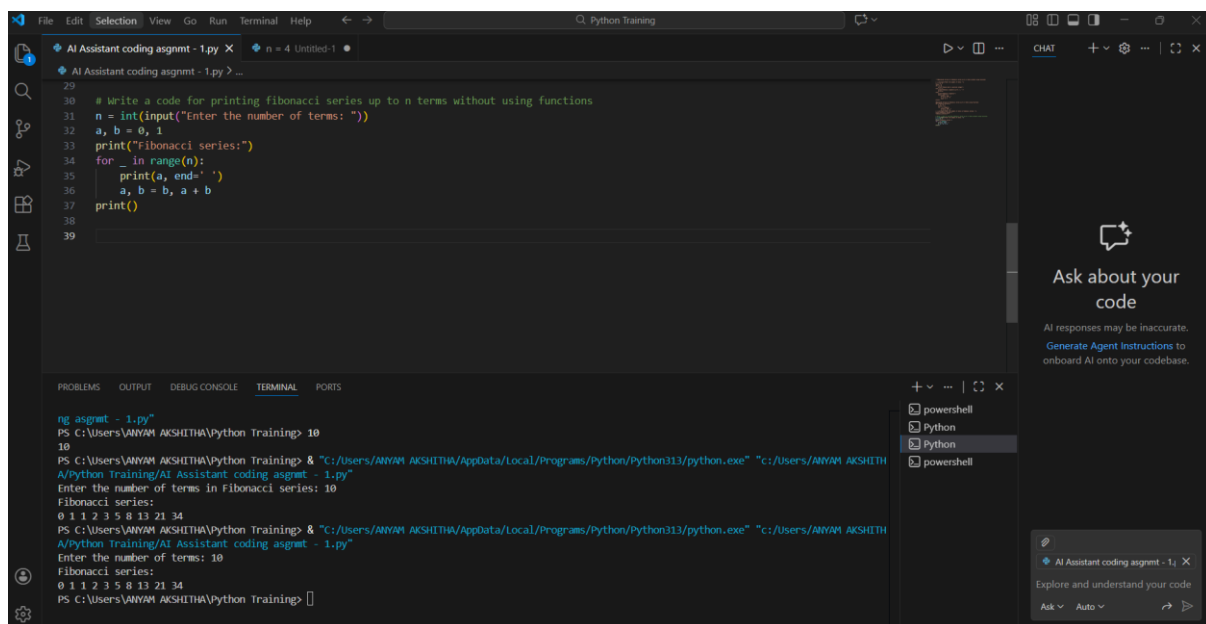
PROMPTS:

Procedural : Write a code for printing fibonacci series up to n terms without using functions

Modular : Optimized version of Fibonacci series up to n terms using functions

CODE:

Procedural:



```
29 # Write a code for printing fibonacci series up to n terms without using functions
30
31 n = int(input("Enter the number of terms: "))
32 a, b = 0, 1
33 print("Fibonacci series:")
34 for _ in range(n):
35     print(a, end=' ')
36     a, b = b, a + b
37 print()
38
39
```

AI Assistant coding asgmt - 1.py X n = 4 Untitled-1

File Edit Selection View Go Run Terminal Help Python Training

AI Assistant coding asgmt - 1.py X

AI Assistant coding asgmt - 1.py ...

29 # Write a code for printing fibonacci series up to n terms without using functions

30

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

32 a, b = 0, 1

33 print("Fibonacci series:")

34 for _ in range(n):

35 print(a, end=' ')

36 a, b = b, a + b

37 print()

38

39

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

ng asgmt - 1.py"

PS C:\Users\ANIAM AKSHITHA\Python Training> 10

10

PS C:\Users\ANIAM AKSHITHA\Python Training> & "C:/Users/ANIAM AKSHITHA/AppData/Local/Programs/python/python313/python.exe" "C:/Users/ANIAM AKSHITHA/Python Training/AI Assistant coding asgmt - 1.py"

A/Python Training/AI Assistant coding asgmt - 1.py

Enter the number of terms in Fibonacci series: 10

Fibonacci series:

0 1 1 2 3 5 8 13 21 34

PS C:\Users\ANIAM AKSHITHA\Python Training> & "C:/Users/ANIAM AKSHITHA/AppData/Local/Programs/python/python313/python.exe" "C:/Users/ANIAM AKSHITHA/Python Training/AI Assistant coding asgmt - 1.py"

A/Python Training/AI Assistant coding asgmt - 1.py

Enter the number of terms: 10

Fibonacci series:

0 1 1 2 3 5 8 13 21 34

PS C:\Users\ANIAM AKSHITHA\Python Training>

Ask about your code

AI responses may be inaccurate.

Generate Agent Instructions to onboard AI onto your codebase.

AI Assistant coding asgmt - 1.py X

Explore and understand your code

Ask Auto

Modular:

```
36     a, b = b, a + b
37     print()
38
39 # Optimized version of Fibonacci series up to n terms using functions
40 def fibonacci_series(n):
41     a, b = 0, 1
42     for _ in range(n):
43         print(a, end=' ')
44         a, b = b, a + b
45 n = int(input("Enter the number of terms in Fibonacci series: "))
46 print("Fibonacci series:")
47 fibonacci_series(n)
48
49
```

```
Enter the number of terms in Fibonacci series: 10
Fibonacci series:
Traceback (most recent call last):
  File "c:\Users\ANAY AKSHITHA\Python Training\AI Assistant coding asgmt - 1.py", line 47, in <module>
    fibonacci_series(n)
  File "c:\Users\ANAY AKSHITHA\Python Training\AI Assistant coding asgmt - 1.py", line 41, in fibonacci_series
    a, b = 0,
ValueError: not enough values to unpack (expected 2, got 1)
PS C:\Users\ANAY AKSHITHA\Python Training> & "c:\Users\ANAY AKSHITHA\AppData\Local\Programs\Python\Python313\python.exe" "c:\Users\ANAY AKSHITHA\Python Training\AI Assistant coding asgmt - 1.py"
Enter the number of terms in Fibonacci series: 10
Fibonacci series:
0 1 1 2 3 5 8 13 21 34
PS C:\Users\ANAY AKSHITHA\Python Training>
```

OUTPUT:
Procedural:

```
29
30 # Write a code for printing fibonacci series up to n terms without using functions
31 n = int(input("Enter the number of terms: "))
32 a, b = 0, 1
33 print("Fibonacci series:")
34 for _ in range(n):
35     print(a, end=' ')
36     a, b = b, a + b
37 print()
38
39
```

```
ng asgmt - 1.py"
PS C:\Users\ANAY AKSHITHA\Python Training> 10
10
PS C:\Users\ANAY AKSHITHA\Python Training> & "c:\Users\ANAY AKSHITHA\AppData\Local\Programs\python\Python313\python.exe" "c:\Users\ANAY AKSHITHA\Python Training\AI Assistant coding asgmt - 1.py"
Enter the number of terms in Fibonacci series: 10
Fibonacci series:
0 1 1 2 3 5 8 13 21 34
PS C:\Users\ANAY AKSHITHA\Python Training> & "c:\Users\ANAY AKSHITHA\AppData\Local\Programs\python\Python313\python.exe" "c:\Users\ANAY AKSHITHA\Python Training\AI Assistant coding asgmt - 1.py"
Enter the number of terms: 10
Fibonacci series:
0 1 1 2 3 5 8 13 21 34
PS C:\Users\ANAY AKSHITHA\Python Training>
```

Modular:

The screenshot shows a Visual Studio Code editor with a Python file named 'AI Assistant coding asgmt - 1.py'. The code defines a function 'fibonacci_series(n)' that calculates the Fibonacci series up to 'n' terms using an iterative approach. The function initializes 'a' and 'b' to 0 and 1, then iterates from 0 to 'n-1', printing the current value of 'a' and updating 'a' and 'b' to 'b' and 'a+b' respectively. The main part of the script prompts the user to enter the number of terms, which is 10, and then calls the 'fibonacci_series' function. The terminal output shows the execution of the script, which prints the Fibonacci series: 0 1 1 2 3 5 8 13 21 34. The terminal also shows a 'ValueError: not enough values to unpack (expected 2, got 1)' error, which is a result of the 'a, b = 0, 1' initialization. The error is resolved by changing the initialization to 'a, b = 0, 1'.

JUSTIFICATION:

By this task , we are able to find the difference between Procedural(without using functions) and Modular(with using functions). The main use of function is

- Reusability of the code
- Easy to Debug
- Code Clarity • Suitable for large systems

By observing, we can analyze that using modular method is a better and clean aproch

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

PROMPTS:

Iterative approach: fibonacci series up to n terms using iterative approach

Recursive approach : fibonacci series up to n terms using recursive approach

CODE:

Recursive approach:

The screenshot shows a Visual Studio Code editor with a Python file named 'AI Assistant coding asgmt - 1.py'. The code implements a Fibonacci series using an iterative approach. The terminal output shows the program running successfully, printing the Fibonacci series for 10 terms: 0 1 1 2 3 5 8 13 21 34.

```
50
51 #fibonacci series up to n terms using iterative approach
52 def fibonacci_series(n):
53     a, b = 0, 1
54     for _ in range(n):
55         print(a, end=' ')
56         a, b = b, a + b
57 n = int(input("Enter the number of terms in Fibonacci series: "))
58 print("Fibonacci series:")
59 fibonacci_series(n)
```

Enter the number of terms in Fibonacci series: 10
Fibonacci series:
0 1 1 2 3 5 8 13 21 34
PS C:\Users\ANYAM AKSHITHA\Python Training>

Iterative approach:

The screenshot shows the same Visual Studio Code editor with a recursive Fibonacci function. The terminal output shows a runtime error: 'NameError: name 'fibonacci' is not defined'. This occurs because the function calls itself using 'fibonacci_recursive(n-1)' instead of 'fibonacci_series(n-1)'.

```
61
62 # fibonacci series up to n terms using recursive approach
63 def fibonacci_recursive(n):
64     if n <= 0:
65         return []
66     elif n == 1:
67         return [0]
68     elif n == 2:
69         return [0, 1]
70     else:
71         series = fibonacci_recursive(n - 1)
72         series.append(series[-1] + series[-2])
73         return series
74 n = int(input("Enter the number of terms in Fibonacci series: "))
75 print("Fibonacci series:")
76 print(' '.join(map(str, fibonacci_recursive(n))))
```

Enter the number of terms in Fibonacci series: 10
Fibonacci series:
Traceback (most recent call last):
File "c:\Users\ANYAM AKSHITHA\Python Training\AI Assistant coding asgmt - 1.py", line 76, in <module>
 print(' '.join(map(str, fibonacci_recursive(n))))

File "c:\Users\ANYAM AKSHITHA\Python Training\AI Assistant coding asgmt - 1.py", line 71, in fibonacci_recursive
 series = fibonacci(n - 1)
NameError: name 'fibonacci' is not defined
PS C:\Users\ANYAM AKSHITHA\Python Training> & "C:\Users\ANYAM AKSHITHA\AppData\Local\Programs\Python\Python313\python.exe" "c:\Users\ANYAM AKSHITHA\Python Training\AI Assistant coding asgmt - 1.py"
Enter the number of terms in Fibonacci series: 10
Fibonacci series:
0 1 1 2 3 5 8 13 21 34
PS C:\Users\ANYAM AKSHITHA\Python Training>

OUTPUT:

Recursive approach:

The screenshot shows a Visual Studio Code editor with a Python file named 'AI Assistant coding asgmt - 1.py'. The code implements an iterative Fibonacci function. The terminal shows the program being executed, with the user entering '10' for the number of terms. The output is the Fibonacci series: '0 1 1 2 3 5 8 13 21 34'. The interface includes a sidebar with icons for Explorer, Search, and Run and Debug, and a bottom status bar showing the file path and language (Python).

```
50
51 #fibonacci series up to n terms using iterative approach
52 def fibonacci_series(n):
53     a, b = 0, 1
54     for _ in range(n):
55         print(a, end=' ')
56         a, b = b, a + b
57 n = int(input("Enter the number of terms in Fibonacci series: "))
58 print("Fibonacci series:")
59 fibonacci_series(n)
```

Enter the number of terms in Fibonacci series: 10
Fibonacci series:
0 1 1 2 3 5 8 13 21 34
PS C:\Users\ANYAM AKSHITHA\Python Training>

A:\Python Training> & "C:\Users\ANYAM AKSHITHA\AppData\Local\Programs\Python\Python313\python.exe" "c:\Users\ANYAM AKSHITHA\A\Python Training\AI Assistant coding asgmt - 1.py"
Enter the number of terms in Fibonacci series: 10
Fibonacci series:
0 1 1 2 3 5 8 13 21 34
PS C:\Users\ANYAM AKSHITHA\Python Training>

Iterative approach:

The screenshot shows the same Visual Studio Code editor with a recursive Fibonacci function. The terminal shows the program being executed, but it results in a 'NameError: name 'fibonacci' is not defined' error. The error message indicates that the function is calling itself recursively, but the variable 'fibonacci' is not defined in the current scope. The output shows the first few terms of the series before the error occurs.

```
61
62 # fibonacci series up to n terms using recursive approach
63 def fibonacci_recursive(n):
64     if n <= 0:
65         return []
66     elif n == 1:
67         return [0]
68     elif n == 2:
69         return [0, 1]
70     else:
71         series = fibonacci_recursive(n - 1)
72         series.append(series[-1] + series[-2])
73         return series
74 n = int(input("Enter the number of terms in Fibonacci series: "))
75 print("Fibonacci series:")
76 print(' '.join(map(str, fibonacci_recursive(n))))
```

Enter the number of terms in Fibonacci series: 10
Fibonacci series:
Traceback (most recent call last):
File "c:\Users\ANYAM AKSHITHA\Python Training\AI Assistant coding asgmt - 1.py", line 76, in <module>
 print(' '.join(map(str, fibonacci_recursive(n))))
File "c:\Users\ANYAM AKSHITHA\Python Training\AI Assistant coding asgmt - 1.py", line 71, in fibonacci_recursive
 series = fibonacci(n - 1)
NameError: name 'fibonacci' is not defined
PS C:\Users\ANYAM AKSHITHA\Python Training> & "C:\Users\ANYAM AKSHITHA\AppData\Local\Programs\Python\Python313\python.exe" "c:\Users\ANYAM AKSHITHA\A\Python Training\AI Assistant coding asgmt - 1.py"
Enter the number of terms in Fibonacci series: 10
Fibonacci series:
0 1 1 2 3 5 8 13 21 34
PS C:\Users\ANYAM AKSHITHA\Python Training>

JUSTIFICATION:

The iterative Fibonacci approach is usually the better choice because it's faster, uses very little memory, and works well even when the number of terms is large. Recursive Fibonacci, on the other hand, makes many function calls, which slows things down and uses extra memory. For bigger inputs, it can even

crash due to recursion limits. That's why, in real-world programs, iteration is generally preferred over recursion.