

LAB-1

H.TNO:2303A51804

Batch:28

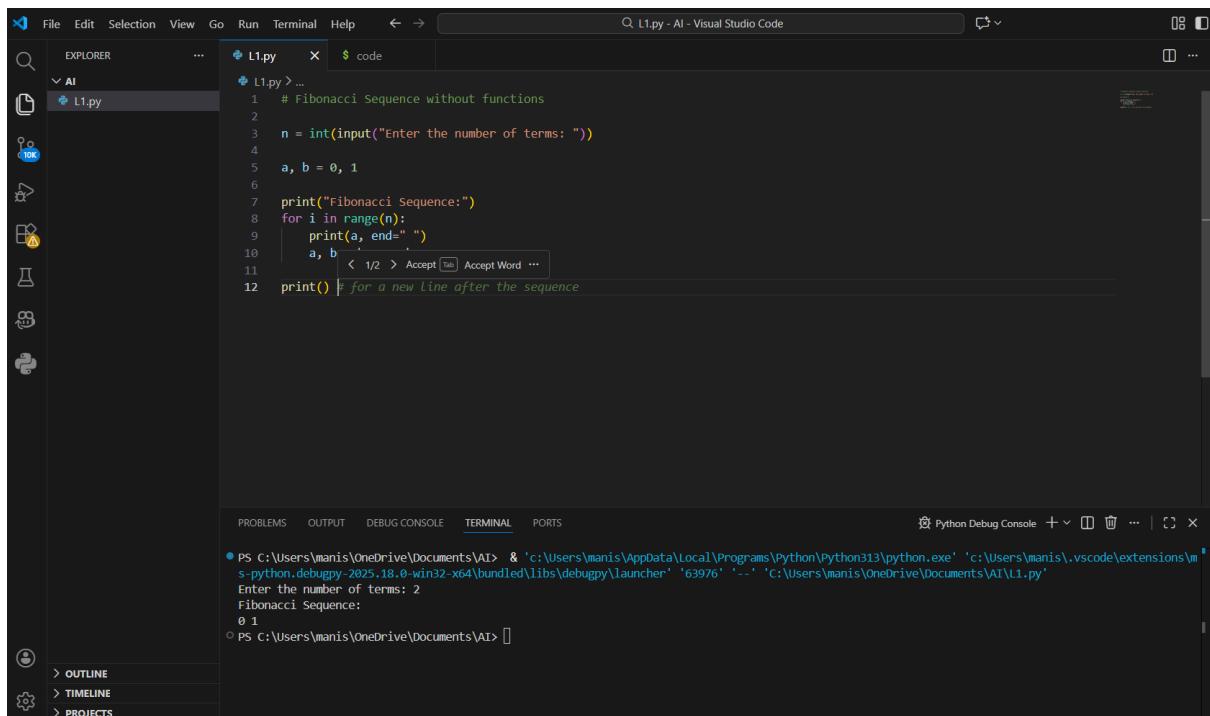
TASK-1

1. AI-Generated Logic Without Modularisation (Fibonacci Sequence Without Functions)

PROMPT:

Fibonacci Sequence without functions

CODE AND OUTPUT:



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

- File Explorer:** Shows a folder named "AI" containing a file "L1.py".
- Code Editor:** Displays the Python code for generating a Fibonacci sequence:

```
# Fibonacci Sequence without functions
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
print() # for a new line after the sequence
```
- Terminal:** Shows the output of running the script:

```
PS C:\Users\manis\OneDrive\Documents\AI> & "c:\Users\manis\AppData\Local\Programs\Python\Python313\python.exe" "c:\Users\manis\.vscode\extensions\ms-python.python-2025.18.0-win32-x64\bundled\libs\debugpy\launcher" 63976 -- "C:\Users\manis\OneDrive\Documents\AI\l1.py"
Enter the number of terms: 2
Fibonacci Sequence:
0 1
```

EXPLANATION:

1. The user inputs a number n , which decides how many Fibonacci values will be displayed.
2. Two variables are initialised, a and b , to represent the Fibonacci sequence 0 and 1.

3. A loop runs n times during each iteration, and the current value of a is printed.
4. After printing each number, the program updates a and b together so that a moves to the next Fibonacci value and b becomes the number after that, allowing the sequence to continue
5. This process continues until the loop finishes, so n Fibonacci values will be printed

TASK-2

AI Code Optimisation & Cleanup (Improving Efficiency)

PROMPT:

Give an optimised version of the above code

#different approach with code optimisation

#Generate code

Optimised Fibonacci Sequence using memorisation

CODE AND OUTPUT:

```

File Edit Selection View Go Run Terminal Help ← → 🔍 L1.py $ code
RUN AND DEBUG ... RUN
Run and Debug To customize Run and Debug create a launch.json file.
Debug using a terminal command or in an interactive chat.
Show automatic Python configurations
RUN
Build with Agent
AI responds may be inaccurate.
Generate Agent Instructions to onboard AI onto your codebase.

L1.py >
8 for i in range(n):
9     print(a, end=" ")
10    a, b = b, a + b
11
12 print()
13 # give a optimized version of the above code
14 #different approach with code optimization
15 #generate code
16 # Optimized Fibonacci Sequence using memorization
17 n = int(input("Enter the number of terms: "))
18 fib_cache = {0: 0, 1: 1}
19 def fibonacci(num):
20     if num not in fib_cache:
21         fib_cache[num] = fibonacci(num - 1) + fibonacci(num - 2)
22     return fib_cache[num]
23 print("Fibonacci Sequence:")
24 for i in range(n):
25     print(fibonacci(i), end=" ")
26
27
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS Python Debug Console + ⚡ ...
python.exe 'c:\Users\manis\vscode\extensions\ms-python.debugger-2025.18.0-win32-x64\bundled\libs\debugger\launcher' '55019' '--' 'C:\Users\manis\OneDrive\Documents\AI> c;; cd 'c:\Users\manis\OneDrive\Documents\AI'; & 'c:\Users\manis\AppData\Local\Programs\Python\Python313\p...
PS C:\Users\manis\OneDrive\Documents\AI>
Fibonacci Sequence:
0 1
Enter the number of terms: 2
Fibonacci Sequence:
0 1

```

Breakpoints: Raised Exceptions, Uncaught Exceptions, User Uncaught Exceptions

Indexing completed.

EXPLANATION:

1. Instead of repeatedly calling a function, the optimised approach simply uses a loop, which makes it faster
2. It only keeps track of the last two Fibonacci numbers, so it doesn't waste extra memory storing many values.
3. By avoiding recursion, it removes the risk of stack overflow and reduces unnecessary overhead.
4. The logic becomes simpler and cleaner, making the program easier to understand and maintain.
5. Overall, this approach gives the Fibonacci sequence quickly, efficiently, and with minimal resource usage

TASK-3

Modular Design Using AI Assistance (Fibonacci Using Functions)

PROMPT:

Give a code of Fibonacci using functions

#Generate code

Fibonacci Sequence using functions

CODE AND OUTPUT:

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

- File Explorer:** Shows a file named "L1.py".
- Code Editor:** Displays the following Python code:

```
26 #give a code of fibonacci using functions
27 #generate code
28 # Fibonacci Sequence using functions
29 def fibonacci(n):
30     a = b = 0
31     sequence = []
32     for i in range(n):
33         sequence.append(a)
34         a, b = b, a+b
35     return sequence
36 n = int(input("Enter the number of terms: "))
37 print("Fibonacci Sequence:")
38 for item in fibonacci(n):
39     print(item)
40 print("\n", end="")
```
- Terminal:** Shows the command to run the script: "python L1.py".
- Output:** Shows the generated Fibonacci sequence from 0 to 9.
- Right Panel:** An AI assistance panel titled "Build with Agent" is visible, with instructions to onboard AI onto the codebase.

EXPLANATION:

1. A Fibonacci number that is responsible for creating the Fibonacci sequence.
2. Inside the function, it starts with two initial values, 0 and 1, and gradually builds the sequence using a loop.
3. Each generated Fibonacci number is stored in a list so the entire sequence can be returned at once.
4. After taking user input, the function is called, and it sends back the completed Fibonacci list.
5. The program prints each number from that list, displaying the Fibonacci series

TASK-4

Analysis – Procedural vs Modular Fibonacci Code

PROMPT:

```
# Give a Python code for Fibonacci using procedural programming  
compare them in terms of readability
```

```
# Fibonacci Sequence using procedural programming
```

CODE AND OUTPUT:

```
File Edit Selection View Go Run Terminal Help < > AI ... Python Debugger: Cu...  
No Con... x code ...  
L1.py > ...  
37 n = int(input("Enter the number of terms: "))  
38 print("Fibonacci Sequence:")  
39 for num in fibonacci(n):  
40     print(num, end=" ")  
41 #give a python code for fibonacci using procedural programming compare them resuability readability  
42 # Fibonacci Sequence using procedural programming  
43 n = int(input("Enter the number of terms: "))  
44 a, b = 0, 1  
45 print("Fibonacci Sequence:")  
46 for i in range(n):  
47     print(a, end=" ")  
48     a, b = b, a + b  
49 print()  
50 # In the procedural programming approach, the code is straightforward and easy to read for small tasks. However, it lack:  
51  
52  
53  
54  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  
PS C:\Users\manis\OneDrive\Documents\AI> & 'c:\Users\manis\AppData\Local\Programs\Python\Python313\python.exe' 'c:\Users\manis\vs_code\extensions\ms-python.debugpy-2025.18.0-win32-x64\bundled\libs\debugpy\launcher' '54841' '--' 'c:\Users\manis\OneDrive\Documents\AI\L1.py'  
Enter the number of terms: 2  
Fibonacci Sequence:  
0 1  
Enter the number of terms: 5
```

EXPLANATION:

1. Procedural code is simple to write and understand for small programs.
2. It places all logic directly in the main program, so everything runs in one flow.
3. But it is not reusable, because the same logic must be rewritten if needed again.
4. The function-based approach is better organised since the logic is placed inside
5. It improves reusability, readability, and maintenance as the function can be called anytime with different inputs.

TASK-5

AI-Generated Iterative vs Recursive Fibonacci Approaches (Different Algorithmic Approaches for the Fibonacci Series)

PROMPT:

Explain the Fibonacci series using two different algorithmic approaches: Iterative and Recursive. Provide Python code for both methods

Iterative Approach

CODE AND OUTPUT:

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

- File Explorer:** Shows a file named "L1.py".
- Code Editor:** Displays the following Python code:

```
1 # Explain the Fibonacci series using two different algorithmic approaches: Iterative and Recursive. Provide Python code for both
2 # methods
3
4 # Iterative Approach
5 def fibonacci_iterative(n):
6     a, b = 0, 1
7     sequence = []
8     for _ in range(n):
9         sequence.append(a)
10        a, b = b, a+b
11    return sequence
12
13 n = int(input("Enter the number of terms for iterative approach: "))
14 print("Fibonacci Sequence (Iterative):")
15 for num in fibonacci_iterative(n):
16     print(num, end=" ")
17
18 print()
19
20
```
- Terminal:** Shows the command line output:

```
PS C:\Users\manis\OneDrive\Documents\VSCode\Python\Exercises> python L1.py
Enter the number of terms: 2
Fibonacci Sequence:
0 1
```
- Output Panel:** Shows the output of the Python debugger.
- Right Sidebar:** Includes a "Build with Agent" feature panel.

EXPLANATION:

1. A function Fibonacci iterative is created to generate the Fibonacci series using a loop-based method
2. It starts with two initial values, 0 and 1, which are the first two Fibonacci numbers.
3. A list is used to store each Fibonacci number as it is calculated inside the loop.
4. The values of a and b keep updating to give the next number in the sequence.
5. After taking user input, the function returns the full sequence, and the program prints it as the Fibonacci series