LAB-1

H.TNO:2303A51804

Batch:28

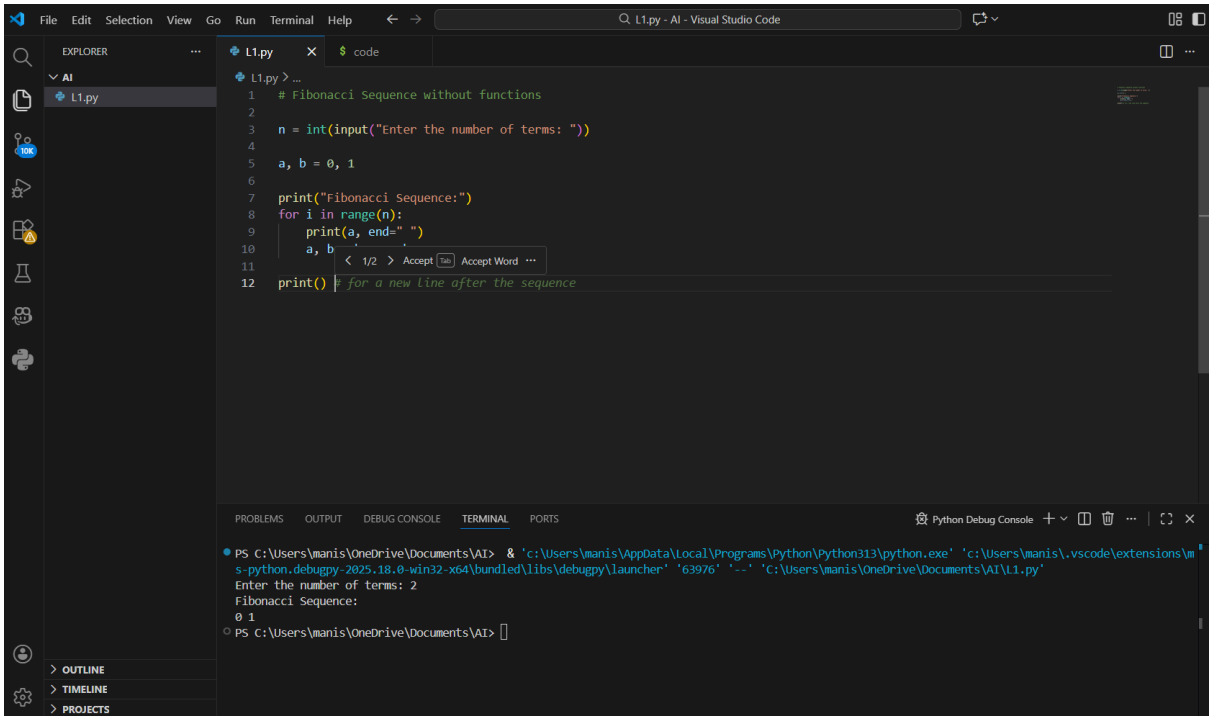
TASK-1

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

PROMPT:

Fibonacci Sequence without functions

CODE AND OUTPUT:



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:

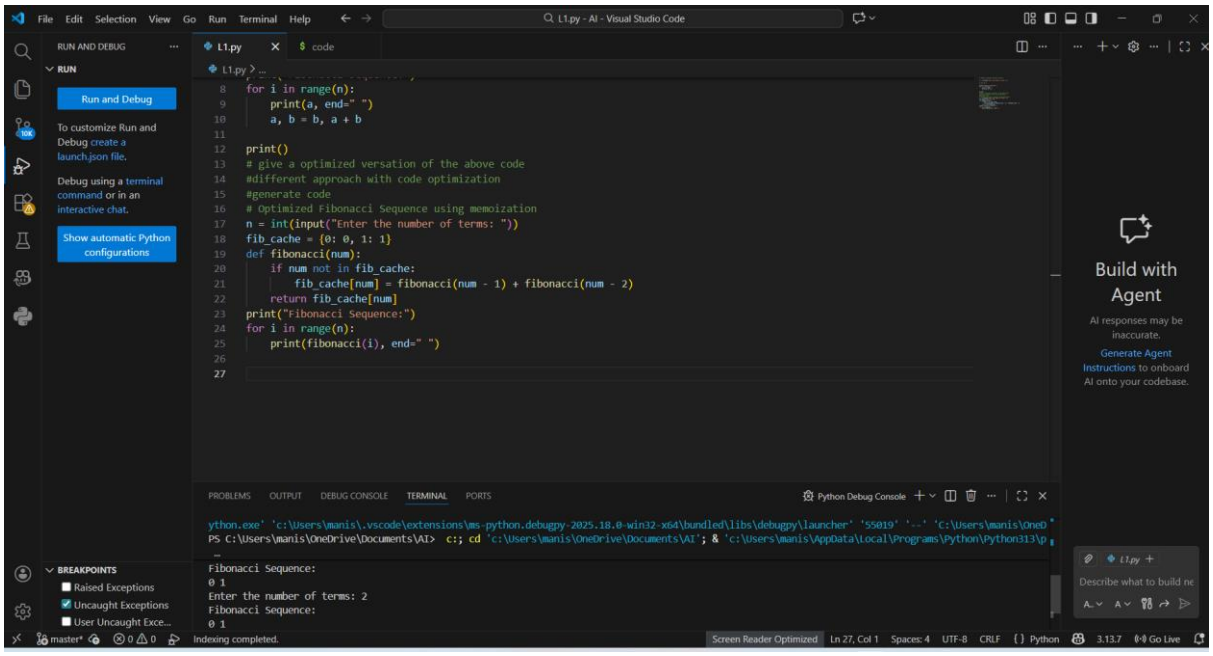
Give an optimised version of the above code

#different approach with code optimisation

#Generate code

Optimised Fibonacci Sequence using memorisation

CODE AND OUTPUT:



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)

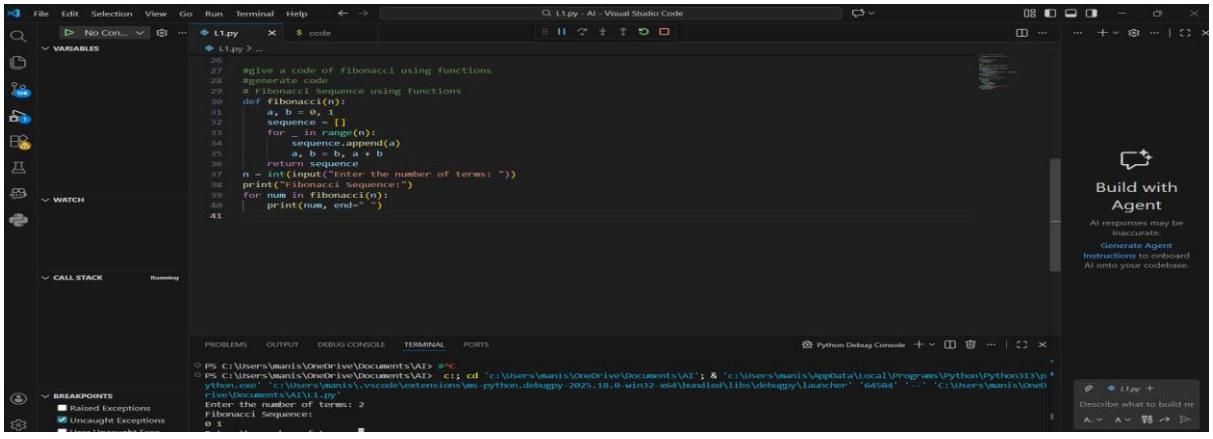
PROMPT:

Give a code of Fibonacci using functions

#Generate code

Fibonacci Sequence using functions

CODE AND OUTPUT:



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

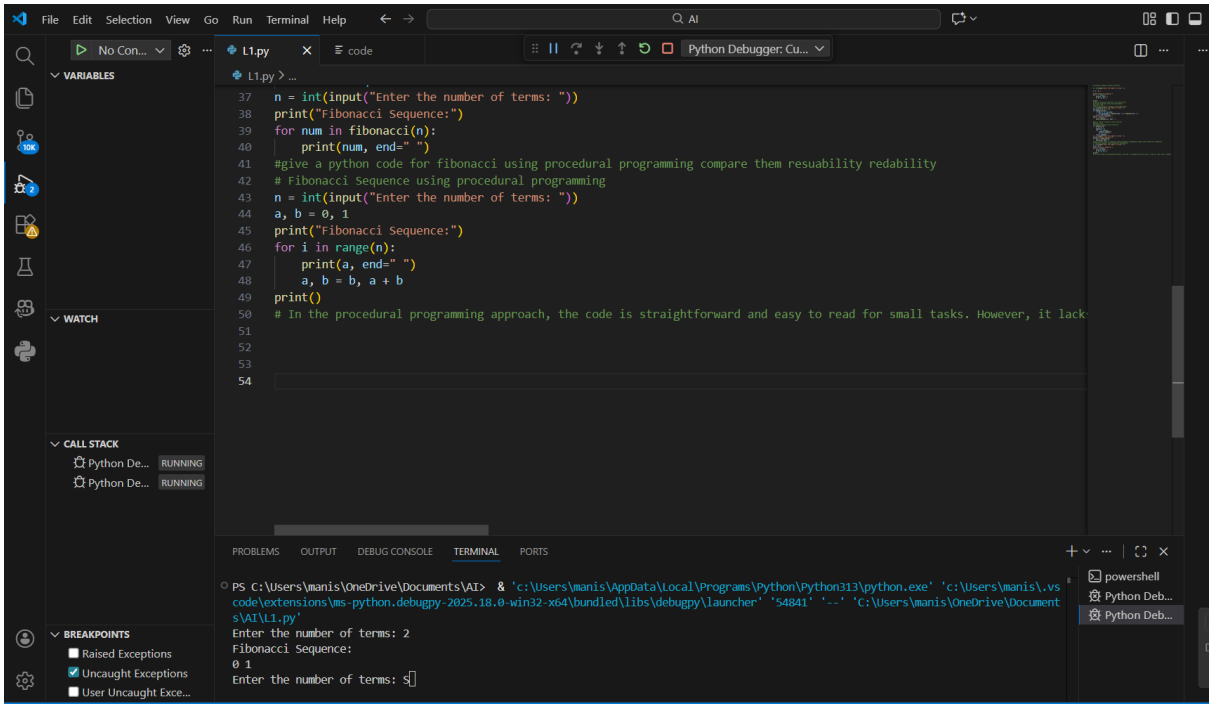
Analysis – Procedural vs Modular Fibonacci Code

PROMPT:

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

Fibonacci Sequence using procedural programming

CODE AND OUTPUT:



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

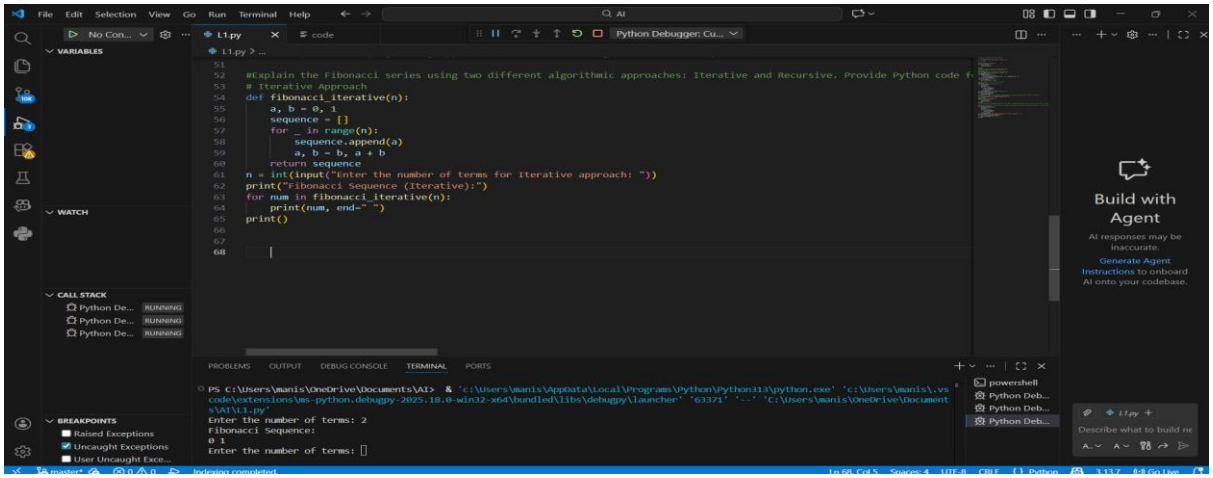
AI-Generated Iterative vs Recursive Fibonacci Approaches (Different Algorithmic Approaches for the Fibonacci Series)

PROMPT:

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

Iterative Approach

CODE AND OUTPUT:



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