Project:

Return a list of first ‘n’ Fibonacci numbers

Team:

Name	SRN	Section
Anish Umesh Naidu	PES1UG25CS074	C3
Arpan Saha	PES1UG25CS096	C3
Bhavin VM	PES1UG25BT010	C3

Problem Statement:

The Fibonacci sequence is a foundational concept in mathematics and computer science, known for its rapid, exponential growth. For learners and analysts, understanding the relationship between the term index (***n***) and the resulting Fibonacci number (***F(n)***) is critical. Manually calculating and comparing these large numbers becomes tedious and inefficient. Develop a program that can efficiently generate the first ***n*** terms of the Fibonacci sequence (starting from 0 and 1) and simultaneously provide a clear, visual representation of this growth.

Approach/Methodology/Data structures used:

Data Structure: List

The core data structure used is the **Python `list`**.

- **Usage:** The `fibonacci` function initializes the sequence as `seq = [0, 1]` and then continuously appends the next calculated term to this list.
- **Advantage:** Lists provide efficient appending (`.append()`), which is ideal for building the sequence step-by-step. They also maintain the **order** of the sequence, allowing easy access to the preceding two elements (`seq[-1]` and `seq[-2]`) needed for the calculation.

Methodology/Algorithm: Iterative Approach

The program uses an **iterative (looping) methodology** to generate the Fibonacci numbers. This is a very efficient and simple way to calculate the sequence.

1. **Initialization:** The function handles the base cases:
    - If n <= 0, it returns an empty list.
    - If n = 1, it returns [0].
    - For n >= 2, it **initializes the list** with the first two terms: [0, 1].
2. **Iteration (The Loop):** A while loop runs as long as the current length of the sequence (len(seq)) is less than the desired number of terms (n).
3. **Calculation:** Inside the loop, the next Fibonacci number is calculated by **summing the last two numbers** in the existing list: seq[-1] + seq[-2].
4. **Growth:** This new number is then **appended** to the list (seq.append(...)).

This iterative approach is computationally light because it only requires **constant time** (O(1)) to calculate each new number after the sequence is initialized, making the overall complexity for generating n terms O(n) (linear time).

Data Visualization

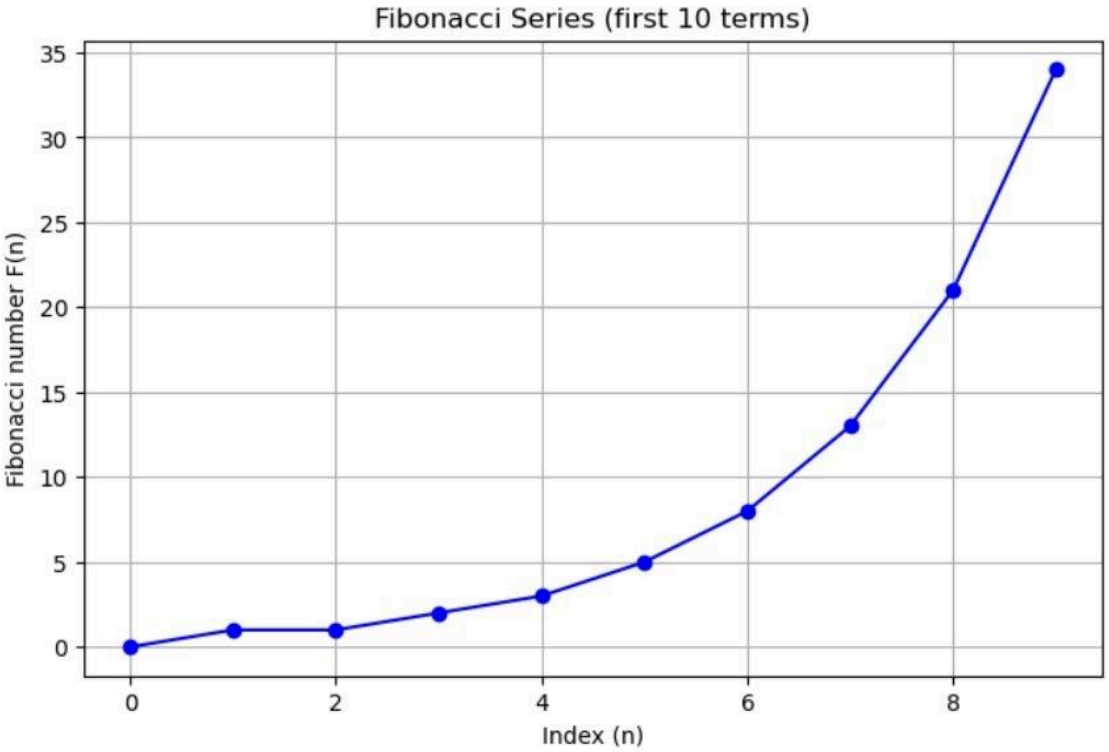
The program utilizes the **matplotlib library** to add a final step: **Data Visualization**.

- The plot_fibonacci function takes the completed list (seq) and plots it on a graph.
- The **x-axis** represents the **index** (or position n) of the number.
- The **y-axis** represents the **value** of the Fibonacci number F(n).
- This visual component clearly illustrates the **exponential growth** characteristic of the Fibonacci sequence

Sample Input:

Enter number of Fibonacci terms to generate: 10

Sample Output:



Challenges faced:

Learning List concepts & matplotlab library functions to plot the graph.