**EXERCISE 7: Financial Forecasting**

**Understanding Recursive Algorithms**

**Concept of Recursion:**

* **Definition:** Recursion is a method of solving a problem where the solution depends on solutions to smaller instances of the same problem. A recursive function calls itself with modified parameters to solve a problem incrementally.
* **Base Case:** The condition under which the recursion stops. This is crucial to prevent infinite recursion.
* **Recursive Case:** The part of the function where it calls itself with a different argument, aiming to reduce the problem size and move towards the base case.

**Benefits of Recursion:**

* **Simplicity:** Recursive solutions can be more intuitive and easier to understand for problems that naturally fit recursive decomposition.
* **Code Reduction:** Can simplify code and reduce the complexity of implementation for problems like tree traversal, divide-and-conquer algorithms, and mathematical computations.

**Analysis**

**Time Complexity:**

* **Time Complexity:** O(n)
* Each recursive call represents a period, and there are nnn recursive calls (where n is the number of periods). Therefore, the time complexity is linear in relation to the number of periods.

**Space Complexity:**

* **Space Complexity:** O(n)
* The space complexity is proportional to the number of recursive calls on the call stack. Each call adds a new frame to the stack, so with n periods, the maximum stack depth is n.

**Optimization Techniques:**

* **Memoization:**
* **Description:** Store the results of expensive function calls and reuse the cached result when the same inputs occur again.
* **Application:** Although not strictly necessary in this case since each period calculation is unique, memoization can be useful for more complex recursive problems to avoid redundant calculations.
* **Iterative Approach:**
* **Description:** Convert the recursive approach to an iterative one to avoid the overhead of recursive calls and stack management.
* **Implementation:** Using a loop to calculate future values would be more efficient in terms of space complexity.