**Recursion** is a programming technique where a function calls itself to solve smaller instances of the same problem. It's especially useful when a task can be divided into similar sub-tasks, such as in mathematical computations or tree traversals.

A recursive function typically includes two parts:

**1. Base Case:**  
This is the condition where recursion stops. It prevents infinite calls and ensures termination.

**2. Recursive Case:**  
This part performs the computation and calls the function again with updated parameters, gradually moving toward the base case.

**Analysis:**  
In our financial prediction example, the recursive solution has a time complexity of **O(n)**, where n is the number of years. While simple and elegant, recursion can lead to stack overflow or performance issues for large inputs.

**Optimization:**  
To improve efficiency, we can use an **iterative approach** instead. A simple loop that multiplies the value over n years avoids the overhead of recursive calls and uses constant stack space.

All these approaches are demonstrated in the Test class for comparison and understanding.