## Basic Recursion

**Concept**: Recursion is a method of solving a problem where the solution depends on solutions to smaller instances of the same problem.

Such problems can generally be solved by iteration, but this needs to identify and index the smaller instances at programming time. Recursion solves such recursive problems by using functions that call themselves from within their own code.

Suppose we have to determine the sum of first n natural numbers, there are several ways of doing that but the simplest approach is simply add the numbers starting from 1 to n. So the function simply looks like,

$$f(n) = 1 + 2 + 3 + \dots\dots.. + n$$

But the problem with this approach is that it is slow. When we are iterating we have to keep the count and a variable to hold the sum. Also, the worst case running time is linear.

The recursive approach to solve this problem is,

$$f(n) = 1 \qquad n=1$$

$$f(n) = n + f(n-1) \quad n>1$$

Recursion is a tradeoff between time complexity and space complexity. When you recursively divide the work, the number of instances corresponding to space complexity will increase but running time significantly decreases.

**Problem Statement**: Implement a method which calculates the value of **nth** fibonacci number.

The Fibonacci sequence, is a sequence such that each number is the sum of the two preceding ones, starting from 0 and 1.

Fibonacci Sequence = 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144,.....

**Sample Test case**

**Input**:

    Nth_fibo = 9 # nth fibonacci value to be computed

**Output**:

    "Fibonacci value at position 9 is 34"

**Pseudocode**

```
Recursion follows a very minimal format, of the form:

Method recursion(parameter):

    If parameter is base_case:
        Return something
    Else:
        #Recurse step
```