Chapter I : Recursivity

Introduction

Recursion is a fundamental concept in computer science where a function calls itself to break down a problem into smaller subproblems. This approach is particularly useful in problems that exhibit **self-similarity** and can be broken down into simpler versions of the same problem.

Recursivity Algorithm

A recursive algorithm is an algorithm that invokes itself during execution with a reduced version of itself, as it proceeds by reducing a problem to the same problem with smaller input. A recursive algorithm consists of the base case and the general case.

1. Basic Concept of Recursion

A recursive function consists of:

1. Base case: A condition that stops the recursion.
2. Recursive case: A rule that breaks the problem into smaller instances.

For example, the factorial function is defined as:

n!=n×(n−1)!n! = n \times (n-1)!n!=n×(n−1)!

with the base case:

0!=10! = 10!=1

Implementation in Python

```python
def factorial(n):
    if n == 0:
        return 1  # Base case
    return n * factorial(n - 1)  # Recursive call

print(factorial(5))  # Output: 120
```

Example -2-

For example, consider this problem statement: Print sum of n natural numbers using recursion. This statement clarifies that we need to formulate a function that will calculate the summation of all natural numbers in the range 1 to n. Hence, mathematically you can represent the function as:

F(n) = 1 + 2 + 3 + 4 + ……..+ (n-2) + (n-1) + n

It can further be simplified as:

$$F(n) = \sum_{k=1}^{k=n} (k)$$

You can breakdown this function into two parts as follows:

Breakdown of Problem Statement

$$F(n) = \sum_{k=0}^{k=n} (k)$$

Base Case

```
If(n == 0)
    {
        Return;
    }
```

Recursive step

F(n) = n + F(n-1)