Recursion is a programming concept where a function solves a problem by calling itself with smaller subproblems until it reaches a base case. It's like solving a large problem by breaking it down into smaller versions of the same problem.

The key conditions required for recursive programming are:

1. **Base Case(s):**

* You must have at least one base case that can be solved directly without recursion
* This prevents infinite recursion and provides a stopping condition
* Example: For calculating factorial, the base case is n=0 or n=1

1. **Recursive Case(s):**

* The problem must be reducible to smaller subproblems of the same type
* Each recursive call should move closer to the base case
* The solution to the original problem can be built from solutions to subproblems

**Example:**

**def** factorial(n):

# Base case

if n == 0 or n == 1:

**return** 1

# Recursive case

**return** n \* factorial(n-1)

In this example:

* Base case: When n is 0 or 1, we return 1
* Recursive case: For any other n, we multiply n by factorial(n-1)
* Each recursive call reduces n by 1, bringing us closer to the base case
* The problem of n! is broken down into calculating (n-1)!