**6.9 Permutation**

**Aim:** The aim is to generate every possible arrangement, known as a permutation, for a given list of distinct numbers.

**Algorithm:**

1. Let the input be a list of distinct numbers:  
  nums = [n₁, n₂, …, nk]

2. The total number of possible permutations is k! (factorial of k).

3. We define a recursive procedure:  
  permute(start, nums, result)  
  - start → the index at which we fix elements  
  - nums → current arrangement of numbers  
  - result → collection of all generated permutations

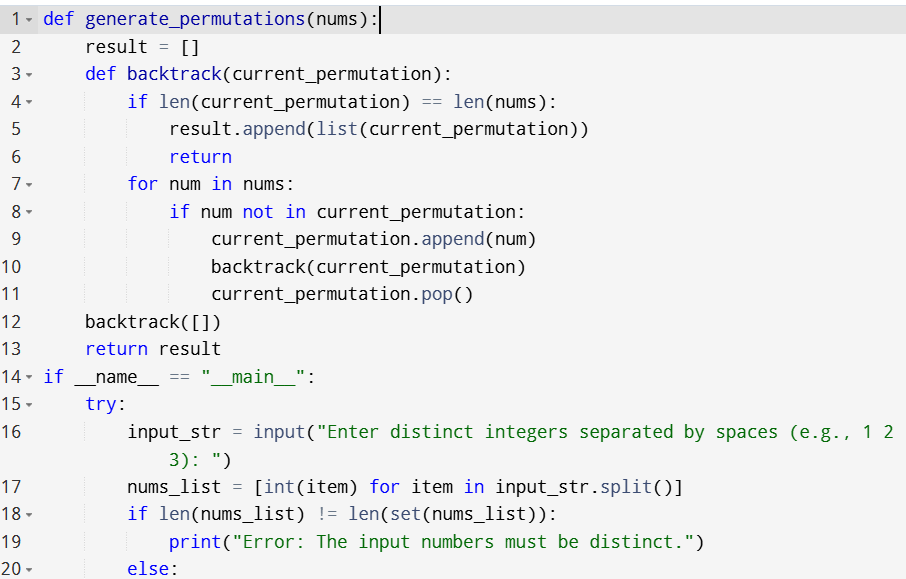
4. Base condition:  
  - If start = k (all positions fixed), add a copy of nums to result.

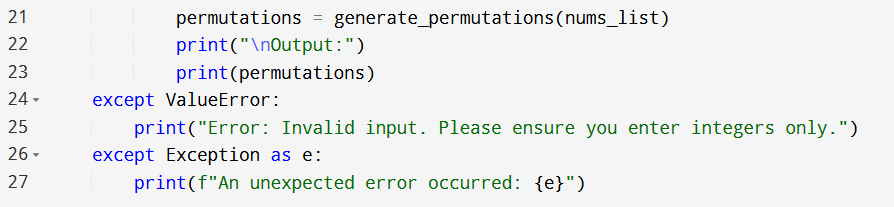
5. Recursive case:  
  - For each index i from start to k−1:  
    a. Swap nums[start] and nums[i] (fix one number at position start).  
    b. Recursively call:  
      permute(start+1, nums, result)  
    c. Swap back nums[start] and nums[i] (backtrack to restore order).

6. Initially, call the procedure as:  
  permute (0, nums, result)

7. The final collection result will contain all possible permutations of the list.

**Program:**

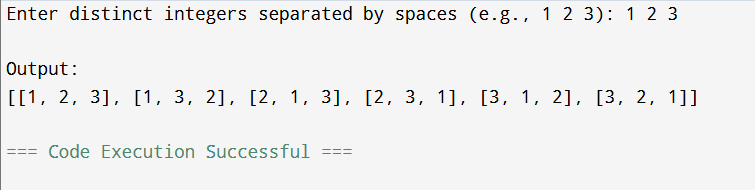
****



**Input:**

* Enter distinct integers separated by spaces (e.g., 1 2 3):

**Output:**

****

**Result:** Thus, the program is executed successfully and output is verified.

**Performance analysis:**

* Time Complexity: O(n\*n)
* Space Complexity: O(n).