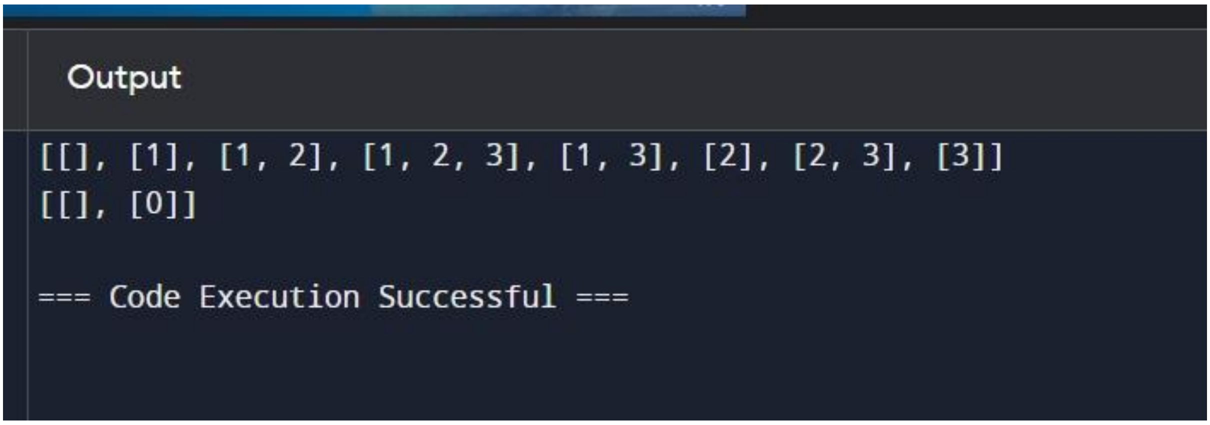
221.Write a program to implement the concept of subset generation. Given a set of unique integers and a specific integer 3, generate all subsets that contain the element 3. Return a list of lists where each inner list is a subset containing the element 3 E = [2, 3, 4, 5], x = 3, The subsets containing 3 :  [3], [2, 3], [3, 4], [3,5], [2, 3, 4], [2, 3, 5], [3, 4, 5], [2, 3, 4, 5] Given an integer array nums of unique elements, return all possible  subsets(the power set). The solution set must not contain duplicate subsets. Return the solution in any order.

Example 1:<br>
Input: nums = [1,2,3]<br>
Output: [[],[1],[2],[1,2],[3],[1,3],[2,3],[1,2,3]]<br>
Example 2:<br>
Input: nums = [0]<br>
Output: [[],[0]]

PROGRAM:-

```
def subsets(nums):
    def backtrack(start, subset):
        # Add the current subset to the results
        result.append(subset[:])

        # Explore all possible subsets that can be formed by adding nums[i] to current subset
        for i in range(start, len(nums)):
            subset.append(nums[i])
            backtrack(i + 1, subset)  # Move to the next element to avoid duplicates
            subset.pop()  # Backtrack: remove nums[i] and try the next element

    result = []
    backtrack(0, [])  # Start backtracking from index 0 with an empty subset
    return result

# Example usage:
nums1 = [1, 2, 3]
print(subsets(nums1))  # Output: [[], [1], [2], [1, 2], [3], [1, 3], [2, 3], [1, 2, 3]]

nums2 = [0]
print(subsets(nums2))  # Output: [[], [0]]
```

OUTPUT:-



Output

```
[[], [1], [1, 2], [1, 2, 3], [1, 3], [2], [2, 3], [3]]
[[], [0]]

=== Code Execution Successful ===
```

TIME COMPLEXITY:- O(2^n),