**2.14 Knapsack Problem**

**Aim**: The aim of this program is to solve the 0–1 Knapsack Problem using the exhaustive search approach.

**Algorithm:**

1. Start

2.Input the list of weights, values, and the knapsack capacity.

3.Generate all possible subsets of items using intercools combinations.

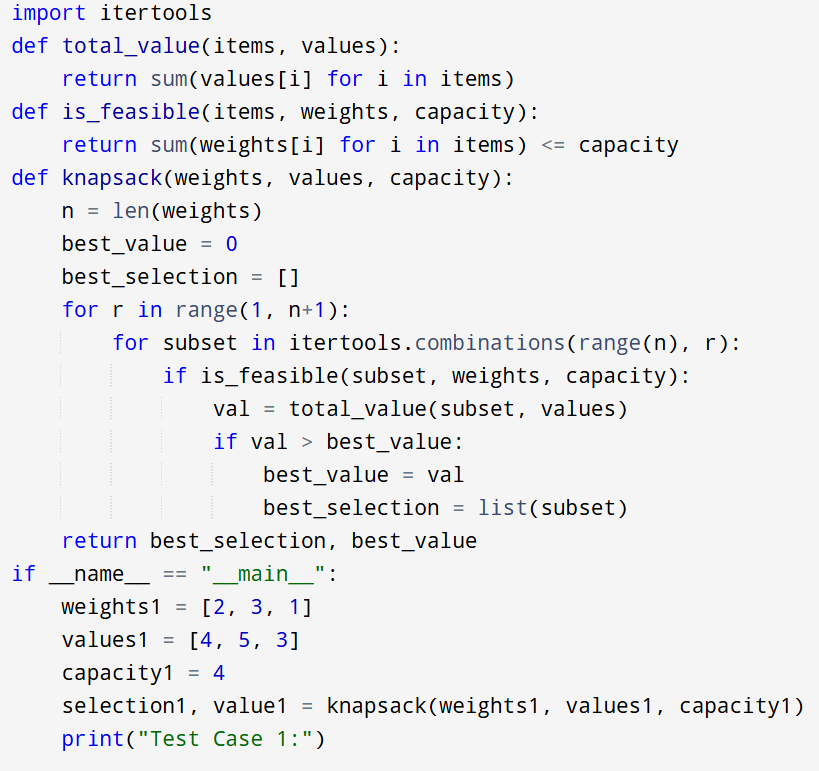
4.For each subset:

* Check feasibility (total weight ≤ capacity).
* Compute total value.
* Keep track of the subset with the maximum total value.

5.Return the optimal selection of items and the maximum value.

6.Stop

**Program:**

****

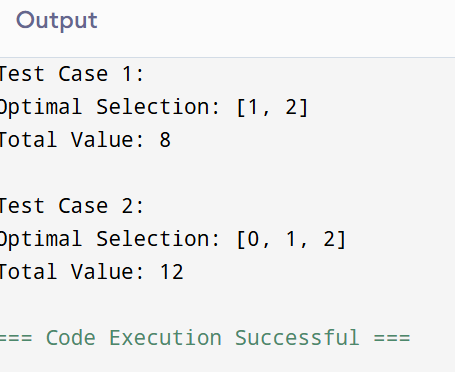
**Input:**

Weights: [2, 3, 1]

Values: [4, 5, 3]

Capacity: 4

**Output:**

****

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

**Performance analysis:**

Time Complexity:

* Generating all subsets: O(2^n)
* Computing weight and value per subset: O(n)
* Total = O(n \* 2^n)

Space Complexity:

* Storing one subset at a time: O(n)
* No extra storage of all subsets (only iterated).
* Overall = O(n)