**2.14** You are given a list of items with their weights and values. Develop a program that utilizes **exhaustive search to solve the 0-1 Knapsack Problem**. The program should:

-Define a function total\_value(items, values) that takes a list of selected items (represented by their indices) and the value list as input. It iterates through the selected items and calculates the total value by summing the corresponding values from the value list.

-Define a function is\_feasible(items, weights, capacity) that takes a list of selected items (represented by their indices), the weight list, and the knapsack capacity as input. It checks if the total weight of the selected items exceeds the capacity.

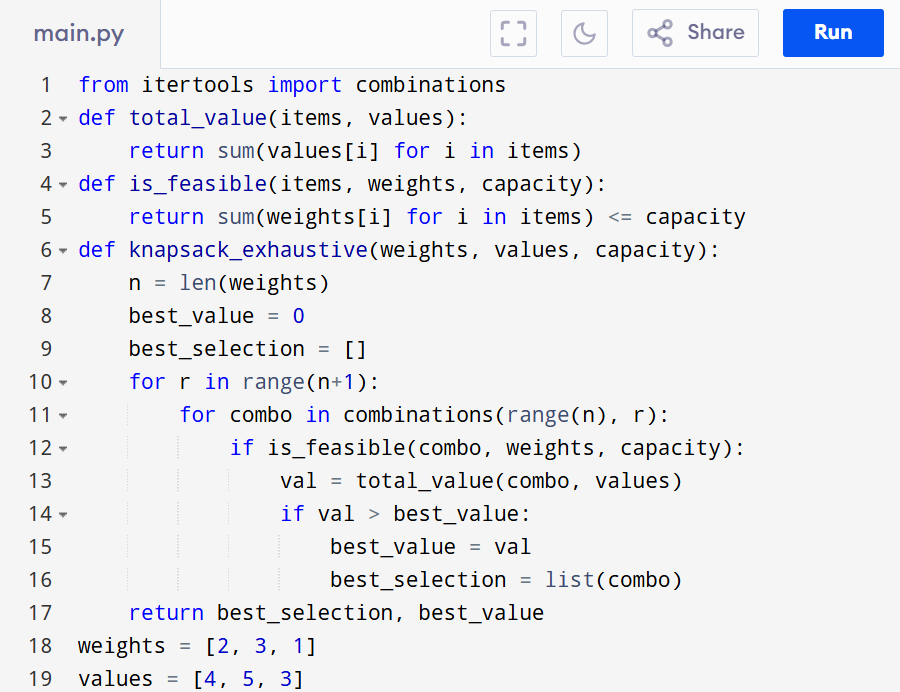
**AIM**

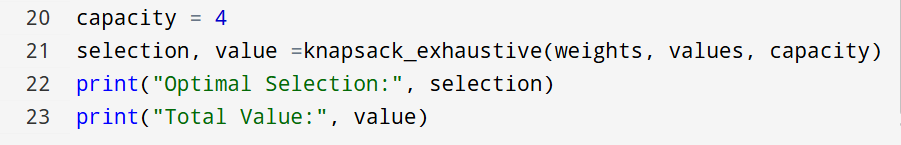
To find the optimal selection of items that maximizes total value without exceeding the knapsack’s weight capacity, using exhaustive search.

**ALGORITHM**

1. **Start**
2. Define total\_value(items, values) to Sum values of selected item indices from the values list.
3. Define is\_feasible(items, weights, capacity) to Calculate total weight of selected item indices.
4. Return True if total weight ≤ capacity, otherwise False.
5. For each combination:-Check feasibility with is\_feasible.
6. If feasible and value is higher than current best, update best selection and value.
7. Return the optimal selection and maximum total value.
8. · **End**

**PROGRAM**





Input:

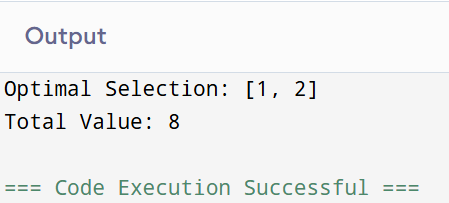
Items: 3 (indices 0, 1, 2)

Weights: [2, 3, 1]

Values: [4, 5, 3]

Capacity: 4

Output:



**RESULT:**

Thus Solving the 0-1 Knapsack Problem using Exhaustive Search is successfully executed and the output is verified.

**PERFORMANCE ANALYSIS:**

· **Time Complexity:** O(2^n)

· **Space Complexity:** O(n)