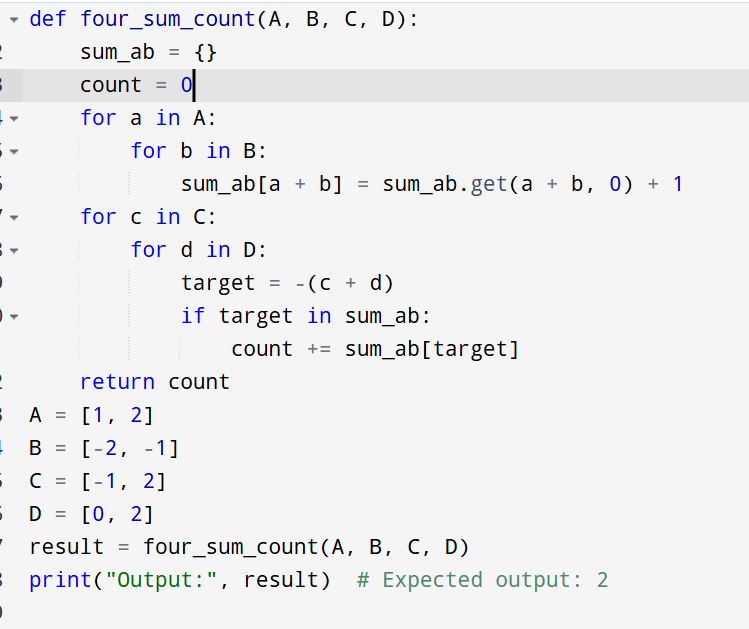
**3.12 Median of Medians Algorithm**

**Aim:** To find the k-th smallest element in an unsorted list **in guaranteed linear time (O(n))**, overcoming the worst-case inefficiency of simpler selection algorithms like Quickselect.

**Algorithm:**

1. Divide arr into groups of 5 elements.
2. Find the median of each group.
3. Recursively find the median of medians.
4. Use this median as pivot to partition arr.
5. Recur on the partition that contains the k-th smallest.

**Program:**

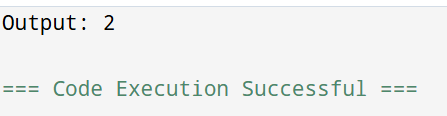
****

**Input:**

arr1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

arr2 = [23, 17, 31, 44, 55, 21, 20, 18, 19, 27]

**Output:**

****

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

**Performance analysis:**

* Time Complexity: The worst-case time complexity of the Median of Medians algorithm is O(n). This is because:
  + Dividing into groups of 5 takes O(n).
  + Finding medians of groups is O(n).
  + Recursively finding median of medians is T(n/5).
  + Partitioning around the pivot is O(n).
  + The recursion call on a smaller subset (at most 7n/10 elements).

This results in the recurrence:

T(n)≤T(n/5)+T(7n/10)+O(n)T(n) \leq T(n/5) + T(7n/10) + O(n)T(n)≤T(n/5)+T(7n/10)+O(n)

which solves to O(n).

* Space Complexity: O(n) due to recursion stack and list slicing.