Quick Sort

Quick Sort is a divide and conquer sorting algorithm.
It works by selecting a pivot element, partitioning the array around the pivot, and then recursively sorting the subarrays.

Steps:

1. Choose a pivot element from the array.
2. Partition the array — elements smaller than pivot go to the left, larger ones go to the right.
3. Recursively apply Quick Sort to left and right subarrays.

Variants:

1. Deterministic Quick Sort:
   - Pivot is chosen in a fixed way (e.g., first or last element).
   - Performance depends on input order.
2. Randomized Quick Sort:
   - Pivot is chosen randomly from the array.
   - Avoids worst-case behavior on already sorted data.

Variant	Best Case	Average Case	Worst Case	Space
Deterministic	O(n log n)	O(n log n)	$O(n^2)$	O(log n)
Randomized	O(n log n)	O(n log n)	O(n log n) (almost always)	O(log n)

code

```
import random

# ----- Deterministic Quick Sort -----

def deterministic_quick_sort(arr):

    if len(arr) <= 1:

        return arr
    pivot = arr[-1]  # last element as pivot
    left = [x for x in arr[:-1] if x <= pivot]
    right = [x for x in arr[:-1] if x > pivot]
    return deterministic_quick_sort(left) + [pivot] + deterministic_quick_sort(right)
# ----- Randomized Quick Sort -----
def randomized_quick_sort(arr):
    if len(arr) <= 1:
        return arr
    pivot = random.choice(arr)  # random pivot
    left = [x for x in arr if x < pivot]
    middle = [x for x in arr if x == pivot]
    right = [x for x in arr if x > pivot]
    return randomized_quick_sort(left) + middle + randomized_quick_sort(right)
# ----- MAIN PROGRAM -----
n = int(input("Enter number of elements: "))
arr = []
for i in range(n):
    arr.append(int(input(f"Enter element {i+1}: ")))
print("\nOriginal Array:", arr)
# Deterministic
sorted_deterministic = deterministic_quick_sort(arr)
print("Sorted using Deterministic Quick Sort:", sorted_deterministic)
# Randomized
sorted_randomized = randomized_quick_sort(arr)
print("Sorted using Randomized Quick Sort:", sorted_randomized)
```