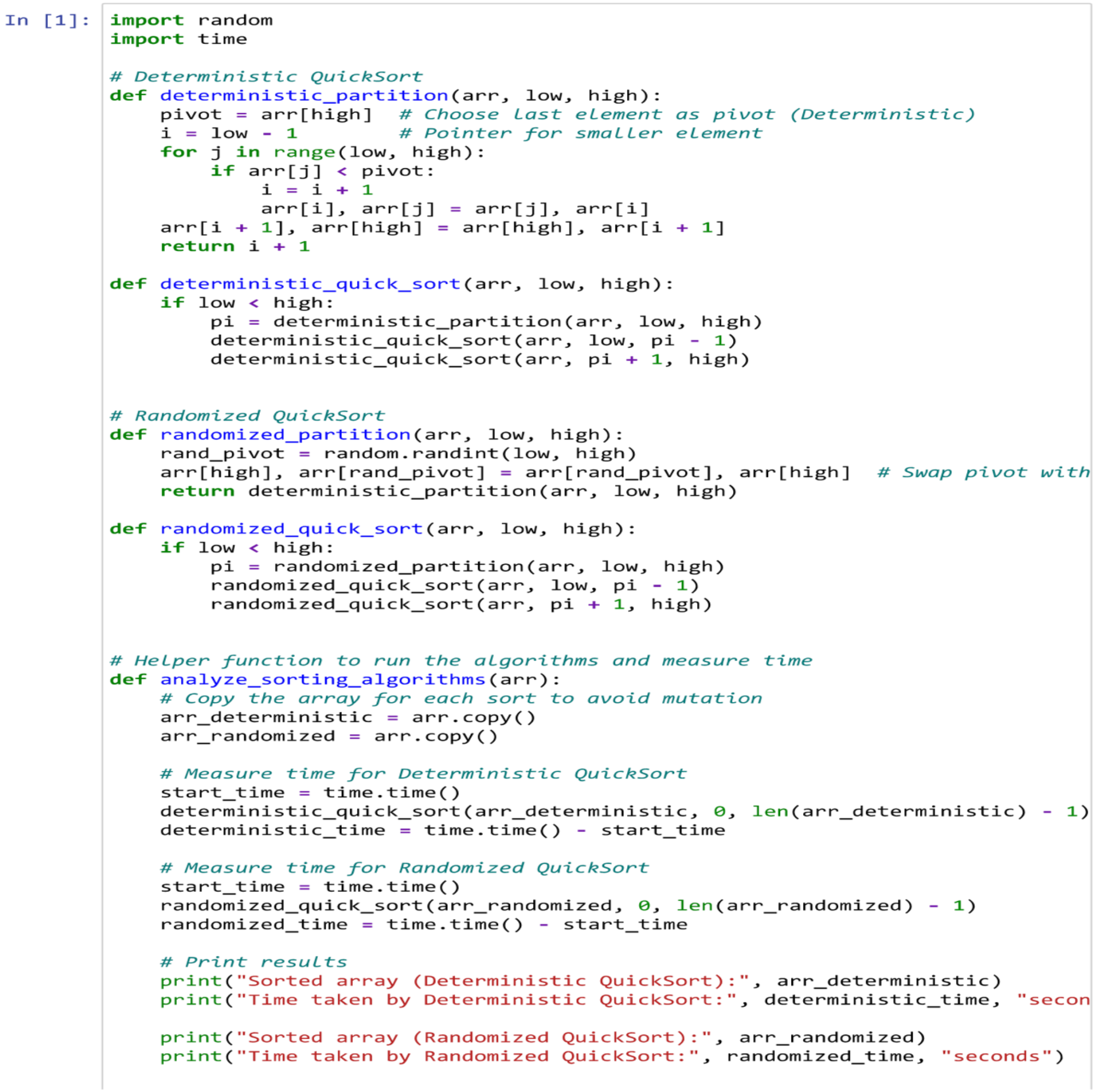
Name: Vaishnavi Pravin Kolse

Class: BE - A

Roll No.37

Practical No.6

Aim: Write a program for analysis of quick sort by using deterministic and randomized variant.



In [1]:

```
import random
import time

# Deterministic QuickSort
def deterministic_partition(arr, low, high):
    pivot = arr[high]  # Choose last element as pivot (Deterministic)
    i = low - 1        # Pointer for smaller element
    for j in range(low, high):
        if arr[j] < pivot:
            i = i + 1
            arr[i], arr[j] = arr[j], arr[i]
    arr[i + 1], arr[high] = arr[high], arr[i + 1]
    return i + 1

def deterministic_quick_sort(arr, low, high):
    if low < high:
        pi = deterministic_partition(arr, low, high)
        deterministic_quick_sort(arr, low, pi - 1)
        deterministic_quick_sort(arr, pi + 1, high)


# Randomized QuickSort
def randomized_partition(arr, low, high):
    rand_pivot = random.randint(low, high)
    arr[high], arr[rand_pivot] = arr[rand_pivot], arr[high]  # Swap pivot with
    return deterministic_partition(arr, low, high)

def randomized_quick_sort(arr, low, high):
    if low < high:
        pi = randomized_partition(arr, low, high)
        randomized_quick_sort(arr, low, pi - 1)
        randomized_quick_sort(arr, pi + 1, high)


# Helper function to run the algorithms and measure time
def analyze_sorting_algorithms(arr):
    # Copy the array for each sort to avoid mutation
    arr_deterministic = arr.copy()
    arr_randomized = arr.copy()

    # Measure time for Deterministic QuickSort
    start_time = time.time()
    deterministic_quick_sort(arr_deterministic, 0, len(arr_deterministic) - 1)
    deterministic_time = time.time() - start_time

    # Measure time for Randomized QuickSort
    start_time = time.time()
    randomized_quick_sort(arr_randomized, 0, len(arr_randomized) - 1)
    randomized_time = time.time() - start_time

    # Print results
    print("Sorted array (Deterministic QuickSort):", arr_deterministic)
    print("Time taken by Deterministic QuickSort:", deterministic_time, "secon

    print("Sorted array (Randomized QuickSort):", arr_randomized)
    print("Time taken by Randomized QuickSort:", randomized_time, "seconds")
```

```python
# Main code to test the analysis
if __name__ == "__main__":
    n = int(input("Enter number of elements: "))
    arr = [random.randint(1, 1000) for _ in range(n)]  # Generating random arr

    print("\nOriginal array:", arr)
    analyze_sorting_algorithms(arr)
```

```
Enter number of elements: 10

Original array: [788, 948, 438, 261, 999, 659, 109, 988, 72, 814]
Sorted array (Deterministic QuickSort): [72, 109, 261, 438, 659, 788, 814, 94
8, 988, 999]
Time taken by Deterministic QuickSort: 0.0 seconds

Sorted array (Randomized QuickSort): [72, 109, 261, 438, 659, 788, 814, 948,
988, 999]
Time taken by Randomized QuickSort: 0.0 seconds
```

In [ ]: