# Practical No: 6(B)

**Practical Title:** Perform the Quicksort on given array and display top five scores

**Aim:** Write a **Python** program to store first year percentage of students in array. Write function for sorting array of floating-point numbers in ascending order using quick sort and display top five scores.

## Theory:

Sorting is a way of arranging items in a systematic manner. Quicksort is the widely used sorting algorithm that makes **n log n** comparisons in average case for sorting an array of n elements. It is a faster and highly efficient sorting algorithm. This algorithm follows the divide and conquer approach. Divide and conquer is a technique of breaking down the algorithms into subproblems, then solving the subproblems, and combining the results back together to solve the original problem.

**Divide:** In Divide, first pick a pivot element. After that, partition or rearrange the array into two sub-arrays such that each element in the left sub-array is less than or equal to the pivot element and each element in the right sub-array is larger than the pivot element.

**Conquer:** Recursively, sort two subarrays with Quicksort.

**Combine:** Combine the already sorted array.

Quicksort picks an element as pivot, and then it partitions the given array around the picked pivot element. In quick sort, a large array is divided into two arrays in which one holds values that are smaller than the specified value (Pivot), and another array holds the values that are greater than the pivot.

After that, left and right sub-arrays are also partitioned using the same approach. It will continue until the single element remains in the sub-array.
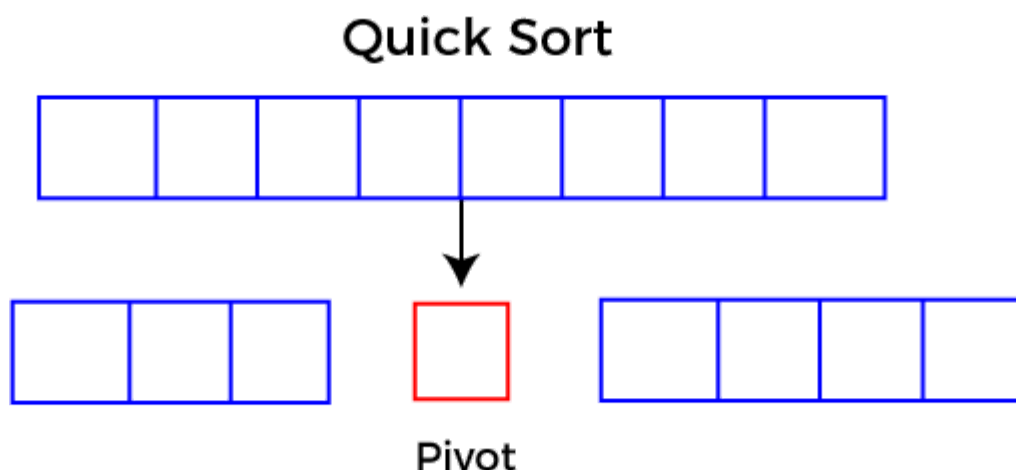


## Choosing the pivot

Picking a good pivot is necessary for the fast implementation of quicksort. However, it is typical to determine a good pivot. Some of the ways of choosing a pivot are as follows –

o   Pivot can be random, i.e. select the random pivot from the given array.

o   Pivot can either be the rightmost element of the leftmost element of the given array.

o   Select median as the pivot element.

## Algorithm:

```
QUICKSORT (array A, start, end)
{
1 if (start < end)
2 {
3 p = partition(A, start, end)
4 QUICKSORT (A, start, p - 1)
5 QUICKSORT (A, p + 1, end)
6 }
}
```

## Partition Algorithm:

The partition algorithm rearranges the sub-arrays in a place.

```
PARTITION (array A, start, end)
{
 1 pivot = A[end]
 2 i = start-1
 3 for j = start to end -1 {
 4 do if (A[j] < pivot) {
5 then i = i + 1
 6 swap A[i] with A[j]
 7 }}
    8   swap A[i+1] with A[end]
 9 return i+1
}
```
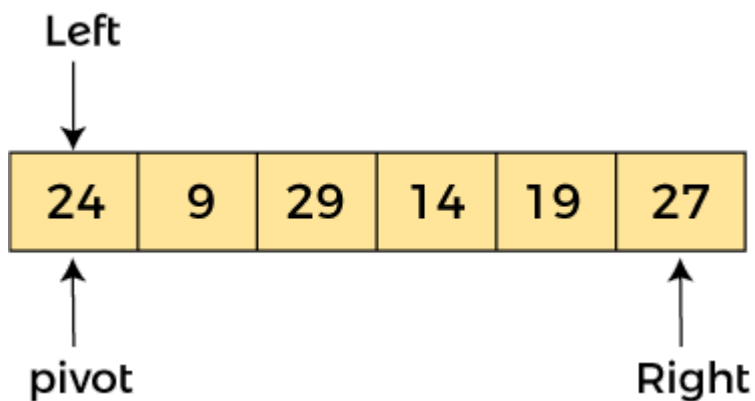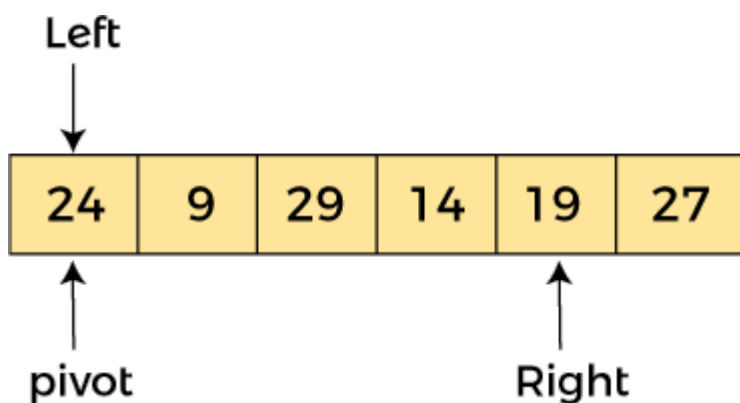
## Example:

Let the elements of array are -

| 24 | 9 | 29 | 14 | 19 | 27 |
| --- | --- | --- | --- | --- | --- |

In the given array, we consider the leftmost element as pivot. So, in this case, a[left] = 24, a[right] = 27 and a[pivot] = 24.

Since, pivot is at left, so algorithm starts from right and move towards left.

**Left**

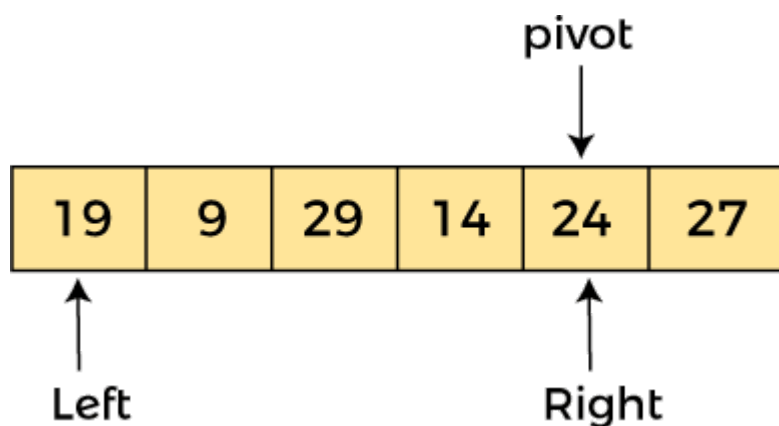| 24 | 9 | 29 | 14 | 19 | 27 |
|----|---|----|----|----|----|

pivot                Right

Now, a[pivot] < a[right], so algorithm moves forward one position towards left, i.e. -

**Left**

| 24 | 9 | 29 | 14 | 19 | 27 |
|----|---|----|----|----|----|

pivot              Right

Now, a[left] = 24, a[right] = 19, and a[pivot] = 24.

Because, a[pivot] > a[right], so, algorithm will swap a[pivot] with a[right], and pivot moves to right, as -

                           **pivot**

| 19 | 9 | 29 | 14 | 24 | 27 |
|----|---|----|----|----|----|

Left                Right
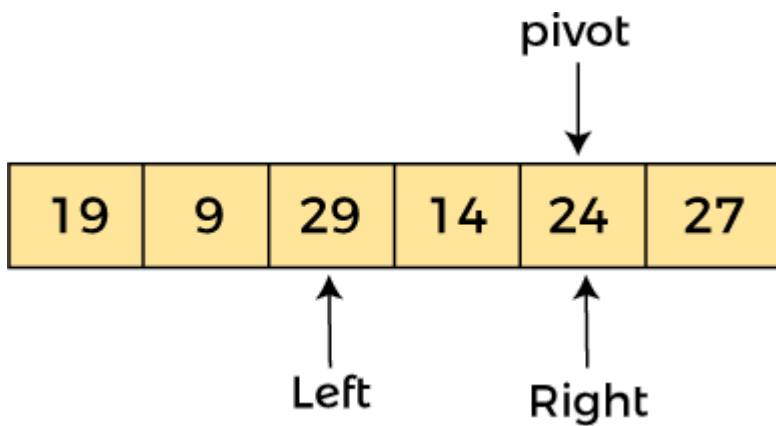
Now, a[left] = 19, a[right] = 24, and a[pivot] = 24. Since, pivot is at right, so algorithm starts from left and moves to right.
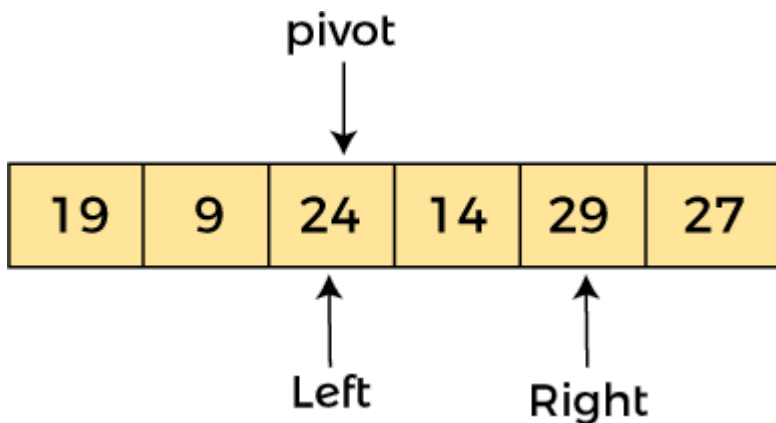
As a[pivot] > a[left], so algorithm moves one position to right as -
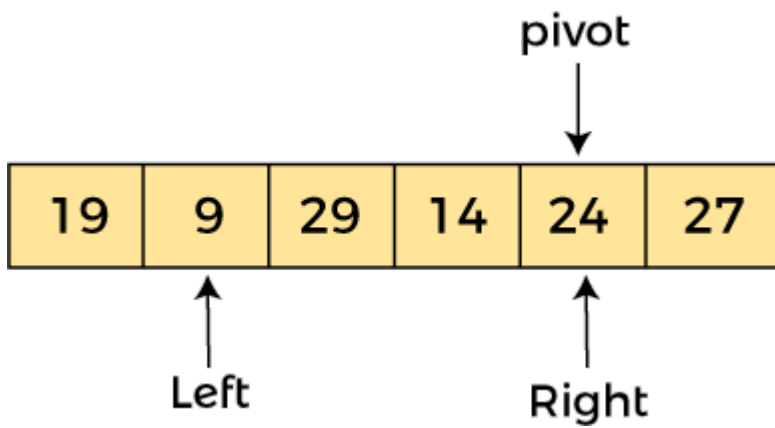
Now, a[left] = 9, a[right] = 24, and a[pivot] = 24. As a[pivot] > a[left], so algorithm moves one position to right as -
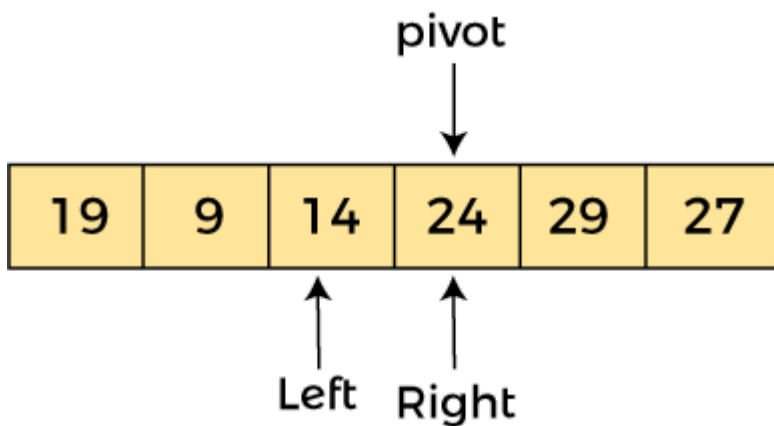


Now, a[left] = 29, a[right] = 24, and a[pivot] = 24. As a[pivot] < a[left], so, swap a[pivot] and a[left], now pivot is at left, i.e. -



Since, pivot is at left, so algorithm starts from right, and move to left. Now, a[left] = 24, a[right] = 29, and a[pivot] = 24. As a[pivot] < a[right], so algorithm moves one position to left, as -
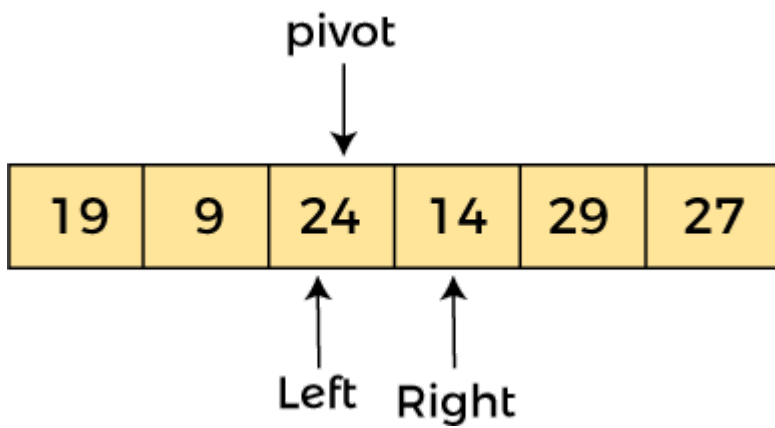
Now, a[pivot] = 24, a[left] = 24, and a[right] = 14. As a[pivot] > a[right], so, swap a[pivot] and a[right], now pivot is at right, i.e. -



Now, a[pivot] = 24, a[left] = 14, and a[right] = 24. Pivot is at right, so the algorithm starts from left and move to right.
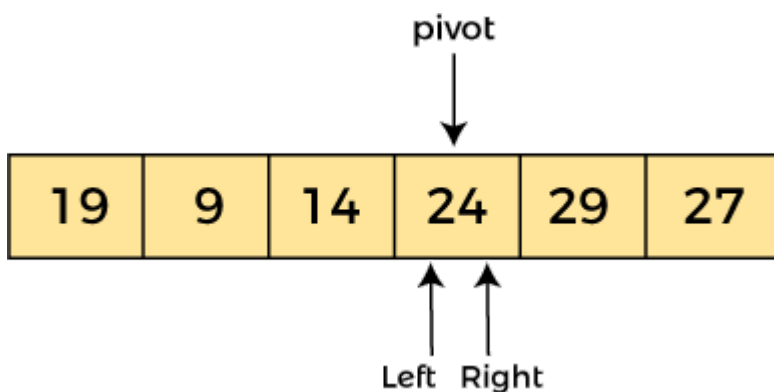


Now, a[pivot] = 24, a[left] = 24, and a[right] = 24. So, pivot, left and right are pointing the same element. It represents the termination of procedure.

Element 24, which is the pivot element is placed at its exact position.

Elements that are right side of element 24 are greater than it, and the elements that are left side of element 24 are smaller than it.

| 19 | 9 | 14 | 24 | 29 | 27 |

Left sub array    Right sub array
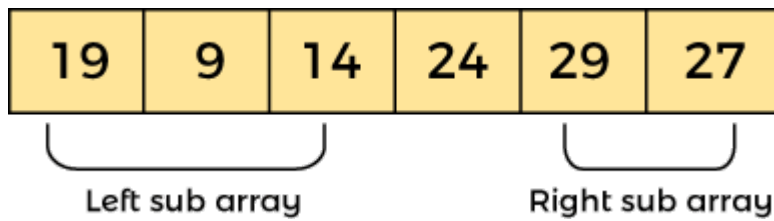
Now, in a similar manner, quick sort algorithm is separately applied to the left and right sub-arrays. After sorting gets done, the array will be -

| 9 | 14 | 19 | 24 | 27 | 29 |
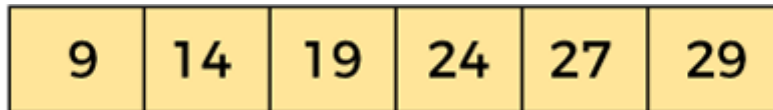
## Conclusion:

In this way, we successfully perform Quicksort on given unsorted array and displayed top five scores