**1. Sorting Functions**

1.1 **quicksort(Percentage, start, end)**

* Implements the quicksort algorithm to sort the **Percentage** list in ascending order.
* Calls the **partition** function to partition the list and recursively sorts the sublists.
* The function takes the list **Percentage** and the indices **start** and **end** as parameters.

1.2 **partition(Percentage, start, end)**

* Selects a pivot element and rearranges the elements in the list such that elements less than the pivot are on the left, and elements greater than the pivot are on the right.
* Returns the index of the pivot after the partitioning process.

**2. Input of Percentages**

* Takes the total number of students (**number**) as input.
* Prompts the user to input the percentage for each student.
* Populates the **Percentage** list with the entered percentages.

**3. Sorting and Displaying Results**

* Calls **quicksort** to sort the **Percentage** list in ascending order.
* Prints the sorted **Percentage** list.

**4. Displaying Top Five Scores**

* Prints the top five scores by accessing the last five elements of the sorted list.
* Uses a loop to iterate over the top five scores, printing each one along with its corresponding rank.

**Note:**

* The code assumes that the user always inputs valid integers for the total number of students and valid floats for their percentages.
* The quicksort algorithm is applied directly to the original **Percentage** list.
* The program outputs the sorted list and the top five scores along with their ranks.
* The index variable is used to display the rank of each top scorer.
* The code could benefit from additional input validation and error handling.
* The **value** variable in the loop for entering percentages is redefined in each iteration, which might be improved for clarity.

Algorithm:Top of Form

1. \*Define the QuickSort function:\*

- quicksort(Percentage, start, end)

- Check if the sublist has more than one element (end - start > 1).

- Call the partition function to find the pivot index (p).

- Recursively apply QuickSort to the left and right sublists.

2. \*Define the Partition function:\*

- partition(Percentage, start, end)

- Choose the pivot element (in this case, it's the first element of the sublist).

- Initialize two pointers i and j to the start and end of the sublist.

- While i is less than or equal to j, find elements on the left and right sides of the pivot that are out of place and swap them.

- If i is not less than or equal to j, swap the pivot with the element at index j and return j.

3. \*Input:\*

- Take input for the total number of students (number).

- Use a loop to input the percentage values for each student and append them to the Percentage list.

4. \*Sort the Percentage list using QuickSort:\*

- Call quicksort(Percentage, 0, len(Percentage)) to sort the list.

5. \*Print the sorted list:\*

- Print the sorted Percentage list.

6. \*Print the top five scores:\*

- Calculate the minimum and maximum indices for the top five scores.

- Use a loop to iterate over the top five scores and print the corresponding information.