3.Write a Python program for sorting a list of elements using selection sort algorithm:

a. Assume two lists: Sorted list- Initially empty and Unsorted List-Given input list.

b.In the first iteration, find the smallest element in the unsorted list and place it in the sorted list.

C. In the second iteration, find the smallest element in the unsorted list and place it in the correct position by comparing with the element in the sorted list.

d. In the third iteration, again find the smallest element in the unsorted list and place it in correct position by comparing with the elements in the sorted list.

e.This process continues till the unsorted list becomes empty.

f. Display the sorted list.

In [38]:
```
def selection_sort(input_list,size):
    sorted_list = list() # Initialize an empty sorted list

    for ind in range(size):
        min_index = ind

        for j in range(ind + 1, size):
            # select the minimum element in every iteration
            if lst[j] < lst[min_index]:
                min_index = j
         # swapping the elements to sort the array
        (lst[ind], lst[min_index]) = (lst[min_index], lst[ind])

    return sorted_list

# Input unsorted list
lst = [200,500,100,55,54,786,111,0.15]
size = len(lst)


# Call the selection_sort function to sort the list
selection_sort(lst,size)

# Print the sorted list
print("Sorted List:", lst)
```

```
Sorted List: [0.15, 54, 55, 100, 111, 200, 500, 786]
```

4.Write a Python program for sorting a list of elements using insertion sort algorithm:

a. Assume two lists: Sorted list- Initially empty and Unsorted List-Given input list.

b. In the first iteration, take the first element in the unsorted list and insert it in Sorted list.

C. In the second iteration, take the second element in the given list and compare with the element in the sorted sub list and place it in the correct position.

d. In the third iteration, take the third element in the given list and compare with the elements in the sorted sub list and place the elements in the correct position.

e. This process continues until the last element is inserted in the sorted sub list.

f. Display the sorted elements.

In [36]:
```
def insertion_sort(input_list1):
    sorted_list1 = list(input_list1) # Create a new list to store the sorted el

    for i in range(1, len(sorted_list1)):
        key = sorted_list1[i] # Current element to be inserted into the sorted
        j = i - 1

        while j >= 0 and key < sorted_list1[j]:
            sorted_list1[j + 1] = sorted_list1[j] # Shift larger elements to th
            j -= 1

        sorted_list1[j + 1] = key # Insert the current element into the correct

    return sorted_list1

# Example usage:
unsorted_list = [200,500,100,55,54,786,111,0.15]
sorted_list1 = insertion_sort(unsorted_list)
print("Sorted List:", sorted_list1)
```

```
Sorted List: [0.15, 54, 55, 100, 111, 200, 500, 786]
```

5.Write a Python program that performs merge sort on a list of numbers:

a.Divide: If the given array has zero or one element, return.

1. Otherwise

2.Divide the input list in to two halves each containing half of t he elements. i.e. left half and right half.

b. Conquer: Recursively sort the two lists (left half and right half).

1.Call the merge sort on left half.

2. Call the merge sort on right half.

C. Combine: Combine the elements back in the input list by merging the two sorted lists into a sorted sequence.

In [39]:
```
def merge_sort(arr):
    if len(arr) <= 1:
        return arr

    # Step 1: Divide the input list into two halves
    mid = len(arr) // 2
    left_half = arr[:mid]
    right_half = arr[mid:]

    # Step 2: Recursively sort the two halves
    left_half = merge_sort(left_half)
    right_half = merge_sort(right_half)

    # Step 3: Combine the two sorted halves
    sorted_arr = merge(left_half, right_half)
    return sorted_arr

def merge(left, right):
    result = []
    left_idx, right_idx = 0, 0

    while left_idx < len(left) and right_idx < len(right):
        if left[left_idx] < right[right_idx]:
            result.append(left[left_idx])
            left_idx += 1
        else:
            result.append(right[right_idx])
            right_idx += 1

    # Add remaining elements from both lists (if any)
    result.extend(left[left_idx:])
    result.extend(right[right_idx:])
    return result


input_list = [12, 11, 13, 5, 6, 7]
sorted_list = merge_sort(input_list)
print("Sorted list:", sorted_list)
```

```
Sorted list: [5, 6, 7, 11, 12, 13]
```

In [ ]: