**Sorting**

## Python Program for Selection Sort

```
def selectionSort(array, size):
    for ind in range(size):
        min_index = ind
        for j in range(ind + 1, size):
            if array[j] < array[min_index]:
                min_index = j
        (array[ind], array[min_index]) = (array[min_index], array[ind])


arr = [-2, 45, 0, 11, -9,88,-97,-202,747]
size = len(arr)
selectionSort(arr, size)
print('The array after sorting in Ascending Order by selection sort is:')
print(arr)
```

Output:

```
The array after sorting in Ascending Order by selection sort is:
[-202, -97, -9, -2, 0, 11, 45, 88, 747]
```

## Python Program for Bubble Sort

```
# Python program for implementation of Bubble Sort

def bubbleSort(arr):
        n = len(arr)
        swapped = False
                for i in range(n-1):
                for j in range(0, n-i-1):
                        if arr[j] > arr[j + 1]:
                                swapped = True
                                arr[j], arr[j + 1] = arr[j + 1], arr[j]

                if not swapped:
                        return



arr = [64, 34, 25, 12, 22, 11, 90]

bubbleSort(arr)

print("Sorted array is:")
for i in range(len(arr)):
        print("% d" % arr[i], end=" ")
```

**Output**

```
Sorted array is:
 11  12  22  25  34  64  90
```

## Python Program for Insertion Sort

```
def insertionSort(arr):



     if (n := len(arr)) <= 1:

     return

     for i in range(1, n):
```

```
                key = arr[i]

                j = i-1

                while j >=0 and key < arr[j] :

                        arr[j+1] = arr[j]

                        j -= 1

                arr[j+1] = key



#sorting the array [12, 11, 13, 5, 6] using insertionSort

arr = [12, 11, 13, 5, 6]

insertionSort(arr)

print(arr)
```

**Output:**

```
Sorted array is:
[5, 6, 11, 12, 13]
```

## Python Program for QuickSort

```
def partition(array, low, high):

              pivot = array[high]


       i = low - 1
```

```
        for j in range(low, high):
                if array[j] <= pivot:

                        i = i + 1

                        # Swapping element at i with element at j
                        (array[i], array[j]) = (array[j], array[i])

        (array[i + 1], array[high]) = (array[high], array[i + 1])


        return i + 1


def quickSort(array, low, high):
        if low < high:
                pi = partition(array, low, high)
                quickSort(array, low, pi - 1)

                quickSort(array, pi + 1, high)


data = [1, 7, 4, 1, 10, 9, -2]
print("Unsorted Array")
print(data)

size = len(data)

quickSort(data, 0, size - 1)

print('Sorted Array in Ascending Order:')
print(data)
```

**Output**

```
Unsorted Array
[1, 7, 4, 1, 10, 9, -2]
Sorted Array in Ascending Order:
[-2, 1, 1, 4, 7, 9, 10]
```

## Python Program for Merge Sort

```
def merge(arr, l, m, r):
	n1 = m - l + 1
	n2 = r - m

	# create temp arrays
	L = [0] * (n1)
	R = [0] * (n2)

	# Copy data to temp arrays L[] and R[]
	for i in range(0, n1):
		L[i] = arr[l + i]

	for j in range(0, n2):
		R[j] = arr[m + 1 + j]

	# Merge the temp arrays back into arr[l..r]
	i = 0	 # Initial index of first subarray
	j = 0	 # Initial index of second subarray
	k = l	 # Initial index of merged subarray

	while i < n1 and j < n2:
		if L[i] <= R[j]:
			arr[k] = L[i]
```

```
                i += 1
            else:
                arr[k] = R[j]
                j += 1
            k += 1

    while i < n1:
        arr[k] = L[i]
        i += 1
        k += 1

    while j < n2:
        arr[k] = R[j]
        j += 1
        k += 1



def mergeSort(arr, l, r):
    if l < r:

        m = l+(r-l)//2

        mergeSort(arr, l, m)
        mergeSort(arr, m+1, r)
        merge(arr, l, m, r)



# Driver code to test above
arr = [12, 11, 13, 5, 6, 7]
n = len(arr)
print("Given array is")
for i in range(n):
    print("%d" % arr[i],end=" ")
```

```
mergeSort(arr, 0, n-1)
print("\n\nSorted array is")
for i in range(n):
       print("%d" % arr[i],end=" ")
```

**Output**

```
Given array is
12 11 13 5 6 7

Sorted array is
5 6 7 11 12 13
```