```
def insertionSort(arr):
        n = len(arr) # Get the length of the array
        if n <= 1:
                return # If the array has 0 or 1 element, it is already sorted, so return
        for i in range(1, n): # Iterate over the array starting from the second element
                key = arr[i] # Store the current element as the key to be inserted in the
                j = i-1
                while j >= 0 and key < arr[j]: # Move elements greater than key one
                        arr[j+1] = arr[j] # Shift elements to the right
                        j -= 1
                arr[j+1] = key # Insert the key in the correct position
# Sorting the array [12, 11, 13, 5, 6] using insertionSort
arr = [12, 11, 13, 5, 6]
print("Array before Sorting")
print(arr)
print()
insertionSort(arr)
print("Array after sorting using Insertion Sort")
print(arr)
print()
print()

def shellSort(arr,n):
        # Start with a big gap, then reduce the gap
        n = len(arr)
        gap = n//2
        # Do a gapped insertion sort for this gap size.
        # The first gap elements a[0..gap-1] are already in gapped
        # order keep adding one more element until the entire array
        # is gap sorted
        while gap > 0:
                for i in range(gap,n):
                        # add a[i] to the elements that have been gap sorted
                        # save a[i] in temp and make a hole at position i
                        temp = arr[i]
                        # shift earlier gap-sorted elements up until the correct
                        # location for a[i] is found
                        j = i
                        while j >= gap and arr[j-gap] >temp:
                                arr[j] = arr[j-gap]
                                j -= gap
                        # put temp (the original a[i]) in its correct location
                        arr[j] = temp
                gap //= 2
# Driver code to test above
arr = [ 12, 34, 54, 2, 3]
n = len(arr)
print ("Array before sorting:")
print(arr)
shellSort(arr,n)
print ("\nArray after sorting using Shell Sort:")
print(arr)
print()
top5=[]
for i in range(-4,1):
        top5.append(arr[-i])
print("top 5 elements",top5)

#########################################################################################

Output : -

ubuntu@ubuntu-OptiPlex-3090:~/Documents/dsl_practicals$ /bin/python3 /home/ubuntu/-
Documents/dsl_practicals/practical5dsl.py
Array before Sorting
[12, 11, 13, 5, 6]
```

```
Array after sorting using Insertion Sort
[5, 6, 11, 12, 13]


Array before sorting:
[12, 34, 54, 2, 3]

Array after sorting using Shell Sort:
[2, 3, 12, 34, 54]

top 5 elements [54, 34, 12, 3, 2]
```