Merge sort is defined as a sorting algorithm that works by dividing an array into smaller subarrays, sorting each subarray, and then merging the sorted subarrays back together to form the final sorted array.

Merge sort is a recursive algorithm that continuously splits the array in half until it cannot be further divide d i.e., the array has only one element left (an array with one element is always sorted). Then the sorted su barrays are merged into one sorted array.

Implementaion for the merge sort in Python:

```
# Python program for implementation of MergeSort


def mergeSort(arr):
 if len(arr) > 1:

  # Finding the mid of the array
  mid = len(arr)//2

  # Dividing the array elements
  L = arr[:mid]

  # Into 2 halves
  R = arr[mid:]

  # Sorting the first half
  mergeSort(L)

  # Sorting the second half
  mergeSort(R)

  i = j = k = 0

  # Copy data to temp arrays L[] and R[]
  while i < len(L) and j < len(R):
   if L[i] <= R[j]:
    arr[k] = L[i]
    i += 1
   else:
    arr[k] = R[j]
    j += 1
   k += 1

  # Checking if any element was left
  while i < len(L):
   arr[k] = L[i]
   i += 1
   k += 1

  while j < len(R):
   arr[k] = R[j]
   j += 1
   k += 1


# Code to print the list
```

```
def printList(arr):
 for i in range(len(arr)):
  print(arr[i], end=" ")
 print()


# Driver Code
if __name__ == '__main__':
 arr = [12, 11, 13, 5, 6, 7]
 print("Given array is")
 printList(arr)
 mergeSort(arr)
 print("\nSorted array is ")
 printList(arr)
```

Advantages of merge sort:
1- Stability: Merge sort is a stable sorting algorithm, which means it maintains the relative order of equal e lements in the input array.
2- Guaranteed worst-case performance: Merge sort has a worst-case time complexity of O(N logN), which means it performs well even on large datasets.
3- Parallelizable: Merge sort is a naturally parallelizable algorithm, which means it can be easily paralleliz ed to take advantage of multiple processors or threads.

Disadvantages of merge sort:
1- Space complexity: Merge sort requires additional memory to store the merged sub-arrays during the so rting process.
2- Not in-place: Merge sort is not an in-place sorting algorithm, which means it requires additional memor y to store the sorted data. This can be a disadvantage in applications where memory usage is a concern.
3- Not always optimal for small datasets: For small datasets, Merge sort has a higher time complexity tha n some other sorting algorithms, such as insertion sort. This can result in slower performance for very sm all datasets.