# Python program for implementation of Quicksort Sort

# This function takes last element as pivot, places

# the pivot element at its correct position in sorted

# array, and places all smaller (smaller than pivot)

# to left of pivot and all greater elements to right

# of pivot

def partition(arr, low, high):

i = (low-1) # index of smaller element

pivot = arr[high] # pivot

for j in range(low, high):

print("j=", j, end=" ")

# If current element is smaller than or

# equal to pivot

if arr[j] <= pivot:

# increment index of smaller element

i = i+1

print("i=",i)

arr[i], arr[j] = arr[j], arr[i]

arr[i+1], arr[high] = arr[high], arr[i+1]

return (i+1)

# The main function that implements QuickSort

# arr[] --> Array to be sorted,

# low --> Starting index,

# high --> Ending index

# Function to do Quick sort

def quickSort(arr, low, high):

if len(arr) == 1:

return arr

if low < high:

print("~~~~~~~~~")

# pi is partitioning index, arr[p] is now

# at right place

pi = partition(arr, low, high)

# Separately sort elements before

# partition and after partition

quickSort(arr, low, pi-1)

quickSort(arr, pi+1, high)

# Driver code to test above

arr = [10, 7, 8, 9, 1, 5]

n = len(arr)

quickSort(arr, 0, n-1)

print("Sorted array is:")

for i in range(n):

print(arr[i], end=" ")

print()

# This code is contributed by Mohit Kumra

#This code in improved by https://github.com/anushkrishnav

"""

[10, 7, 8, 9, 1, 5]

-quick(arr, 0, 5)

-pi=part(arr, 0, 5) = 2

- i=-1

- pivot=5

- for j 0->4

- j=0

- j=1

- j=2

- j=3

- j=4

arr[j] = 1

arr[j] < 5

i=0

arr[0] <-> arr[4]

[1,7,8,9,10,5]

- arr[1] <-> arr[5]

[1,5,8,9,10,7]

- return 2

-quick(arr,0, 1)

-pi=part(arr,0,1)

-quick(arr,3, 5)

"""