Python Program to find largest element in an array

Given an array, find the largest element in it.

Input : arr[] = {10, 20, 4}

Output : 20

Input : arr[] = {20, 10, 20, 4, 100}

Output : 100

# Python3 program to find maximum

# in arr[] of size n

# python function to find maximum

# in arr[] of size n

def largest(arr,n):

    # Initialize maximum element

    max = arr[0]

    # Traverse array elements from second

    # and compare every element with

    # current max

    for i in range(1, n):

        if arr[i] > max:

            max = arr[i]

    return max

# Driver Code

arr = [10, 324, 45, 90, 9808]

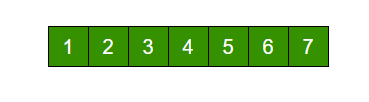
n = len(arr)

Ans = largest(arr,n)

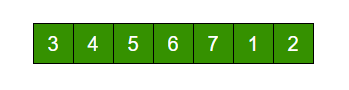
print ("Largest in given array is",Ans)

# This code is contributed by Smitha Dinesh Semwal

# Python Program for array rotation

Write a function rotate(ar[], d, n) that rotates arr[] of size n by d elements.  
  


Rotation of the above array by 2 will make array



**METHOD 1 (Using temp array)**

Input arr[] = [1, 2, 3, 4, 5, 6, 7], d = 2, n =7

1) Store d elements in a temp array

temp[] = [1, 2]

2) Shift rest of the arr[]

arr[] = [3, 4, 5, 6, 7, 6, 7]

3) Store back the d elements

arr[] = [3, 4, 5, 6, 7, 1, 2]

**Time complexity :** O(n)  
**Auxiliary Space :**O(d)

Method:-

|  |
| --- |
| Function to left rotate arr[] of size n by d\*/  def leftRotate(arr, d, n):      for i in range(d):          leftRotatebyOne(arr, n)    #Function to left Rotate arr[] of size n by 1\*/  def leftRotatebyOne(arr, n):      temp = arr[0]      for i in range(n-1):          arr[i] = arr[i+1]      arr[n-1] = temp      # utility function to print an array \*/  def printArray(arr,size):      for i in range(size):          print ("%d"% arr[i],end=" ")      # Driver program to test above functions \*/  arr = [1, 2, 3, 4, 5, 6, 7]  leftRotate(arr, 2, 7)  printArray(arr, 7)    # This code is contributed by Shreyanshi Arun |

Output :

3 4 5 6 7 1 2

**Time complexity :** O(n \* d)  
**Auxiliary Space :** O(1)

Write a Program to reverse an array:-

# Python program for reversal algorithm of array rotation

# Function to reverse arr[] from index start to end

def rverseArray(arr, start, end):

    while (start < end):

        temp = arr[start]

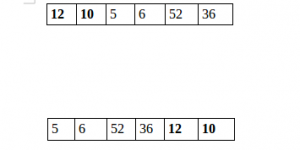
        arr[start] = arr[end]

        arr[end] = temp

        start += 1

        end = end-1

# Python Program to Split the array and add the first part to the end

There is a given an array and split it from a specified position, and move the first part of array add to the end.  


|  |
| --- |
| # Python program to split array and move first  # part to end.    def splitArr(arr, n, k):      for i in range(0, k):          x = arr[0]          for j in range(0, n-1):              arr[j] = arr[j + 1]            arr[n-1] = x      # main  arr = [12, 10, 5, 6, 52, 36]  n = len(arr)  position = 2    splitArr(arr, n, position)    for i in range(0, n):      print(arr[i], end = ' ')    # Code Contributed by Mohit Gupta\_OMG <(0\_o)> |

**Output:**

5 6 52 36 12 10

# Python Program to check if given array is Monotonic

Given an array **A** containing **n** integers. The task is to check whether the array is **Monotonic** or not. An array is monotonic if it is either **monotone increasing** or **monotone decreasing**.  
An array A is monotone increasing if for all i <= j, **A[i] <= A[j]**. An array A is monotone decreasing if for all i <= j, **A[i] >= A[j]**.  
Return “**True**” if the given array A is monotonic else return “**False**” (without quotes).

Examples:

Input : 6 5 4 4

Output : true

Input : 5 15 20 10

Output : false

# Python3 program to find sum in Nth group

# Check if given array is Monotonic

def isMonotonic(A):

    return (all(A[i] <= A[i + 1] for i in range(len(A) - 1)) or

            all(A[i] >= A[i + 1] for i in range(len(A) - 1)))

# Driver program

A = [6, 5, 4, 4]

# Print required result

print(isMonotonic(A))

# This code is written by

# Sanjit\_Prasad

Output:

True

**Time Complexity:** O(N), where N is the length of array.

# Python program to print all Prime numbers in an Interval

# Python program to print all

# prime number in an interval

start = 11

end = 25

for val in range(start, end + 1):

   # If num is divisible by any number

   # between 2 and val, it is not prime

   if val > 1:

       for n in range(2, val):

           if (val % n) == 0:

               break

       else:

           print(val)

# Python program to check whether a number is Prime or not

|  |
| --- |
| # Python program to check if  # given number is prime or not    num = 11    # If given number is greater than 1  if num > 1:       # Iterate from 2 to n / 2     for i in range(2, num//2):           # If num is divisible by any number between         # 2 and n / 2, it is not prime         if (num % i) == 0:             print(num, "is not a prime number")             break     else:         print(num, "is a prime number")    else:     print(num, "is not a prime number") |

**Output:**

11 is a prime number

# Python Program for n-th Fibonacci number

# Function for nth Fibonacci number

def Fibonacci(n):

    if n<0:

        print("Incorrect input")

    # First Fibonacci number is 0

    elif n==1:

        return 0

    # Second Fibonacci number is 1

    elif n==2:

        return 1

    else:

        return Fibonacci(n-1)+Fibonacci(n-2)

# Driver Program

print(Fibonacci(9))

# Program to print ASCII Value of a character

**Examples :**

Input : a

Output : 97

Input : D

Output : 68

**Python code using ord function :**

|  |
| --- |
| # Python program to print  # ASCII Value of Character    # In c we can assign different  # characters of which we want ASCII value    c = 'g'  # print the ASCII value of assigned character in c  print("The ASCII value of '" + c + "' is", ord(c)) |

Output:

The ASCII value of g is 103

# Python Program for Sum of squares of first n natural numbers

Given a positive integer **N**. The task is to find 12 + 22 + 32 + ….. + N2.

# Python3 Program to

# find sum of square

# of first n natural

# numbers

# Return the sum of

# square of first n

# natural numbers

def squaresum(n) :

    # Iterate i from 1

    # and n finding

    # square of i and

    # add to sum.

    sm = 0

    for i in range(1, n+1) :

        sm = sm + (i \* i)

    return sm

# Driven Program

n = 4

print(squaresum(n))

# Python | Find missing numbers in a sorted list range

**Input :** [1, 2, 4, 6, 7, 9, 10]

**Output :** [3, 5, 8]

**Input :** [5, 6, 10, 11, 13]

**Output :** [7, 8, 9, 12]

|  |
| --- |
| # Python3 program to Find missing  # integers in list    def find\_missing(lst):      return [x for x in range(lst[0], lst[-1]+1)                                 if x not in lst]    # Driver code  lst = [1, 2, 4, 6, 7, 9, 10]  print(find\_missing(lst)) |

**Output:**

[3, 5, 8]

# Given a sorted and rotated array, find if there is a pair with a given sum

Input: arr[] = {11, 15, 6, 8, 9, 10}, x = 16

Output: true

There is a pair (6, 10) with sum 16

# Python3 program to find a

# pair with a given sum in

# a sorted and rotated array

# This function returns True

# if arr[0..n-1] has a pair

# with sum equals to x.

def pairInSortedRotated( arr, n, x ):

    # Find the pivot element

    for i in range(0, n - 1):

        if (arr[i] > arr[i + 1]):

            break;

    # l is now index of smallest element

    l = (i + 1) % n

    # r is now index of largest element

    r = i

    # Keep moving either l

    # or r till they meet

    while (l != r):

        # If we find a pair with

        # sum x, we return True

        if (arr[l] + arr[r] == x):

            return True;

        # If current pair sum is less,

        # move to the higher sum

        if (arr[l] + arr[r] < x):

            l = (l + 1) % n;

        else:

            # Move to the lower sum side

            r = (n + r - 1) % n;

    return False;

# Driver program to test above function

arr = [11, 15, 26, 38, 9, 10]

sum = 16

n = len(arr)

if (pairInSortedRotated(arr, n, sum)):

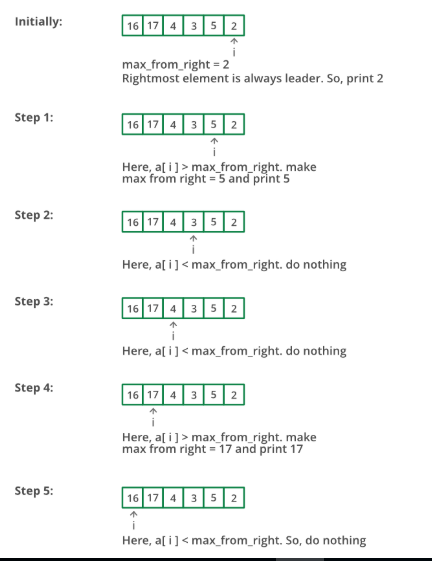
    print ("Array has two elements with sum 16")

else:

    print ("Array doesn't have two elements with sum 16 ")

# Leaders in an array

Write a program to print all the LEADERS in the array. An element is leader if it is greater than all the elements to its right side. And the rightmost element is always a leader. For example int the array {16, 17, 4, 3, 5, 2}, leaders are 17, 5 and 2.



# Python function to print leaders in array

def printLeaders(arr, size):

    max\_from\_right = arr[size-1]

    print max\_from\_right,

    for i in range( size-2, 0, -1):

        if max\_from\_right < arr[i]:

            print arr[i],

            max\_from\_right = arr[i]

# Driver function

arr = [16, 17, 4, 3, 5, 2]

printLeaders(arr, len(arr))

# Print all triplets in sorted array that form AP

Input : arr[] = { 2, 6, 9, 12, 17, 22, 31, 32, 35, 42 };

Output :

6 9 12

2 12 22

12 17 22

2 17 32

12 22 32

9 22 35

2 22 42

22 32 42

# python 3 program to print all triplets in given

# array that form Arithmetic Progression

# Function to print all triplets in

# given sorted array that forms AP

def printAllAPTriplets(arr, n):

    for i in range(1, n - 1):

        # Search other two elements of

        # AP with arr[i] as middle.

        j = i - 1

        k = i + 1

        while(j >= 0 and k < n ):

            # if a triplet is found

            if (arr[j] + arr[k] == 2 \* arr[i]):

                print(arr[j], "", arr[i], "", arr[k])

                # Since elements are distinct,

                # arr[k] and arr[j] cannot form

                # any more triplets with arr[i]

                k += 1

                j -= 1

            # If middle element is more move to

            # higher side, else move lower side.

            elif (arr[j] + arr[k] < 2 \* arr[i]):

                k += 1

            else:

                j -= 1

# Driver code

arr = [ 2, 6, 9, 12, 17,

        22, 31, 32, 35, 42 ]

n = len(arr)

printAllAPTriplets(arr, n)

# This article is contributed

# by Smitha Dinesh Semwal

# Find subarray with given sum | Set 1 (Nonnegative Numbers)

Given an unsorted array of nonnegative integers, find a continuous subarray which adds to a given number.

**Examples :**

***Input****: arr[] = {1, 4, 20, 3, 10, 5}, sum = 33****Ouptut****: Sum found between indexes 2 and 4*

***Input****: arr[] = {1, 4, 0, 0, 3, 10, 5}, sum = 7****Ouptut****: Sum found between indexes 1 and 4*

***Input****: arr[] = {1, 4}, sum = 0****Output****: No subarray found*

|  |
| --- |
| # Returns true if the  # there is a subarray  # of arr[] with sum  # equal to 'sum'  # otherwise returns  # false. Also, prints  # the result  def subArraySum(arr, n, sum):        # Pick a starting      # point      for i in range(n):          curr\_sum = arr[i]            # try all subarrays          # starting with 'i'          j = i+1          while j <= n:                if curr\_sum == sum:                  print ("Sum found between")                  print("indexes %d and %d"%( i, j-1))                    return 1                if curr\_sum > sum or j == n:                  break                curr\_sum = curr\_sum + arr[j]              j += 1        print ("No subarray found")      return 0    # Driver program  arr = [15, 2, 4, 8, 9, 5, 10, 23]  n = len(arr)  sum = 23    subArraySum(arr, n, sum)    # This code is Contributed by shreyanshi\_arun. |

**Output :**

Sum found between indexes 1 and 4

**Time Complexity :** O(n^2) in worst case.

Two sum problem

The two sum problem is a common interview question, and it is a variation of the [subset sum](https://en.wikipedia.org/wiki/Subset_sum_problem) problem

For example, if the array is [3, 5, 2, -4, 8, 11] and the sum is 7, your program should return [[11, -4], [2, 5]] because 11 + -4 = 7 and 2 + 5 = 7.

## Naive solution

# our two sum function which will return

# all pairs in the list that sum up to S

def twoSum(arr, S):

sums = []

# check each element in array

for i in range(0, len(arr)):

# check each other element in the array

for j in range(i+1, len(arr)):

# determine if these two elements sum to S

if (arr[i] + arr[j] == S):

sums.append([arr[i], arr[j]])

# return all pairs of integers that sum to S

return sums

print twoSum([3, 5, 2, -4, 8, 11], 7)

## Faster solution

# our two sum function which will return

# all pairs in the list that sum up to S

def twoSum(arr, S):

sums = []

hashTable = {}

# check each element in array

for i in range(0, len(arr)):

# calculate S minus current element

sumMinusElement = S - arr[i]

# check if this number exists in hash table

# if so then we found a pair of numbers that sum to S

if sumMinusElement in hashTable:

sums.append([arr[i], sumMinusElement])

# add the current number to the hash table

hashTable[arr[i]] = arr[i]

# return all pairs of integers that sum to S

return sums

print twoSum([3, 5, 2, -4, 8, 11], 7)

# Rearrange an array so that arr[i] becomes arr[arr[i]] with O(1) extra space

Input: arr[] = {3, 2, 0, 1}

Output: arr[] = {1, 0, 3, 2}

**1)** Increase every array element arr[i] by (arr[arr[i]] % n)\*n.  
**2)** Divide every element by n.

Let us understand the above steps by an example array {3, 2, 0, 1}

In first step, every value is incremented by (arr[arr[i]] % n)\*n

After first step array becomes {7, 2, 12, 9}.

***The important thing is, after the increment operation***

***of first step, every element holds both old values and new values.***

***Old value can be obtained by arr[i]%n and new value can be obtained***

***by arr[i]/n.***

In second step, all elements are updated to new or output values

by doing arr[i] = arr[i]/n.

After second step, array becomes {1, 0, 3, 2}

# Python3 program to Rearrange

# an array so that arr[i] becomes

# arr[arr[i]]

# The function to rearrange an

# array in-place so that arr[i]

# becomes arr[arr[i]].

def rearrange(arr, n):

    # First step: Increase all values

    # by (arr[arr[i]] % n) \* n

    for i in range(0, n):

        arr[i] += (arr[arr[i]] % n) \* n

    # Second Step: Divide all values

    # by n

    for i in range(0, n):

        arr[i] = int(arr[i] / n)

# A utility function to print

# an array of size n

def printArr(arr, n):

    for i in range(0, n):

        print (arr[i], end =" ")

    print ("")

# Driver program

arr = [3, 2, 0, 1]

n = len(arr)

print ("Given array is")

printArr(arr, n)

rearrange(arr, n);

print ("Modified array is")

printArr(arr, n)

# Sort an array of 0s, 1s and 2s

Given an array A[] consisting 0s, 1s and 2s, write a function that sorts A[]. The functions should put all 0s first, then all 1s and all 2s in last.

Examples:

Input : {0, 1, 2, 0, 1, 2}

Output : {0, 0, 1, 1, 2, 2}

Input : {0, 1, 1, 0, 1, 2, 1, 2, 0, 0, 0, 1}

Output : {0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2}

# Python C++ program to sort an array of 0s

# 1s and 2s.

import math

def sort012(arr, n):

    # Variables to maintain the count of 0's,

    # 1's and 2's in the array

    count0 = 0

    count1 = 0

    count2 = 0

    for i in range(0,n):

        if (arr[i] == 0):

            count0=count0+1

        if (arr[i] == 1):

            count1=count1+1

        if (arr[i] == 2):

            count2=count2+1

    # Putting the 0's in the array in starting.

    for i in range(0,count0):

        arr[i] = 0

    # Putting the 1's in the array after the 0's.

    for i in range( count0, (count0 + count1)) :

        arr[i] = 1

    # Putting the 2's in the array after the 1's

    for i in range((count0 + count1),n) :

        arr[i] = 2

    return

# Prints the array

def printArray( arr,  n):

    for i in range(0,n):

        print( arr[i] , end=" ")

    print()

# Driver code

arr = [ 0, 1, 1, 0, 1, 2, 1, 2, 0, 0, 0, 1 ]

n = len(arr)

sort012(arr, n)

printArray(arr, n)

# Count Inversions in an array

*Inversion Count*for an array indicates – how far (or close) the array is from being sorted. If array is already sorted then inversion count is 0. If array is sorted in reverse order that inversion count is the maximum.  
Formally speaking, two elements a[i] and a[j] form an inversion if a[i] > a[j] and i < j

**Example:**  
The sequence 2, 4, 1, 3, 5 has three inversions (2, 1), (4, 1), (4, 3).

# Python3 program to count

# inversions in an array

def getInvCount(arr, n):

    inv\_count = 0

    for i in range(n):

        for j in range(i + 1, n):

            if (arr[i] > arr[j]):

                inv\_count += 1

    return inv\_count

# Driver Code

arr = [1, 20, 6, 4, 5]

n = len(arr)

print("Number of inversions are",

              getInvCount(arr, n))

# Majority Element

Write a function which takes an array and prints the majority element (if it exists), otherwise prints “No Majority Element”. A ***majority element*** in an array A[] of size n is an element that appears more than n/2 times (and hence there is at most one such element).  
**Examples :**

Input : {3, 3, 4, 2, 4, 4, 2, 4, 4}

Output : 4

Input : {3, 3, 4, 2, 4, 4, 2, 4}

Output : No Majority Element

# Python program for finding out majority

# element in an array

def findMajority(arr, size):

    m = {}

    for i in range(size):

        if arr[i] in m:

            m[arr[i]] += 1

        else:

            m[arr[i]] = 1

    count = 0

    for key in m:

        if m[key] > size / 2:

            count = 1

            print("Majority found :-",key)

            break

    if(count == 0):

        print("No Majority element")

# Driver code

arr = [2, 2, 2, 2, 5, 5, 2, 3, 3]

n = len(arr)

# Function calling

findMajority(arr, n)

# This code is contributed by ankush\_953

# Split an array into two equal Sum subarrays

Given an array of integers greater than zero, find if it is possible to split it in two subarrays (without reordering the elements), such that the sum of the two subarrays is the same. Print the two subarrays.

**Examples :**

Input : Arr[] = { 1 , 2 , 3 , 4 , 5 , 5 }

Output : { 1 2 3 4 }

{ 5 , 5 }

Input : Arr[] = { 4, 1, 2, 3 }

Output : {4 1}

{2 3}

Input : Arr[] = { 4, 3, 2, 1}

Output : Not Possible

# Python3 program to split

# an array into Two

# equal sum subarrays

# Returns split point.

# If not possible,

# then return -1.

def findSplitPoint(arr, n) :

    # traverse array element and

    # compute sum of whole array

    leftSum = 0

    for i in range(0, n) :

        leftSum += arr[i]

    # again traverse array and

    # compute right sum and also

    # check left\_sum equal to

    # right sum or not

    rightSum = 0

    for i in range(n-1, -1, -1) :

        # add current element

        # to right\_sum

        rightSum += arr[i]

        # exclude current element

        # to the left\_sum

        leftSum -= arr[i]

        if (rightSum == leftSum) :

            return i

    # if it is not possible

    # to split array into

    # two parts.

    return -1

# Prints two parts after

# finding split point

# using findSplitPoint()

def printTwoParts(arr, n) :

    splitPoint = findSplitPoint(arr, n)

    if (splitPoint == -1 or splitPoint == n ) :

        print ("Not Possible")

        return

    for i in range (0, n) :

        if(splitPoint == i) :

            print ("")

        print (arr[i], end = " ")

# Driver Code

arr = [1, 2, 3, 4, 5, 5]

n = len(arr)

printTwoParts(arr, n)

# A Product Array Puzzle

Given an array arr[] of n integers, construct a Product Array prod[] (of same size) such that prod[i] is equal to the product of all the elements of arr[] except arr[i]. Solve it **without division operator and in O(n)**.

**Example :**  
arr[] = {10, 3, 5, 6, 2}  
prod[] = {180, 600, 360, 300, 900}

# Python3 program for A Product Array Puzzle

def productArray(arr, n):

    # Base case

    if n == 1:

        print(0)

        return

    i, temp = 1, 1

    # Allocate memory for the product array

    prod = [1 for i in range(n)]

    # Initialize the product array as 1

    # In this loop, temp variable contains product of

    # elements on left side excluding arr[i]

    for i in range(n):

        prod[i] = temp

        temp \*= arr[i]

    # Initialize temp to 1 for product on right side

    temp = 1

    # In this loop, temp variable contains product of

    # elements on right side excluding arr[i]

    for i in range(n - 1, -1, -1):

        prod[i] \*= temp

        temp \*= arr[i]

    # Print the constructed prod array

    for i in range(n):

        print(prod[i], end = " ")

    return

# Driver Code

arr = [10, 3, 5, 6, 2]

n = len(arr)

print("The product array is: n")

productArray(arr, n)

# Find the Number Occurring Odd Number of Times

Given an array of positive integers. All numbers occur even number of times except one number which occurs odd number of times. Find the number in O(n) time & constant space.

**Examples :**

Input : arr = {1, 2, 3, 2, 3, 1, 3}

Output : 3

Input : arr = {5, 7, 2, 7, 5, 2, 5}

Output : 5

# Python3 program to find the element

# occurring odd number of times

# function to find the element

# occurring odd number of times

def getOddOccurrence(arr,size):

    # Defining HashMap in C++

    Hash=dict()

    # Putting all elements into the HashMap

    for i in range(size):

        Hash[arr[i]]=Hash.get(arr[i],0) + 1;

    # Iterate through HashMap to check an element

    # occurring odd number of times and return it

    for i in Hash:

        if(Hash[i]% 2 != 0):

            return i

    return -1

# Driver code

arr=[2, 3, 5, 4, 5, 2, 4,3, 5, 2, 4, 4, 2]

n = len(arr)

# Function calling

print(getOddOccurrence(arr, n))

# Sort an array in wave form

Given an unsorted array of integers, sort the array into a wave like array. An array ‘arr[0..n-1]’ is sorted in wave form if arr[0] >= arr[1] <= arr[2] >= arr[3] <= arr[4] >= …..

Examples:

Input: arr[] = {10, 5, 6, 3, 2, 20, 100, 80}

Output: arr[] = {10, 5, 6, 2, 20, 3, 100, 80} OR

{20, 5, 10, 2, 80, 6, 100, 3} OR

any other array that is in wave form

Input: arr[] = {20, 10, 8, 6, 4, 2}

Output: arr[] = {20, 8, 10, 4, 6, 2} OR

{10, 8, 20, 2, 6, 4} OR

any other array that is in wave form

# Python function to sort the array arr[0..n-1] in wave form,

# i.e., arr[0] >= arr[1] <= arr[2] >= arr[3] <= arr[4] >= arr[5]

def sortInWave(arr, n):

    #sort the array

    arr.sort()

    # Swap adjacent elements

    for i in range(0,n-1,2):

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

# Driver progrM

arr = [10, 90, 49, 2, 1, 5, 23]

sortInWave(arr, len(arr))

for i in range(0,len(arr)):

    print arr[i],

# Python function to sort the array arr[0..n-1] in wave form,

# i.e., arr[0] >= arr[1] <= arr[2] >= arr[3] <= arr[4] >= arr[5]

def sortInWave(arr, n):

    # Traverse all even elements

    for i in range(0, n, 2):

        # If current even element is smaller than previous

        if (i> 0 and arr[i] < arr[i-1]):

            arr[i],arr[i-1] = arr[i-1],arr[i]

        # If current even element is smaller than next

        if (i < n-1 and arr[i] < arr[i+1]):

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

# Driver program

arr = [10, 90, 49, 2, 1, 5, 23]

sortInWave(arr, len(arr))

for i in range(0,len(arr)):

    print arr[i],

# Maximum difference between two elements such that larger element appears after the smaller number

Given an array arr[] of integers, find out the maximum difference between any two elements such that larger element appears after the smaller number.

**Examples :**

Input : arr = {2, 3, 10, 6, 4, 8, 1}

Output : 8

Explanation : The maximum difference is between 10 and 2.

def maxDiff(arr, arr\_size):

    max\_diff = arr[1] - arr[0]

    for i in range( 0, arr\_size ):

        for j in range( i+1, arr\_size ):

            if(arr[j] - arr[i] > max\_diff):

                max\_diff = arr[j] - arr[i]

    return max\_diff

# Driver program to test above function

arr = [1, 2, 90, 10, 110]

size = len(arr)

print ("Maximum difference is", maxDiff(arr, size))

**Method 2 (Tricky and Efficient)**  
In this method, instead of taking difference of the picked element with every other element, we take the difference with the minimum element found so far. So we need to keep track of 2 things:  
1) Maximum difference found so far (max\_diff).  
2) Minimum number visited so far (min\_element).

# Python 3 code to find Maximum difference

# between two elements such that larger

# element appears after the smaller number

# The function assumes that there are

# at least two elements in array.

# The function returns a negative

# value if the array is sorted in

# decreasing order. Returns 0 if

# elements are equal

def maxDiff(arr, arr\_size):

    max\_diff = arr[1] - arr[0]

    min\_element = arr[0]

    for i in range( 1, arr\_size ):

        if (arr[i] - min\_element > max\_diff):

            max\_diff = arr[i] - min\_element

        if (arr[i] < min\_element):

            min\_element = arr[i]

    return max\_diff

# Driver program to test above function

arr = [1, 2, 6, 80, 100]

size = len(arr)

print ("Maximum difference is",

        maxDiff(arr, size))

s

# Find the smallest positive integer value that cannot be represented as sum of any subset of a given array

Given a sorted array (sorted in non-decreasing order) of positive numbers, find the smallest positive integer value that cannot be represented as sum of elements of any subset of given set.  
Expected time complexity is O(n).

Examples:

Input: arr[] = {1, 3, 6, 10, 11, 15};

Output: 2

Input: arr[] = {1, 1, 1, 1};

Output: 5

Input: arr[] = {1, 1, 3, 4};

Output: 10

Input: arr[] = {1, 2, 5, 10, 20, 40};

Output: 4

Input: arr[] = {1, 2, 3, 4, 5, 6};

Output: 22

# Python3 program to find the smallest

# positive value that cannot be

# represented as sum of subsets

# of a given sorted array

# Returns the smallest number

# that cannot be represented as sum

# of subset of elements from set

# represented by sorted array arr[0..n-1]

def findSmallest(arr, n):

    res = 1 #Initialize result

    # Traverse the array and increment

    # 'res' if arr[i] is smaller than

    # or equal to 'res'.

    for i in range (0, n ):

        if arr[i] <= res:

            res = res + arr[i]

        else:

            break

    return res

# Driver program to test above function

arr1 = [1, 3, 4, 5]

n1 = len(arr1)

print(findSmallest(arr1, n1))

arr2= [1, 2, 6, 10, 11, 15]

n2 = len(arr2)

print(findSmallest(arr2, n2))

arr3= [1, 1, 1, 1]

n3 = len(arr3)

print(findSmallest(arr3, n3))

**Leet Code problem:-**

Given the array nums, for each nums[i] find out how many numbers in the array are smaller than it. That is, for each nums[i] you have to count the number of valid j's such that j != i **and** nums[j] < nums[i].

Return the answer in an array.

Example:-

**Input:** nums = [8,1,2,2,3]

**Output:** [4,0,1,1,3]

**Explanation:**

For nums[0]=8 there exist four smaller numbers than it (1, 2, 2 and 3).

For nums[1]=1 does not exist any smaller number than it.

For nums[2]=2 there exist one smaller number than it (1).

For nums[3]=2 there exist one smaller number than it (1).

For nums[4]=3 there exist three smaller numbers than it (1, 2 and 2).

class Solution:

def smallerNumbersThanCurrent(self, nums):

d = {}

for i,v in enumerate(sorted(nums)):

if v not in d:

d[v]=i

return [d[i] for i in nums]

obj = Solution()

print(obj.smallerNumbersThanCurrent([8,1,2,2,3]))

**Sort Array By Parity**

Given an array A of non-negative integers, return an array consisting of all the even elements of A, followed by all the odd elements of A.

You may return any answer array that satisfies this condition.

**Example 1:**

**Input:** [3,1,2,4]

**Output:** [2,4,3,1]

The outputs [4,2,3,1], [2,4,1,3], and [4,2,1,3] would also be accepted.

class Solution:

def sortArrayByParity(self, A: List[int]) -> List[int]:

even = []

odd = []

for num in A:

if num % 2 == 0:

even.append(num)

else:

odd.append(num)

return even + odd

from collections import deque

class Solution:

def sortArrayByParity(self, A: List[int]) -> List[int]:

integers = deque([])

for integer in A:

if integer%2 == 0:

integers.appendleft(integer)

else:

integers.append(integer)

return integers

**Create Target Array in the Given Order**

Given two arrays of integers nums and index. Your task is to create *target* array under the following rules:

* Initially *target* array is empty.
* From left to right read nums[i] and index[i], insert at index index[i] the value nums[i] in *target* array.
* Repeat the previous step until there are no elements to read in nums and index.

Return the *target* array.

It is guaranteed that the insertion operations will be valid.

**Example 1:**

**Input:** nums = [0,1,2,3,4], index = [0,1,2,2,1]

**Output:** [0,4,1,3,2]

**Explanation:**

nums index target

0 0 [0]

1 1 [0,1]

2 2 [0,1,2]

3 2 [0,1,3,2]

4 1 [0,4,1,3,2]

**Example 2:**

**Input:** nums = [1,2,3,4,0], index = [0,1,2,3,0]

**Output:** [0,1,2,3,4]

**Explanation:**

nums index target

1 0 [1]

2 1 [1,2]

3 2 [1,2,3]

4 3 [1,2,3,4]

0 0 [0,1,2,3,4]

**Example 3:**

**Input:** nums = [1], index = [0]

**Output:** [1]

class Solution:

def createTargetArray(self, nums, index):

target = []

for i,value in zip(index, nums):

target.insert(i,value)

return target

obj = Solution()

print(obj.createTargetArray([0,1,2,3,4],[0,1,2,2,1]))

**Replace Elements with Greatest Element on Right Side**

Given an array arr, replace every element in that array with the greatest element among the elements to its right, and replace the last element with -1.

After doing so, return the array.

**Example 1:**

**Input:** arr = [17,18,5,4,6,1]

**Output:** [18,6,6,6,1,-1]

**Explanation:**

- index 0 --> the greatest element to the right of index 0 is index 1 (18).

- index 1 --> the greatest element to the right of index 1 is index 4 (6).

- index 2 --> the greatest element to the right of index 2 is index 4 (6).

- index 3 --> the greatest element to the right of index 3 is index 4 (6).

- index 4 --> the greatest element to the right of index 4 is index 5 (1).

- index 5 --> there are no elements to the right of index 5, so we put -1.

sclass Solution:

def replaceElements(self, arr: List[int]) -> List[int]:

output = []

for i in range(len(arr)-1):

output.append(max(arr[i+1:]))

output.append(-1)

return output

**Shortest Unsorted Continuous Subarrays**

Given an integer array nums, you need to find one **continuous subarray** that if you only sort this subarray in ascending order, then the whole array will be sorted in ascending order.

Return *the shortest such subarray and output its length*.

**Example 1:**

**Input:** nums = [2,6,4,8,10,9,15]

**Output:** 5

**Explanation:** You need to sort [6, 4, 8, 10, 9] in ascending order to make the whole array sorted in ascending order.

class Solution:

def findUnsortedSubarray(self, nums: List[int]) -> int:

maxSeq = 0

mx = -float('inf')

for i in range(len(nums)):

if nums[i] >= mx:

mx = nums[i]

maxSeq += 1

else:

maxSeq = 0

if maxSeq == len(nums):

return 0

minSeq = 0

mn = float('inf')

for i in reversed(range(len(nums))):

if nums[i] <= mn:

mn = nums[i]

minSeq += 1

else:

minSeq = 0

return len(nums)-(minSeq+maxSeq)

**First Missing Positive**

Given an unsorted integer array nums, find the smallest missing positive integer.

**Follow up:** Could you implement an algorithm that runs in O(n) time and uses constant extra space.?

**Example 1:**

**Input:** nums = [1,2,0]

**Output:** 3

**Example 2:**

**Input:** nums = [3,4,-1,1]

**Output:** 2

def first\_missing\_positive:

if len(l)==0:

return 1

m = max(l)

i = 1

while i<=m:

if i not in l:

return i

else:

i+=1

return i

**The K Weakest Rows in a Matrixs**

Given a m \* n matrix mat of *ones* (representing soldiers) and *zeros* (representing civilians), return the indexes of the k weakest rows in the matrix ordered from the weakest to the strongest.

A row ***i*** is weaker than row ***j***, if the number of soldiers in row ***i*** is less than the number of soldiers in row ***j***, or they have the same number of soldiers but ***i*** is less than ***j***. Soldiers are **always** stand in the frontier of a row, that is, always *ones* may appear first and then *zeros*.

**Example 1:**

**Input:** mat =

[[1,1,0,0,0],

[1,1,1,1,0],

[1,0,0,0,0],

[1,1,0,0,0],

[1,1,1,1,1]],

k = 3

**Output:** [2,0,3]

**Explanation:**

The number of soldiers for each row is:

row 0 -> 2

row 1 -> 4

row 2 -> 1

row 3 -> 2

row 4 -> 5

Rows ordered from the weakest to the strongest are [2,0,3,1,4]

**Example 2:**

**Input:** mat =

[[1,0,0,0],

 [1,1,1,1],

 [1,0,0,0],

 [1,0,0,0]],

k = 2

**Output:** [0,2]

**Explanation:**

The number of soldiers for each row is:

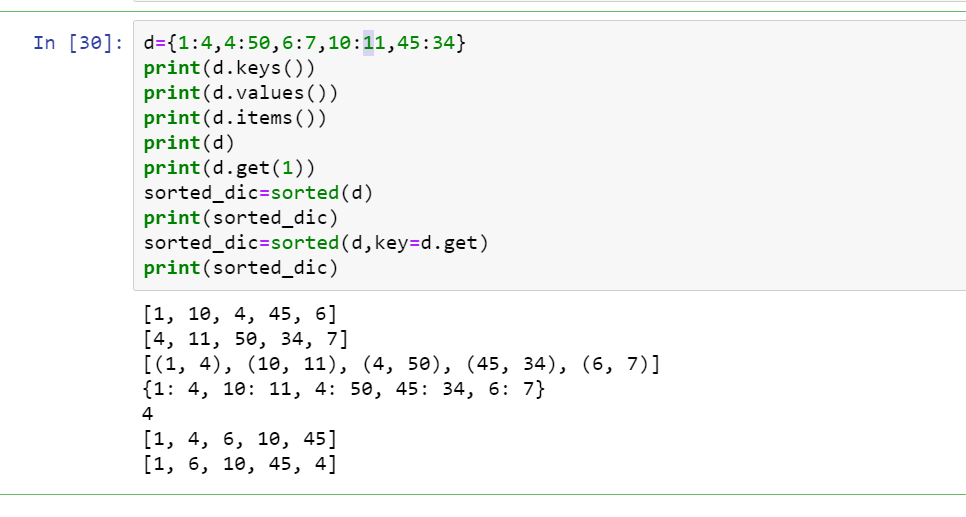
row 0 -> 1

row 1 -> 4

row 2 -> 1

row 3 -> 1

Rows ordered from the weakest to the strongest are [0,2,3,1]



class Solution:

def kWeakestRows(self, mat: List[List[int]], k: int) -> List[int]:

dic={}

for i in range(len(mat)):

dic[i]=sum(mat[i])

sorted\_dic=sorted(dic,key=dic.get)

return sorted\_dic[:k]

class Solution:

def kWeakestRows(self, mat: List[List[int]], k: int) -> List[int]:

return list(map(lambda t: t[1], sorted(

[(sum(row), index) for index, row in enumerate(mat)]

)))[:k]

**Count Negative Numbers in a Sorted Matrix**

Given a m x n matrix grid which is sorted in non-increasing order both row-wise and column-wise, return *the number of****negative****numbers in* grid.

**Example 1:**

**Input:** grid = [[4,3,2,-1],[3,2,1,-1],[1,1,-1,-2],[-1,-1,-2,-3]]

**Output:** 8

**Explanation:** There are 8 negatives number in the matrix.

class Solution:

def countNegatives(self, grid: List[List[int]]) -> int:

negative = 0

for list in grid:

for digit in list:

if digit < 0:

negative += 1

return negative

**Median of Two Sorted Arrayss**

Given two sorted arrays nums1 and nums2 of size m and n respectively, return **the median** of the two sorted arrays.

**Follow up:** The overall run time complexity should be O(log (m+n)).

**Example 1:**

**Input:** nums1 = [1,3], nums2 = [2]

**Output:** 2.00000

**Explanation:** merged array = [1,2,3] and median is 2.

**Example 2:**

**Input:** nums1 = [1,2], nums2 = [3,4]

**Output:** 2.50000

**Explanation:** merged array = [1,2,3,4] and median is (2 + 3) / 2 = 2.5.

class Solution:  
def findMedianSortedArrays(self, nums1: List[int], nums2: List[int]) -> float:  
total = []  
for i in nums1:  
total.append(i)  
for i in nums2:  
total.append(i)  
total = sorted(total)  
if len(total) % 2 == 0 :  
x = len(total) - 2  
z = x//2  
ans = (total[z] + total[z+1])/2  
return ans  
elif len(total) % 2 != 0:  
x = len(total) - 1  
z = x//2  
ans = total[z]  
return ans

**Set Matrix Zeroes**

Given an *m* x *n* matrix. If an element is **0**, set its entire row and column to **0**. Do it [**in-place**](https://en.wikipedia.org/wiki/In-place_algorithm).

**Follow up:**

* A straight forward solution using O(*mn*) space is probably a bad idea.
* A simple improvement uses O(*m* + *n*) space, but still not the best solution.
* Could you devise a constant space solution?

numrows = len(input) # 3 rows in your example

numcols = len(input[0]) # 2 columns in your example

**Example 1:**



**Input:** matrix = [[1,1,1],[1,0,1],[1,1,1]]

**Output:** [[1,0,1],[0,0,0],[1,0,1]]

**Example 2:**



**Input:** matrix = [[0,1,2,0],[3,4,5,2],[1,3,1,5]]

**Output:** [[0,0,0,0],[0,4,5,0],[0,3,1,0]]

def setZeroes(self, matrix: List[List[int]]) -> None:

"""

Do not return anything, modify matrix in-place instead.

"""

row = set()

col = set()

n = len(matrix)

m = len(matrix[0])

for i in range(n):

for j in range(m):

if(matrix[i][j] == 0):

row.add(i)

col.add(j)

for i in range(n):

for j in range(m):

if(i in row or j in col):

matrix[i][j] = 0

q = []

for i in range(len(matrix)):

for j in range(len(matrix[0])):

if matrix[i][j] == 0:

q.append((i, j))

used\_i = set()

used\_j = set()

while q:

i, j = q.pop()

if i not in used\_i:

for jj in range(len(matrix[0])):

matrix[i][jj] = 0

used\_i.add(i)

if j not in used\_j:

for ii in range(len(matrix)):

matrix[ii][j] = 0

used\_j.add(j)

s