**Question 1**  
Given three integer arrays arr1, arr2 and arr3 **sorted** in **strictly increasing** order, return a sorted array of **only** the integers that appeared in **all** three arrays.

**Example 1:**

Input: arr1 = [1,2,3,4,5], arr2 = [1,2,5,7,9], arr3 = [1,3,4,5,8]

Output: [1,5]

**Explanation:** Only 1 and 5 appeared in the three arrays.

**Answer:**

def findCommonElements(arr1, arr2, arr3):

ptr1 = ptr2 = ptr3 = 0

result = []

while ptr1 < len(arr1) and ptr2 < len(arr2) and ptr3 < len(arr3):

if arr1[ptr1] == arr2[ptr2] == arr3[ptr3]:

result.append(arr1[ptr1])

ptr1 += 1

ptr2 += 1

ptr3 += 1

elif arr1[ptr1] < min(arr2[ptr2], arr3[ptr3]):

ptr1 += 1

elif arr2[ptr2] < min(arr1[ptr1], arr3[ptr3]):

ptr2 += 1

elif arr3[ptr3] < min(arr1[ptr1], arr2[ptr2]):

ptr3 += 1

return result

**Question 2**

Given two **0-indexed** integer arrays nums1 and nums2, return *a list* answer *of size* 2 *where:*

* answer[0] *is a list of all* ***distinct*** *integers in* nums1 *which are* ***not*** *present in* nums2\*.\*
* answer[1] *is a list of all* ***distinct*** *integers in* nums2 *which are* ***not*** *present in* nums1.

**Note** that the integers in the lists may be returned in **any** order.

**Example 1:**

**Input:** nums1 = [1,2,3], nums2 = [2,4,6]

**Output:** [[1,3],[4,6]]

**Explanation:**

For nums1, nums1[1] = 2 is present at index 0 of nums2, whereas nums1[0] = 1 and nums1[2] = 3 are not present in nums2. Therefore, answer[0] = [1,3].

For nums2, nums2[0] = 2 is present at index 1 of nums1, whereas nums2[1] = 4 and nums2[2] = 6 are not present in nums2. Therefore, answer[1] = [4,6].

**Answer:**  
  
def findDistinctIntegers(nums1, nums2):

set1 = set(nums1)

set2 = set(nums2)

return [list(set1 - set2), list(set2 - set1)]

nums1 = [1, 2, 3]

nums2 = [2, 4, 6]

result = findDistinctIntegers(nums1, nums2)

print(result)

**Question 3**

Given a 2D integer array matrix, return *the* ***transpose*** *of* matrix.

The **transpose** of a matrix is the matrix flipped over its main diagonal, switching the matrix's row and column indices.

**Example 1:**

Input: matrix = [[1,2,3],[4,5,6],[7,8,9]]

Output: [[1,4,7],[2,5,8],[3,6,9]]

**Answer:**

def transpose(matrix):

rows = len(matrix)

cols = len(matrix[0])

transpose = [[0 for \_ in range(rows)] for \_ in range(cols)]

for row in range(rows):

for col in range(cols):

transpose[col][row] = matrix[row][col]

return transpose

matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

result = transpose(matrix)

print(result)

**Question 4**

Given an integer array nums of 2n integers, group these integers into n pairs (a1, b1), (a2, b2), ..., (an, bn) such that the sum of min(ai, bi) for all i is **maximized**. Return *the maximized sum*.

**Example 1:**

Input: nums = [1,4,3,2]

Output: 4

**Explanation:** All possible pairings (ignoring the ordering of elements) are:

1. (1, 4), (2, 3) -> min(1, 4) + min(2, 3) = 1 + 2 = 3
2. (1, 3), (2, 4) -> min(1, 3) + min(2, 4) = 1 + 2 = 3
3. (1, 2), (3, 4) -> min(1, 2) + min(3, 4) = 1 + 3 = 4

So the maximum possible sum is 4.

**Answer:**   
def arrayPairSum(nums):

nums.sort()

max\_sum = 0

for i in range(0, len(nums), 2):

max\_sum += nums[i]

return max\_sum

nums = [1, 4, 3, 2]

result = arrayPairSum(nums)

print(result)

**Question 5** You have n coins and you want to build a staircase with these coins. The staircase consists of k rows where the ith row has exactly i coins. The last row of the staircase **may be** incomplete.

Given the integer n, return *the number of* ***complete rows*** *of the staircase you will build*.

**Example 1:**

**Input:** n = 5

**Output:** 2

**Explanation:** Because the 3rd row is incomplete, we return 2.

**Answer:**

def arrangeCoins(n):

complete\_rows = 0

i = 1

while n >= i:

n -= i

complete\_rows += 1

i += 1

return complete\_rows

**Question 6** Given an integer array nums sorted in **non-decreasing** order, return *an array of* ***the squares of each number*** *sorted in non-decreasing order*.

**Example 1:**

Input: nums = [-4,-1,0,3,10]

Output: [0,1,9,16,100]

**Explanation:** After squaring, the array becomes [16,1,0,9,100]. After sorting, it becomes [0,1,9,16,100]

**Answer:**

def sortedSquares(nums):

squared\_nums = []

for num in nums:

squared\_nums.append(num \* num)

return sorted(squared\_nums)

**Question 7** You are given an m x n matrix M initialized with all 0's and an array of operations ops, where ops[i] = [ai, bi] means M[x][y] should be incremented by one for all 0 <= x < ai and 0 <= y < bi.

Count and return *the number of maximum integers in the matrix after performing all the operations*

**Example 1:**

**Input:** m = 3, n = 3, ops = [[2,2],[3,3]]

**Output:** 4

**Explanation:** The maximum integer in M is 2, and there are four of it in M. So return 4.

**Answer:**

def maxCount(m, n, ops):

min\_row = m

min\_col = n

for op in ops:

min\_row = min(min\_row, op[0])

min\_col = min(min\_col, op[1])

return min\_row \* min\_col

**Question 8** Given two [sparse matrices](https://en.wikipedia.org/wiki/Sparse_matrix) mat1 of size m x k and mat2 of size k x n, return the result of mat1 x mat2. You may assume that multiplication is always possible.

**Example 1:**

**Input:** mat1 = [[1,0,0],[-1,0,3]], mat2 = [[7,0,0],[0,0,0],[0,0,1]]

**Output:**

[[7,0,0],[-7,0,3]]

**Answer:**

def multiplySparseMatrices(mat1, mat2):

m, k = len(mat1), len(mat1[0])

n = len(mat2[0])

# Initialize the result matrix with zeros

result = [[0] \* n for \_ in range(m)]

# Create a transpose of mat2 for efficient column access

mat2T = [[mat2[j][i] for j in range(n)] for i in range(k)]

# Perform matrix multiplication

for i in range(m):

for j in range(n):

for p in range(k):

if mat1[i][p] != 0 and mat2T[j][p] != 0:

result[i][j] += mat1[i][p] \* mat2T[j][p]

return result