**Day-1**

1.First palindrome string in the array.

def palindrome(words):

for i in words:

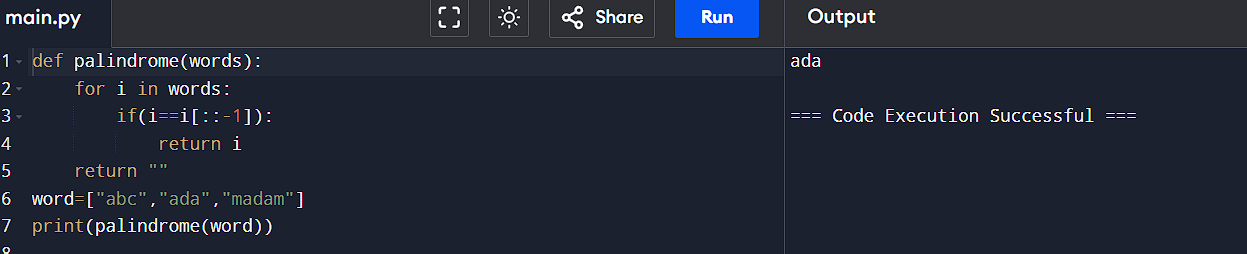
if(i==i[::-1]):

return i

return ""

word=["abc","ada","madam"]

print(palindrome(word))



2. You are given two integer arrays nums1 and nums2 of sizes n and m, respectively. Calculate the following values: answer1 : the number of indices i such that nums1[i] exists in nums2. answer2 : the number of indices i such that nums2[i] exists in nums1 Return [answer1,answer2].

a=[2,3,2]

b=[2,1]

c,d=0,0

i=0

for i in a:

if(i in b):

c+=1

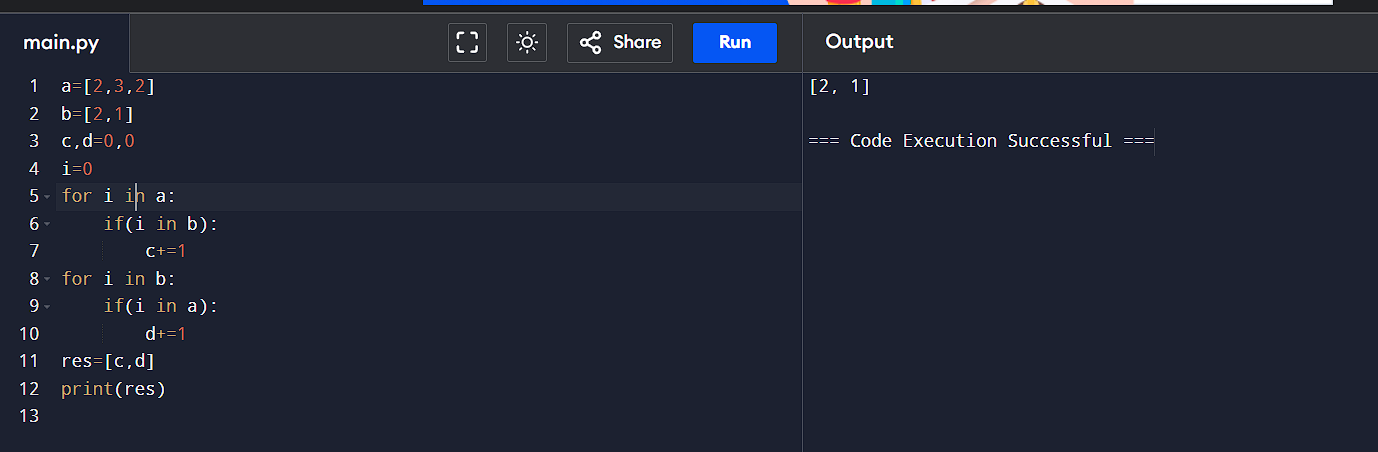
for i in b:

if(i in a):

d+=1

res=[c,d]

print(res)



3. You are given a 0-indexed integer array nums. The distinct count of a subarray of nums is defined as: Let nums[i..j] be a subarray of nums consisting of all the indices from i to j such that 0 <= i <= j < nums.length. Then the number of distinct values in nums[i..j] is called the distinct count of nums[i..j]. Return the sum of the squares of distinct counts of all subarrays of nums. A subarray is a contiguous non-empty sequence of elements within an array.

a=[1,2,1]

n=len(a)

s=0

for i in range(n):

d=set()

for j in range(i,n):

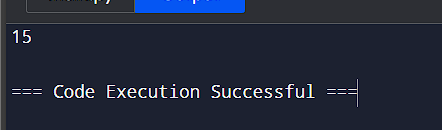
d.add(a[j])

count=len(d)

s+=count\*\*2

print(s)

Output:



4. Given a 0-indexed integer array nums of length n and an integer k, return the number of pairs (i, j) where 0 <= i < j < n, such that nums[i] == nums[j] and (i \* j) is divisible by k. Example 1: Input: nums = [3,1,2,2,2,1,3], k = 2 Output: 4

num=[3,1,2,2,2,1,3]

k=2

count=0

n=len(num)

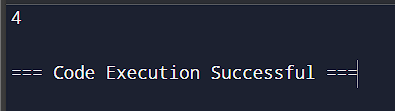
for i in range(n):

for j in range(i+1,n):

if(num[i]==num[j] and (i\*j)%k==0):

count+=1

print(count)



5. Write a program FOR THE BELOW TEST CASES with least time complexity Test Cases: - 1) Input: {1, 2, 3, 4, 5} Expected Output: 5

def maxi(arr):

max\_i=arr[0]

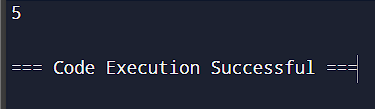
for i in arr:

if(i>max\_i):

max\_i=i

return max\_i

print(maxi([1,2,3,4,5]))



6. You have an algorithm that process a list of numbers. It firsts sorts the list using an efficient sorting algorithm and then finds the maximum element in sorted list. Write the code for the same.

def quick(arr):

if len(arr) <= 1:

return arr

p = arr[len(arr) // 2]

l = [x for x in arr if x < p]

m = [x for x in arr if x == p]

r = [x for x in arr if x > p]

return quick(l) + m + quick(r)

def maxi(arr):

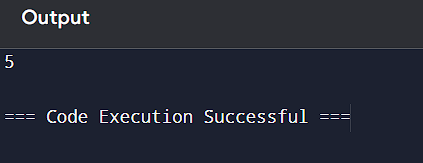
sorti = quick(arr)

max\_i = sorti[-1]

return max\_i

print(maxi([1, 2, 3, 4, 5]))

Output:



7. Write a program that takes an input list of n numbers and creates a new list containing only the unique elements from the original list. What is the space complexity of the algorithm?

def unique(arr):

a=[]

for i in arr:

b=True

for j in a:

if i==j:

b=False

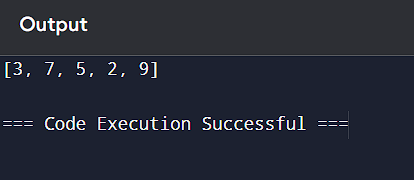
break

if b:

a.append(i)

return a

print(unique([3, 7, 3, 5, 2, 5, 9, 2]))



8. Sort an array of integers using the bubble sort technique. Analyze its time complexity using Big-O notation. Write the code.

def bubble(arr):

n = 0

for \_ in arr:

n+=1

for i in range(n):

swapped=False

for j in range(0,n-i-1):

if arr[j]>arr[j+1]:

arr[j],arr[j+1]=arr[j+1],arr[j]

swapped=True

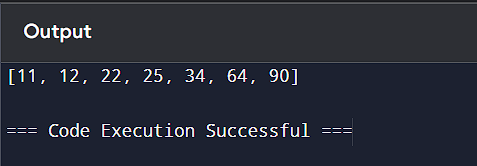
if not swapped:

break

arr1 = [64,34,25,12,22,11,90]

bubble(arr1)

print(arr1)



9. Checks if a given number x exists in a sorted array arr using binary search. Analyze its time complexity using Big-O notation.

def binary\_search(arr, x):

arr.sort()

low = 0

high = len(arr) - 1

while low <= high:

mid = (low + high) // 2

if arr[mid] == x:

return mid

elif arr[mid] < x:

low = mid + 1

else:

high = mid - 1

return -1

arr = [3, 4, 6, -9, 10, 8, 9, 30]

key = 10

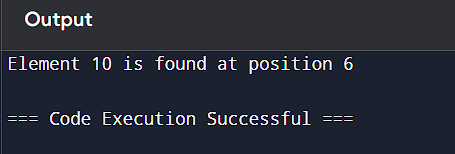
result = binary\_search(arr, key)

if result != -1:

print(f"Element {key} is found at position {result}")

else:

print(f"Element {key} is not found")



10. Given an array of integers nums, sort the array in ascending order and return it. You must solve the problem without using any built-in functions in O(nlog(n)) time complexity and with the smallest space complexity possible.

def merge(left, right):

merged = []

i = j = 0

while i < len(left) and j < len(right):

if left[i] <= right[j]:

merged.append(left[i])

i += 1

else:

merged.append(right[j])

j += 1

while i < len(left):

merged.append(left[i])

i += 1

while j < len(right):

merged.append(right[j])

j += 1

return merged

def merge\_s(arr):

if len(arr) <= 1:

return arr

mid = len(arr) // 2

left = arr[:mid]

right = arr[mid:]

return merge(merge\_s(left), merge\_s(right))

nums = [3, 4, 6, -9, 10, 8, 9, 30]

sorted\_nums = merge\_s(nums)

print("Sorted array:", sorted\_nums)

