1.

def first\_palindromic\_string(words):

for word in words:

if word == word[::-1]:

return word

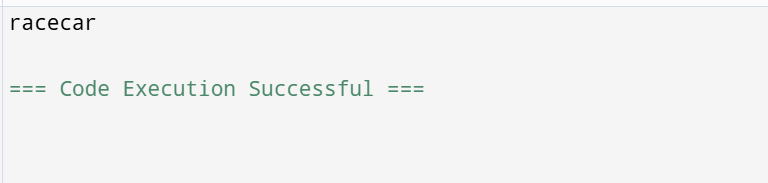
return ""

words = ["abc", "car", "racecar", "cool"]

result = first\_palindromic\_string(words)

print(result)

OUT PUT:



2.

def count\_common\_indices(nums1, nums2):

set1 = set(nums1)

set2 = set(nums2)

answer1 = sum(1 for num in nums1 if num in set2)

answer2 = sum(1 for num in nums2 if num in set1)

return [answer1, answer2]

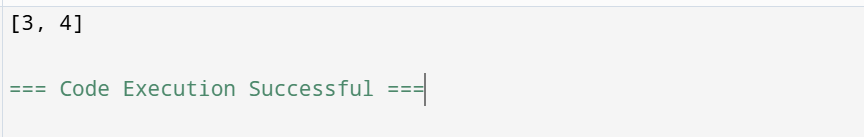
nums1 = [4,3,2,3,1]

nums2 = [2,2,5,2,3,6]

output = count\_common\_indices(nums1, nums2)

print(output)

OUT PUT:



3.

def sum\_of\_squares\_of\_distinct\_counts(nums):

n = len(nums)

total\_sum = 0

for i in range(n):

distinct\_elements = set()

for j in range(i, n):

distinct\_elements.add(nums[j])

distinct\_count = len(distinct\_elements)

total\_sum += distinct\_count \*\* 2

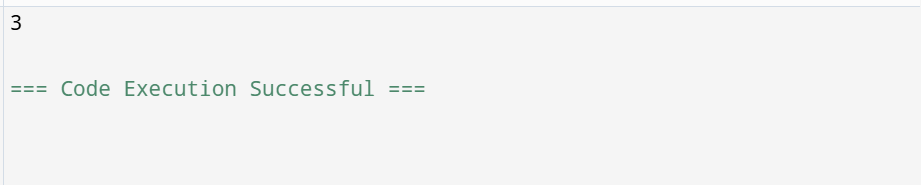
return total\_sum

nums = [1, 1]

output = sum\_of\_squares\_of\_distinct\_counts(nums)

print(output)

OUT PUT:



4.

def count\_pairs(nums, k):

count = 0

n = len(nums)

for i in range(n):

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

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

count += 1

return count

nums = [3, 1, 2, 2, 2, 1, 3]

k = 2

result = count\_pairs(nums, k)

print(result)

OUT PUT:



5.

def find\_maximum(input\_set):

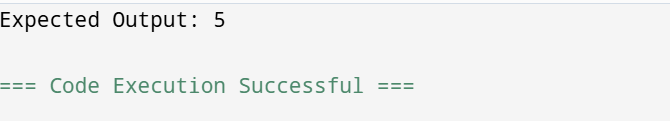
return max(input\_set)

input\_data = {1, 2, 3, 4, 5}

output = find\_maximum(input\_data)

print(f"Expected Output: {output}")

OUT PUT:



6.

def find\_max\_in\_sorted\_list(numbers):

if not numbers:

return "The list is empty."

sorted\_numbers = sorted(numbers)

return sorted\_numbers[-1]

print(find\_max\_in\_sorted\_list([]))

OUT PUT:



7.

def unique\_elements(input\_list):

return list(set(input\_list))

input\_data = [3, 7, 3, 5, 2, 5, 9, 2]

output\_data = unique\_elements(input\_data)

print(output\_data)

OUT PUT:



8.

def bubble\_sort(arr):

n = len(arr)

for i in range(n):

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

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

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

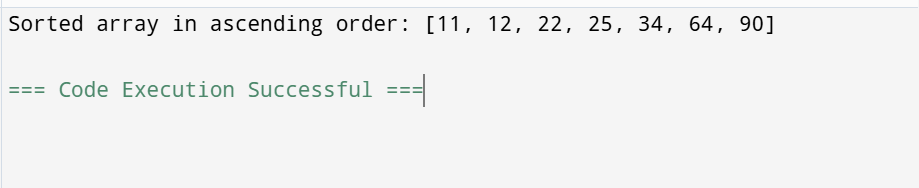
return arr

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

sorted\_array = bubble\_sort(array)

print("Sorted array in descending order:", sorted\_array)

OUT PUT:



9.

def binary\_search(arr, key):

left, right = 0, len(arr) - 1

while left <= right:

mid = (left + right) // 2

if arr[mid] == key:

return f"Element {key} found at index {mid}"

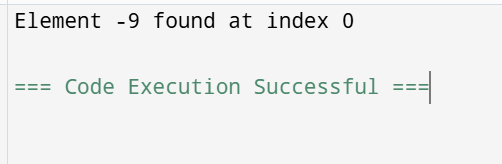
left, right = (mid + 1, right) if arr[mid] < key else (left, mid - 1)

return f"Element {key} not found"

X = sorted([3, 4, 6, -9, 10, 8, 9, 30])

print(binary\_search(X, -9))

OUT PUT:



10.

def merge\_sort(nums):

if len(nums) <= 1:

return nums

mid = len(nums) // 2

left\_half = merge\_sort(nums[:mid])

right\_half = merge\_sort(nums[mid:])

return merge(left\_half, right\_half)

def merge(left, right):

sorted\_array = []

i = j = 0

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

if left[i] < right[j]:

sorted\_array.append(left[i])

i += 1

else:

sorted\_array.append(right[j])

j += 1

sorted\_array.extend(left[i:])

sorted\_array.extend(right[j:])

return sorted\_array

nums = [5, 2, 9, 1, 5, 6]

sorted\_nums = merge\_sort(nums)

print(sorted\_nums)

OUT PUT:

