1. def deepest\_nesting(lst):

max\_depth = 0

stack = [(lst, 1)] # Each element is a tuple (current\_list, current\_depth)

while stack:

current, depth = stack.pop()

max\_depth = max (max\_depth, depth)

for item in current:

if isinstance(item, list):

stack.append((item, depth + 1))

return max\_depth

1. lst = [('a', 1), ('b', 2), ('a', 3)]

result = {}

for key, value in lst:

if key in result:

result[key].append(value)

else:

result[key] = [value]

print(result)

**OUTPUT**

{'a': [1, 3], 'b': [2]}

1. def compress\_string(s):

if not s:

return ""

result = []

i = 0

while i < len(s):

count = 1

# Count consecutive duplicates

while i + 1 < len(s) and s[i] == s[i + 1]:

count += 1

i += 1

# If 3 or more repetitions, compress

if count >= 3:

result.append(f"{s[i]}{count}")

else:

result.extend(s[i] \* count)

i += 1

return ''.join(result)

OUTPUT

print(compress\_string("aaabbcccc")) # Output: "a3bbc4"

print(compress\_string("abc")) # Output: "abc"

print(compress\_string("aaa")) # Output: "a3"

print(compress\_string("aabbbccccdd")) # Output: "aa b3 c4 dd"

1. # Define two sample lists

list1 = [1, 2, 3, 4, 5, 6]

list2 = [4, 5, 6, 7, 8, 9]

# Find the intersection without duplicates

intersection = list(set(list1) & set(list2))

print ("Intersection without duplicates:", intersection)

OUTPUT

Intersection without duplicates: [4, 5, 6]

**10.** def find\_missing\_number(nums):

n = len(nums) + 1 # Since one number is missing

total = n \* (n + 1) // 2

return total - sum(nums)

nums = [1, 2, 4, 5]

missing = find\_missing\_number(nums)

print ("Missing number is:", missing)

**OUTPUT**

Missing number is: 3