**FIND THE DIFFERENCE**

# You are given two strings s and t.  
# String t is generated by random shuffling string s and then add one more letter at a random position.  
# Return the letter that was added to t.  
  
# Var 1  
  
s = input("S: ")  
t = input("T: ")  
  
d = {}  
for i in range(65, 91):  
 d[chr(i)] = 0  
 d[chr(i+32)] = 0  
  
for element in t:  
 d[element] += 1  
  
for element in s:  
 d[element] -= 1  
  
for element in d:  
 if d[element] == 1:  
 print(f"Difference: {element}")  
  
# Var 2  
  
s = input("S: ")  
t = input("T: ")  
  
d = {}  
for i in range(65, 91):  
 d[chr(i)] = i  
 d[chr(i+32)] = i+32  
  
sum\_s = 0  
sum\_t = 0  
for i in range(len(s)):  
 sum\_s += d[s[i]]  
 sum\_t += d[t[i]]  
sum\_t += d[t[-1]]  
  
  
difference = sum\_t - sum\_s  
for letter, code in d.items():  
 if code == difference:  
 print(f"Difference: {letter}")

**4SUM**

# Given four integer arrays a, b, c, and d all of length n,  
# return the number of tuples (i, j, k, l) such that:  
# -------> 0 <= i, j, k, l < n  
# -------> a[i] + b[j] + c[k] + d[l] == 0  
  
n = int(input("N: "))  
  
a = []  
for i in range(n):  
 x = int(input("Element A: "))  
 a.append(x)  
  
b = []  
for i in range(n):  
 x = int(input("Element B: "))  
 b.append(x)  
  
c = []  
for i in range(n):  
 x = int(input("Element C: "))  
 c.append(x)  
  
d = []  
for i in range(n):  
 x = int(input("Element D: "))  
 d.append(x)  
  
scd = []  
for nc in c:  
 for nd in d:  
 scd.append(nc+nd)  
sab = []  
for na in a:  
 for nb in b:  
 sab.append(na+nb)  
  
my\_dict = {}  
for element in sab:  
 my\_dict[element] = 0  
  
for element in sab:  
 my\_dict[element] += 1  
  
number = 0  
for element in scd:  
 if element in my\_dict.keys():  
 number += my\_dict[element]  
  
print(f"Number: {number}")

**SUBSTRING CONCATENATION ALL WORDS**

# You are given a string s and an array of strings words of the same length.  
# Return all starting indices of substring(s) in s that is a concatenation  
# of each word in words exactly once, in any order, and without any intervening characters.  
# You can return the answer in any order.  
  
  
def backtracking(niv):  
 global st, n, words, final\_words  
 for val in words:  
 st[niv] = val  
 if st[niv] not in st[:niv]:  
 if niv == n:  
 final\_words.append(st[1:])  
 else:  
 backtracking(niv+1)  
  
  
s = input("S: ")  
words = input("Words: ").split()  
  
n = len(words)  
st = [0]\*(n+1)  
final\_words = []  
backtracking(1)  
  
find\_words = []  
for word in final\_words:  
 aux = ''  
 for mini\_word in word:  
 aux += mini\_word  
 find\_words.append(aux)  
  
solution = []  
for word in find\_words:  
 p = s.find(word)  
 if p == -1:  
 continue  
 else:  
 while p != -1:  
 solution.append(p)  
 p = s.find(word, p + len(word))  
  
print("Positions:", \*solution)