Week-7

1. Given a tuple and a positive integer k, the task is to find the count of distinct pairs in the tuple whose sum is equal to **K**.

**Examples:**

**Input:** t = (5, 6, 5, 7, 7, 8 ), K = 13   
**Output:** 2   
**Explanation:**   
Pairs with sum K( = 13) are  {(5, 8), (6, 7), (6, 7)}.   
Therefore, distinct pairs with sum K( = 13) are { (5, 8), (6, 7) }.   
Therefore, the required output is 2.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 1,2,1,2,5  3 | 1 |
| 1,2  0 | 0 |

Program:

t=tuple(input().split(','))

t=tuple(int(i)for i in t)

k=int(input())

result=[]

for i in t:

for j in t:

if(i+j==k and not ([i,j] in result or [j,i]in result)):

result.append([i,j])

print(len(result))

output:

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 5,6,5,7,7,8  13 | 2 | 2 |  |
|  | 1,2,1,2,5  3 | 1 | 1 |  |
|  | 1,2  0 | 0 | 0 |  |

1. The **DNA sequence** is composed of a series of nucleotides abbreviated as 'A', 'C', 'G', and 'T'.

* For example, "ACGAATTCCG" is a **DNA sequence**.

When studying **DNA**, it is useful to identify repeated sequences within the DNA.

Given a string s that represents a **DNA sequence**, return all the **10-letter-long** sequences (substrings) that occur more than once in a DNA molecule. You may return the answer in **any order**.

**Example 1:**

**Input:** s = "AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT"

**Output:** ["AAAAACCCCC","CCCCCAAAAA"]

**Example 2:**

**Input:** s = "AAAAAAAAAAAAA"

**Output:** ["AAAAAAAAAA"]

**For example:**

| **Input** | **Result** |
| --- | --- |
| AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT | AAAAACCCCC  CCCCCAAAAA |

Program:

def findRepeatedDnaSequences(s):

sequences = {}

result = []

for i in range(len(s) - 9):

sequence = s[i:i+10]

if sequence in sequences:

if sequences[sequence] == 1:

result.append(sequence)

sequences[sequence] += 1

else:

sequences[sequence] = 1

return "\n".join(result)

s = input()

print(findRepeatedDnaSequences(s))

output:

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT | AAAAACCCCC  CCCCCAAAAA | AAAAACCCCC  CCCCCAAAAA |  |
|  | AAAAAAAAAAAAA | AAAAAAAAAA | AAAAAAAAAA |  |

1. There is a malfunctioning keyboard where some letter keys do not work. All other keys on the keyboard work properly.

Given a string text of words separated by a single space (no leading or trailing spaces) and a string brokenLetters of all distinct letter keys that are broken, return the number of words in text you can fully type using this keyboard.

Example 1:

Input: text = "hello world", brokenLetters = "ad"

Output:

1

Explanation: We cannot type "world" because the 'd' key is broken.

**For example:**

| **Input** | **Result** |
| --- | --- |
| hello world  ad | 1 |
| Faculty Upskilling in Python Programming  ak | 2 |

Program:

def canType(text, brokenLetters):

words = text.split(' ')

count = 0

broken\_set = set(brokenLetters)

for word in words:

if not any(char.lower() in broken\_set for char in word):

count += 1

return count

text = input()

brokenLetters = input()

print(canType(text, brokenLetters))

output:

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | hello world  ad | 1 | 1 |  |
|  | Welcome to REC  e | 1 | 1 |  |
|  | Faculty Upskilling in Python Programming  ak | 2 | 2 |  |

1. Coders here is a simple task for you, Given string str. Your task is to check whether it is a binary string or not by using python set.

Examples:

Input: str = "01010101010"

Output: Yes

Input: str = "REC101"

Output: No

**For example:**

| **Input** | **Result** |
| --- | --- |
| 01010101010 | Yes |
| 010101 10101 | No |

Program:

s = str(input())

set\_s = set(s)

if set\_s == {'0', '1'}:

print('Yes')

else:

print('No')

output:

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 01010101010 | Yes | Yes |  |
|  | REC123 | No | No |  |
|  | 010101 10101 | No | No |  |

1. Given an array of integers nums containing n + 1 integers where each integer is in the range [1, n] inclusive.There is only **one repeated number** in nums, return *this repeated number*. Solve the problem using set.

**Example 1:**

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

**Output:** 2

**Example 2:**

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

**Output:** 3

**For example:**

| **Input** | **Result** |
| --- | --- |
| 1 3 4 4 2 | 4 |

Program:

def find\_duplicate(nums):

unique\_nums = set()

for num in nums:

if num in unique\_nums:

return num

unique\_nums.add(num)

return -1

nums = list(input().split())

print(find\_duplicate(nums))

output:

| **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- |
|  | 1 3 4 4 2 | 4 | 4 |  |
|  | 1 2 2 3 4 5 6 7 | 2 | 2 |  |