# 08 – Tuple/Set

Examples:

Input: str = "01010101010"

Output: Yes

Input: str = "REC101"

Output: No

For example:

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

Ex. No. : 8.1 Date:

Register No.: Name:

Binary String

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.

mySet = ('0','1')

str = input()

check = True

for i in str:

if i not in mySet:

check = False

break

if check:

print("Yes")

else:

print("No")

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 |

Ex. No. : 8.2 Date:

Register No.: Name:

Check Pair

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.

t = set(map(int,input().split(",")))

k = int(input())

t = list(t)

count = []

for i in range(len(t)):

for j in range(i,len(t)):

if t[i] + t[j] == k:

temp = (t[i],t[j])

count.append(temp)

print(len(count))

Example 1:

Input: s = "AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT"

Output: ["AAAAACCCCC","CCCCCAAAAA"]

Example 2:

Input: s = "AAAAAAAAAAAAA"

Output: ["AAAAAAAAAA"]

For example:

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

Ex. No. : 8.3 Date:

Register No.: Name:

DNA Sequence

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.

def find(s):

if len(s) <= 10:

return []

sequences = {}

result = set()

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

seq = s[i:i+10]

if seq in sequences:

result.add(seq)

else:

sequences[seq] = 1

return list(result)

dnaseq = input()

dna = find(dnaseq)

for i in dna:

print(i)

**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 |

Ex. No. : 8.4 Date:

Register No.: Name:

Print repeated no

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](http://118.185.187.137/moodle/mod/resource/view.php?id=734).

def findDuplicate(nums):

visited = set()

for num in nums:

if num in visited:

return num

visited.add(num)

nums = list(map(int,input().split()))

print(findDuplicate(nums))

[Sample](https://www.rajalakshmicolleges.net/moodle/mod/quiz/view.php?id=5127) Input:

5 4

1 2 8 6 5

2 6 8 10

[Sample](https://www.rajalakshmicolleges.net/moodle/mod/quiz/view.php?id=5127) Output:

1 5 10

3

[Sample](https://www.rajalakshmicolleges.net/moodle/mod/quiz/view.php?id=5127)  Input:

5 5

1 2 3 4 5

1 2 3 4 5

[Sample](https://www.rajalakshmicolleges.net/moodle/mod/quiz/view.php?id=5127) Output:

NO SUCH ELEMENTS

For example:

| Input | Result |
| --- | --- |
| 5 4  1 2 8 6 5  2 6 8 10 | 1 5 10  3 |

Ex. No. : 8.5 Date:

Register No.: Name:

Remove repeated

Write a program to eliminate the common elements in the given 2 arrays and print only the non-repeating elements and the total number of such non-repeating elements.

Input Format:

The first line contains space-separated values, denoting the size of the two arrays in integer format respectively.

The next two lines contain the space-separated integer arrays to be compared.

a,b = input().split()

tup1 =tuple(map(int,input().split()))

tup2= tuple(map(int,input().split()))

temp = []

for i in tup1:

if i not in tup2:

temp.append(i)

for i in tup2:

if i not in tup1:

temp.append(i)

for i in range(len(temp)):

if i< len(temp)-1:

print(temp[i],end=" ")

else:

print(temp[i])

print(len(temp))

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 |

Ex. No. : 8.6 Date:

Register No.: Name:

Malfunctioning Keyboard

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.

string = input()

Bword = input()

count = 0

for i in Bword:

if i in string:

count +=1

print(count)



Example 1:

Input: words = ["Hello","Alaska","Dad","Peace"]

Output: ["Alaska","Dad"]

Example 2:

Input: words = ["omk"]

Output: []

Example 3:

Input: words = ["adsdf","sfd"]

Output: ["adsdf","sfd"]

For example:

| Input | Result |
| --- | --- |
| 4  Hello  Alaska  Dad  Peace | Alaska  Dad |

Ex. No. : 8.7 Date:

Register No.: Name:

American keyboard

Given an array of strings words, return the words that can be typed using letters of the alphabet on only one row of American keyboard like the image below.

In the American keyboard:

* the first row consists of the characters "qwertyuiop",
* the second row consists of the characters "asdfghjkl", and
* the third row consists of the characters "zxcvbnm".

n = int(input())

temp = []

count = []

for i in range(n):

temp.append(input())

upper = set("qwertyuiop")

middle = set("asdfghjkl")

lower = set("zxcvbnm")

for i in temp:

check = set(i.lower())

if check.issubset(upper) or check.issubset(middle) or check.issubset(lower):

count.append(i)

if len(count):

for i in count:

print(i)

else:

print("No words")