

08 – Tuple/Set



Ex. No. : 8.1

Date:

Register No.: 230701348

Name: N SUBRAMANIAN

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.

```
a=input()
flag=0
for i in a:
    if i not in "01":
        flag=1
        break
if flag:
    print("No")
else:
    print("Yes")
```



Examples:

Input: str = "01010101010"

Output: Yes

Input: str = "REC101"

Output: No

For example:

Input	Result
01010101010	Yes
010101 10101	No



Ex. No. : 8.2

Date:

Register No.: 230701348

Name: N SUBRAMANIAN

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**.

```
def count_distinct_pairs(t, K):  
    distinct_pairs = set()  
    for i in range(len(t)):  
        for j in range(i + 1, len(t)):  
            if t[i] + t[j] == K:  
                distinct_pairs.add((min(t[i],  
t[j]), max(t[i], t[j])))  
    return len(distinct_pairs)  
t_input = input()  
t = tuple(map(int, t_input.split(',')))  
K = int(input())  
print(count_distinct_pairs(t, 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



Ex. No. : 8.3

Date:

Register No.: 230701348

Name: N SUBRAMANIAN

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 findRepeatedSequences(s):  
    sequences =  
    {}  
    result = []  
    for i  
in range(len(s) - 9):  
        seq = s[i:i+10]  
        sequences[seq] =  
sequences.get(seq, 0) + 1  
        if  
sequences[seq] ==  
2:  
            result.append(seq)  
    return result  
s1 = input()  
for i in  
findRepeatedSequences(s1):  
    print(i)
```



Example 1:

Input: s = "AAAAACCCCCAAAAACCCCCAAAAAGGGTTT"

Output: ["AAAAACCCCC","CCCCAAAAA"]

Example 2:

Input: s = "AAAAAAAAAAAAA"

Output: ["AAAAAAAAAAAA"]

For example:

Input	Result
AAAAACCCCCAAAAACCCCCAAAAAGGGTTT	AAAAACCCCC CCCCAAAAA



Ex. No. : 8.4

Date:

Register No.: 230701348

Name: N SUBRAMANIAN

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](#).

```
a=input()
b=a.split(' ')
for i in range(len(b)):
    b[i]=int(b[i])
for i in b:
    if b.count(b[i])>1:
        print(b[i])
        break
```



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. : 8.5

Date:

Register No.: 230701348

Name: N SUBRAMANIAN

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=input()
b=input()
c=input()
d=[]
s1=b.split(' ')
s2=c.split(' ')
for i in s1:
    if i not in s2:
        d.append(i)
for j in s2:
    if j not in s1:
        d.append(j)
for m in d:
    print(m,end=' ')
print("\n",len(d),sep="")
```



Sample Input:

5 4

1 2 8 6 5

2 6 8 10

Sample Output:

1 5 10

3

Sample Input:

5 5

1 2 3 4 5

1 2 3 4 5

Sample Output:

NO SUCH ELEMENTS

For example:

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



Ex. No. : 8.6

Date:

Register No.: 230701348

Name: N SUBRAMANIAN

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.

```
a=input()
a=a.lower()
b=input()
count=0
words=a.split(' ')
for i in range(len(words)):
    for j in range(len(b)):
        if b[j] in words[i]:
            count+=1
uniquecount=len(words)-count
print(uniquecount)
```



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. : 8.7

Date:

Register No.: 230701348

Name: N SUBRAMANIAN

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".

```
a=int(input())
b=[]
c=[]
for i in range(a):
    b.append(input())
for j in b:
    x,y,z=0,0,0
    for k in j:
        if k in "qwertyuiopQWERTYUIOP":
            x+=1
        elif k in "asdfghjklASDFGHJKL":
            y+=1
        elif k in "zxcvbnmZXCVBNM":
            z+=1
    if (x>0 and y==0 and z==0) or (x==0 and y>0 and z==0) or (x==0 and y==0 and z>0):
        c.append(j)
if len(c)==0:
    print("No words");
else:
    for i in c:
        print(i)
```



~ `	! 1	@ 2	# 3	\$ 4	% 5	^ 6	& 7	* 8	(9) 0	- _	+ =	← Backspace
Tab ↔	Q	W	E	R	T	Y	U	I	O	P	{ [}]	 \ _
Caps Lock ↑	A	S	D	F	G	H	J	K	L	:	" '	Enter ↵	
Shift ↑	Z	X	C	V	B	N	M	< ,	> .	? /	Shift ↑		
Ctrl	Win Key	Alt								Alt	Win Key	Menu	Ctrl

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