

08 – Tuple/Set

Ex. No. : 8.1

Date: 25.05.24

Register No.: 231901020

Name: KAVIYA.V

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.

CODING:

```
a=input()
for i in range(len(a)):
    if a[i] not in ['0','1']:
        print('No')
        exit()
    elif a[i]==" ":
        print('No')
        exit()
    elif i==len(a)-1:
        print('Yes')
        exit()
    else:
        continue
```

	Input	Expected	Got	
✓	01010101010	Yes	Yes	✓
✓	REC123	No	No	✓
✓	010101 10101	No	No	✓

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: 25.05.24

Register No.: 231901020

Name: KAVIYA.V

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.

CODING:

```
t=tuple(map(int,input().split(',')))
```

```
k=int(input())
```

```
s=set(t)
```

```
count=0
```

```
for x in s:
```

```
    if k-x in s:
```

```
        count+=1
```

```
result=count//2
```

```
print(result)
```

	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	✓

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: 25.05.24

Register No.: 231901020

Name: KAVIYA.V

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**.

CODING:

```
s=input()
```

```
sc={} 
```

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

```
    substring=s[i:i+10]
```

```
    sc[substring]=sc.get(substring,0)+1
```

```
rs=[substring for substring,count in sc.items() if count>1]
```

```
for substring in rs:
```

```
    print(substring)
```

	Input	Expected	Got	
✓	AAAAACCCCCAAAAACCCCCAAAAAGGGTTT	AAAAACCCCC CCCCAAAAA	AAAAACCCCC CCCCAAAAA	✓
✓	AAAAAAAAAAAAA	AAAAAAAAAA	AAAAAAAAAA	✓

Example 1:

Input: s = "AAAAACCCCCAAAAACCCCCAAAAAGGGTTT"

Output: ["AAAAACCCCC","CCCCCAAAA"]

Example 2:

Input: s = "AAAAAAAAAAAA"

Output: ["AAAAAAAAAA"]

For example:

Input	Result
AAAAACCCCCAAAAACCCCCAAAAAGGGTTT	AAAAACCCCC CCCCCAAAA

Ex. No. : 8.4

Date: 25.05.24

Register No.: 231901020

Name: KAVIYA.V

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](#).

CODING:

```
n=list(map(int,input().split()))
```

```
ns=set()
```

```
for i in n:
```

```
    if i in ns:
```

```
        print(i)
```

```
        break
```

```
    else:
```

```
        ns.add(i)
```

	Input	Expected	Got	
✓	1 3 4 4 2	4	4	✓
✓	1 2 2 3 4 5 6 7	2	2	✓

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: 25.05.24

Register No.: 231901020

Name: KAVIYA.V

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.

CODING:

```
n, m = map(int, input().split()) # Sizes of arrays
arr1 = list(map(int, input().split())) # First array
arr2 = list(map(int, input().split())) # Second array
set1 = set(arr1)
set2 = set(arr2)

non_repeating_elements =
(set1.symmetric_difference(set2)).difference(set1.intersection(set2))

if non_repeating_elements:
    print(*sorted(non_repeating_elements))
    print(len(non_repeating_elements))
else:
```

print("NO SUCH ELEMENTS")

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 2 8 6 5 2 6 8 10	1 5 10 3

Ex. No. : 8.6

Date:

Register No.: 231901020

Name: KAVIYA.V

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.

CODING:

```
n=input()
```

```
bl=input()
```

```
words=n.split()
```

```
vw=0
```

```
for word in words:
```

```
    if any(letter in bl for letter in word):
```

```
        continue
```

```
    else:
```

```
        vw+=1
```

```
print(vw)
```

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.: 231901020

Name: KAVIYA.V

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".

CODING:

```
r1=set('qwertyuiop')
```

```
r2=set('asdfghjkl')
```

```
r3=set('zxcvbnm')
```

```
nw=int(input())
```

```
found=False
```

```
for i in range(nw):
```

```
    w=input()
```

```
    wl=w.lower()
```

```
    if all(char in r1 for char in wl) or \
```

```
    all(char in r2 for char in wl) or \
```

```
    all(char in r3 for char in wl):
```

```
        print(w)
```

found=True

if not found:

print("No words")

	Input	Expected	Got	
✓	4 Hello Alaska Dad Peace	Alaska Dad	Alaska Dad	✓
✓	1 omk	No words	No words	✓
✓	2 adsfd afd	adsfd afd	adsfd afd	✓

~ `	!	@	#	\$	%	^	&	*	()	-	+	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