Skip to main content

REC-OCATS-1

CS23336-Introduction to Python Programming

Started on	Monday, 21 October 2024, 8:52 PM
State	Finished
Completed on	Monday, 21 October 2024, 10:27 PM
Time taken	1 hour 35 mins
Marks	10.00/10.00
Grade	**100.00** out of 100.00

Question 1

Correct
Mark 1.00 out of 1.00
□ Flag question

Question text

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

Answer:(penalty regime: 0 %)

```
a=input()
b=int(input())
a=set(a)
a.remove(',')
a=tuple(a)
res=0
for i in a:
    for j in
range(a.index(i),len(a)
):
        if
int(i)+int(a[j])==b:
            res+=1

print(res)
```

Feedback

	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

Passed all tests!

Correct
Marks for this submission: 1.00/1.00.

Question 2

Correct
Mark 1.00 out of 1.00
□ Flag question

Question text

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.

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

5 5
1 2 3 4 5
1 2 3 4 5
	NO SUCH ELEMENTS

Answer:(penalty regime: 0 %)

```
import re
a=input()
b=input()
c=input()
b=(re.findall(r'[0-9]+',b))
c=(re.findall(r'[0-9]+',c))
b=set(b)
c=set(c)
d=b^c
b={0}
for i in d:
   b.add(int(i))
b.discard(0)
b=list(b)
b.sort()
if len(b)==0:
```

Feedback

Input	Expected	Got

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

3 3
10 10 10
10 11 12
	11 12
	2
		11 12
		2

5 5
1 2 3 4 5
1 2 3 4 5
	NO SUCH ELEMENTS	NO SUCH ELEMENTS

Passed all tests!

Correct
Marks for this submission: 1.00/1.00.

Question 3

Correct
Mark 1.00 out of 1.00
□ Flag question

Question text

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`

Answer:(penalty regime: 0 %)

```
a=input()
b=set(a)
c={'1','0'}
if c==b:
    print("Yes")
else:
    print("No")
```

Feedback

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

Passed all tests!

Correct
Marks for this submission: 1.00/1.00.

Question 4

Correct
Mark 1.00 out of 1.00
□ Flag question

Question text

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

Answer:(penalty regime: 0 %)

```
s=input()
n=set()
p=set()
for i in range(len(s)-9):
    c=s[i:i+10]
    if c in n:
        p.add(c)
    else:
        n.add(c)
s=list(p)
for i in range(len(s)-1,-1,-1):
    print(s[i])
```

Feedback

Input	Expected	Got
AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT	AAAAACCCCC CCCCCAAAAA	AAAAACCCCC CCCCCAAAAA
AAAAAAAAAAAAA	AAAAAAAAAA	AAAAAAAAAA

Passed all tests!

Correct
Marks for this submission: 1.00/1.00.

Question 5

Correct
Mark 1.00 out of 1.00
□ Flag question

Question text

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"`.

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
2 adsfd afd	adsfd afd

Answer:(penalty regime: 0 %)

```
def function(word,rows):
    l=word.lower()
    for row in rows:
        if all(char in row for char in l):
            return True
    return False
def find(words):
    rows=["qwertyuiop","asdfghjkl","zxcvbnm"]
    res=[]
    for word in words:
        if function(word,rows):
            res.append(word)
```

Feedback

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

Passed all tests!

Correct
Marks for this submission: 1.00/1.00.

Question 6

Correct
Mark 1.00 out of 1.00
□ Flag question

Question text

Program to print all the distinct elements in an array. Distinct elements are nothing but the unique (non-duplicate) elements present in the given array.

Input Format:

First line take an Integer input from stdin which is array length n.

Second line take n Integers which is inputs of array.

Output Format:

Print the Distinct Elements in Array in single line which is space Separated

Example Input:

5

1 2 2 3 4

Output:

1 2 3 4

Example Input:

6

1 1 2 2 3 3

Output:

1 2 3

For example:

Input	Result
5 1 2 2 3 4	1 2 3 4

Answer:(penalty regime: 0 %)

```
a=int(input())
b=[]
for i in range(a):
    f=int(input())
    b.append(f)
c=set(b)
b=tuple(c)
print(*b)
```

Feedback

Input	Expected	Got
5 1 2 2 3 4	1 2 3 4	1 2 3 4

6 1 1 2 2 3 3	1 2 3	1 2 3
5 11 22 11 22 11	11 22	11 22
10 1 2 3 4 5 1 2 3 4 5	1 2 3 4 5	1 2 3 4 5

Passed all tests!

Correct
Marks for this submission: 1.00/1.00.

Question 7

Correct
Mark 1.00 out of 1.00
□ Flag question

Question text

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

Answer:(penalty regime: 0 %)

```
import re
a=input()
a=a.lower()
b=input()
b=b.lower()
c=re.findall(r'[a-z]+',a)
d=re.findall(r'[a-z]',b)
res=0
for i in d:
    for j in c:
        if i not in j:
            pass
        else:
            c.remove(j)
print(len(c))
```

Feedback

Input	Expected	Got

`hello world`
`ad`
	1	1

`Welcome to REC`
`e`
	1	1

`Faculty Upskilling in Python Programming`
`ak`
	2	2

Passed all tests!

Correct
Marks for this submission: 1.00/1.00.

Question 8

Correct
Mark 1.00 out of 1.00
□ Flag question

Question text

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

Answer:(penalty regime: 0 %)

```
a=input()
a=tuple(a)
n=tuple(i for i in a if
i.strip())
b=set(a)
for i in b:
    if n.count(i)>=2:
        print(i)
        break
```

Feedback

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

Passed all tests!

Correct
Marks for this submission: 1.00/1.00.

Question 9

Correct
Mark 1.00 out of 1.00
□ Flag question

Question text

Check if a set is a subset of another set.

Example:

Sample Input1:

mango apple

mango orange

mango

output1:

yes

set3 is subset of set1 and set2

```
input2:

mango orange

banana orange

grapes

output2:
```

no

For example:

Test	Input	Result
1	mango apple mango orange mango	yes set3 is subset of set1 and set2
2	mango orange banana orange grapes	No

Answer:(penalty regime: 0 %)

```
s1=set(input().strip().split())
s2=set(input().strip().split())
s3=set(input().strip().split())
if s3.issubset(s1) and s3.issubset(s2):
    print('yes')
    print('set3 is subset of set1 and set2')
else:
    print('No')
```

Feedback

Test	Input	Expected	Got
1	mango apple mango orange mango	yes set3 is subset of set1 and set2	yes set3 is subset of set1 and set2
2	mango orange banana orange grapes	No	No

Passed all tests!

Correct
Marks for this submission: 1.00/1.00.

Question 10

Correct
Mark 1.00 out of 1.00
□ Flag question

Question text

You are given an integer tuple `nums` containing distinct numbers. Your task is to perform a sequence of operations on this tuple until it becomes empty. The operations are defined as follows:

1. If the first element of the tuple has the smallest value in the entire tuple, remove it.
2. Otherwise, move the first element to the end of the tuple.

You need to return an integer denoting the number of operations required to make the tuple empty.

Constraints

- The input tuple `nums` contains distinct integers.
- The operations must be performed using tuples and sets to maintain immutability and efficiency.

- Your function should accept the tuple `nums` as input and return the total number of operations as an integer.

Example:

Input: nums = (3, 4, -1)
Output: 5

Explanation:
Operation 1: [3, 4, -1] -> First element is not the smallest, move to the end -> [4, -1, 3]
Operation 2: [4, -1, 3] -> First element is not the smallest, move to the end -> [-1, 3, 4]
Operation 3: [-1, 3, 4] -> First element is the smallest, remove it -> [3, 4]
Operation 4: [3, 4] -> First element is the smallest, remove it -> [4]
Operation 5: [4] -> First element is the smallest, remove it -> []
Total operations: 5

For example:

Test	Result
`print(count_operations((3, 4, -1)))`	5

Answer:(penalty regime: 0 %)

```
def count_operations(nums: tuple):
    # Your implementation here
    op=0
    nums=list(nums)
    while len(nums):
        if nums[0]==min(nums):
            nums.remove(nums[0])
            op+=1
        else:
            t=nums[0]
```

Reset answer

Feedback

Test	Expected	Got
`print(count_operations((3, 4, -1)))`	5	5
`print(count_operations((1, 2, 3, 4, 5)))`	5	5
`print(count_operations((5, 4, 3, 2, 1)))`	15	15
`print(count_operations((42, )))`	1	1
`print(count_operations((-2, 3, -5, 4, 1)))`	11	11

Passed all tests!

Correct
Marks for this submission: 1.00/1.00.

Save the state of the flags

Finish review