Dashboard / My courses / PSPP/PUP / Experiments based on Tuples, Sets and its operations / Week7_Coding

Started on	Thursday, 23 May 2024, 9:40 PM
State	Finished
Completed on	Thursday, 23 May 2024, 9:55 PM
Time taken	15 mins 30 secs
Marks	5.00/5.00
Grade	**100.00** out of 100.00

Question **1**

Correct

Mark 1.00 out of 1.00

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 %)

```
def find_duplicate(nums):
    seen = set()
    for num in nums:
        if num in seen:
            return num
        seen.add(num)
if __name__ == "__main__":
    nums = list(map(int, input().split()))
    duplicate = find_duplicate(nums)
    print(f"{duplicate}")
```

	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 **2**

Correct

Mark 1.00 out of 1.00

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 %)

```
def findRepeatedDnaSequences(s):
    if len(s) < 10:
        return []

    seen, repeated, order = set(), set(), []

    for i in range(len(s) - 9):
        substring = s[i:i+10]
        if substring in seen:
            if substring not in repeated:
                repeated.add(substring)
                order.append(substring)
        else:
            seen.add(substring)

    return order
input_str = input()
repeated_sequences = findRepeatedDnaSequences(input_str)
for sequence in repeated_sequences:
    print(sequence)

```

	Input	Expected	Got	
✓	AAAAACCCCCAAAAACCCCCCAAAAAGGGTTT	AAAAACCCCC CCCCCAAAAA	AAAAACCCCC CCCCCAAAAA	✓
✓	AAAAAAAAAAAAA	AAAAAAAAAA	AAAAAAAAAA	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **3**

Correct

Mark 1.00 out of 1.00

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

Answer: (penalty regime: 0 %)

```
size1, size2, = map(int, input().split())
arr1 = list(map(int, input().split()))
arr2 = list(map(int, input().split()))
non_repeating = [x for x in arr1 if x not in arr2] + [x for x in arr2 if x not in arr1]
if non_repeating:
    non_repeating = sorted(set(non_repeating))
    print(*non_repeating)
    print(len(non_repeating))
else:
    print("NO SUCH ELEMENTS")
```

	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	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

Question **4**

Correct

Mark 1.00 out of 1.00

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 %)

```
def is_binary_string(s):
    binary_set = {'0', '1'}
    input_set = set(s)
    if input_set.issubset(binary_set):
        return "Yes"
    else:
        return "No"

input_str = input()
print(is_binary_string(input_str))


```

	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 **5**

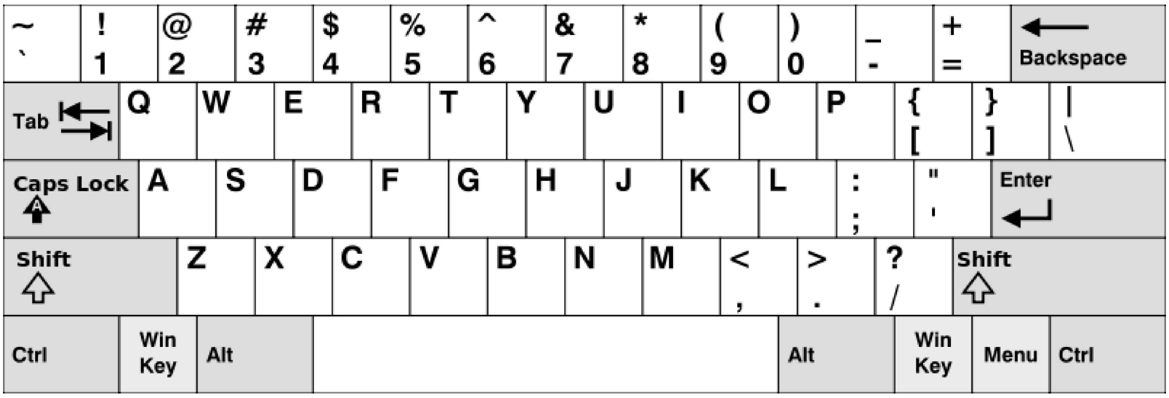
Correct

Mark 1.00 out of 1.00

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 %)

```
a = int(input())
c = []
for i in range(a):
    c.append(input())
d = []
r1 = "qwertyuiop"
r2 = "asdfghjkl"
r3 = "zxcvbnm"
for i in c:
    l = ""
    for j in i.lower():
        if l=="":
            if j in r1:l=r1
            elif j in r2:l=r2
            else:l=r3
        if j not in l:
            d.append(i)
            break
k = 1
for i in c:
    if i not in d:
        k = 0
        print(i)
if k:
    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	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

◄ Week7_MCQ

Jump to...

Dictionary ►