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

Started on	Friday, 24 May 2024, 9:28 AM
State	Finished
Completed on	Sunday, 26 May 2024, 9:28 AM
Time taken	2 days
Marks	4.00/5.00
Grade	**80.00** 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 %)

```python
def find_duplicate(nums):
    seen = set()
    for num in nums:
        if num in seen:
            return num
        seen.add(num)
nums = list(map(int, input().split()))
print(find_duplicate(nums))


```

	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

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'}
    return set(s).issubset(binary_set)

def main():
    s = input().strip()
    if is_binary_string(s):
        print("Yes")
    else:
        print("No")

if __name__ == "__main__":
    main()
```

	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 **3**

Correct

Mark 1.00 out of 1.00

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 %)

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

seen = {}
distinct_pairs = set()

for num in t:
    complement = K - num
    if complement in seen and seen[complement] > 0:
        distinct_pairs.add((min(num, complement), max(num, complement)))
        seen[complement] -= 1
    else:
        seen[num] = seen.get(num, 0) + 1

print(len(distinct_pairs))
```

	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 **4**

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 %)

```
n, m = map(int, input().split())
array1 = list(map(int, input().split()))
array2 = list(map(int, input().split()))
set1 = set(array1)
set2 = set(array2)
symmetric_diff = set1.symmetric_difference(set2)
non_repeating_elements = [x for x in symmetric_diff if x not in set1 or x not in set2]
if non_repeating_elements:
    print(*non_repeating_elements)
    print(len(non_repeating_elements))
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 **5**

Incorrect

Mark 0.00 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 %)

```python
s = input()
A = set()
B = set()
for i in range(len(s) - 9):
    C = s[i:i + 10]
    if C in A:
        B.add(C)
    else:
        A.add(C)
for seq in B:
    print(seq)
```

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

Your code must pass all tests to earn any marks. Try again.

Show differences

Incorrect

Marks for this submission: 0.00/1.00.

◄ Week7_MCQ

Jump to...

Dictionary ►