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

Started on	Friday, 24 May 2024, 8:39 AM
State	Finished
Completed on	Friday, 24 May 2024, 9:11 AM
Time taken	31 mins 58 secs
Marks	5.00/5.00
Grade	**100.00** out of 100.00

Question **1**

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
1
2  def count_distinct_pairs(t, k):
3      count = 0
4      seen_pairs = set()  # To keep track of seen pairs
5      for i in range(len(t)):
6          for j in range(i + 1, len(t)):
7              if t[i] + t[j] == k:
8                  # Check if the pair is not already seen
9                  if (t[i], t[j]) not in seen_pairs and (t[j], t[i]) not in seen_pairs:
10                     count += 1
11                     seen_pairs.add((t[i], t[j]))  # Add the pair to seen pairs
12     return count
13
14 # Get input from the user for the tuple
15 t = tuple(map(int, input().split(',')))
16
17 # Get input from the user for the target sum
18 k = int(input())
19
20 # Call the function and print the result
21 print(count_distinct_pairs(t, k))
22
```

	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

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 %)

```
1  a=input()
2  b=input()
3  c=[]
4  for char in a:
5      if char in b and char not in c:
6          c.append(char)
7  results="".join(c)
8  res=len(c)
9  print(res)
```

	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 **3**

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
1  def find_duplicate(nums):
2      seen=set()
3      for num in nums:
4          if num in seen:
5              return num
6          seen.add(num)
7  nums=list(map(int,input().split()))
8  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 **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 %)

```python
1  def is_binary_string(s):
2      # Create a set of characters in the string
3      char_set = set(s)
4      # Check if the set contains only '0' and '1'
5      return char_set == {'0', '1'}
6
7  # Test the function
8  str1 = str(input())
9
10
11 print("Yes" if is_binary_string(str1) else "No")  # Output: Yes
12
```

	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

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 = "AAAAACCCCCAAAAACCCCCAAAAAGGGTTT"
Output: ["AAAAACCCCC","CCCCCAAAAA"]
```

Example 2:

```
Input: s = "AAAAAAAAAAAAA"
Output: ["AAAAAAAAAA"]
```

For example:

Input	Result
AAAAACCCCCAAAAACCCCCAAAAAGGGTTT	AAAAACCCCC
	CCCCCAAAAA

Answer: (penalty regime: 0 %)

```
 1  s = input()
 2  A = set()
 3  B = set()
 4  for i in range(len(s) - 9):
 5      C = s[i:i + 10]
 6      if C in A:
 7          B.add(C)
 8      else:
 9          A.add(C)
10  for seq in B:
11      print(seq)
```

	Input	Expected	Got	
✓	AAAAACCCCCAAAAACCCCCAAAAAGGGTTT	AAAAACCCCC	AAAAACCCCC	✓
		CCCCCAAAAA	CCCCCAAAAA	

	Input	Expected	Got	
✓	AAAAAAAAAAAA	AAAAAAAAAA	AAAAAAAAAA	✓

Passed all tests! ✓

Correct

Marks for this submission: 1.00/1.00.

◄ Week7_MCQ

Jump to...

Dictionary ►