1. Python – Sort Dictionary key and values List
2. Handling missing keys in Python dictionaries
3. Python dictionary with keys having multiple inputs
4. Print anagrams together in Python using List and Dictionary
5. K’th Non-repeating Character in Python using List Comprehension and OrderedDict
6. Check if binary representations of two numbers are anagram
7. Python Counter to find the size of largest subset of anagram words
8. Python | Remove all duplicates words from a given sentence
9. Python Dictionary to find mirror characters in a string
10. Counting the frequencies in a list using dictionary in Python

Solve

Let's go through each problem step by step.

### 1. Python – Sort Dictionary Key and Values List

To sort a dictionary by keys and then sort each of its corresponding list values:

```python

# Example dictionary

data = {'c': [3, 2, 1], 'a': [3, 1, 2], 'b': [2, 3, 1]}

# Sorting dictionary by keys and then sorting values lists

sorted\_dict = {k: sorted(v) for k, v in sorted(data.items())}

print(sorted\_dict)

```

### 2. Handling Missing Keys in Python Dictionaries

To handle missing keys in a dictionary, use `dict.get()` or `collections.defaultdict`:

```python

# Using dict.get()

my\_dict = {'a': 1, 'b': 2}

print(my\_dict.get('c', 'Default Value'))

# Using defaultdict

from collections import defaultdict

my\_default\_dict = defaultdict(lambda: 'Default Value')

my\_default\_dict['a'] = 1

print(my\_default\_dict['c'])

```

### 3. Python Dictionary with Keys Having Multiple Inputs

If you want a dictionary where each key can store multiple values:

```python

# Using a list of tuples

data = [('a', 1), ('b', 2), ('a', 3)]

multi\_dict = {}

for key, value in data:

multi\_dict.setdefault(key, []).append(value)

print(multi\_dict)

```

### 4. Print Anagrams Together in Python Using List and Dictionary

To group anagrams together:

```python

from collections import defaultdict

words = ['listen', 'silent', 'enlist', 'hello', 'ohell']

anagram\_dict = defaultdict(list)

for word in words:

sorted\_word = ''.join(sorted(word))

anagram\_dict[sorted\_word].append(word)

print(list(anagram\_dict.values()))

```

### 5. K’th Non-repeating Character in Python Using List Comprehension and OrderedDict

To find the K'th non-repeating character:

```python

from collections import OrderedDict

def kth\_non\_repeating(string, k):

char\_order = OrderedDict()

for ch in string:

char\_order[ch] = char\_order.get(ch, 0) + 1

non\_repeating\_chars = [ch for ch, count in char\_order.items() if count == 1]

return non\_repeating\_chars[k - 1] if k <= len(non\_repeating\_chars) else None

print(kth\_non\_repeating('geeksforgeeks', 3))

```

### 6. Check if Binary Representations of Two Numbers Are Anagram

To check if the binary representations of two numbers are anagrams:

```python

from collections import Counter

def binary\_anagram(num1, num2):

bin1 = bin(num1)[2:]

bin2 = bin(num2)[2:]

return Counter(bin1) == Counter(bin2)

print(binary\_anagram(5, 3)) # binary of 5 is '101', binary of 3 is '011'

```

### 7. Python Counter to Find the Size of Largest Subset of Anagram Words

To find the largest subset of anagram words:

```python

from collections import Counter

def largest\_anagram\_subset(words):

anagram\_counter = Counter()

for word in words:

sorted\_word = ''.join(sorted(word))

anagram\_counter[sorted\_word] += 1

return max(anagram\_counter.values())

words = ["listen", "silent", "enlist", "hello", "ohell"]

print(largest\_anagram\_subset(words))

```

### 8. Python | Remove All Duplicate Words from a Given Sentence

To remove duplicate words from a sentence:

```python

def remove\_duplicates(sentence):

words = sentence.split()

seen = set()

result = []

for word in words:

if word.lower() not in seen:

seen.add(word.lower())

result.append(word)

return ' '.join(result)

sentence = "This is a test Test sentence with with duplicates."

print(remove\_duplicates(sentence))

```

### 9. Python Dictionary to Find Mirror Characters in a String

To find mirror characters:

```python

def mirror\_characters(string, mirror\_index):

alphabet = 'abcdefghijklmnopqrstuvwxyz'

mirror = {alphabet[i]: alphabet[25 - i] for i in range(26)}

result = ''.join(mirror[ch] if ch in mirror else ch for ch in string[:mirror\_index + 1])

result += string[mirror\_index + 1:]

return result

string = "hello"

print(mirror\_characters(string, 2)) # Mirror characters up to index 2

```

### 10. Counting the Frequencies in a List Using Dictionary in Python

To count the frequency of each element in a list:

```python

def count\_frequencies(lst):

freq\_dict = {}

for item in lst:

freq\_dict[item] = freq\_dict.get(item, 0) + 1

return freq\_dict

lst = [1, 2, 2, 3, 3, 3]

print(count\_frequencies(lst))

```

These solutions cover the given problems. You can run and test them in your Python environment. Let me know if you need further assistance!