1. **Python – Sort Dictionary key and values List**

```
test_dict = {'gfg': [7, 6, 3],
                        'is': [2, 10, 3],
                        'best': [19, 4]}

# printing original dictionary
print("The original dictionary is : " + str(test_dict))

# Sort Dictionary key and values List
# Using loop + dictionary comprehension
res = dict()
for key in sorted(test_dict):
        res[key] = sorted(test_dict[key])

# printing result
print("The sorted dictionary : " + str(res))
```

2. **Handling missing keys in Python dictionaries**

```
d = { 'a' : 1 , 'b' : 2 }

# trying to output value of absent key
print ("The value associated with 'c' is : ")
print (d['c'])
```

3. **Python dictionary with keys having multiple inputs**

```
import random as rn
# creating an empty dictionary
dict = {}
# Insert first triplet in dictionary
x, y, z = 10, 20, 30
dict[x, y, z] = x + y - z;
# Insert second triplet in dictionary
x, y, z = 5, 2, 4
dict[x, y, z] = x + y - z;
```

```
# print the dictionary
print(dict)
```

4. **Print anagrams together in Python using List and Dictionary**

```
def allAnagram(input):
        dict = {}

        # traverse list of strings
        for strVal in input:

                key = ''.join(sorted(strVal))

                if key in dict.keys():
                        dict[key].append(strVal)
                else:
                        dict[key] = []
                        dict[key].append(strVal)

        # traverse dictionary and concatenate values
        # of keys together
        output = ""
        for key,value in dict.items():
                output = output + ' '.join(value) + ' '

        return output

# Driver function
if __name__ == "__main__":
        input=['cat', 'dog', 'tac', 'god', 'act']
        print (allAnagram(input))
```

5. **K’th Non-repeating Character in Python using List Comprehension and OrderedDict**

```
from collections import OrderedDict

def kthRepeating(input,k):
```

```
        dict=OrderedDict.fromkeys(input,0)

        for ch in input:
                dict[ch]+=1
        nonRepeatDict = [key for (key,value) in dict.items() if value==1]
        if len(nonRepeatDict) < k:
                return 'Less than k non-repeating characters in input.'
        else:
                return nonRepeatDict[k-1]

# Driver function
if __name__ == "__main__":
        input = "geeksforgeeks"
        k = 3
        print (kthRepeating(input, k))
```

6. **Check if binary representations of two numbers are anagram**
   from collections import Counter

```
def checkAnagram(num1,num2):

        bin1 = bin(num1)[2:]
        bin2 = bin(num2)[2:]

        # append zeros in shorter string
        zeros = abs(len(bin1)-len(bin2))
        if (len(bin1)>len(bin2)):
                bin2 = zeros * '0' + bin2
        else:
                bin1 = zeros * '0' + bin1

        # convert binary representations
        # into dictionary
        dict1 = Counter(bin1)
        dict2 = Counter(bin2)
```

```
    # compare both dictionaries
    if dict1 == dict2:
        print('Yes')
    else:
        print('No')

# Driver program
if __name__ == "__main__":
    num1 = 8
    num2 = 4
    checkAnagram(num1,num2)
```

7. **Python Counter to find the size of the largest subset of anagram words**

```
from collections import Counter

def maxAnagramSize(input):

    # split input string separated by space
    input = input.split(" ")

    # sort each string in given list of strings
    for i in range(0,len(input)):
        input[i]=''.join(sorted(input[i]))

    # now create dictionary using counter method
    # which will have strings as key and their
    # frequencies as value
    freqDict = Counter(input)

    # get maximum value of frequency
    print (max(freqDict.values()))

# Driver program
if __name__ == "__main__":
    input = 'ant magenta magnate tan gnamate'
    maxAnagramSize(input)
```

8. **Python | Remove all duplicated words from a given sentence**

```
from collections import Counter

def remov_duplicates(input):

        # split input string separated by space
        input = input.split(" ")

        # now create dictionary using counter method
        # which will have strings as key and their
        # frequencies as value
        UniqW = Counter(input)

        # joins two adjacent elements in iterable way
        s = " ".join(UniqW.keys())
        print (s)

# Driver program
if __name__ == "__main__":
        input = 'Python is great and Java is also great'
        remov_duplicates(input)
```

9. **Python Dictionary to find mirror characters in a string**
10. **Counting the frequencies in a list using dictionary in Python**

```
def CountFrequency(my_list):

        # Creating an empty dictionary

        freq = {}

        for item in my_list:

                if (item in freq):

                        freq[item] += 1

                else:
```

```
                freq[item] = 1
        for key, value in freq.items():
                print ("% d : % d"%(key, value))
# Driver function
if __name__ == "__main__":
        my_list =[1, 1, 1, 5, 5, 3, 1, 3, 3, 1, 4, 4, 4, 2, 2, 2, 2]
        CountFrequency(my_list)
```