1. Python – Sort Dictionary key and values List

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

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

res = dict()
for key in sorted(test_dict):
    res[key] = sorted(test_dict[key])

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

```
The original dictionary is : {'good': [7, 6, 3], 'is': [2, 10, 3], 'best': [19, 4]}
The sorted dictionary : {'best': [4, 19], 'good': [3, 6, 7], 'is': [2, 3, 10]}
```

2. Handling missing keys in Python dictionaries

In [3]:
```
country_code = {'India' : '0091',
                'Australia' : '0025',
                'Nepal' : '00977'}

print(country_code.get('India', 'Not Found'))

print(country_code.get('Japan', 'Not Found'))
```

```
0091
Not Found
```

3. Python dictionary with keys having multiple inputs

In [4]:
```
import random as rn

dict = {}

x, y, z = 10, 20, 30
dict[x, y, z] = x + y - z;

x, y, z = 5, 2, 4
dict[x, y, z] = x + y - z;

print(dict)
```

```
{(10, 20, 30): 0, (5, 2, 4): 3}
```

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

In [5]:
```
def allAnagram(input):

    dict = {}

    for strVal in input:

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

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

    output = ""
    for key,value in dict.items():
        output = output + ' '.join(value) + ' '

    return output

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

```
cat tac act dog god
```

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

In [7]:
```
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]

if __name__ == "__main__":
    input = "goodsforgood"
    k = 3
    print (kthRepeating(input, k))
```

```
r
```

6.Check if binary representations of two numbers are anagram.

In [8]:
```
from collections import Counter

def checkAnagram(num1,num2):

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

    zeros = abs(len(bin1)-len(bin2))
    if (len(bin1)>len(bin2)):
        bin2 = zeros * '0' + bin2
    else:
        bin1 = zeros * '0' + bin1

    dict1 = Counter(bin1)
    dict2 = Counter(bin2)

    if dict1 == dict2:
        print('Yes')
    else:
         print('No')

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

```
Yes
```

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

In [9]:
```
from collections import Counter

def maxAnagramSize(input):

    input = input.split(" ")

    for i in range(0,len(input)):
         input[i]=''.join(sorted(input[i]))

    freqDict = Counter(input)

    print (max(freqDict.values()))

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

```
3
```

8. Python | Remove all duplicates words from a given sentence.

In [11]:
```
from collections import Counter

def remov_duplicates(input):

    input = input.split(" ")

    UniqW = Counter(input)

    s = " ".join(UniqW.keys())
    print (s)

if __name__ == "__main__":
    input = 'good is great and god is also great'
    remov_duplicates(input)
```

```
good is great and god also
```

10. Counting the frequencies in a list using dictionary in Python.

In [19]:
```
def CountFrequency(my_list):

    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))

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)
```

```
 1 :  5
 5 :  2
 3 :  3
 4 :  3
 2 :  4
```

In [ ]:
```

```