1. Python program to sort Python Dictionaries by Keys

In [1]:

```
MLB_team = dict([('Colorado', 'Rockies'),('Boston', 'Red Sox'),('Minnesota', 'Twins'),('Sea
dict(sorted(MLB_team.items()))
```

Out[1]:

```
{'Boston': 'Red Sox',
 'Colorado': 'Rockies',
 'Milwaukee': 'Brewers',
 'Minnesota': 'Twins',
 'Seattle': 'Mariners'}
```

2. Python program to sort Python Dictionaries by Values

In [2]:

```
MLB_team = dict([('Colorado', 'Rockies'),('Boston', 'Red Sox'),('Minnesota', 'Twins'),('Sea
dict(sorted(MLB_team.items(),key=lambda x:x[1]))
```

Out[2]:

```
{'Milwaukee': 'Brewers',
 'Seattle': 'Mariners',
 'Boston': 'Red Sox',
 'Colorado': 'Rockies',
 'Minnesota': 'Twins'}
```

3. Python program to find the sum of all items in a dictionary

In [3]:

```
MLB = dict([(1,100),(2,200),(3,300),(4,400)])
sum(MLB_team.keys())
sum(MLB_team.values())
sum(MLB_team.keys())+sum(MLB_team.values())
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-3-6c4065fef5ea> in <module>
      1 MLB = dict([(1,100),(2,200),(3,300),(4,400)])
----> 2 sum(MLB_team.keys())
      3 sum(MLB_team.values())
      4 sum(MLB_team.keys())+sum(MLB_team.values())

TypeError: unsupported operand type(s) for +: 'int' and 'str'
```

4. Python program to remove a key from a dictionary

In [ ]:

```
MLB_team = dict([('Colorado', 'Rockies'),('Boston', 'Red Sox'),('Minnesota', 'Twins'),('Sea
MLB_team.pop('Colorado')
MLB_team
```

5. Python program to merge two Dictionaries

In [ ]:

```
MLB_team = dict([('Colorado', 'Rockies'),('Boston', 'Red Sox'),('Minnesota', 'Twins'),('Sea
MLB = dict([(1,100),(2,200),(3,300),(4,400)])
# MLB_team.update(MLB)
for i in MLB.keys():
    MLB_team[i]=MLB[i]
MLB_team
```

6. Program to create grade calculator in Python

In [21]:

```python
jack = { "name":"Jack Frost",
         "assignment" : [80, 50, 40, 20],
         "test" : [75, 75],
         "lab" : [78.20, 77.20]}

james = { "name":"James Potter",
          "assignment" : [82, 56, 44, 30],
          "test" : [80, 80],
          "lab" : [67.90, 78.72]}

dylan = { "name" : "Dylan Rhodes",
          "assignment" : [77, 82, 23, 39],
          "test" : [78, 77],
          "lab" : [80, 80]}

jess = { "name" : "Jessica Stone",
         "assignment" : [67, 55, 77, 21],
         "test" : [40, 50],
         "lab" : [69, 44.56]}

tom = { "name" : "Tom Hanks",
        "assignment" : [29, 89, 60, 56],
        "test" : [65, 56],
        "lab" : [50, 40.6]}

def get_avg(name):
    ans={}
    ans['name']=name['name']
    ans['avg_assignment']=0.1*sum(name['assignment'])/len(name['assignment'])
    ans['avg_test']=0.7*sum(name['test'])/len(name['test'])
    ans['avg_lab']=0.2*sum(name['lab'])/len(name['lab'])
    return sum(list(ans.values())[1:])

class_avg=sum([get_avg(jack),get_avg(james),get_avg(dylan),get_avg(jess),get_avg(tom)])/5
print("Average marks of {} is : {} ".format(jack['name'],round(get_avg(jack),2)))
print("Average marks of {} is : {} ".format(james['name'],round(get_avg(james),2)))
print("Average marks of {} is : {} ".format(dylan['name'],round(get_avg(dylan),2)))
print("Average marks of {} is : {} ".format(jess['name'],round(get_avg(jess),2)))
print("Average marks of {} is : {} ".format(tom['name'],round(get_avg(tom),)))
print("Average marks of {} is : {} ".format('class',round(class_avg,2)))
```

```
Average marks of Jack Frost is : 72.79
Average marks of James Potter is : 75.96
Average marks of Dylan Rhodes is : 75.78
Average marks of Jessica Stone is : 48.36
Average marks of Tom Hanks is : 57
Average marks of class is : 66.03
```

7. Print anagrams together in Python using List and Dictionary

In [22]:

```
arr = ['cat', 'dog', 'tac', 'god', 'act','listen','silent']
def anagrams(arr):
    ansd={}
    for i in range(len(arr)-1):
        for j in arr[i+1:]:
            if sorted(arr[i])==sorted(j):
                if arr[i] in ansd.keys():
                    ansd[arr[i]]+=[j]
                else:
                    ansd[arr[i]]=[j]
    return ansd
print(anagrams(arr))
```

```
{'cat': ['tac', 'act'], 'dog': ['god'], 'tac': ['act'], 'listen': ['silen
t']}
```

8. Check if binary representations of two numbers are an anagram

In [ ]:

```
a,b=13,11
if sorted(bin(a)[2:])==sorted(bin(b)[2:]):
    print('binary representations of these two numbers are an anagram')
else:
    print('binary representations of these two numbers are not an anagram')
bin(a),bin(b)
```

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

In [24]:

```
arr = ['cat', 'dog', 'tac', 'god', 'act','listen','silent','cider','cried','dicer','riced',
       'below','bowel','elbow','state','taste','tates','teats','testa','dusty','study','nig
       'garb','grab','bored','orbed','robed','angel','angle','genal','glean']
ansd={}
ansc={}
for i in range(len(arr)-1):
    for j in arr[i+1:]:
        if sorted(arr[i])==sorted(j):
            if arr[i] in ansd.keys():
                ansd[arr[i]]+=[j]
            else:
                ansd[arr[i]]=[j]
for i in ansd.keys():
    ansc[i]=len(ansd[i])
max(ansc,key=ansc.get)
max([len(i) for i in ansd.values()])
ansc
```

Out[24]:

```
{'cat': 2,
 'dog': 1,
 'tac': 1,
 'listen': 1,
 'cider': 3,
 'cried': 2,
 'dicer': 1,
 'art': 2,
 'rat': 1,
 'below': 2,
 'bowel': 1,
 'state': 4,
 'taste': 3,
 'tates': 2,
 'teats': 1,
 'dusty': 1,
 'night': 1,
 'brag': 2,
 'garb': 1,
 'bored': 2,
 'orbed': 1,
 'angel': 3,
 'angle': 2,
 'genal': 1}
```

10. Python Dictionary to find mirror characters in a string

In [26]:

```
l1=[chr(i) for i in range(97,123)]
l2=[chr(i) for i in range(97,123)][::-1]
d=dict(zip(l1,l2))
st1 = 'python'
st2 = ''
for i in st1:
    st2+=d[i]
print(st2)
```

```
kbgslm
```

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

In [ ]:

```
st='In this tutorial, we have seen what are mirror characters and for a given string how to
d={}
for i in st:
    if i in d:
        d[i]+=1
    else:
        d[i]=1
print(d)
```

12. Python program to convert a list of Tuples into Dictionary

In [ ]:

```
l = [("akash", 10), ("gaurav", 12), ("anand", 14), ("suraj", 20), ("akhil", 25), ("ashish",
#1
d=dict(l)
print(d)
#2
d={}
for tp in l:
    d[tp[0]]=tp[1]
print(d)
```

13. Scraping And Finding Ordered Words In A Dictionary using Python

In [ ]:

```
```

14. Create a list of tuples from the given list having a number and its cube in each tuple

In [27]:

```
l=[i for i in range(1,11)]
l3=[i**3 for i in l]
ans=[]
for i in range(10):
        ans.append((l[i],l3[i]))
ans2=list(zip(l,l3))
print(ans,'\n',ans2)
```

```
[(1, 1), (2, 8), (3, 27), (4, 64), (5, 125), (6, 216), (7, 343), (8, 512),
(9, 729), (10, 1000)]
 [(1, 1), (2, 8), (3, 27), (4, 64), (5, 125), (6, 216), (7, 343), (8, 512),
(9, 729), (10, 1000)]
```

15. Sort a list of tuples by the second Item

In [9]:

```
l1=list('abcdefghij')[::-1]
l3=[i**3 for i in range(1,11)]
l=list(zip(l3,l1))
ans=sorted(l,key=lambda x:x[1])
ans
```

Out[9]:

```
[(1000, 'a'),
 (729, 'b'),
 (512, 'c'),
 (343, 'd'),
 (216, 'e'),
 (125, 'f'),
 (64, 'g'),
 (27, 'h'),
 (8, 'i'),
 (1, 'j')]
```

16. Python Program for Insertion Sort

In [4]:

```
def insertionSort(arr):
    for i in range(1, len(arr)):
        key = arr[i]
        j = i-1
        while j >=0 and key < arr[j] :
                arr[j+1] = arr[j]
                j -= 1
        arr[j+1] = key
arr = [12, 11, 13, 5, 6]
insertionSort(arr)
print ("Sorted array is:",arr)
# for i in range(len(arr)):
#     print ("%d" %arr[i])
```

```
Sorted array is: [5, 6, 11, 12, 13]
```

17. Python Program for SelectionSort

In [7]:

```
import sys
A = [64, 25, 12, 22, 11]
for i in range(len(A)):
    min_idx = i
    for j in range(i+1, len(A)):
        if A[min_idx] > A[j]:
            min_idx = j
    A[i], A[min_idx] = A[min_idx], A[i]

print ("Sorted array is: ",A)
# for i in range(len(A)):
#     print("%d" %A[i]),
```

```
Sorted array is:  [11, 12, 22, 25, 64]
```

18. Python Program for Bubble Sort

In [8]:

```
def bubbleSort(arr):
    n = len(arr)
    for i in range(n-1):
        for j in range(0, n-i-1):
            if arr[j] > arr[j + 1] :
                arr[j], arr[j + 1] = arr[j + 1], arr[j]

arr = [64, 34, 25, 12, 22, 11, 90]

bubbleSort(arr)

print ("Sorted array is: ",arr)
# for i in range(len(arr)):
#     print ("% d" % arr[i]),
```

```
Sorted array is:  [11, 12, 22, 25, 34, 64, 90]
```

19. Python Program for Merge Sort

In [12]:

```python
def merge(arr, l, m, r):
    n1 = m - l + 1
    n2 = r - m
    L = [0] * (n1)
    R = [0] * (n2)
    for i in range(0, n1):
        L[i] = arr[l + i]

    for j in range(0, n2):
        R[j] = arr[m + 1 + j]
    i = 0
    j = 0
    k = l

    while i < n1 and j < n2:
        if L[i] <= R[j]:
            arr[k] = L[i]
            i += 1
        else:
            arr[k] = R[j]
            j += 1
        k += 1
    while i < n1:
        arr[k] = L[i]
        i += 1
        k += 1
    while j < n2:
        arr[k] = R[j]
        j += 1
        k += 1
def mergeSort(arr, l, r):
    if l < r:
        m = l+(r-l)//2
        mergeSort(arr, l, m)
        mergeSort(arr, m+1, r)
        merge(arr, l, m, r)
arr = [12, 11, 13, 5, 6, 7]
n = len(arr)
print("Given array is :",arr)
# for i in range(n):
#     print("%d" % arr[i]),

mergeSort(arr, 0, n-1)
print("\n\nSorted array is :", arr)
# for i in range(n):
#     print("%d" % arr[i]),
```

```
Given array is : [12, 11, 13, 5, 6, 7]


Sorted array is : [5, 6, 7, 11, 12, 13]
```

20. Python Program for QuickSortSort

In [13]:

```python
def partition(arr, low, high):
    i = (low-1)
    pivot = arr[high]
    for j in range(low, high):
        if arr[j] <= pivot:
            i = i+1
            arr[i], arr[j] = arr[j], arr[i]
    arr[i+1], arr[high] = arr[high], arr[i+1]
    return (i+1)

def quickSort(arr, low, high):
    if len(arr) == 1:
        return arr
    if low < high:
        pi = partition(arr, low, high)
        quickSort(arr, low, pi-1)
        quickSort(arr, pi+1, high)

arr = [10, 7, 8, 9, 1, 5]
n = len(arr)
quickSort(arr, 0, n-1)
print("Sorted array is:",arr)
# for i in range(n):
#     print("%d" % arr[i]),
```

```
Sorted array is: [1, 5, 7, 8, 9, 10]
```