1. Python program to sort Python Dictionaries by Keys

In [4]:
```
def sorting():
    a={"abc":123,"efg":456,"xyz":789,"bcd":987,"hij":654}
    b=a
    dict1={}
    for i in sorted(a):
        dict1[i]=a[i]
    return dict1

sorting()
```

Out[4]:
```
{'abc': 123, 'bcd': 987, 'efg': 456, 'hij': 654, 'xyz': 789}
```

2. Python program to sort Python Dictionaries by Values

In [5]:
```
def sorting():
    a={"abc":123,"efg":456,"xyz":789,"bcd":987,"hij":654}
    dict1={}
    for i in sorted(a.values()):
        for j in a:
            if i==a[j]:
                dict1[j]=i
    return dict1

sorting()
```

Out[5]:
```
{'abc': 123, 'efg': 456, 'hij': 654, 'xyz': 789, 'bcd': 987}
```

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

In [12]:
```
def addition():
    a={1:1,2:4,3:9,4:16,5:25,6:36,7:49,8:64,9:81,10:100}
    sum_keys=0
    sum_values=0
    for i in a:
        sum_keys+=i
    print("Addition of keys are =",sum_keys)
    for j in a.values():
        sum_values+=j
    print("Addition of values are =",sum_values)
    print("Sum of all items in a dictionary are=",sum_keys+sum_values)

addition()
```

```
addition of keys are = 55
addition of values are = 385
Sum of all items in a dictionary are=  440
```

4. Python program to remove a key from a dictionary

In [4]:
```
def remove():
    a={"abc":123,"efg":456,"xyz":789,"bcd":987,"hij":654}
    n=input("Enter a kye which you want to remove= ")
    del a[n]
    return a

remove()
```

```
Enter a kye which you want to remove= xyz
{'abc': 123, 'efg': 456, 'bcd': 987, 'hij': 654}
```

5. Python program to merge two Dictionaries

In [9]:
```
def merge():
    a={"abc":123,"efg":456,"xyz":789,"bcd":987}
    b={1:1,2:4,3:9,4:16,5:25,6:36,7:49,8:64,9:81}
    a.update(b)
    print(a,end="")

merge()
```

```
{'abc': 123, 'efg': 456, 'xyz': 789, 'bcd': 987, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81}
```

6. Program to create grade calculator in Python

In [10]:
```
def grade_calculator():
    m=int(input())
    if m>=90:
        print("'congratulation'- A Grade")
    elif m<90 and m>80:
        print("B Grade")
    elif m<80 and m>70:
        print("C Grade")
    elif m<70 and m>60:
        print("D Grade")
    else:
        print("E Grade")


grade_calculator()
```

```
90
'congratulation'- A Grade
```

7. Print anagrams together in Python using List and Dictionary

In [14]:
```
a=["eat","tea","tan","ate","nat","bat"]
dict1={}
for i in a:
    s=''.join(sorted(i))
    if s in dict1:
        dict1[s].append(i)
    else:
        dict1[s]=[i]
print(list(dict1.values()))
```

```
[['eat', 'tea', 'ate'], ['tan', 'nat'], ['bat']]
```

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

In [16]:
```
def anagram():
    n=int(input("Enter first  number= "))
    m=int(input("Enter second  number= "))
    a=bin(n)[2:]
    b=bin(m)[2:]

    if a.count("0")==b.count("0"):
        if a.count("1")==b.count("1"):
            print(f"a={a},b={b}")
            print(f"{N} and {M} are anagram number")
        else:
            print("Number is not anagram")

    else:
        print(f"{N} and {M} are not anagram ")


anagram()
```

```
Enter first  number= 9
Enter second  number= 12
a=1001,b=1100
9 and 12 are anagram number
```

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

In [20]:
```
a=["eat","tea","tan","ate","nat","bat"]
dict1={}
for i in a:
    sort=''.join(sorted(i))
    if sort in dict1:
        dict1[sort].append(i)
    else:
        dict1[sort]=[i]
print(list(dict1.values()))
print("Largest subset of anagram words is= ",max(dict1.values(),key=len))
```

```
[['eat', 'tea', 'ate'], ['tan', 'nat'], ['bat']]
Largest subset of anagram words is=  ['eat', 'tea', 'ate']
```

10. Python Dictionary to find mirror characters in a string

In [12]:
```
a="abcdefghijklmnopqrstuvwxyz"
b="zyxwvutsrqponmlkjihgfedcba"                    # tl will gives dict like this
dict1=dict(zip(a,b))                              #{'a': 'z', 'b': 'y', 'c': 'x',....,'y': 'b', 'z': 'a'}

m=input("Enter a string= ")
n=int(input("Enter a number from which you want to mirrorn= "))
x=m[:n-1]
y=m[n-1:]
#x=""

z=''
for i in y:
    z+=dict1[i]
print("your string with mirror character is= ",x+z)
```

```
Enter a string= anagram
Enter a number from which you want to mirror= 4
your string with mirror character is=  anaziz
```

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

In [13]:
```
a=[1,2,11,3,1,5,2,3,6,4,5,4,8,6,3,2,1,2,5]
dict1={}
for i in a:
    dict1[i]=a.count(i)
print("Count of frequency is= ",dict1)
```

```
Count of frequency is=  {1: 3, 2: 4, 11: 1, 3: 3, 5: 3, 6: 2, 4: 2, 8: 1}
```

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

In [15]:
```
a=[(1,1),(2,4),(3,9),(4,16),(5,25),(6,36)]
dict1={}
for i in a:
    dict1.update({i[0]:i[1]})
print("your list of tuple converted into dictionary= ",dict1)
```

```
your list of tuple converted into dictionary=  {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36}
```

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

In [15]:
```
a="abc is beautifull and xyz is honest"
b=a.split()
for i in b:
    for n in range(len(i)-1):
        if ord(i[n])>ord(i[n+1]):
            break
    else:
        print(f"'{i}'\" is ordered word in n{a}")
```

```
"abc" is ordered word in nabc is beautifull and xyz is honest
"is" is ordered word in nabc is beautifull and xyz is honest
"xyz" is ordered word in nabc is beautifull and xyz is honest
"is" is ordered word in nabc is beautifull and xyz is honest
```

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

In [21]:
```
a=[1,2,3,4,5,6]
d=[]
for i in a:
    d.append((i,i**3))
print(d)
```

```
[(1, 1), (2, 8), (3, 27), (4, 64), (5, 125), (6, 216)]
```

15. Sort a list of tuples by the second Item

In [17]:
```
a=[(6,36),(2,4),(1,1),(4,16),(5,25),(3,9)]
for i in range(len(a)-1):
    for j in range(len(a)-i):
        if a[j][1]>a[j+1][1]:
            a[j],a[j+1]=a[j+1],a[j]

print("your list of tuple after sorting= ",a)
```

```
your list of tuple after sorting=  [(1, 1), (2, 4), (3, 9), (4, 16), (5, 25), (6, 36)]
```

16. Python Program for Insertion Sort

In [6]:
```
a=[4,1,6,9,2,7,3]
for i in range(len(a)):
    insertion_value=a[i]   #i=1,
    j=i-1;
    while j>=0 and insertion_value < a[j]:#a[0] 1<4
        a[j+1]=a[j]
        j=j-1
    else:
        a[j+1]=insertion_value
        print("Sorted list= ",a)
```

```
Sorted list=  [4, 1, 6, 9, 2, 7, 3]
Sorted list=  [1, 4, 6, 9, 2, 7, 3]
Sorted list=  [1, 4, 6, 9, 2, 7, 3]
Sorted list=  [1, 2, 4, 6, 9, 7, 3]
Sorted list=  [1, 2, 4, 6, 9, 7, 3]
Sorted list=  [1, 2, 4, 6, 7, 9, 3]
Sorted list=  [1, 2, 3, 4, 6, 7, 9]
```

17. Python Program for SelectionSort

In [12]:
```
lis=[5,9,4,3,6,9,15,2]
for i in range(len(lis)):
    min_value=i
    for j in range(i,len(lis)):
        if lis[j]<lis[min_value]:
            min_value=j

    x=lis[i]
    lis[i]=lis[min_value]
    lis[min_value]=x

    print(lis)
```

```
[2, 9, 4, 3, 6, 9, 15, 5]
[2, 3, 4, 9, 6, 9, 15, 5]
[2, 3, 4, 9, 6, 9, 15, 5]
[2, 3, 4, 5, 6, 9, 15, 9]
[2, 3, 4, 5, 6, 9, 15, 8]
[2, 3, 4, 5, 6, 8, 15, 9]
[2, 3, 4, 5, 6, 8, 9, 15]
[2, 3, 4, 5, 6, 8, 9, 15]
```

18. Python Program for Bubble Sort

In [26]:
```
a=[11,2,55,5,4,9,6,1,77]
for i in range(len(a)):
    print(i)
    for j in range(len(a)-1):
#         print(j)
        if a[j]>a[j+1]:
            a[j],a[j+1]=a[j+1],a[j]
            print("List after updating= ",a)

print(a)
```

```
0
List after updating=  [2, 11, 55, 3, 4, 9, 6, 1, 77]
List after updating=  [2, 11, 3, 55, 4, 9, 6, 1, 77]
List after updating=  [2, 11, 3, 4, 55, 9, 6, 1, 77]
List after updating=  [2, 11, 3, 4, 9, 55, 6, 1, 77]
List after updating=  [2, 11, 3, 4, 9, 6, 55, 1, 77]
List after updating=  [2, 11, 3, 4, 9, 6, 1, 55, 77]
1
List after updating=  [2, 3, 11, 4, 9, 6, 1, 55, 77]
List after updating=  [2, 3, 4, 11, 9, 6, 1, 55, 77]
List after updating=  [2, 3, 4, 9, 11, 6, 1, 55, 77]
List after updating=  [2, 3, 4, 9, 6, 11, 1, 55, 77]
List after updating=  [2, 3, 4, 9, 6, 1, 11, 55, 77]
2
List after updating=  [2, 3, 4, 6, 9, 1, 11, 55, 77]
List after updating=  [2, 3, 4, 6, 1, 9, 11, 55, 77]
List after updating=  [2, 3, 4, 6, 9, 11, 55, 77]
3
List after updating=  [2, 3, 4, 1, 6, 9, 11, 55, 77]
4
List after updating=  [2, 3, 1, 4, 6, 9, 11, 55, 77]
5
List after updating=  [2, 3, 4, 6, 9, 11, 55, 77]
6
List after updating=  [1, 2, 3, 4, 6, 9, 11, 55, 77]
7
8
[1, 2, 3, 4, 6, 9, 11, 55, 77]
```

19. Python Program for Merge Sort

In [35]:
```
lis1=[5,2,4,6,6,3,2,12]
def merge(lis1):
#     lis1=[5,2,4,6,6,3,2,12]
    if len(lis1)>=1:
        return lis1

    m=len(lis1)//2
    left=lis1[:m]
    right=lis1[m:]

    print("left=",left,"  right= {right}")

    left=merge(left)
    right=merge(right)
    return merged_list(left,right)

def merged_list(a,b):
    lis2=[]
    i=j=0
    while i<len(a) and j<len(b):
        if a[i]<=b[j]:
            lis2.append(a[i])
            i+=1
        else:
            lis2.append(b[j])
            j+=1


    while i<len(a):
        lis2.append(a[i])
        i+=1
    while j<len(b):
        lis2.append(b[j])
        j+=1
    print(lis2)
    return lis2


result=merge(lis1)
print("sorted list will be= ",result)
```

```
left= [5, 2, 4, 6]  right= {6, 3, 2, 12}
left= [5, 2]  right= {4, 6}
left= [5]  right= [2]
[2, 5]
left= [4]  right= [6]
[4, 6]
[2, 4, 5, 6]
left= [6, 3]  right= [2, 12]
left= [6]  right= [3]
[3, 6]
left= [2]  right= [12]
[2, 12]
[2, 3, 6, 12]
[2, 2, 3, 4, 5, 6, 6, 12]
sorted list will be=  [2, 2, 3, 4, 5, 6, 6, 12]
```

20. Python Program for QuickSortSort

In [77]:
```
def quicksort(lis):
    flag=0
    start=1
    end=len(lis)-1

    while start<=len(lis) and end>1 and start<=end:
        if lis[start]<lis[flag]:
            start+=1
        elif lis[end]>lis[flag]:
            end-=1
        else:
            lis[start],lis[end]=lis[end],lis[start]

    if start>end:
        lis[flag],lis[end]=lis[end],lis[flag]
    print(f"list after Quick sorting= ",lis)


lis=[15,9,12,45,7]
quicksort(lis)          # This is not the complect ans. currently i don't know the the logic
```

```
list after Quick sorting=  [7, 9, 12, 15, 45]
```

In [ ]:
```

```