In [1]:
```
#31.create a list of tuple from given list having number and its cube in each tuple
def cubeoflist(li):
    result=[(num, num**3) for num in li]
    return result
li = [3, 4, 1, 2]
print(cubeoflist(li))
```

```
[(3, 27), (4, 64), (1, 1), (2, 8)]
```

In [3]:
```
#32.python|sort python dictionaries by key or value
mydict={'ravi':10,'rajesh':9,'sanjeev':15,'yash':2,'suraj':32}
mykeys=list(mydict.keys())
mykeys.sort()
sorted_dict={i:mydict[i] for i in mykeys}
print(sorted_dict)
```

```
{'rajesh': 9, 'ravi': 10, 'sanjeev': 15, 'suraj': 32, 'yash': 2}
```

In [9]:
```
#33.python dictionary with keys having multiple value
dic = {}

a,b,c= 5, 3, 10

p,q,r= 12, 6, 9
dic["x-y+z"] = [a-b+c,p-q+r]
print(dic)
```

```
{'x-y+z': [12, 15]}
```

In [10]:
```
#34.python program to find
dic={ 'x':455, 'y':223, 'z':300, 'p':908 }

print("Dictionary: ", dic)

#using sum() and values()
print("sum: ",sum(dic.values()))
```

```
Dictionary:  {'x': 455, 'y': 223, 'z': 300, 'p': 908}
sum:  1886
```

In [15]:
```
#35.python program to find the size of dictionary
import sys
dic1={"A":1,"B":2,"C":3}
dic2={"Geek1":"Raju","Geek2":"Nikhil","Geek3":"Deepanshu"}
dic3={1:"Lion",2:"Tiger",3:"Fox",4:"Wolf"}
print("Size of dic1:"+str(sys.getsizeof(dic1))+"bytes")
print("Size of dic2:"+str(sys.getsizeof(dic2))+"bytes")
print("Size of dic3:"+str(sys.getsizeof(dic3))+"bytes")
```

```
Size of dic1:232bytes
Size of dic2:232bytes
Size of dic3:232bytes
```

In [16]:
```
#36.find the size of the set in python
import sys
set1={"A",1,"B",2,"C",3}
set2={"Geek1","Raju","Geek2","Nikhil","Geek3","Deepanshu"}
set3={1,"Lion"),(2,"Tiger"),(3,"Fox"),(4,"Wolf"))
print("Size of dic1:"+str(sys.getsizeof(set1))+"bytes")
print("Size of dic2:"+str(sys.getsizeof(set2))+"bytes")
print("Size of dic3:"+str(sys.getsizeof(set3))+"bytes")
```

```
Size of dic1:472bytes
Size of dic2:472bytes
Size of dic3:216bytes
```

In [22]:
```
#37.iterate over a set in python
test_set=set("geEks")
for val in test_set:
    print(val)
```

```
g
e
E
s
k
```

In [28]:
```
#38.python maximum and minimum in list
def MAX(sets):
    return (max(sets))
sets = set([8, 16, 24, 1, 25, 3, 10, 65, 55])
print("the maximum element in the set:",MAX(sets))

def MIN(sets):
    return(min(sets))
print("the minimum element in the set:",MIN(sets))
```

```
the maximum element in the set: 65
the minimum element in the set: 1
```

In [31]:
```
#39.python remove items from the set
languages={'puthon','java','engish','C','c++','tami','hindi'}
languages.remove('C')
print(languages)
```

```
{'hindi', 'tami', 'java', 'c++', 'engish', 'puthon'}
```

In [33]:
```
#40.python check if two lists have atleast one element in common
def common_data(list1,list2):
    result=False
    for x in list1:
        for y in list2:
            if x==y:
                result=True
                return result
    return result
a=[1,2,3,4,5]
b=[5,6,7,8,9]
print(common_data(a,b))
a=[1,2,3,4,5]
b=[6,7,8,9]
print(common_data(a,b))
```

```
True
False
```

In [34]:
```
#41.python assigning subsequent rows to matrix first row elements
test_list=[[5,8,9],[2,0,9],[5,4,2],[2,3,9]]
print("the original list:"+str(test_list))
res={test_list[0][ele]:test_list[ele+1] for ele in range(len(test_list)-1)}
print("the assigned matrix:"+str(res))
```

```
the original list:[[5, 8, 9], [2, 0, 9], [5, 4, 2], [2, 3, 9]]
the assigned matrix:{5: [2, 0, 9], 8: [5, 4, 2], 9: [2, 3, 9]}
```

In [43]:
```
#42.adding and subtracting matrix in python
import numpy as np
A = np.array([[1,2], [3,4]])
B = np.array([[4,5], [6,7]])
print("Printing elements of first matrix")
print(A)
print("Printing elements of second matrix")
print(B)
print("Addition of two matrix")
print(np.add(A,B))
print("subtraction of two matrix")
print(np.subtract(A,B))
```

```
Printing elements of first matrix
[[1 2]
 [3 4]]
Printing elements of second matrix
[[4 5]
 [6 7]]
Addition of two matrix
[[ 5  7]
 [ 9 11]]
subtraction of two matrix
[[-3 -3]
 [-3 -3]]
```

In [45]:
```
#43puthon-.group similar elements into matrix
from itertools import groupby
test_list=[1,3,5,1,3,2,5,4,2]
print("the original list:"+str(test_list))
res=[list(val) for key,val in groupby(sorted(test_list))]
print("Matrix after grouping:"+str(res))
```

```
the original list:[1, 3, 5, 1, 3, 2, 5, 4, 2]
Matrix after grouping:[[1, 1], [2, 2], [3, 3], [4], [5, 5]]
```

In [1]:
```
# 44.python-row wise element addition in tuple matrix
test_list=[[('Gfg',3), ('is',3)], [('best',1)], [('for',5), ('geeks',1)]]
print("The original list is:"+ str(test_list))
cus_eles=[6,7,8]
res=[[sub+(cus_eles[idx],) for sub in val] for idx, val in enumerate(test_list)]
print("The matrix after row elements addition :"+str(res))
```

```
The original list is:[[('Gfg', 3), ('is', 3)], [('best', 1)], [('for', 5), ('geeks', 1)]]
The matrix after row elements addition :[[('Gfg', 3, 6), ('is', 3, 6)], [('best', 1, 7)], [('for', 5, 8), ('geeks', 1, 8)]]
```

In [18]:
```
#45.create an n x n matrix,where all the sub matrix has the sum of opposite corner elements as even

import itertools


def sub_mat_even(n):

    temp = itertools.count(1)



    l = [[next(temp)for i in range(n)]for i in range(n)]
    if n%2 == 0:
        for i in range(0,len(l)):
            if i%2 == 1:
                l[i][:] = l[i][::-1]


    for i in range(n):
        for j in range(n):
            print(l[i][j],end=" ")
        print()

n = 4
sub_mat_even(n)
```

```
1 2 3 4
8 7 6 5
9 10 11 12
16 15 14 13
```

In [15]:
```
#46)How to get list of parameters name from a function in Python?

def fun(a, b):
    return a**b


import inspect


print(inspect.signature(fun))
```

```
(a, b)
```

In [20]:
```
#47)How to Print Multiple Arguments in Python?
def GFG(name, num="25"):
    print("Hello from", name + ', ' + num)



GFG("gfg")
GFG("gfg","26")
```

```
Hello from gfg, 25
Hello from gfg, 26
```

In [5]:
```
#48)Python program to find the power of a number using recursion
def power(N, P):
    if P == 0:
        return 1
    return (N*power(N, P-1))



if __name__ == '__main__':
    N = 5
    P = 2

    print(power(N, P))
```

```
25
```

In [16]:
```
#49)Sorting objects of user defined class in Python

class GFG:
    def __init__(self, a, b):
        self.a = a
        self.b = b

    def __repr__(self):
        return str((self.a, self.b))

gfg = [GFG("geeks", 1),
    GFG("computer", 3),
    GFG("for", 2),
    GFG("geeks", 4),
    GFG("science", 3)]

print(sorted(gfg, key=lambda x: x.b))
```

```
[('geeks', 1), ('for', 2), ('computer', 3), ('science', 3), ('geeks', 4)]
```

In [19]:
```
#50)Functions that accept variable length key value pair as arguments
def printKwargs(**kwargs):
    print(kwargs)

if __name__ == "__main__":
    printKwargs(Argument_1='gfg', Argument_2='GFG')
```

```
{'Argument_1': 'gfg', 'Argument_2': 'GFG'}
```