1. sort the list without inbuilt method

```
k = [13,6,7,89]
for i in range(len(k)):
     for j in range(i+1,len(k)):
         if(k[i]>k[j]):
            k[i],k[j] = k[j],k[i]
print(k)
```

output:

```
[6,7,13,89]
```

2.using bubble sort (sum of two lowest numbers)

```
def raja(nums):
    for i in range(len(nums)):
        for j in range(i+1,len(nums)):
            if(nums[i]>nums[j]):
                nums[i],nums[j] = nums[j],nums[j]
    return nums[0]+nums[1]
nums = [-14, 15, -10, -11, -12, -13, 16, 17, 18, 19, 20]
print(raja(nums))
```

output:

```
-27
```

3.Two numbers sum equal to target sum.

```
def prob(d,sum1):
    for A in range(len(d)):
        for B in range(A+1,len(d)):
            if d[A]+d[B] == sum1 :
                 print(d[A],d[B])
d = [13,10,23,15,34,17,12]
sum1 = 27
prob(d,sum1)
```

4.

```
s = "ABCD"
for i in range(len(s)):
    for j in range(i+1,len(s)+1):
        print(s[i:j])
```

output:

```
A
AB
ABC
ABCD
B
BC
BCD
C
CD
D
```

8.longest palidrom from given string:

```
def long_pali(s):
    max_len = 0
    max_str = ""
    for i in range(len(s)):
        for j in range(i+1,len(s)+1):
            word = s[i:j]
            if word == word[::-1]:
                if max_len<len(word):
                    max_len = len(word)
                    max_str = word
    return max_str
print(long_pali("llababaioop"))

#longest non repeating string
def long_non_repeating(s):
    max_len = 0
    max_str = ""
    for i in range(len(s)):
        word= s[i]
        for j in range(i+1,len(s)):
            if s[j] not in word:
                word += s[j]
                if max_len<len(word):
                    max_len = len(word)
                    max_str = word
            else:
                break
    return max_str,len(max_str)
print(long_non_repeating("ABDEFGABEFKABDEFG"))

7.move the zeros to end of the list
k =[0,41,3,15,0]
#o/p:[41,3,15,0,0]
k1= []
k2=[]
for i in k:
    if(i>0):
        k1.append(i)
    else:
        k2.append(i)
print(k2+k1)


8.find second max element  from list
k = [-14, 15, -10, -11, -12, -13, 16, 17, 18, 19, 20]
max = k[0]
secondmax = k[1]
for i in range(len(k)):
    if(max<k[i]):
        secondmax = max
```

```
        max = k[i]
    elif(secondmax<k[i]):
        secondmax = k[i]
    else:
        pass
print(secondmax)

10.string conv to list
k = "100:200:300"
k1 =k.split(":")
print(list(map(int,k1)))
output:
[100,200,300]

11. how to filter the even and odd number using list compression
even = []
odd = []
k1 = [even.append(i) if i%2==0 else odd.append(i) for i in range(10)]
print(even)
print(odd)

12. remove duplicates from list
k1 = [1,1,2,3,4,5,6,7]
k = []
obj = [k.append(i) for i in k1 if i not in k]
print(k)

#2ndmethod
for i in k1:
        if i not in k:
           k.append(i)
print(k)

#3rd method
print(list(set(k1))

14.Python3 code to program to find occurrence
to each character in given string
inp_str = "Rajasekharreddy"
freq = {}
for ele in inp_str:
    if ele in freq:
        freq[ele] = freq[ele]+1
    else:
        freq[ele] = 1
print(freq)

#2nd method:
k ="viratkohili"
d ={}
```

```
for i in k:
    d.update({i:k.count(i)})
print(d)


from collections import Counter
d = "viratkohili"
print(Counter(d))

15.list compression using find cubes of even and squares of odd
print([i**3 if i%2 == 0 else i**2 for i in range(10)])

-------------------------------------------------------------------------------
l = [1,2,3,4,5,6,7,8,9,10]
print([l[i:i+2] for i in range(0,len(l),2)])
[[1, 2], [3, 4], [5, 6], [7, 8], [9, 10]]
-------------------------------------------------------------------------------
16.sorted the nested list
k =[[1, 2, 3], [2, 4, 5], [1, 1, 1]]
print([i for i in sorted(k)])

17. sort the second element of inner tuple.
k1 = [(8,2),(7,4),(5,2)]
k1.sort(key = lambda x:x[1])
print(k1)
o/p:[(8, 2), (5, 2), (7, 4)]
or
l = sorted(k1, key = lambda x:x[1])
print(l)

18.sort the values of dict
k1 = {"ab":23,"ba":13,"ac":90}
print({k:v for k,v in sorted(k1.items(),key = lambda x:x[1])})
o/p:{'ba': 13, 'ab': 23, 'ac': 90}

similarly:
sort the key by using x[0]

19.sort the keys values of dict
k1 = {"c":[3],"b":[13,4],"ac":[3,1]}
print({k:sorted(v) for k,v in sorted(k1.items(),key = lambda x:x[1])})
o/p:{'c': [3], 'ac': [1, 3], 'b': [4, 13]}

20. “aaaabbbccaabbd" converted into 4a3b2c2a2b1d
s = "aaaabbbccaabbd"
c = 1 #initially taken count = 1
previous = s[0]
c = 1 # represents index
output = ""
for i in range(1,len(s)):
```

```
    if s[i] ==  previous :
        c = c+1
    else:
        output = output+str(c)+previous #o/p = ""+4+a=4a
        previous = s[i] #b
        c = 0
        c = c+1
    if i == len(s)-1:#printed the last element(like d)
        output = output+str(c)+previous
print(output)
#0/p 4a3c2c2a2b1d
```

21. "a4b3c2d1" converted into aaaabbbccd

```
s = "a4b3c2d1"
output =''
for i in s:
    if i.isalpha():
        x = i
    else:
        d = int(i)
        output = output+x*d
print(output)
```

o/p:aaaabbbccd

22. "4a2b2c" converted into “aaaabbcc”

```
s = "4a2b2c"
output =''
for i in s:
    if i.isdigit():
        x = int(i)
    else:
        output = output+i*x
print(output)
```

o/p:aaaabbcc

23.find the sum of diagonal matrix:

```
k = [[1,2,3],[4,5,6],[7,8,9]]
sum = 0
for i in range(len(k)):
    sum = sum+k[i][i]
    #   = 0+k[0][0]+-----
print(sum)
```

output:

15

24.prite charcters at odd position and Even position for thr string:

```
s = input("Enter Some String:")
print(s[::2])
print(s[1::2])
```

25.write a program for only revered odd positions words from given string

```
s = "virat kohili is a good batsman"
#o/p: virat ilihok is a good namstab
s = "virat kohili is a good batsman"
s1 = s.split()
s2=[]
for i in range(0,len(s1)):
    if i%2 == 0:
        s2.append(s1[i])
    else:
        s2.append(s1[i][::-1])
print(" ".join(s2))
```

26.Find The missing elements in a list:

```
L=[1,2,3,4,5,7,8,9,10,13]
missing=[]
for i in range(L[0],L[-1]):#1,13
    if i not in L:
        missing.append(i)
print(missing)
```

output;

```
[6,11,12]
```

27.reverse the list list with out using built and as well as [::-1]

```
k = [1,2,34,7,6]
l = []
for i in range(1,len(k)+1):
    l.append(k[-i])
print(l)
```

28.find the list monotonic or not

```
l=list(map(int,input(“enter:”).split()))
c,d=0,0
for i in range(len(l)-1):
    if l[i]<l[i+1]:
        c=c+1
    elif l[i]>l[i+1]:
        d=d+1
if c==len(l)-1:#count increments equal to length
    print("inc mono")
elif d==len(l)-1:
    print("dec mono")
else:
    print("not monotonic")
```

29.ip address matching

```
import re
```

```
s = input("enter ip address:")
if re.fullmatch("\d{1,3}.\d{1,3}.\d{1,3}.\d{1,3}",s):
    print("match is availble")
else:
    print("match is not avilble")
```

30.matching phone number

```
import re
s = input("enter phone number:")
if re.fullmatch("\d{10}",s):
    print("match is availble")
else:
    print("match is not avilble")
```

```
#note---------------
[+][9][1]\d{10} #matching phone number starting with +91
[+][9][1]\s\d{10} #matching phone number after +91 with space

Adhar card matching:\d{4}\s\d{4}\s\d{4}#4556 7888 7094
PAN CARD MATCHING: [A-Z]{5}\d{4}[A-Z]{1}#FEEPK5533M
------------------------------------------------------------
```

31.gmail matching

```
import re
s = input("enter a gmail:")
if re.fullmatch("[\w._-]+[@][a-zA-Z]{1,5}[.][a-zA-Z]{1}",s):
    print("match is availble")
else:
    print("match is not avilble")
```

32.fibnacci WITHOUT recursive function

```
first=0
second=1
n = int(input("enter number:"))
for i in range(n):
    print(first)
    temp = first
    first = second
    second = second+temp
```

32.fibanacci series using recursion functiom

```
def fib(n):
    if(n<0):
        raise ValueError("must be positive")
    elif(n==0):
        return 0
    elif(n==1):
        return 1
    else:
        return fib(n-1)+fib(n-2)
```

```
n = int(input("enter number:"))
for i in range(n):
    print(fib(i))
```

34.factotial usung recursive function

```
def fact(n):
    if(n==0 or n==1):
        return 1
    else:
        return n*fact(n-1)
print(fact(3))
```

35.factotial without usung recursive function

```
n = int(input("enter a number:"))
r = 1
for i in range(1,n+1):
    r = r*i
print("factorial of {}".format(r))
```

36.Python program to check if the number is an Armstrong number or not

```
num = int(input("Enter a number: "))
sum = 0
temp = num
while temp > 0:
   digit = temp % 10
   sum += digit ** 3
   temp //= 10
if num == sum:
   print(num,"is an Armstrong number")
else:
   print(num,"is not an Armstrong number")
```

```
logic:153%10|  153 --3 is removed
10)153(15       10)15(1
     150             10
------           =====
remaining 3      reamiming 5
0+3**3+          27+125+1
```

```
finally:
15 5 is remove
```

38.prime numbers:

```
num = int(input("enter number:"))
if num >1:
    for i in range(2,num):
        if num%i == 0:
            print("not prime")
```

```
            break
    else:
        print("prime")
# for range
for num in range(20):
    if num>1:
        for i in range(2,num):
            if num%i == 0:
                break
        else:
            print(num)
or:
for n in range(1,100,2):
    if n>1:
        for i in range(2,n):
            if n%i == 0:
                break
        else:
            print(n)
```

40.patterns:

```
for i in range(1,7):
    print("*"*i)
```

o/p:

```
*
**
***
****
*****
```

```
for i in range(1,7):
    print((str(i)+" ")*i)
```

o/p:

```
1
2 2
3 3 3
4 4 4 4
5 5 5 5 5
6 6 6 6 6 6
------------------------------
```

```
for i in range(1,7):
    print((chr(64+i)+" ")*i)
```

```
A
B B
```

```
C C C
D D D D
E E E E E
F F F F F F

----------------------------------------


n =int(input("enter number:"))
for i in range(1,n+1):
    print("*"*n)

****
****
****
****

-----------------------

n =int(input("enter number:"))
for i in range(1,n+1):
    print((str(i)+" ")*n)

1111
2222
3333
4444

------------------------------------------------------------
n = int(input("enter---"))
for i in range(1,n+1):
    print(' '*(n-i)+("*"+" ")*i)
      *
     * *
    * * *
   * * * *
  * * * * *
 * * * * * *

NOTE: print(' ' * (n - i) + (str(i) + " ") * i) #for numbers printing
      print(' ' * (n - i) + (chr(64+i) + " ") * i) #for alphabital printing

------------------------------------------------------------------------------
n = int(input("enter no of rows:"))
for i in range(1,n+1):
    print(" "*(n-i)+"*"*i)



      *
```

```
     **
    ***
   ****
  *****
 ******
*******
```

```
n = int(input("enter no of rows:"))
for i in range(1,n+1):
    for j in range(1,i+1):
        print(j,end = " ")
    print("\r")
```

```
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5
1 2 3 4 5 6
1 2 3 4 5 6 7
```

```
n = int(input("enter no of rows:"))
for i in range(1,n+1):
    print(" "*(n-i),end = " ")#no of spaces
    for j in range(1,i+1):
        print((str(j)+" "),end = " ")
    print("\r")
```

```
       1
      1  2
     1  2  3
    1  2  3  4
   1  2  3  4  5
  1  2  3  4  5  6
 1  2  3  4  5  6  7
```

```
n = int(input("enter no of rows:"))
for i in range(1,n+1):
    print(" "*(n-i),end = " ")#no of spaces
    for j in range(1,i+1):
        print((chr(64+j)+" "),end = " ")
    print("\r")
```

```
       A
      A  B
     A  B  C
    A  B  C  D
```

```
   A  B  C  D  E
  A  B  C  D  E  F
 A  B  C  D  E  F  G
```

41.threading example

```
import threading
def print_one():
    for i in range(10):
        print(i)

def print_two():
    for i in range(10):
        print(2)
t1 = threading.Thread(target =print_one())
t2 = threading.Thread(target =print_two())
t1.start()
t2.start()

#2nd example
def print_one(x,y):
    return x+y

def print_two(x,y):
    return x*y
t1 = threading.Thread(target =print_one,args = (3,7))
t2 = threading.Thread(target =print_two,args = (3,7))
t1.start()
t2.start()
```

43. class method

```
class details:
    company = "TCS"
    @classmethod
    def method(cls,name):
        print("{} is working on {}".format(name,cls.company))
details.method("raja")
details.method("raja1")
```

44.encapsulation method(private,protected,public)

```
class Test:
    x = "virat"
    _y = "Anushka"
    __z = "kohili"

    def data(self):
        print(Test.x,Test._y,Test.__z)

obj = Test()
#print(Test.x,Test._y,Test.__z)
```

```
obj.data()
print(Test.x,Test._y,obj._Test__z)
```

45.IS PYTHON PASS BY VALUE OR PASS BY REFERENCE?
Python by default pass by reference

pass by reference:
The pass by reference change the original value of variable passed as arguments

```
def add_more(list):
        list.append(50)
        print("Inside Function", list)
list = [10,20,30,40]
add_more(list)
print("Outside Function:", list)
```

o/p:Inside Function [10, 20, 30, 40, 50]
Outside Function: [10, 20, 30, 40, 50]

pass by value:
The pass by value does'nt change the original value of variable passed as arguments

```
def sample(var):
    var = "Rajasekhar"
    print("inside function:",var)
var = "reddy"
sample(var)
print("outside function:",var)
```

o/p:inside function: Rajasekhar
outside function: reddy

```
def add_more(list):
        list = [1,34,90]
        print("Inside Function", list)
list = [10,20,30,40]
add_more(list)
print("Outside Function:", list)
```

46.
USER DEFINED EXCEPTIONS:

```
class TransitionError(Exception):

        # Raised when an operation attempts a state
        # transition that's not allowed.
```

```
        def __init__(self, prev, next, msg):
                self.prev = prev
                self.next = next

                # Error message thrown is saved in msg
                self.msg = msg


try:
        raise(TransitionError(2, 3*2, "Not Allowed"))

# Value of Exception is stored in error
except TransitionError as error:
        print('Exception occurred: ', error.next)
-------------------------------------------------------------
class ZeroException(Exception):
        "Raised when the input value b is  0"
        pass

a = int(input("enter first value1:"))
b = int(input("enter first value2:"))
try:

        if b == 0:
                raise ZeroException
        else:
                print("valid:",a/b)
except ZeroException:
        print("Exception occurred: Invalid Age")

-------------------------------------------------------------

class vote(Exception):
    pass

n = int(input("enter a age:"))
try:
    if n<18:
        raise vote
    else:
        print("valid age:",n)
except vote as e:
    print("invalid age:",n)



lambda problums:
-------------------

l = [('English', 88), ('Science', 90), ('Maths', 97), ('Social sciences', 82)]
```

```
k = sorted(l,key=lambda x:x[1])
print(k)
l.sort(key=lambda x:x[1])
print(l)

s = {"english":88,"science":90,"mathes":82}
print({k:v for k,v in sorted(s.items(),key=lambda x:x[1])})

colors_list = ["Red", "Green", "Blue", "White", "Black"]
result = list(map(lambda x:x[::-1], colors_list))
print(result)

#remove common elements
l = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
l1= [2, 4, 6, 8]
print(list(filter(lambda x:x not in l1,l)))
output:
[1,3,5,7,9,10]

l = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
l1= [2, 4, 6, 8,78]
print(list(filter(lambda x:x in l1,l)))
output:
[2,4,6,8]

#filter palindromes
s = ["php", "w3r", "Python", "abcd", "Java", "aaa"]
print(list(filter(lambda x:x == x[::-1],s)))

#remove null values
s = [12, 0, None, 23, None, -55, 234, 89, None, 0, 6, -12]
print(list(filter(lambda x:x != None,s)))

#filter divide by 13 or 19
s = [19, 65, 57, 39, 152, 639, 121, 44, 90, 190]
print(list(filter(lambda x:x%19==0 or x%13==0,s)))

#filter the length equal to 6
s = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
print(list(filter(lambda x:len(x)==6,s)))


--------------------------------------
#vowels find
s = "pratapcapgemini"
print(list(filter(lambda x:x in "aeiou",s)))


decorators:(decorator is a function that can add additional functionality to an
existing function)
```

```
---------------
1.login_problum
---------------
def login_required(f1):#f1 defines the funtion names(home,webpage)
    def inner(islogin):#matching parameters with function(home,webpage) parameters
        if islogin == False:
            print("kindly login")
            return
        else:
            return f1(islogin)#it calls the decorator functions
    return inner

@login_required
def home(islogin):
    print("welocome to home page")
@login_required
def webpage(islogin):
    print("welocome to webpage page")
#inputs
home(False)
webpage(True)

2.intergers sum
---------------
def sample(f1):
    def inner(a,b):
        if type(a) == type(b):
            return f1(a,b)#calls the decorators
        else:
            return "please provide correct values"
    return inner

@sample
def normal(a,b):
    return a+b
#print(normal(7,"virat"))
print(normal(7,10))

3.string upper conversion
-------------------------
def conversion(f1):
    def inner(a,b):
        if type(a) == type(b):
            return f1(a,b)
        else:
            return "plz correct data provide"
    return inner

@conversion
def data(a,b):
```

```
    return a.upper()+" "+b.lower()
print(data("virat","kohili"))
print(data("virat",8))
```

4.zero devision

```
---------------
def zero_div(f1):
    def inner(a,b):
        if b == 0:
            return "please provide proper values"
        else:
            return f1(a,b)
    return inner
@zero_div
def div(a,b):
    return a/b
print(div(7,0))
print(div(7,7))
```

Genertors:

```
-------------

def data():
    return "virat1"           ------>
    return "virat2"
    return "virat3"
print(data())
#only one return executed.
```

using yield:

```
------------
def data():
    yield "virat1"
    yield "virat2"
    yield "virat3"
obj = data() #All yield data stored in object.
for data in obj:
    print(data)
```

output:

```
-------
virat1
virat2
virat3

print(next(obj))#it holds one data
print(obj.__iter__())#same as abve next
```