```
"""
ZIP()
"""
x=[1,2,3,4]
y=[5,'d',6,8]
l=list(zip(x,y))

x = [1,2,3]
y = [4,5,6,7,8]
z=['one','two','three']
list(zip(x,y,z))

coordinate = ['x', 'y', 'z']
value = [3, 4, 5, 0, 9]

result = zip(coordinate, value)
resultList = list(result)
print(resultList)
#zip*zippedlist will unzip it
c, v =  zip(*resultList)
print('c =', c)
print('v =', v)
#---------------------------------
"""
ANY()
"""
l = [1, 3, 4, 10]
print(any(l))
#return True, if all value s true

l = [0, False]
print(any(l))
#return false if all values are false: 0 is false

l = [0, False, 5]
print(any(l))
#return true if one value is true

l = []
print(any(l))
#empty list means false

s = "This is good"
print(any(s))

# 0 is False
# '0' is True
s = '000'
print(any(s))

s = ''
print(any(s))

d = {0: 'False'}
print(any(d))

d = {0: 'False', 1: 'True'}
```

```
print(any(d))

d = {0: 'False', False: 0}
print(any(d))

d = {}
print(any(d))

# 0 is False
# '0' is True
d = {'0': False}
print(any(d))
#-----------------------------
"""
All()
"""
# all values true
l = [1, 3, 4, 5]
print(all(l))

# all values false
l = [0, False]
print(all(l))

# one false value
l = [1, 3, 4, 0]
print(all(l))

# one true value
l = [0, False, 5]
print(all(l))

# empty iterable
l = []
print(all(l))
#---------------------------------

class Student1():
    Sclass='CA'

ss=Student1.Sclass



class Student():
    university="Punjabi University"
    def __init__(self, name,class_s,rollnumber,news):#userdefined
attributes
        self.name=name
        self.class_s=class_s
        self.rollnumber=rollnumber
        #wat it to be a boolean attribute
        self.news=news
    #methods
    def message(self):
        if self.news==False:
            print("Welcome back at {}!".format(self.university))
        else:
```

```
            print("{}! welcome to the class of {} at
{}.".format(self.name, self.class_s, Student.university))

    def grades(self, a1,a2,pro,t1,t2):
        self.total= a1+a2+pro+t1+t2
        return self.total

s= Student(name='Pooja',class_s='ds',rollnumber=45, news=False)
s.message()
print(s.university)
print(s.grades(12,23,12,32,32))

s2= Student(name='Arti',class_s='CA',rollnumber=25, news=True)
s2.message()
print(s2.university)



#inheritence
class Animal():#main class
    def __init__(self):
        print('This is an Animal Class')
#cat=Animal()
    def eat(self):
        print('I need food')

    def walk(self):
        print("I want to go out on walk!")


#derived class
class cat(Animal):
    def __init__(self):
        Animal.__init__(self)
        print("I'm a little kitty")

    def eat(self):
        print("I eat tuna!")#overwite

    def sleep(self):
        print("I sleep alot!")

catty=cat()
catty.eat()
catty.walk()
catty.sleep()

#polymorphism

class Dog:
    def __init__(self, name):
        self.name = name

    def speak(self):
        return self.name+' says Woof!'

class Cat:
    def __init__(self, name):
```

```
        self.name = name

    def speak(self):
        return self.name+' says Meow!'

dog1 = Dog('Niko')
cat1 = Cat('Felix')

print(dog1.speak())
print(cat1.speak())

for pet in [dog1,cat1]:
    print(pet.speak())


def pet_speak(pet):
    print(pet.speak())

pet_speak(dog1)
pet_speak(cat1)
print(dog)

#special methos dunders
class Book:
    def __init__(self, title, author, pages):
        print("A book is created")
        self.title = title
        self.author = author
        self.pages = pages

    def __str__(self):
        return "Title: %s, author: %s, pages: %s" %(self.title,
self.author, self.pages)

    def __len__(self):
        return self.pages

    def __del__(self):
        print("A book is destroyed")
b=Book('Python','Pooja',180)
print(b)

l=[1,2,3,5,6,7]
import numpy as np
a=np.array(l)

l1=[[1,2,3],[11,12,13],[21,22,23]]
a1=np.array(l1)

a2=np.arange(0,11,2)
a3=np.zeros(3)
a4=np.ones((3,2))

a5=np.linspace(0,20,50)
aa=np.full((3,4),np.random.randint(4,56))
a6=np.eye(4)
a7=np.random.rand(5) #uniform dist
a8=np.random.randn(2)
```

```
a_7=np.random.rand(5,4) #uniform dist
a_8=np.random.randn(2,3)
a9=np.random.randint(5,10)
a10=np.random.randint(5,100,4)

arr=np.arange(25)
arr11=arr.reshape(5,5)
arr11.shape

arr1=np.random.randint(5,100,4)
arr1.max()
arr1.min()
#index location of max n min

arr1.argmin()
arr1.argmax()

arr1[2]#element at index 2?
a=np.arange(0,50)
a_slice=a[:6]
a_slice[:]=80#so numpy doesnot created a new array for a_slice.

#for copy
slice_a_copy=a[:6].copy()
slice_a_copy[:]=67


l1=[[1,2,3],[11,12,13],[21,22,23]]
a1=np.array(l1)

a1[0][2]
#or
a1[0,2]

a1[0:2,1:]

#conditional selection
a1_bool=a1>6
a1[a1_bool]
#or simply
a1[a1>6]

a1.sum()
a1.sum(axis=0)
a1.std()
a1.mean()


a1=[1,2,3,4]
a2=[10,11,12,13]
#a2-a1

a1=np.array(a1)
a2=np.array(a2)
a2-a1
a1-a2
a1*a2
a1=a1*2
```

```
a1/a1
np.sqrt(a1)
np.exp(a1)

import numpy as np
import pandas as pd


labels=['a','b','c','d']

data=[10,20,30,40]

arr=np.array(data)

d={'a':10,'b':20,'c':30,'d':40}


pd.Series(data)

pd.Series(data, index=labels)
#or
pd.Series(data,labels)



data=pd.Series([23,34,45,56], ['a','b','c','d'])

pd.Series(arr)
#or
pd.Series(arr, labels)


pd.Series(d)

ser1 = pd.Series([1,2,3,4],index = ['USA', 'Germany','USSR', 'Japan'])
ser2 = pd.Series([1,2,5,4],index = ['USA', 'Germany','Italy', 'Japan'])
ser1['USA']
ser1[0:3]
ser1 + ser2


from numpy.random import randn
df = pd.DataFrame(randn(5,4),index='A B C D E'.split(),columns='W X Y
Z'.split())

import pandas as pd
I=pd.Series['a','b','c','d','e']
I=['a','b','c','d','e']
data=np.array([1,2,3,4,5])
dD=pd.Series(data,I)
dD
dic={'Name':'Pooja', 'profession':'trainer',
'EMAIL':'poojachoudhary80@gmail.com'}
Dic=pd.Series(dic)
Dic
dD=pd.Series(10,I)
dD
ind=['a','b','c','d','e','f']
```

```
val=np.array([10,90,200,500,505,260])
dD1=pd.Series(val,ind)
dD1
ind['d']
ind[3]
val[3]
val[2:5]
val[a,c,f]
val[0,2,6]
val[0,2,5]
val[::2]
val[-1,::-2]
val[-3]

max(dD1)
```