```
# -*- coding: utf-8 -*-
"""
Created on Mon Jun  4 00:24:03 2018

@author: pooja
"""

def my_fun():
    '''
    this is just a function
    '''
    print('Hi i am good')

def my_fun1(name):
    print('Hello'+ ' '+name)

def myfunc(name):
    print('Hello {}'.format(name))

def my_fun2(name='ALL'):
    print('Hello'+' '+name)

st=my_fun2()#none type st as function doesnot return anything

def my_fun3():
    return 'Hello'

def myfunc(a):
    if a==True:
        return 'Hello'
    else:
        return 'Goodbye'

def myfunc(x,y,z):
    if z==True:
        return x
    else:
        return y

def myfunc(num1,num2):
    return num1+num2

def is_even(num):
    if num%2==0:
        return True
    else:
        return False

def is_greater(num1,num2):
    if num1>num2:
        return True
    else:
        return False
'''
*args and **kwargs
simple arguments and keyword arguments
arg is passed as tuple:arbitrary number of positional parameters
```

```
kargs returns back dictionary
'''

def p(*args):
    return sum(args)
p(23,45,67)

def f(**kwargs):
    print(kwargs)
    if 'fruits' in kwargs:
        print("I like {}".format(kwargs['fruits']))
    else:
        print('No fruits')
f(fruits='apple', veggies='mushrooms')

def both_arg(*args, **kwargs):
    print("I would like to have {} and
{}".format(args[0],kwargs['food']))

both_arg('tea','coffee','juice',fruits='orange', food='pasta')



def myfunc(*args):
    l=[]
    for i in args:
        if i%2==0:
            l.append(i)
    return l

def myfunc(string):
    ss=[]
    for i in range(len(string)):
        if i%2==0:
            ss.append(string[i].upper())
        else:
           ss.append(string[i].lower())
    strings=''.join(ss)
    return strings







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}

#creating series with list
pd.Series(data)

#creatinf series with list and index values
pd.Series(data, index=labels)
#or
pd.Series(data,labels)

#creating series in  line filling data and index values

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

#creating list with an array
pd.Series(arr)
#or
pd.Series(arr, labels)

#creating dictionary using zip()
my_dict=dict(zip(data,labels))

#creating list with dictionary
pd.Series(d)

ser1 = pd.Series([1,2,3,4],index = ['USA',India,'Italy', 'China'])
ser2 = pd.Series([1,2,5,4],index = ['USA', 'Germany','India', 'China'])

#accessing values using indexing and index names
ser1['Italy']
ser1[0:3]
ser1[3]
ser1 + ser2



from numpy.random import randn
randn(5,4)
#creating dataframe using random with index values and column names
df = pd.DataFrame(randn(5,4),index=['A', 'B', 'C', 'D',
'E'],columns=['W', 'X', 'Y', 'Z'])
#checking for type od dataframe and series in it
type(df)
type(df['W'])

#applying conditional slicing on dataframe
df>0
df[df>0]#dataframe where values are greater than 0
df[df['W']>0]#dataframe where values in column 'W' are greater than 0
df[df['W']>0]['Y']#column y of dataframe where values in column 'W' are
greater than 0
df[df['W']>0][['Y','X']]#column X and y of dataframe where values in
column 'W' are greater than 0
df[(df['W']>0) & (df['Y'] > 1)]# For two conditions you can use | and &
with parenthesis:
# ## More Index Details# # Let's discuss some more features of indexing,
including resetting the index or setting it something else. We'll also
talk about index hierarchy!
```

```
#reset index values back to 0 1 2....
df.reset_index()
#adding a new column in data frame
newind = ['P','O','W','Y']
df['St'] = newind
#setting a column values as index
df.set_index('St')

df.set_index('St',inplace=True)
```