**Introduction to Numpy**

Numpy is a fundamental package in Python that is used for scientific computational operations.

Installation of Numpy-

* pip install numpy/sudo pip install numpy

numpy package provides

* a powerful N dimensional array object
* sophisticated functions
* linear algerbra, random numbers, fourier transform etc.

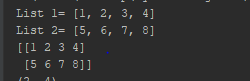
Examples

1. creating an array

import numpy as np  
list1= [1,2,3,4]  
list2=[5,6,7,8]  
myArray=np.array([list1,list2])  
print("List 1=",list1)  
print("List 2=",list2)  
print(myArray)

The above code creates two arrays list1 and list2 and combines them to form a single array.

Output:



1. arange()- used to create an array in sequential order

syntax- a=np.arange(length)

bydefault, the array starts from 0

syntax- a=np.arange(start,end,increment)

import numpy as np  
a=np.arange(0,12)  
print(a)

Output:



1. shape and dtype fuction

shape - indicates the dimension of an array/list

dtype - indicates the type of data stored in an array

print(myArray.shape)  
print(myArray.dtype)

Output:



1. zeros() , ones() and eye()

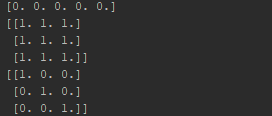
zeros()- creates an array of specified length with all elements as 0.

ones()- crates an array of specified length with all elements as 0.

eye()- creates an identity matrix of specified length.

new\_array=np.zeros(5)  
new\_array1=np.ones([3,3])  
new\_array2=np.empty(5)  
new\_array3=np.eye(3)  
new\_array4=np.arange(0,20,2)  
print(new\_array)  
print(new\_array1)  
#print(new\_array2)  
print(new\_array3)  
print(new\_array4)

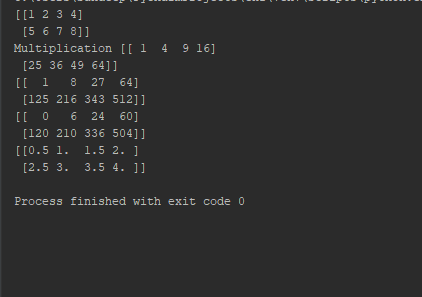
Output:



1. Matrix Multiplication, subtraction, Exponential

from \_\_future\_\_ import division  
import numpy as np  
arr1=np.array([[1,2,3,4],[5,6,7,8]])  
print(arr1)  
  
#multiplication  
print('Multiplication',arr1\*arr1)  
#exponential multiplication  
arr2=arr1 \*\* 3  
print(arr2)  
  
#subtraction  
s=arr2-arr1  
print(s)  
  
d=arr1/2  
print(d)

Output:



1. Array Indexes(usage of :)

import numpy as np  
a=np.arange(0,12)  
print(a)  
  
print(a[0]) #prints the first element  
print(a[0:5]) #prints the first 5 elements  
a[0:5]=20 #assigns value 20 to the first 5 elements of the array  
acopy=a.copy()  
print(a[0:5])  
print(a)  
#interesting thing  
a2=a[0:6]  
print(a2)  
a2[:]=29 #all the elements  
print(a2)  
print(a)  
#no extra copy is created when using numpy  
  
#creating new array copy  
print(acopy)  
acopy=a.copy()

a:b where a-row, b-column. Accesses all the elements from a to b.

a.copy()-creates a copy of an array.

**Note-** While using numpy, no copy of array is created implicitly.

We have to make use of copy() function.

Ex. In the above example- as you can see array a2 is derived from array a. Making any changes to a2 will also affect the value to a.

To avoid any changes in the original matrix, we use copy()

1. Multidimensional Arrays

import numpy as np  
a=np.array([[1,2,3],[4,5,6],[7,8,9]])  
print(a)  
print(a[0]) #prints the first row

Output:



Accessing each element of an array:

#accessing each element  
print(a[0][2])

Output:

3

Slices in an array

#slices in an array  
slice1=a[0:1,0:2]  
print("Slice1=",slice)  
  
slice2=a[:2,2:]  
print("Slice2=",slice1)

Output:



Slice 1 contains 1st row 1st element and 1st row 2nd element.

b=[[0,0,0],[0,0,0],[0,0,0]]  
a\_len=a.shape[0]  
print(a\_len)  
for i in range(a\_len):  
 b[i]=a[i]\*a[i]  
print(b)  
#another way of accessing elements  
print(a[[0,2]])

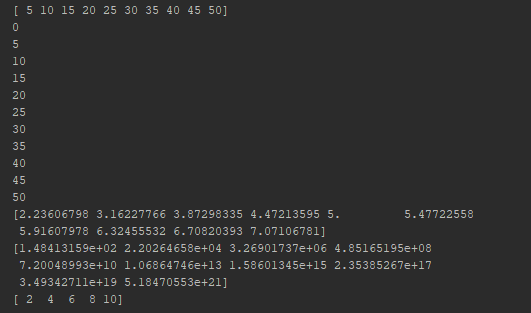
a.shape[0] returns the number of rows in an array.

1. Some basic functions in numpy

* Arange- creates a sequential array
* Sqrt- returns the square root
* Exp- returns e^the element, e is Eulers number
* Add- adds two elements
* Max – returns the maxinum value

import numpy as np  
#arange  
#sqrt  
#random  
#exp  
#add  
#max  
a=np.arange(5,55,5)  
a1=[1,2,3,4,5]  
print(a)  
  
i=0  
n=5  
for i in range(11):  
 print(n\*i)  
  
b=np.sqrt(a)  
print(b)  
#b1=np.sqrt(25)  
#print(b1)  
c=np.exp(a) #eulers number raised to array  
print(c)  
  
d=np.add(a1,a1)  
print(d)  
  
#e=np.max(a,b)  
#print(e)

Output:



1. Saving and Loading elements from external library

import numpy as np  
a=np.arange(10)  
print(a)  
  
#save numpy array  
np.save("saved",a)  
#new file is craeted with name saved.npy  
  
new\_a=np.load("saved.npy") #load the saved file  
print(new\_a)  
  
#saving multiple arrays as zip or archive file  
a1=np.arange(25)  
a2=np.arange(30)  
np.savez("saved\_archive.npz",x=a1,y=a2) #savez is used for saving multiple arrays  
load\_npz=np.load("saved\_archive.npz")  
print(load\_npz['x'])  
print(load\_npz['y'])  
  
#save the arrays to textfile  
  
np.savetxt('text.txt',a1,delimiter=',')  
#loading of txt files  
  
load\_txt=np.loadtxt('text.txt',delimiter=',')  
print("Text File=",load\_txt)  
#convets an integrer into a float number

np.save()- saves an array

np.load()-loads an array

np.savez()- saves multiple arrays

np.savetxt()- saves an array in text format

np.loadtxt()- loads the array stored in text format

Saving and Loading files in external memory helps **to save the memory**.

It stores the data in a zip or archive file, whose size is comparatively smaller.

1. Matplotlib – it is python library used for plotting figures, graphs etc.

import numpy as np  
import matplotlib.pyplot as plt  
a=np.arange(-100,100,10)  
print(a)  
  
dx, dy = np.meshgrid(a,a) #groups the point with every other point  
#print("Dx",dx)  
#print("Dy",dy)  
function=2\*dx+3\*dy  
function2=np.cos(dx)+np.cos(dy)  
print("Function",function)  
print("Function 2",function2)  
plt.imshow(function)  
plt.title('Function Plot of 2dx+3dy')  
plt.colorbar()  
plt.savefig('MyFig.png')  
  
plt.imshow(function2)  
plt.title('Function of Cos(dx)+Cos(dy)')  
plt.colorbar()  
plt.savefig('MyFig2.png')

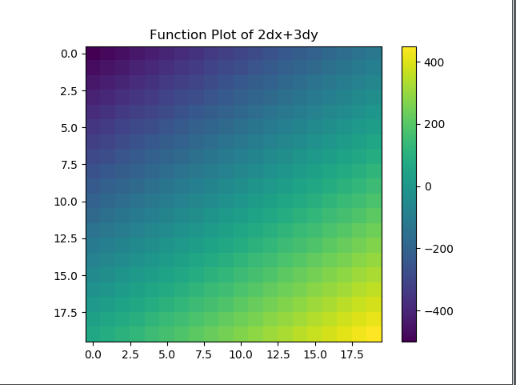
.meshgrid()- group every point with the corresponding point in other element.

.imshow()- used for displaying values/images

.title()- assigns a title for the result

.colorbar()- Adds a colorbar to a plot

.savefig(‘MyFig.png’)- saves the plot as a png file



1. Conditional Clauses

import numpy as np  
x=np.array([100,200,300,400]) #each member as a  
y=np.array([400,500,600,700]) #each member as b  
condition=np.array([True,True,False,False]) #each member cond  
#if true take x else y  
z= [a if cond else b for a,cond, b in zip(x,condition,y)]  
print(z)  
  
#same opeartion using numpy  
#np.where(#condition,#value for yes,#value for no)  
z2=np.where(condition,x,y)  
  
z3=np.where(x>200,0,1) #if x is > 200 replace it by 0 else 1  
print(z3)

1. Various Function in numpy

#sum  
print(x.sum())  
  
n=np.array([[10,20,30],[40,50,60],[70,80,90]])  
print("N=",n)  
#colomun sum  
print("Sum of Coloums=",n.sum(0))  
#row sum  
print("Sum of Rows=",n.sum(1))  
#mean  
mean=x.mean()  
print("Mean=",mean)  
#std deviation  
std\_dev=x.std()  
print("Standard Deviation=",std\_dev)  
  
#varinace  
var=x.var()  
print("Variane=",var)  
  
#logical operations  
  
condition2=np.array([True,False,True])  
print("OR operator",condition2.any())  
print("AND operator",condition2.all())  
  
#sorting in numpy arrays  
unsorted=np.array([2,5,9,4,-6,3,1])  
print("Unsorted Array=",unsorted)  
unsorted.sort()  
print("Sorted Array=",unsorted)  
  
s=np.array(["Solid","Solid","Liquid","Liquid","Liquid","Gas","Gas"])  
n=np.array([1,2,3,1,2,4,4,8,9,9,3,4,4])  
print(np.unique(s))  
print(np.unique(n))  
  
print(np.in1d([2,6,9],n))

np.in1d([2,6,9],n) – returns a Boolean value to indicate if the given elements are present in the array.

**Introduction to Pandas**

Pandas is a data manipulation library used for data analysis.

1. Series-

syntax- pandas.Series(data,index)

.values()-prints the values of the series

.index()- prints the indexes of the series

s=Series([5,10,15,20])  
print("Series=",s)  
print("Values=",s.values)  
print("Index=",s.index)

We can also create Series by passing an array to it.

Ex-

data=np.array(['a','b','c'])  
s1=Series(data)  
print(s1)

we can create our own set of indexes(both numeric and text based)

s1=Series(data,index=[100,101,102])  
print("Customized Series Index=",s1)  
  
s=Series(data,index=['A','E','I'])  
print(s)

Conditions can be used within a Series to manipulate data

rev=Series([20,80,40,35],index=['Ola','Uber','grap','gojek'])  
print(rev)  
  
#using index to access values  
print(rev['Ola'])  
#usinh condition within series  
  
print(rev[rev>30])  
  
#using boolean conditions  
  
print( 'Ola' in rev)  
print('Car' in rev)

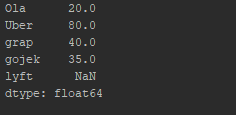
‘ola’ in rev determines if it is present in the Series and returns a Boolean value.

We can convert a Series in to a Dictionary by using .to\_dict()

rev\_dict=rev.to\_dict()  
print(rev\_dict)

#nan values(not available values)  
index2=['Ola','Uber','grap','gojek','lyft']  
rev2=Series(rev,index2)  
print(rev2)

Output:



.isnull()- returns true if null value is present else false.

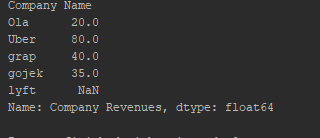
.notnull()- returns true if the value is not present else false.

print(pd.isnull(rev2))  
print(pd.notnull(rev2))

#addition of series  
print(rev+rev2)  
  
#assigning names  
rev2.name="Company Revenues"  
rev2.index.name="Company Name"  
print(rev2)

Series can be added by using + operator.

Names can be assigned to Series and the index values of the series.



1. DataFrames

DataFrame is 2 dimensional tabular structure with two axes(rows and cols) is a container for Series objects.

import pandas as pd  
import numpy as np  
from pandas import Series,DataFrame  
  
#revenue of companies  
  
df=pd.read\_clipboard()  
print(df)  
#access indexes and columns  
print(df.columns)  
print(df['Industry'])  
  
#multiple columns  
print(DataFrame(df,columns=['Rank','Industry','Name']))  
#nan values  
df2=DataFrame(df,columns=['Rank','Industry','Name','Profit'])  
print("New dataFrame=")  
print(df2)  
  
#head and tail  
  
print(df2.head(4)) #prints first 5 rows  
print(df2.tail(4)) #prints last 5 rows  
  
#access rows in dataframe  
#print(df.ix[0]) #does not work  
print(df.iloc[0]) #first row  
print(df.loc[5]) #5th row  
  
#assign values to dataframe  
#using numpy  
a1=np.array([1,2,3,4,5,6,7,8])  
df2['Profit']=a1  
print(df2)  
  
#using series  
  
profit=Series([900,100],index=[3,5])  
df2['Profit']=profit  
print(df2)  
  
#deletion  
del df2['Profit']  
print(df2)  
  
#use dictionary with dataframe  
  
sample= {  
 'Company':['A','B'],  
 'Profit':[1000,5000]  
}  
print(sample)  
dict\_df=DataFrame(sample)  
print(dict\_df)

read\_clipboard()- read the value for the data frame copied on the clipboard.

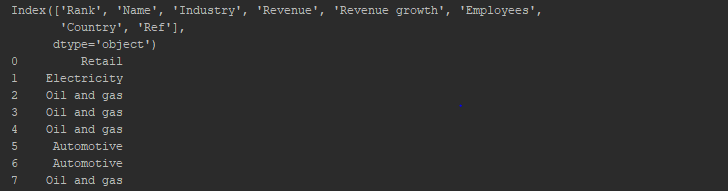
df=pd.read\_clipboard()  
print(df)

df.columns- displays all the columns in the datframe

individual colums can be accessed by specifying the name of the coloum

#access indexes and columns  
print(df.columns)  
print(df['Industry'])

Output:



Multiple columns can also be accessed

#multiple columns  
print(DataFrame(df,columns=['Rank','Industry','Name']))

.head(n)- return the first n values in the dataframe.

.tail(n)- returns the last n values in the dataframe.

Individual rows can be accssed using iloc() or loc() function

#head and tail  
print(df2.head(4)) #prints first 5 rows  
print(df2.tail(4)) #prints last 5 rows  
#access rows in dataframe  
#print(df.ix[0]) #does not work  
print(df.iloc[0]) #first row  
print(df.loc[5]) #5th row

#assign values to dataframe  
#using numpy  
a1=np.array([1,2,3,4,5,6,7,8])  
df2['Profit']=a1  
print(df2)

The above code assigns the values of array a1 to the Profit column.

#using series  
  
profit=Series([900,100],index=[3,5])  
df2['Profit']=profit  
print(df2)

Values 900 and 100 are assigned at the index 3 and 5.

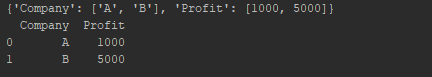
#deletion  
del df2['Profit']  
print(df2)

Del- deletes the entire column from the dataframe.

#use dictionary with dataframe  
  
sample= {  
 'Company':['A','B'],  
 'Profit':[1000,5000]  
}  
print(sample)  
dict\_df=DataFrame(sample)  
print(dict\_df)

The above code converts a dictionary sample into a dataframe. This can be done by passing dict in the DataFrame()

Output:



1. Indexes

s1=Series([10,20,30,40],index=['A','B','C','D'])  
print(s1)  
  
index1=s1.index  
print(index1)

.index – returns the index values

#index operations  
print(index1[2])  
print(index1[2:])

Returns the value at index 2 and value from index 2 to the last index, respectively.

Negative Indexes

#negative indexes  
print(index1[-2:]) #ignores the first 2 elemnts and prints the rest  
print(index1[:-2]) #ignores the last 2 elemnts and prints the rest

Range of Indexes

print(index1[1:4])

prints all the indexes from 1 to 3(index starts from 0)

Note- index once assigned cannot be changed using = operator.

1. Re-indexing

#create a new series  
s1=Series([1,2,3,4],index=['e','f','g','h'])  
print(s1)  
  
#creating new indexes using reindex  
  
s2=s1.reindex(['e','f','g','h','i','j'])  
print(s2)

reindex()- is used to change the index values of Series.

#using fill value  
s2=s2.reindex(['e','f','g','h','i','j','k'],fill\_value=10)  
print(s2)  
  
#using ffill  
cars=Series(['Audi','BMW','Honda'],index=[0,4,8])  
print(cars)  
ranger=range(13)  
print(ranger)  
cars=cars.reindex(ranger,method="ffill")  
print(cars)

fill()- fills in the given value to the new index

ffill()- forward fill is used to fill the missing values.

#create new df suing random  
#reindex rows of data frame  
df1=DataFrame(randn(25).reshape(5,5),index=['a','b','c','d','e'],columns=['c1','c2','c3','c4','c5'])  
print(df1)  
df2=df1.reindex(['a','b','c','d','e','f'])  
print(df2)

.randn(n)- provides n random numbers.

.reshape()- reshapes the given values into rows and cols.

The above code creates a new dataframe using random values.

#reindex cols  
df3=df2.reindex(columns=['c1','c2','c3','c4','c5','c6'])  
print(df3)

We can also re-index the column values using reindex()

df4=df1.loc[['a','b','c','d','e','f'],['c1','c2','c3','c4','c5','c6']]  
print(df4)

We can re-index both rows and cols simultaneously by using ix[] or loc[] or iloc[]

1. Dropping entries in Series and DataFrame

import pandas as pd  
import numpy as np  
from pandas import Series,DataFrame  
  
cars=Series(['BMW','Audi','Honda'],index=['a','b','c'])  
print(cars)  
  
#drop enteries  
cars=cars.drop('a')  
print(cars)  
#df  
cars\_df=DataFrame(np.arange(9).reshape(3,3),index=['BMW','Audi','Honda'],columns=['rev','profit','expenses'])  
print(cars\_df)  
  
#drop rows  
cars\_df=cars\_df.drop('BMW',axis=0)  
print(cars\_df)  
#drop cols  
cars\_df=cars\_df.drop('profit',axis=1) #axis=0 for index  
print(cars\_df) #axis=1 for columns

drop series values

syntax- series\_name.drop(‘index\_name’)

drops the elements of the given index

drop rows and cols in a dataframe

dataframe\_name.drop(‘index\_name’,axis=0) // for index

dataframe\_name.drop(‘column\_name’,axis=1)//for columns

axis=0;index

axis=1;columns

1. Handing null values

import numpy as np  
import pandas as pd  
from pandas import Series,DataFrame  
s1=Series(['A','B','C','D','E',np.nan])  
print(s1)  
#validate  
print(s1.isnull())  
#drop unavailbale values  
print(s1.dropna())  
  
df=DataFrame([[1,2,3],[4,5,np.nan],[7,np.nan,10],[np.nan,np.nan,np.nan]])  
print(df)  
#drona in dataframe  
#print(df.dropna()) #delets the entire row that has atleast one nan entry  
print(df.dropna(how="all"))  
  
#dropna corresponding to columns  
  
print(df.dropna(axis=1)) #column vise deletion  
  
df2=DataFrame([[1,2,3,np.nan],[4,5,6,7],[8,9,np.nan,np.nan],[12,np.nan,np.nan,np.nan]])  
print(df2)  
  
print(df2.dropna(thresh=3)) #should contain atleast 3 data values (for rows)  
print(df2.dropna(thresh=3,axis=1)) #for cols  
  
#fillna  
  
print(df2.fillna('-')) #fills all the null values by 0  
print(df2.fillna({0:0,1:50,2:100,3:200})) #replaces null value in col 1 by 0, col2 by 50, col3 by 100 and col4 by 200

.isnull()- returns true if the entry is null else false

.dropna()- drops null values

Note- .dropna() drops every row with atleast one null entry

np.nan()-creates a null entry.

.dropna(how="all")- drops only those rows in which all the entries are null.

.dropna(axis=1)- drops the elements column wise.

.dropna(thresh=3)- drops the row if it contains less than 3 values

.dropna(thresh=3,axis=1)- works column wise.

.fillna('')- fills the null values with the specified value.

1. Accessing elements of Series and datframe

import pandas as pd  
import numpy as np  
from pandas import Series, DataFrame  
  
s1= Series([100,200,300],index=['A','B','C'])  
print(s1)  
#access element of series  
  
print(s1['A'])  
#access multiple elements  
print(s1[['A','B']])

elements of a series can be accessed by specifying the index of the series.

Multiple elements of a series can also accessed by passing multiple index values.

#number indexes  
print(s1[0]) #equivalent to s1['A']  
#aceesing multiple elements  
print(s1[0:2])  
print(s1[0:4])  
  
#conditional indexes  
print(s1[s1>150])  
print(s1[s1==300])

We can also pass the number as index instead of index values to save time.

Conditional statements can also be used with Series as shown above.

df1=DataFrame(np.arange(9).reshape(3,3),index=['Car','Bike','Cycle'],columns=['A','B','C'])  
print(df1)  
  
#col wise  
print(df1['A'])  
print(df1[['A','B']]) #multiple values  
  
print(df1>5)  
  
#access elements using ix function  
  
print(df1.loc['Bike'])

elements of a dataframe can be accessed by in a similar way as that of Series by passing column values.

df1[]>5- returns true if the values are greater than 5 else false

1. Alignment of elements

import pandas as pd  
import numpy as np  
from pandas import Series,DataFrame  
s1= Series([100,200,300],index=['A','B','C'])  
s2= Series([300,400,500,600],index=['A','B','C','D'])  
print("Sum of Series=",s1+s2)  
  
#datframe  
df1=DataFrame(np.arange(4).reshape(2,2),index=['Car','Bike'],columns=['A','B'])  
df2=DataFrame(np.arange(9).reshape(3,3),index=['Car','Bike','Boat'],columns=['A','B','C'])  
print("Df1\n",df1)  
print("Df2\n",df2)  
print("Sum of DataFrames\n",df1+df2)  
  
df1=df1.add(df2,fill\_value=0)  
print(df1)  
  
s3=df2.iloc[0]  
print(df2-s3)

‘+’ operator when used on the 2 series adds the corresponding indexes of the series.

Null values are not considered and output is –nan

‘+’ operator when used on the 2 dataframes adds the corresponding indexes of the series.

Null values are not considered as 0.

1. Ranking and Sorting

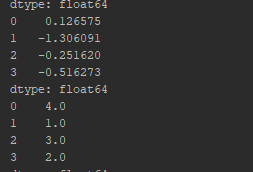
import numpy as np  
import pandas as pd  
from pandas import Series,DataFrame  
from numpy.random import randn  
  
s1=Series([500,1000,1500],index=['a','c','b'])  
print(s1)  
  
#sorting by index  
print(s1.sort\_index())  
  
#sort by values  
print(s1.sort\_values())  
  
#ranking of series  
print(s1.rank())  
s2=Series(randn(4))  
print(s2)  
  
print(s2.rank())

.sort\_index- sorts the series by its index value

.sort\_value- sorts the series by the value of elements

.rank()- first sorts the series by its value(elements) and then assigns rank accordingly

Ex:



1. Statistics

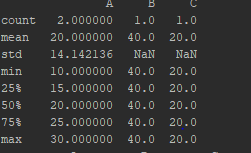
from pandas import Series,DataFrame  
import numpy as np  
from numpy.random import randn  
import matplotlib.pyplot as plt  
  
array1 = np.array([[10,np.nan,20],[30,40,np.nan]])  
print (array1)  
df1 = DataFrame(array1,index=[1,2],columns=list('ABC'))  
print (df1)  
  
#sum()  
print (df1.sum()) #sums along each column  
print(df1.sum(axis=1)) #sum along indexes  
  
print (df1.min())  
print (df1.max())  
  
print(df1.idxmax())  
print (df1.cumsum())  
print (df1.describe())  
  
df2 = DataFrame(randn(9).reshape(3,3),index=[1,2,3],columns=list('ABC'))  
print (df2)  
  
plt.plot(df2)  
plt.legend(df2.columns,loc="lower right")  
plt.savefig('samplepic.png')  
plt.show()

.min() / .max()- returns min and max value in each column of the datframe respectively.

.idxmax()- retuns the index containing highest value in the dataframe.

.cumsum()- returns the cumulative sum of the columns of datframe.

.describe()- returns values such as count, mean , min, max etc. As shown below



.plot()- used to plot the dataframe values.

Plots can be made as line, bar , box, hist etc.

Ex- .plot(hist)-plots a histogram

Legend- is used to identify each data element uniquely.

legend(df2.columns,loc="lower right")- displays the legend object at the lower right corner.

.show()-displays the plot.

ser1 = Series(list('abcccaabd'))  
print (ser1.unique())  
  
print (ser1.value\_counts())

.unique()-prints only the distinct values in the series.

.value\_counts()- prints the count of each element in the series.