# NumPy: Numerical python by Travis Oliphant

For working with arrays

import numpy as np

1. To define an array:

1-D -> a = np.array( [ … ] )

2-D -> a = np.array( […] ,[…] )

3-D -> a = np.array( […] ,[…],[…] )

1. To get type -> type(a)
2. To get size(No.of Elements in array) -> a.size
3. To get shape of array -> a.shape
4. To get no of dimensions -> a.ndim
5. To get type of elements -> a.dtype
6. To transpose -> a.transpose()
7. To get random array of size -> np.empty((rows,col),dtype = \_ )
8. To find an element -> x=np.where( condition )
9. To sort the array -> np.sort(a)

|  |  |
| --- | --- |
| flatten() | ravel() |
| 1)return the view of original array | 1)returns the copy of original array |
| 2)when this is modified it will not | 2)when this is modified it will change |
| change values of original array so it | values of original array so it is faster |
| is slower as it consumes more space |  |
| 3)it is method of ndarray obj | 3)it is library level function |

1. To get an array of 1s -> np.ones((rows,cols) , dtype = \_ )

To get an array of 0s -> np.zeros((rows,cols) , dtype = \_ )

Dtypes are str / int / float / bool

1. To get an array of number of range -> a = np.arange(start,end,step) not arrange it is arrange
2. To reshape it into require shape -> a.reshape((rows,cols))
3. To convert to n-D to 1-D -> a.flatten() and a.ravel()
4. To access nth row -> a[n-1]
5. To access mth element of nth row -> a[n-1][m-1]
6. To get i to j th row -> a[ i : j ]
7. To get n to m elements in i to j th row -> a[ i-1:j ,n-1:m ]
8. To arithmetic operations:

1. Add -> a+b / np.add(a,b)

2. Subtract -> a-b / np.subtract(a,b)

3. Multiply(not matrix multiplication) -> a\*b / np.multiply(a,b)

4. Division -> a/b / np.division(a,b)

5. Matrix multiplication -> a@b / a.dot(b)

6. Min in array -> a.min()

7. Max in array -> a.max()

8. Max value index -> a.argmax()

9. Sum of all elements -> np.sum(a)

To get sum of row-wise -> np.sum(a, axis=1)

To get sum of col-wise -> np.sum(a, axis=0)

10. mean -> np.mean(a)

11. square root -> np.sqrt(a)

12. standard deviation -> np.std(a)

13. log -> np.log(a)

14. for pi value -> np.pi

15. for sin -> np.sin(a) or np.sin(angle)

16. llry for cos tan

17. for random generated array -> np.random.random(shape)

shape=(rows,cols)

18. for random in range -> np.random.randint(start,end ,shape)

19. for random float values -> np.random.rand(shape)

20. incase of -ve values -> np.random.randn(shape)

21. to select randomly -> np.random.choice( array\_name )

1. Let s1 and s2 are 2 strings

a. To add the strings -> np.char.add(s1,s2)

b. To convert to uppercase -> np.char.upper(s1)

c. To convert to lowercase -> np.char.lower(s2)

d. To split at space -> np.char.split(s1)

e. To split at lines -> np.char.splitlines(s1)

f. To replace -> np.char.replace(s1,old\_str ,new\_str)

g. To print at center with surrounding => np.char.center(str ,size ,

remaining\_place\_str )

# Pandas :Panel Data by Wes McKinney

To work with data sets and analyze big data

Import pandas as pd

1) To check the version -> pd.\_\_version\_\_

2)Series is like a col in table = 1D array

Create using Series ->

pd.Series(list\_name,index = [ … ],name =‘Name\_of\_series’)

For empty series -> pd.Series([])

3) Create using dict

To create series -> pd.Series(dict\_name)

The key will become indexes for the values

4) DataFrames => Creating using csv /xslx file or manually

CSV = comma separated files

Using list -> pd.DataFrame(list/array/dict)

Eg list = {

Col\_name: [ vals ] , …..

}

Using csv -> pd.read\_csv(“location\_of\_file”)

5) Let df = pd.read\_csv(‘ … ’)

a. To get columns -> df.columns

b. To get no.of rows and cols -> df.shape

c. To get no.of cells -> df.cells

d. To get first n rows -> df.head(n)

e. To get last n rows -> df.tail(n)

f. To get description of numerical data -> df.describe()

g. To get info of this -> df.info()

6) To check for null values -> df.is.null()

a. To get total no.of null values ->

df.isnull().sum() row wise add .sum() to get total no. of nulls

7) To drop rows with atleast one null ->

df.dropna(axis = ) 0 for row 1 for col

a. To drop row with any one with null -> df.dropna(how = ‘any’)

b. To drop row with complete empty -> df.dropna(how =’all’)

c. df after removing null values -> df,dropna(inplace =True)

d. To fill null value with n -> df.fillna(n)

e. To fill null value acc to column name -> df.fillna({col1:value\_be\_replaced\_with …})

f. To fill with previous value -> df.fillna(method = ‘ffill’) if next val bfill

g. To replace -> df.replace(to\_replace = value/list , value = replacing\_val)

h. To replace all char -> df.replace(‘[A-Za-z]’,replacing\_val regex=True)

i. To group -> df.groupby(by = val )

j. To merge to dataframes -> pd.merge(df1,df2 , on =on\_what\_bases )

k. To merge one dataframe to outer -> pd.merge(df1,df2 , how = case )

i. outer -> all data ii. left -> on basis of df1

iii. right -> on basis of df2

l. to append df1 to df2 -> df1.append(df2 ,ignore\_index= True)

Changing type of a column using .astype(newType)

read loc() and iloc()

m. loc means to return one or more specified rows details ->

df.loc[row\_number]

# Matplotlib : import matplotlib.pyplot as plt

1) plt.style.use(‘dark\_background’) -> for black background

2) plt.figure(figsize = (row,cols)) -> to give graph a size

3) plt.plot(arrays/list\_values/….,’\_color\_’,linestyle =’ ‘ ,linewidth =

val) -> to plot

4) plt.show() -> to show graph

5) plt.title(‘….’) -> to give title

6) plt.xlabel(‘…’) -> to give x-axis a title & llry for y

7) to make sub graphs -> plt.subplot(row,cols,index) above each graph

stmts

8) plt.scatter(list,list,color= ‘green …,marker = ‘ ‘,marksize =val

,alpha=opacity\_val)

9) plt.xticks(np.arange(…)) -> to marks val on x-axis llry for y-axis

10) plt.bar(data , color= ,width = ,edgecolor = linestyle = ) -> for bar chart

11) plt.hist(data,bin = ,color = ,rwidth= ,histtype= ) -> for histogram

12) plt.pie(data , labels =,colors ,autopct = ‘%0.1f%%’,explode = val,

shadow = True ,radius = ,textprops = ,wedgeprops

eg: x\_cos=np.arrage(0,2\*pi,0.1)

y\_cos = np.cos(x\_cos)

plt.figure(figsize(6,6))

plt.plot(x\_cos,y\_cos)

plt.show()

# Seaborn: import seaborn as sns

1. Boxplot

We load pandas and matplot

df = sns.load\_dataset(“tips”) ie: tips is inbuilt

df.boxplot(by = ‘ x-axis row selection ‘ ,column = [ name of any col ] , grid = true/false)

1. Distplot

df=sns.load\_dataset(“titanic”)

age1=df[‘age’].dropna()

sns.distplot(age1,bins=30,kde=false)

plt.show()

1. Edgeplot

Data=sns.load\_dataset(‘mpg’)

sns.regplot(x=’mpg’,y=’acceleration’,data=data)

plt.show()

# VLC Module : import vlc

1. To initialize playing file

media = vlc.MediaPlayer(“file\_name.mp4”)

1. To start player -> media.play()
2. To end or stop -> media.stop()
3. To create a instance of vlc player

vlc\_obj = vlc.Instance()

1. Create media player

vlcplayer = vlc\_obj.media\_player\_new()

1. Creating the media player

vlc\_media = vlc\_obj.media\_new(“file\_name.mp4”)

1. To set media to player

vlcplayer.set\_media(vlc\_media)

1. To play the set media file

vlcplayer.play()

1. To get the video duration of media

vlcplayer.get\_length()

# Speedtest Module: import speedtest

1. To get the tester

test = speedtest.SpeedTest()

1. To get the servers available

test.get\_servers()

1. To get the best sever available

test.get\_best\_server()

1. To get download speed for the network

test.download()

1. To get upload speed

test.upload()

**To get the speed in mbps divide the value by 220**

1. To get ping value

test.results.ping

# Pyforest Module: import pyforest

1. active\_imports() => to display library loaded to memory
2. lazy\_imports() => to display all the libraries that are frequently loaded to memory