In [ ]:
```
#Day-17
#Agenda  :    Python Bulit-in Libraries:
                  1. Numpy
                  2. Pandas
                  3. MatPolt Lib
#Numpy:
Numpy is an Open Source Library available in Python.

#It provides High -level functions to perform mathematical operations on numerical data.
Author: Travis Oliphant
First Release : 1995
Written in- Python Programming,C

#Features
Numpy stands - for Numrical Python (Numpy-num+py)
1. Its contains powerful N-dimensional Object
2. Fourier Transforms and shapes manipulation.
3. Linear Algebra and random number generation.
```

In [ ]:
```
#How to Install Numpy Package in your System:
pip install numpy   - #IDLE Users

conda install -c anaconda numpy  - #anaconda users

#How to import and use numpy
import numpy as np
```

In [2]:
```
#How to import and use numpy
import numpy as np

#to check version
print(np.__version__)
```

```
1.16.4
```

In [14]:
```
#how to create Numpy arrays: (single-dimensional arrays)
#We create Numpy arrays using Lists or Nested lists.
import numpy as np
a = np.array([1,2,3])
print(a)
print(a.shape)  #to find the shape of numpy array
print(a.dtype)  # to find datatype of numpy array
a
```

```
[1 2 3]
(3,)
int32
```

Out[14]:
```
array([1, 2, 3])
```

In [13]:
```
#numpy arrays with float values
b = np.array([1.1,3.5,6.8])
print(b.dtype)
print(b.shape)
```

```
float64
(3,)
```

In [18]:
```
#How to create 2D numpy Arrays:
import numpy as np
d = np.array([[1,2,3],[4,5,6],[7,8,9]])
print(d)
print(d.shape)
print(d.dtype)
```

```
[[1 2 3]
 [4 5 6]
 [7 8 9]]
(3, 3)
int32
```

In [38]:
```
#Numpy operations:
#zeros: (its used to create a matrix full of zeros)
#syntax:
#numpy.zeros(shape,dtype=float,order="C")
import numpy as np
print(np.zeros((2,2)))
print(np.zeros((4,4)))
```

```
[[0. 0.]
 [0. 0.]]
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
```

In [32]:
```
#np.ones() functions:
#syntax:
#numpy.zeros(shape,dtype=float,order="C")

print(np.ones([1,2,3]))   # To give the rows and coloums order
print(np.ones((1,2,3),dtype=np.int16))
```

```
[[[1. 1. 1.]
  [1. 1. 1.]]]
[[[1 1 1]
  [1 1 1]]]
```

In [42]:
```
print(np.ones((2,3)))
print(np.ones((3,2)))
```

```
[[1. 1. 1.]
 [1. 1. 1.]]
[[1. 1.]
 [1. 1.]
 [1. 1.]]
```

In [49]:
```
#reshape of numpy arrays
#reshape():
#syntax:
             #numpy.reshape(a,newshape,order="C")
             #a - array that you what to change
import numpy as np
a = np.array([[1,2,3],[5,6,8]])
print(a)
print(a.shape)
b=a.reshape(3,2)
print(b)
print(b.shape)
print(b.flatten())
```

```
[[1 2 3]
 [5 6 8]]
(2, 3)
[[1 2]
 [3 5]
 [6 8]]
(3, 2)
[1 2 3 5 6 8]
```

In [52]:
```
#Hstack and Vstack: (for appending the data in horizontal and vertically)
#Hstack:
import numpy as np
a = np.array([[1,3,5]])
b = np.array([[4,6,8]])
print(np.hstack((a,b)))
print(np.vstack((a,b)))
```

```
[[1 3 5 4 6 8]]
[[1 3 5]
 [4 6 8]]
```

In [54]:
```
#Generate random numbers:
#numpy.random.normal(loc,scale,size)
# Loc- mean of distribution
#scale - standard deviation
#size- number of retuns
# To generate random number from normal distribution.
normal_array = np.random.normal(5,0.5,10)
print(normal_array)
```

```
[4.41895406 5.1965508  4.91275274 4.90870586 4.66131123 5.02896264
 4.55874946 4.73970913 5.27904588 4.63092246]
```

In [ ]:
```
#arange() function:
Its used to returns an ndarray(n-dimensional array) object that contains evenly
spaced values within a defined internal.
#syatax:

numpy.arange(start,stop,size)
```

In [67]:
```
import numpy as np
print(np.arange(1,11))

print(np.arange(1,11,2))
print(np.linspace(1.0,5.0,num=10,endpoint=False)) #for generationg even spaced samples
print(np.logspace(3.0,4.0,num=4))   #for even space samples on log scale
```

```
[ 1  2  3  4  5  6  7  8  9 10]
[1 3 5 7 9]
[1.  1.4 1.8 2.2 2.6 3.  3.4 3.8 4.2 4.6]
[ 1000.          2154.43469003  4641.58883361 10000.        ]
```

In [81]:
```
#slicing on numpy arrays:
import numpy as np
a=np.array([(1,2,3,5),(3,5,6,8),[50,60,90,100]])
print(a[0,2])
print(a[1,3])
print(a[0, 2], a[1, 2])   #boomika logic
print(a[0:,2])      #my logic
print(a[0:,1])  #prasanna kumar logic
print(a[1,1],a[1,3])   #naveen logic
print(a[0:2,3])
```

```
3
8
3 6
[ 3  6 90]
[ 2  5 60]
5 8
[5 8]
```

In [98]:
```
#Max/min:
import numpy as np
a = np.array([[1,2,3,4,5],[4,5,6,7,8]])
b = np.array([[1,2,3,4,5],[4,5,6,7,8]])
print(a.max())
print(a.min())
print(a.sum())
print(a.sum(axis=0))  #column based
print(a.sum(axis=1)) #row based
print(np.sqrt(a))
print(np.std(a))
print(a+b)
print(a-b)
print(a*b)
print(a/b)
```

```
8
1
45
[ 5  7  9 11 13]
[15 30]
[[1.         1.41421356 1.73205081 2.         2.23606798]
 [2.         2.23606798 2.44948974 2.64575131 2.82842712]]
2.0615528128088303
[[ 2  4  6  8 10]
 [ 8 10 12 14 16]]
[[0 0 0 0 0]
 [0 0 0 0 0]]
[[ 1  4  9 16 25]
 [16 25 36 49 64]]
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
```

In [ ]:
```
#Pandas in Python:
Pandas is an opensource library in python that allows to you perform
                (i)    Data manipuation,
                (ii)   Data analysis
                (iii)  Data Cleaning
1. its built on top of Numpy Packages,Pandas need to Numpy arrays data to operate
2. its a elegant solution for time series Data
            1. Population of Data
            2. any Company production Growth
    #Author : Wes McKinney
    #First Release: version 0.23 July,2018
    #Written in : Python

#Why use pandas?
 Data Scientics use pandas for its
     1. easily handles missing data
     2. flexible way to slice the data.
     3. Provides powerful time series tool to work with Data Frames.
```

In [ ]:
```
#How to install pandas:
pip install pandas
```

In [ ]:
```
#What is Data frame:(Pandas Data Structures (Data Frame and Data Series))
Df is 2d- array ,with labeled axes (rows and columns) to store the data.
 its contains tabular data.
   #ex: item     price
    0   A        50
    1   B       100
#Data Series:
 A series is a 1- D data structure. It can have any data type values
     like integer, float, and string
    #ex:
    0    1.0
    1   2.0
    2   3.0
    dtype: float64
```

In [107]:
```
# How to create Data series:
import pandas as pd
s1 =pd.Series([1,2,3])
s2= pd.Series([1.,2.,3.],index=["a","b","c"])
s3 = pd.Series([1,2,np.nan])   #missing value as "NaN"
print(s1)
print(s2)
print(s3)
```

```
0    1
1    2
2    3
dtype: int64
a    1.0
b    2.0
c    3.0
dtype: float64
0    1.0
1    2.0
2    NaN
dtype: float64
```

In [113]:
```
#How to create Data Frames:
#Numpy to Pandas
import numpy as np
#a = [[1,3],[5,6]]
df = pd.DataFrame([[1,3],[5,6]])
print(df)
df
#pandas to numpy
df_to_numpy = np.array(df)
print(df_to_numpy)
```

```
   0  1
0  1  3
1  5  6
[[1 3]
 [5 6]]
```

```
In [118]: #dataframe with Dictionary Values:
          import pandas as pd
          dic = {'Name':["surya","apssdc","python"],"values":[50,60,100]}
          pd.DataFrame(data=dic,index=["1","2","3"])
```

```
Out[118]:
             Name  values
          1  surya     50
          2  apssdc    60
          3  python   100
```

```
In [ ]:
```