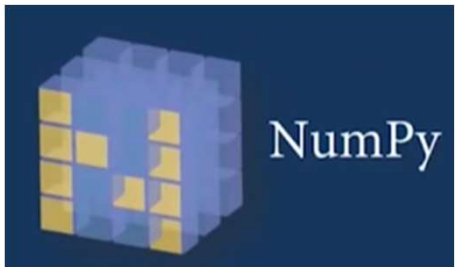
Python Library

- Python library is a collection of functions and methods that allows you/user to perform many action without writing complex code.



Python libraries for statistical analysis

- NumPy (numerical computing / complex mathematical computation)
  - Scientific Computations
  - Multi-dimensional array objects
  - Data manipulation
- SciPy
  - Mathematical computations
  - Provide a collection of sub-packages
  - Advanced linear algebra functions
  - Support for signal processing
- Pandas (data manipulation with pandas)
  - Data frame objects
  - Process large data sets
  - Complex data analysis
  - Time series data

Python libraries for data visualization

- Matplotlib (Bar plot, graph chart, scatter plot)
  - Plot a variety of graphs
  - Extract quantitative info
  - Pyplot module (Similar to MATLAB)
  - Integrate with tools
- Seaborn
  - Compatible with various data formats
  - Support for automated statistical estimation
  - High level abstractions
  - Support for build-in themes (graph themes)
- Plotly
  - In-build API
- Bokeh
- Create complex statistical graph
- Integration with flask and django

Python libraries for machine learning

- Scikit-learn
  - Provides a set of standard datasets
  - Machine learning algorithm
  - In-build functions for feature extraction and selection
  - Model evaluation
- XGBoost
  - eXtreme Gradient Boosting
  - Fast data processing
  - Support parallel computation
  - Provides internal parameters for evaluation
  - Higher Accuracy
- Eli5

Python library for Deep learning

- TensorFlow
  - Build and train multiple neural networks
  - Statistical analysis
  - In-build functions to improve the accuracy
  - Comes with an in-build visualizer
- Keras
  - Support various type of neural networks
  - Advanced neural networks computatons
  - Provide pre-processed dataset
  - Easily extensible
- Pytorch
  - APIs to integrate with ML/Data science frameworks
  - Support for multi-dimensional arrays
  - 200+ mathematical operations
  - Build Dynamic computation graphs

NumPy

- Numerical Python (Core library for numeric and scientific computing)
- It consists of multi- dimensional array objects and a collection of routines (methods) for processing those arrays.

Work with multi-dimensional array and provide methods to process it….

Creating NumPy Array

Single-dimensional Array

```
In [3]: import numpy as np

        n1=np.array([10,20,30,40])
        n1

Out[3]: array([10, 20, 30, 40])
```

→Load library

(Method array () of numpy, pass list of element inside array)

(Method array of numpy, pass **list of list** of element inside array) (2 row ad 4 column)

Multi-dimensional Array

```
In [6]: import numpy as np

        n2=np.array([[10,20,30,40],[40,30,20,10]])
        n2

Out[6]: array([[10, 20, 30, 40],
               [40, 30, 20, 10]])
```

```
In [13]: import numpy as np

In [14]: l1 = [1,2,3,4]

In [16]: n1=np.array(l1)

In [17]: n1
Out[17]: array([1, 2, 3, 4])

In [18]: type(n1)
Out[18]: numpy.ndarray

In [19]: n2=np.array([[1,2,3,4],[4,3,2,1]])

In [20]: n2
Out[20]: array([[1, 2, 3, 4],
                [4, 3, 2, 1]])
```

```
[8]  l1=[1,2,3,4,5]
```

← create list

```
[10] np.array(l1)

     array([1, 2, 3, 4, 5])
```

Check type of array

```
[14] type(n1)

     numpy.ndarray
```

Initializing NumPy Array with zeros
--Call zeors(row, col) method of NumPy

```
[15] n1=np.zeros((1,2))
```

```
[16] n1

     array([[0., 0.]])
```

```
[17] n1=np.zeros((5,5))
```

```
[18] n1

     array([[0., 0., 0., 0., 0.],
            [0., 0., 0., 0., 0.],
            [0., 0., 0., 0., 0.],
            [0., 0., 0., 0., 0.],
            [0., 0., 0., 0., 0.]])
```

Initializing NumPy Array with some number
--Call full((dimensions),value) method of NumPy
full((row,col),val)

```
In [38]: import numpy as np
         n1=np.full((2,2),10)
         n1

Out[38]: array([[10, 10],
                [10, 10]])
```

```
In [22]: n3 = np.zeros((2,3))

In [23]: n3

Out[23]: array([[0., 0., 0.],
                [0., 0., 0.]])

In [24]: type(n3)

Out[24]: numpy.ndarray
```

```
In [25]: n3 = np.zeros((10,10))

In [26]: n3

Out[26]: array([[0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
                [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
                [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
                [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
                [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
                [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
                [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
                [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
                [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.],
                [0., 0., 0., 0., 0., 0., 0., 0., 0., 0.]])
```

```
In [27]: n4 = np.full((3,3),55)

In [28]: n4

Out[28]: array([[55, 55, 55],
                [55, 55, 55],
                [55, 55, 55]])
```

Initializing NumPy Array within a range

--Call arange(initial value , final value) method of NumPy

--Final value is not included in range

```
In [34]: import numpy as np
         n1=np.arange(10,20)
         n1

Out[34]: array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
```

```
In [29]: n1 = np.arange(50,101)

In [30]: n1

Out[30]: array([ 50,  51,  52,  53,  54,  55,  56,  57,  58,  59,  60,  61,  62,
                63,  64,  65,  66,  67,  68,  69,  70,  71,  72,  73,  74,  75,
                76,  77,  78,  79,  80,  81,  82,  83,  84,  85,  86,  87,  88,
                89,  90,  91,  92,  93,  94,  95,  96,  97,  98,  99, 100])
```

--If find values given in series (steps/gaps)

```
In [35]: import numpy as np
         n1=np.arange(10,50,5)
         n1

Out[35]: array([10, 15, 20, 25, 30, 35, 40, 45])
```

```
In [31]: n1 = np.arange(50,500,10)

In [32]: n1

Out[32]: array([ 50,  60,  70,  80,  90, 100, 110, 120, 130, 140, 150, 160, 170,
                180, 190, 200, 210, 220, 230, 240, 250, 260, 270, 280, 290, 300,
                310, 320, 330, 340, 350, 360, 370, 380, 390, 400, 410, 420, 430,
                440, 450, 460, 470, 480, 490])
```

Final value is excluded ..

Initializing NumPy Array with random numbers

--Call random.randint(initial value , final value, no of random int) method of NumPy

```
np.random.randint(100,200,10)

array([139, 146, 152, 194, 143, 119, 171, 120, 112, 151])
```

```
In [46]: import numpy as np
         n1=np.random.randint(1,100,5)
         n1

Out[46]: array([95, 88, 26, 22, 76])
```

Random is sub module inside NumPy and randint() method of random.

Checking the shape of NumPy arrays

```
In [4]: import numpy as np
        n1=np.array([[1,2,3],[4,5,6]])
        n1.shape

Out[4]: (2, 3)
```

Change the shape of NumPy arrays

```
In [5]: n1.shape = (3,2)
        n1.shape

Out[5]: (3, 2)
```

```
[12] n1=np.array([[1,2,3,4],[5,6,7,8]])
```

```
[13] n1

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

```
[14] n1.shape

     (2, 4)
```

```
n1.shape=(4,2)
n1.shape

(4, 2)
```

```
[16] n1

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

```
n1.shape = (8,1)
n1.shape

(8, 1)
```

```
n1

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

```
In [38]: import numpy as np

In [39]: n1 = np.array([[1,2,3,4],[4,3,2,1]])

In [40]: n1

Out[40]: array([[1, 2, 3, 4],
                [4, 3, 2, 1]])

In [42]: n1.shape = (4,2)

In [43]: n1

Out[43]: array([[1, 2],
                [3, 4],
                [4, 3],
                [2, 1]])

In [45]: n1.shape = (8,1)

In [46]: n1

Out[46]: array([[1],
                [2],
                [3],
                [4],
                [4],
                [3],
                [2],
                [1]])
```

Join NumPy arrays
-vstack()--vertical
-hstack()--horizontal
-column_stack()

```
In [32]: import numpy as np
         n1=np.array([10,20,30])
         n2=np.array([40,50,60])

         np.vstack((n1,n2))

Out[32]: array([[10, 20, 30],
                [40, 50, 60]])
```

```
In [33]: import numpy as np
         n1=np.array([10,20,30])
         n2=np.array([40,50,60])

         np.hstack((n1,n2))

Out[33]: array([10, 20, 30, 40, 50, 60])
```

```
In [34]: import numpy as np
         n1=np.array([10,20,30])
         n2=np.array([40,50,60])

         np.column_stack((n1,n2))

Out[34]: array([[10, 40],
                [20, 50],
                [30, 60]])
```

```
In [50]: import numpy as np

In [51]: n1 = np.array([1,2,3])

In [52]: n2 = np.array([4,5,6])

In [53]: np.vstack((n1,n2))

Out[53]: array([[1, 2, 3],
                [4, 5, 6]])

In [54]: np.hstack((n1,n2))

Out[54]: array([1, 2, 3, 4, 5, 6])

In [55]: np.column_stack((n1,n2))

Out[55]: array([[1, 4],
                [2, 5],
                [3, 6]])
```

```
In [10]: import numpy as np
         n1=np.array([10,20,30,40,50,60])
         n2=np.array([50,60,70,80,90])

In [11]: np.intersect1d(n1,n2)
Out[11]: array([50, 60])
```

```
In [10]: import numpy as np
         n1=np.array([10,20,30,40,50,60])
         n2=np.array([50,60,70,80,90])

In [23]: np.setdiff1d(n1,n2)
Out[23]: array([10, 20, 30, 40])
```

NumPy Intersection and Difference

--Intersection: Find common in two array

--Difference: Find unique in two array

```
n1=np.array([10,20,30,40,50,60])
```

```
n2=np.array([50,60,70,80,90])
```

```
np.intersect1d(n1,n2)
```

```
array([50, 60])
```

```
np.setdiff1d(n1,n2)
```

n1-n2

```
array([10, 20, 30, 40])
```

```
np.setdiff1d(n2,n1)
```

n2-n1

```
array([70, 80, 90])
```

One id

```
In [56]: import numpy as np

In [58]: n1=np.array([10,20,30,40,50,60])

In [59]: n2 = np.array([50,60,70,80,90])

In [60]: np.intersect1d(n1,n2)
Out[60]: array([50, 60])

In [61]: np.setdiff1d(n1,n2)
Out[61]: array([10, 20, 30, 40])

In [62]: np.setdiff1d(n2,n1)
Out[62]: array([70, 80, 90])
```

NumPy Array Mathematics
--Addition of NumPy Arrays

```
n1=np.array([10,20])
```

```
n2=np.array([30,40])
```

```
np.sum([n1,n2])
```

```
100
```

--Add column values (axis=0)

```
np.sum([n1,n2],axis=0)
```

```
array([40, 60])
```

--Add row values (axis=1)

```
np.sum([n1,n2],axis=1)
```

```
array([30, 70])
```

Basic Addition

```
In [4]: import numpy as np
        n1=np.array([10,20,30])
        n1=n1+1
        n1
Out[4]: array([11, 21, 31])
```

Basic Multiplication

```
In [6]: import numpy as np
        n1=np.array([10,20,30])
        n1=n1*2
        n1
Out[6]: array([20, 40, 60])
```

Basic Subtraction

```
In [5]: import numpy as np
        n1=np.array([10,20,30])
        n1=n1-1
        n1
Out[5]: array([ 9, 19, 29])
```

Basic Division

```
In [7]: import numpy as np
        n1=np.array([10,20,30])
        n1=n1/2
        n1
Out[7]: array([ 5., 10., 15.])
```

NumPy Math Functions

Mean = {Sum of Observation} ÷ {Total numbers of Observations}

Mean

```
In [14]: import numpy as np
         n1=np.array([10,20,30,40,50,60])
         np.mean(n1)

Out[14]: 35.0
```

Median

```
In [16]: import numpy as np
         n1=np.array([11,44,5,96,67,85])
         np.median(n1)

Out[16]: 55.5
```

5, 11, 44, 67, 85, 96

Standard Deviation

```
In [17]: import numpy as np
         n1=np.array([1,5,3,100,4,48])
         np.std(n1)

Out[17]: 36.59424666377065
```

1, 3, 3, **6**, 7, 8, 9

Median = 6

1, 2, 3, **4**, **5**, 6, 8, 9

Median = (4 + 5) ÷ 2

= 4.5

```
n1=np.random.randint(1,50,10)
```

```
n1
```

```
array([17, 44, 11, 12, 45, 37, 10,  9,  2,  8])
```

```
np.mean(n1)
```

```
19.5
```

```
np.median(n1)
```

```
11.5
```

```
np.std(n1)
```

```
15.26597523907333
```

NumPy Save and Load
--Saving NumPy Array
-- Loading NumPy Array

```
In [13]: import numpy as np
         n1=np.array([10,20,30,40,50,60])
         np.save('my_numpy',n1)
```

Saving Numpy Array

```
In [17]: n2=np.load('my_numpy.npy')
         n2
Out[17]: array([10, 20, 30, 40, 50, 60])
```

Loading Numpy Array

```
n1
```

```
array([17, 44, 11, 12, 45, 37, 10,  9,  2,  8])
```

```
np.save('myarray',n1)
```

```
new_n1=np.load('myarray.npy')
```

```
new_n1
```

```
array([17, 44, 11, 12, 45, 37, 10,  9,  2,  8])
```