Numpy:

NumPy does not come with Python by default so it needs to be installed

importing a library means loading it into the memory and then it’s there for you to work with. To import NumPy you need to write the following code:

import numpy as np

# Working with NumPy:

>NumPy works with Python objects called multidimensional **arrays.**

**>** Arrays are basically collections of values, and they have one or more dimensions

>NumPy array data structure is also called ndarray, short for n-dimensional array.

>An array with one dimension is called a **vector**and an array with two dimensions is called a **matrix.**

**Reading a CSV by NP:**

>From a CSV file

**np\_file\_read=np.genfromtxt("nba.csv",delimiter=",")| Reading a CSV**

 >From a text file

np.loadtxt('t1.txt')

>Save an array to a text file.

np.savetxt('file.txt',arr,delimiter=' ') | Writes to a text file

>Save an array to a csv file

np.savetxt('file.csv',arr,delimiter=',') | Writes to a CSV file

>save : Save an array to a binary file in NumPy ``.npy`` format

np.savetxt("nba1.npy",a,delimiter=",")

savez : Save several arrays into an uncompressed ``.npz`` archive

savez\_compressed : Save several arrays into a compressed ``.npz`` archive

## Creating Arrays:

## np.array([1,2,3]) | One dimensional array np.array([(1,2,3),(4,5,6)]) | Two dimensional array np.zeros(3) | 1D array of length 3 all values 0 np.ones((3,4)) | 3x4 array with all values 1 np.eye(5) | 5x5 array of 0 with 1 on diagonal (Identity matrix) np.linspace(0,100,6) | Array of 6 evenly divided values from 0 to 100 np.arange(0,10,3) | Array of values from 0 to less than 10 with step 3 (eg [0,3,6,9]) np.full((2,3),8) | 2x3 array with all values 8 np.random.rand(4,5) | 4x5 array of random floats between 0–1 np.random.rand(6,7)\*100 | 6x7 array of random floats between 0–100 np.random.randint(5,size=(2,3)) | 2x3 array with random ints between 0–4

## Properties:

## arr.size | Returns number of elements in arr arr.shape | Returns dimensions of arr (rows,columns) arr.dtype | Returns type of elements in arr arr.astype(dtype) | Convert arr elements to type dtype arr.tolist() | Convert arr to a Python list np.info(np.eye) | View documentation for np.eye

## Copying/sorting/reshaping

np.copy(arr) | Copies arr to new memory  
arr.view(dtype) | Creates view of arr elements with type dtype  
arr.sort() | Sorts arr  
arr.sort(axis=0) | Sorts specific axis of arr  
two\_d\_arr.flatten() | Flattens 2D array two\_d\_arr to 1D  
arr.T | Transposes arr (rows become columns and vice versa)  
arr.reshape(3,4) | Reshapes arr to 3 rows, 4 columns without changing data  
arr.resize((5,6)) | Changes arr shape to 5x6 and fills new values with 0

## Adding/removing Elements

np.append(arr,values) | Appends values to end of arr  
np.insert(arr,2,values) | Inserts values into arr before index 2  
np.delete(arr,3,axis=0) | Deletes row on index 3 of arr  
np.delete(arr,4,axis=1) | Deletes column on index 4 of arr

**Python NumPy Array v/s List**

**Why NumPy is used in Python?**

We use python NumPy array instead of a list because of the below three reasons:

1. Less Memory
2. Fast
3. Convenient

#Data-type consisting of more than one element:

