# NumPy:

```
NumPy in Python is a library that is used to work with arrays and was created in 2
005.

NumPy is an open-source library that can be used freely.

NumPy library in Python has functions for working in domain of Linear algebra, Mat
rices, Fourier transfrom.

NumPy stands for Numerical Python.

NumPy Supports basic operations such as average, minimum, maximum, standard deviat
ion, variance, and many more.
```

The key concept in NumPy is the NumPy array datatype. A NumPy array may have one or more dimensions:

```
* One dimention array (1D) represent vectors.

* Two-dimension array (2D) represent matrices.

* and higher diemensional arrays represent tensors.
```

NumPy is an important library for:

```
* Data Science

* Machine Learning

* Signal and image processing

* Scientific and engineer computing
```

In [1]:

```
# Importing NumPy and see the version
import numpy as np
print(np.__version__)
```

```
1.21.5
```

In [2]:

```
# Create 1D array of number from 0 to 10

a = np.arange(11)
a
```

Out[2]:

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10])
```

In [3]:

```
# Convert list into array
l = [1,2,3,4,5]
np.array(l)
```

Out[3]:

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

In [4]:

```
# Convert list into array with all type elements
l1 = [1,2,3,4,'raj', 23.54, True]
np.array(l1)
```

Out[4]:

```
array(['1', '2', '3', '4', 'raj', '23.54', 'True'], dtype='<U32')
```

Here list is converted into array but internally it is converted all elements into string because list contains string element also.

In [10]:

```
# Extract all odd  numbers from list
l2 = [1,2,3,4,5,6,7,8,9,10,12,15,16]
odd = np.array(l2)
print('All odd numbers')
odd[odd % 2 == 1]

```

```
All odd numbers
```

Out[10]:

```
array([ 1,  3,  5,  7,  9, 15])
```

In [11]:

```
even = np.array(l2)
print("All even numbers")
even[even % 2 == 0]
```

```
All even numbers
```

Out[11]:

```
array([ 2,  4,  6,  8, 10, 12, 16])
```

In [14]:

```
# Replace all odd numbers by 0 in list
zero = np.array(l2)
zero[zero % 2 == 1] = 0
zero
```

Out[14]:

```
array([ 0,  2,  0,  4,  0,  6,  0,  8,  0, 10, 12,  0, 16])
```

In [17]:

```
# Convert a 1D array to a 2D array with 2 rows

# Convert a 1D array to a 2D array with 2 rows
arr = np.arange(10)
arr.reshape(2, -1)
```

Out[17]:

```
array([[0, 1, 2, 3, 4],
       [5, 6, 7, 8, 9]])
```

In [19]:

```
# Get common elements in two list
a = np.array([1,2,5,48,9,6,2,3,4,7,1,2,3,6,4])
b = ([5,2,1,4,5,2,1,2])
np.intersect1d(a,b)
```

Out[19]:

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

In [21]:

```
# From array a remove all items present in array b
a = np.array([1,5,2,4,3,4])
b = ([1,2,9,2,7])
np.setdiff1d(a,b)
```

Out[21]:

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

In [23]:

```
# Get the position where elements of a and b match
a = np.array([1,4,5,2,4,6,7])
b = np.array([0,4,8,2,9,6,7])
np.where(a == b)
```

Out[23]:

```
(array([1, 3, 5, 6], dtype=int64),)
```

In [26]:

```
# Addition of to arrays
a = np.array([[1,2,3],[4,8,9]])
b = np.array([[2,5,4],[5,6,3]])

c = a + b
print(c)
```

```
[[ 3  7  7]
 [ 9 14 12]]
```

In [28]:

```
# Multiply the numpy array a(matrix) by 3

a = np.array([[1,4,7,2],[2,3,4,5]])
b = a*3
print(b)
```

```
[[ 3 12 21  6]
 [ 6  9 12 15]]
```

In [30]:

```
# Carete an indentity matrix dimension 4-by-4
a = np.eye(4)
a
```

Out[30]:

```
array([[1., 0., 0., 0.],
       [0., 1., 0., 0.],
       [0., 0., 1., 0.],
       [0., 0., 0., 1.]])
```

In [33]:

```
# Convert a 1D array to a 3D array
a = np.array([x for x in range(27)])
o = a.reshape((3,3,3))
print(o)
```

```
[[[ 0  1  2]
  [ 3  4  5]
  [ 6  7  8]]

 [[ 9 10 11]
  [12 13 14]
  [15 16 17]]

 [[18 19 20]
  [21 22 23]
  [24 25 26]]]
```

In [35]:

```
# Convert all the elements of a numpy array from float to integer datatype

a = np.array([[2.4,6.1,25.8],[5.0,1.2,7.6]])
i = a.astype('int')
print(i)
```

```
[[ 2  6 25]
 [ 5  1  7]]
```

In [37]:

```
# Convert a binary numpy array (containing only 0s and 1s) to a boolen numpy array

a = np.array([[[1,0,1,0],[1,0,1,1],[1,1,0,0],[0,1,0,1]]])
b = a.astype('bool')
print(b)
```

```
[[[ True False  True False]
  [ True False  True  True]
  [ True  True False False]
  [False  True False  True]]]
```

In [40]:

```
# Horizontal Stacking of Numyp Arraya
a = ([[5,4,7,6],[1,2,4,3]])
b = ([[4,7,2,3],[2,4,5,6]])
h = np.hstack((a,b))
print(h)
```

```
[[5 4 7 6 4 7 2 3]
 [1 2 4 3 2 4 5 6]]
```

In [43]:

```
# Vertically Stacking of Numpy Arrays

a = ([[1,2,3,4],[2,4,5,6]])
b = ([[2,4,7,9],[5,7,1,2]])
v = np.vstack((a,b))
print(v)
```

```
[[1 2 3 4]
 [2 4 5 6]
 [2 4 7 9]
 [5 7 1 2]]
```

In [45]:

```
# All 0 values
z = np.zeros((5,5))
z
```

Out[45]:

```
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.]])
```

In [46]:

```
# All one values
o = np.ones((5,6))
o
```

Out[46]:

```
array([[1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1.],
       [1., 1., 1., 1., 1., 1.]])
```

In [48]:

```
# Adding values
o+12
```

Out[48]:

```
array([[13., 13., 13., 13., 13., 13.],
       [13., 13., 13., 13., 13., 13.],
       [13., 13., 13., 13., 13., 13.],
       [13., 13., 13., 13., 13., 13.],
       [13., 13., 13., 13., 13., 13.]])
```

In [50]:

```
# Adding with array
o + np.array([1,2,3,4,5,6])
```

Out[50]:

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

In [53]:

```
# Random numbers
a1 = np.random.randint(1,100,(5,6))
a1
```

Out[53]:

```
array([[35,  7, 33, 24, 55, 43],
       [59, 29, 34, 32, 51, 55],
       [80,  6, 42, 29, 69, 45],
       [10, 78, 38, 79, 98, 98],
       [48, 35, 32,  8, 58, 41]])
```

In [54]:

```
# greater than
a1[a1>50]
```

Out[54]:

```
array([55, 59, 51, 55, 80, 69, 78, 79, 98, 98, 58])
```

In [56]:

```
# Square root
np.sqrt(a1)
```

Out[56]:

```
array([[5.91607978, 2.64575131, 5.74456265, 4.89897949, 7.41619849,
        6.55743852],
       [7.68114575, 5.38516481, 5.83095189, 5.65685425, 7.14142843,
        7.41619849],
       [8.94427191, 2.44948974, 6.4807407 , 5.38516481, 8.30662386,
        6.70820393],
       [3.16227766, 8.83176087, 6.164414  , 8.88819442, 9.89949494,
        9.89949494],
       [6.92820323, 5.91607978, 5.65685425, 2.82842712, 7.61577311,
        6.40312424]])
```

In [58]:

```
# Exponential
np.exp(a1)
```

Out[58]:

```
array([[1.58601345e+15, 1.09663316e+03, 2.14643580e+14, 2.64891221e+10,
        7.69478527e+23, 4.72783947e+18],
       [4.20121040e+25, 3.93133430e+12, 5.83461743e+14, 7.89629602e+13,
        1.40934908e+22, 7.69478527e+23],
       [5.54062238e+34, 4.03428793e+02, 1.73927494e+18, 3.93133430e+12,
        9.25378173e+29, 3.49342711e+19],
       [2.20264658e+04, 7.49841700e+33, 3.18559318e+16, 2.03828107e+34,
        3.63797095e+42, 3.63797095e+42],
       [7.01673591e+20, 1.58601345e+15, 7.89629602e+13, 2.98095799e+03,
        1.54553894e+25, 6.39843494e+17]])
```

In [59]:

```
# Log function
np.log10(a1)
```

Out[59]:

```
array([[1.54406804, 0.84509804, 1.51851394, 1.38021124, 1.74036269,
        1.63346846],
       [1.77085201, 1.462398  , 1.53147892, 1.50514998, 1.70757018,
        1.74036269],
       [1.90308999, 0.77815125, 1.62324929, 1.462398  , 1.83884909,
        1.65321251],
       [1.        , 1.8920946 , 1.5797836 , 1.89762709, 1.99122608,
        1.99122608],
       [1.68124124, 1.54406804, 1.50514998, 0.90308999, 1.76342799,
        1.61278386]])
```