Manaranjan Pradhan
manaranjan@enablecloud.com
*This notebook is given as part of **Data Science for everyone** workshop.*

Numpy Overview

Importing numpy library

In [64]:

```
import numpy as np
```

Creating numpy arrays and initializing

In [65]:

```
## Create one dimensional array
a = np.array([1, 2, 3])
a
```

Out[65]:

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

In [66]:

```
## Find the type of the object
type(a)
```

Out[66]:

```
numpy.ndarray
```

In [67]:

```
## Find the dimension of the array
a.shape
```

Out[67]:

```
(3,)
```

In [68]:

```
print( a[1] )
```

```
2
```

In [69]:

```
# Create a two dimensional array
b = np.array([[1,2,3],[4,5,6]])   # Create a rank 2 array
print( b )
```

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

In [70]:

```
print( b.shape )
```

```
(2, 3)
```

In [71]:

```
print( b[0, :] )
```

```
[1 2 3]
```

In [72]:

```
print( b[:, 1] )
```

```
[2 5]
```

In [73]:

```
b
```

Out[73]:

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

In [74]:

```
b[:]
```

Out[74]:

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

Special Initializing functions

In [75]:

```
a = np.zeros((2,2))  # Create an array of all zeros
# Prints "[[ 0.  0.]
#          [ 0.  0.]]"
a
```

Out[75]:

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

In [76]:

```
b = np.ones((1,2)) # Create an array of all ones
#  [[ 1  1]]
b
```

Out[76]:

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

In [77]:

```
d = np.eye(2)
```

In [78]:

```
d
```

Out[78]:

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

In [85]:

```
c = np.full((2,2), 7)
```

In [86]:

```
c
```

Out[86]:

```
array([[ 7.,  7.],
       [ 7.,  7.]])
```

In [87]:

```
e = np.random.random((2,2) )
```

In [88]:

```
e
```

Out[88]:

```
array([[ 0.73155607,  0.82598003],
       [ 0.72991034,  0.42626126]])
```

In [89]:

```
f = np.random.randint(10, size = (4,4) )
```

In [90]:

```
f
```

Out[90]:

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

Slicing & Indexing an array

In [91]:

```
# Get first row
f[0,:]
```

Out[91]:

```
array([0, 2, 2, 2])
```

In [92]:

```
# Get 1 and 2 row
f[0:2,:]
```

Out[92]:

```
array([[0, 2, 2, 2],
       [6, 3, 6, 0]])
```

In [93]:

```
f[:, 1]
```

Out[93]:

```
array([2, 3, 7, 7])
```

In [94]:

```
# Get first column
f[:,0]
```

Out[94]:

```
array([0, 6, 9, 1])
```

In [95]:

```
f[:,0:2]
```

Out[95]:

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

In [96]:

```
## slicing an array
## All
b = f[:2, 1:3]
```

In [53]:

```
b
```

Out[53]:

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

In [54]:

```
## Get specific elements
np.array([f[0,1], f[2,2]])
```

Out[54]:

```
array([4, 9])
```

In [55]:

```
## Boolean indexing
f>2
```

Out[55]:

```
array([[ True,  True, False,  True],
       [ True,  True, False, False],
       [False, False,  True,  True],
       [ True,  True, False, False]], dtype=bool)
```

In [56]:

```
g = f[f>2]
```

In [57]:

```
g
```

Out[57]:

```
array([3, 4, 3, 3, 3, 9, 6, 8, 4])
```

Reshaping an array

In [58]:

```
np.reshape( f, ( 8,2 ) )
```

Out[58]:

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

Numpy Maths

Adding, substracting, multiplying, tranposing arrays

In [59]:

```
x = np.random.randint( 100, size = (5,5) )
y = np.random.randint( 100, size = (5,5) )
```

In [60]:

```
x
```

Out[60]:

```
array([[64, 47,  0, 55,  9],
       [94, 70, 46,  7, 91],
       [95, 41, 95, 83,  9],
       [47, 46, 40, 85, 18],
       [50, 12, 43, 29, 26]])
```

In [61]:

```
y
```

Out[61]:

```
array([[35, 67,  9, 74, 56],
       [89, 78, 55, 54, 50],
       [29, 89, 56,  6, 22],
       [47, 68,  7, 24, 38],
       [65, 46, 46, 92, 51]])
```

In [62]:

```
## Add two matrices x + y or np.add( x, y )
x + y
```

Out[62]:

```
array([[ 99, 114,   9, 129,  65],
       [183, 148, 101,  61, 141],
       [124, 130, 151,  89,  31],
       [ 94, 114,  47, 109,  56],
       [115,  58,  89, 121,  77]])
```

In [63]:

```
np.add( x, y )
```

Out[63]:

```
array([[ 99, 114,   9, 129,  65],
       [183, 148, 101,  61, 141],
       [124, 130, 151,  89,  31],
       [ 94, 114,  47, 109,  56],
       [115,  58,  89, 121,  77]])
```

In [39]:

```
# np.substract( x, y )
x - y
```

Out[39]:

```
array([[-28,   7,  12,  -3,   4],
       [ -2, -23,  46,   1, -26],
       [ -1,  28, -18,  32, -21],
       [-14, -35,  -4,   5,  -8],
       [ 47,  22,  13, -74, -29]])
```

In [40]:

```
# np.multiply( x, y )
x * y
```

Out[40]:

```
array([[1404, 6960,  589, 5548, 4352],
       [ 728, 3408, 2720,  812, 1431],
       [  12, 1325, 6003, 2448, 2752],
       [5576, 5394, 1845, 3654, 4080],
       [ 440,  968,  140, 2475,  660]])
```

In [41]:

```
# Matrix Transpose
x.T
```

Out[41]:

```
array([[26, 26,  3, 68, 55],
       [87, 48, 53, 58, 44],
       [31, 80, 69, 41, 20],
       [73, 29, 68, 63, 25],
       [68, 27, 43, 60, 15]])
```

In [42]:

```
np.sum( x )
```

Out[42]:

```
1180
```

In [43]:

```
np.sum( x, axis = 0 )
```

Out[43]:

```
array([178, 290, 241, 258, 213])
```

In [44]:

```
np.sum( x, axis = 1 )
```

Out[44]:

```
array([285, 210, 236, 290, 159])
```

Linear Algebra.. Advanced Matrix Operation

In [45]:

```

```

Solving a set of linear equations

```
from numpy import linalg
```

- 2x + 2y = 5
- 3x + y = 7

In [46]:

```
a = np.array([[2,2], [3,1]])
b = np.array([5,7])
x = np.linalg.solve(a, b)
x
```

Out[46]:

```
array([ 2.25,  0.25])
```

Matrix Inversion

In [47]:

```
a = np.array([[1, 2], [3, 4]])
linalg.inv( a )
```

Out[47]:

```
array([[-2. ,  1. ],
       [ 1.5, -0.5]])
```

Calculating an eigen value and vector for a matrix

In [48]:

```
m1 = np.diag((1, 2, 3))
m1
```

Out[48]:

```
array([[1, 0, 0],
       [0, 2, 0],
       [0, 0, 3]])
```

In [49]:

```
eigval, eigvec = linalg.eig( m1 )
```

In [50]:

```
eigval
```

Out[50]:

```
array([ 1.,  2.,  3.])
```

In [51]:

```
eigvec
```

Out[51]:

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