Numpy Library

In [1]:
```
# Importing numpy library

import numpy as np
```

1-D Array

In [2]:
```
# One-dimensional Array

a = np.array([10,10,10])
a
```

Out[2]:
```
array([10, 10, 10])
```

In [3]:
```
# Checking Type
type(a)
```

Out[3]:
```
numpy.ndarray
```

In [4]:
```
# Length of 1-D Array
len(a)
```

Out[4]:
```
3
```

In [5]:
```
# Indexing
a[0]
```

Out[5]:
```
10
```

In [6]:
```
a[2]
```

Out[6]:
```
10
```

In [7]:
```
a[0:]
```

Out[7]:
```
array([10, 10, 10])
```

2-D Array

In [8]:
```
# Two-dimensional Array

b = np.array([[1,1,1], [5,5,5], [3,3,3]])
b
```

Out[8]:
```
array([[1, 1, 1],
       [5, 5, 5],
       [3, 3, 3]])
```

In [9]:
```
# Length of 2-D Array

len(b)
```

Out[9]:
```
3
```

In [10]:
```
# Indexing

b[0:]
```

Out[10]:
```
array([[1, 1, 1],
       [5, 5, 5],
       [3, 3, 3]])
```

In [11]:
```
b[1:3]
```

Out[11]:
```
array([[5, 5, 5],
       [3, 3, 3]])
```

In [12]:
```
b[2,1:3]
```

Out[12]:
```
array([3, 3])
```

In [13]:
```
b[1:,1:]
```

Out[13]:
```
array([[5, 5],
       [3, 3]])
```

In [14]:
```
# Zero Matrix

zero_array = np.zeros([3,3])
zero_array
```

Out[14]:
```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

In [15]:
```
# Unit Matrix

unit_array = np.ones([2,2])
unit_array
```

Out[15]:
```
array([[1., 1.],
       [1., 1.]])
```

In [16]:
```
# Empty Maatrix

empty_array = np.empty([2,2])
empty_array
```

Out[16]:
```
array([[1., 1.],
       [1., 1.]])
```

Arange Function

In [17]:
```
# Arange function - array with specific range of elements from 0 to specified range

x = np.arange(9)
x
```

Out[17]:
```
array([0, 1, 2, 3, 4, 5, 6, 7, 8])
```

In [18]:
```
# Arange function - array with specific range of elements from one specified range to a

y = np.arange(10,19)
y
```

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

In [19]:
```
# Arange function with step size

z = np.arange(1,15,3)
z
```

Out[19]:
```
array([ 1,  4,  7, 10, 13])
```

In [20]:
```
# Linearly spaced arrays - equally spaced array from one specified range to another spe

s = np.linspace(2,20,4)
s
```

Out[20]:
```
array([ 2.,  8., 14., 20.])
```

In [21]:
```
# specific data type array - int data type

c = np.ones(4, dtype=np.int8)
c
```

Out[21]:
```
array([1, 1, 1, 1], dtype=int8)
```

In [22]:
```
# specific data type array - float data type

d = np.ones(4, dtype=np.float64)
d
```

Out[22]:
```
array([1., 1., 1., 1.])
```

3-D Array

In [23]:

```
f = np.arange(24).reshape(2, 3, 4)
f
```

Out[23]:
```
array([[[ 0,  1,  2,  3],
        [ 4,  5,  6,  7],
        [ 8,  9, 10, 11]],

       [[12, 13, 14, 15],
        [16, 17, 18, 19],
        [20, 21, 22, 23]]])
```

Practicing Numpy Library

In [24]:
```
price = np.array([23,45,67,86])
price.mean()
```

Out[24]:
```
55.25
```

In [25]:
```
# Table of 9

table = np.arange(0,91,9)
table
```

Out[25]:
```
array([ 0,  9, 18, 27, 36, 45, 54, 63, 72, 81, 90])
```

Array Functions

1-D Array

In [26]:
```
h = np.array([1,67,98,546,23,987,33])
h
```

Out[26]:
```
array([  1,  67,  98, 546,  23, 987,  33])
```

In [27]:
```
# Sort Function

h.sort()
h
```

Out[27]:
```
array([  1,  23,  33,  67,  98, 546, 987])
```

In [28]:
```
# all function

h.all()
```

Out[28]:
```
True
```

In [29]:
```
# Copy function
```

```
g = h.copy()
g
```

Out[29]:
```
array([  1,  23,  33,  67,  98, 546, 987])
```

In [30]:
```
# dtype

h.dtype
```

Out[30]:
```
dtype('int32')
```

In [31]:
```
# Concatenation

x = np.array([2,4,5,98,23])
y = np.array([45,22,98,10,1])

z = np.concatenate((x,y))
z
```

Out[31]:
```
array([ 2,  4,  5, 98, 23, 45, 22, 98, 10,  1])
```

2-D Array

In [32]:
```
# Concatenate

a = np.array([[1,2,3], [6,7,8]])
b = np.array([[45,87,22], [56,78,90]])

c = np.concatenate((a,b), axis=0)
c
```

Out[32]:
```
array([[ 1,  2,  3],
       [ 6,  7,  8],
       [45, 87, 22],
       [56, 78, 90]])
```

In [33]:
```
np.concatenate((a,b), axis=1)
```

Out[33]:
```
array([[ 1,  2,  3, 45, 87, 22],
       [ 6,  7,  8, 56, 78, 90]])
```

3-D Array

In [34]:
```
x = np.array([[[1,2,3,4],[5,6,7,8]], [[1,2,3,4],[5,6,7,8]]])
x
```

Out[34]:
```
array([[[1, 2, 3, 4],
        [5, 6, 7, 8]],

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

In [35]:
```
# ndim attribute - Finding no. of dimensions
```

```
x.ndim
```

Out[35]: 3

In [36]:
```
# Size attribute - finding no. of elements in an array

x.size
```

Out[36]: 16

In [37]:
```
# Shape attribute

x.shape
```

Out[37]: (2, 2, 4)

In [38]:
```
a = np.arange(6)
a
```

Out[38]: array([0, 1, 2, 3, 4, 5])

In [39]:
```
# Resize function - changes any dimensional matrix into another dimensional matrix

a.resize(3,2)
a
```

Out[39]:
```
array([[0, 1],
       [2, 3],
       [4, 5]])
```

In [40]:
```
a = np.arange(6)
a
```

Out[40]: array([0, 1, 2, 3, 4, 5])

In [41]:
```
# Row-wise 2D

b = a[np.newaxis, :]
b
```

Out[41]: array([[0, 1, 2, 3, 4, 5]])

In [42]:
```
# Column-wise 2D

c = a[:, np.newaxis]
c
```

Out[42]:
```
array([[0],
       [1],
       [2],
```

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

In [43]:
```
a
```

Out[43]:
```
array([0, 1, 2, 3, 4, 5])
```

In [44]:
```
a*6
```

Out[44]:
```
array([ 0,  6, 12, 18, 24, 30])
```

In [45]:
```
a+6
```

Out[45]:
```
array([ 6,  7,  8,  9, 10, 11])
```

In [46]:
```
# Sum Function

a.sum()
```

Out[46]:
```
15
```