Numpy Library

How to use

In [1]:
```
import numpy as np
```

How it looks like?

In [2]:
```
a = np.arange(6)
a2 = a[np.newaxis, :]
a2.shape
```

Out[2]:
```
(1, 6)
```

Whatni s an array?

In [7]:
```
import numpy as np
a = np.array([5,5,5,5])
a
```

Out[7]:
```
array([5, 5, 5, 5])
```

Examples:

In [8]:
```
import numpy as np
a = np.array([5,5,5])
a
```

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

In [9]:
```
type(a)
```

Out[9]:
```
numpy.ndarray
```

In [10]:
```
# lists of lists
b = np.array([[5,5,5],[5,5,5],[5,5,5]])
b
```

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

Vector?

A vector is an array with a single dimension (there is no difference b/w row and column vectors)

Matrix?

A matrix refers to an array with two dimensions

1-D Array:

In [12]:
```
import numpy as np
a = np.array([3,3,3])
a
```

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

In [13]:
```
type(a)
```

Out[13]:
```
numpy.ndarray
```

In [14]:
```
len(a)
```

Out[14]:
```
3
```

In [15]:
```
a[0]
```

Out[15]:
```
3
```

In [17]:
```
a[0:]
```

Out[17]:
```
array([3, 3, 3])
```

2-D Array:

The NumPy ndarray class is used to represent both matrices and vectors.

In [23]:
```
# lists of lists
b = np.array([[5,5,5],[3,3,5],[5,5,5]])
b
```

Out[23]:
```
array([[5, 5, 5],
       [3, 3, 5],
       [5, 5, 5]])
```

In [24]:
```
type(b)
```

Out[24]:
```
numpy.ndarray
```

In [25]:
```
len(b)
```

Out[25]:
```
3
```

In [29]:
```
# Indexing in array
b[0]
```

Out[29]:
```
array([3, 3, 5])
```

In [28]:
```
b[1]
```

Out[28]:
```
array([3, 3, 5])
```

In [ ]:
```

```

In [30]:
```
b[0:]
```

Out[30]:
```
array([[5, 5, 5],
       [3, 3, 5],
       [5, 5, 5]])
```

3-D or higher:

Tensor?

For 3-D or higher dimensional arrays, the term tensor is also commonly used

Array attributes:

- Dimensions are called axis

In [31]:
```
c = np.array([[3,3,3],[4,4,4]])
c
```

Out[31]:
```
array([[3, 3, 3],
       [4, 4, 4]])
```

2-axis

- First axis has length = 2
- Second axis has length = 3

How to create an array?

- ## 1-D Array ## Examples

In [32]:
```
import numpy as np
a = np.array([1,2,3,4,5])
a
```

Out[32]:
```
array([1, 2, 3, 4, 5])
```

In [33]:
```
b = np.zeros(2)
b
```

Out[33]:
```
array([0., 0.])
```

In [34]:
```
c = np.ones(3)
c
```

Out[34]:
```
array([1., 1., 1.])
```

In [37]:
```
# Create an empty array with 2 elements
d = np.empty(3)
d
```

Out[37]:
```
array([1., 1., 1.])
```

In [38]:
```
# With range of elements
e = np.arange(6)
e
```

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

In [39]:
```
# With specific range of elements
f = np.arange(3,20)
f
```

Out[39]:
```
array([ 3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19])
```

In [40]:
```
# Continue....
g = np.arange(3,30,3)
g
```

Out[40]:
```
array([ 3,  6,  9, 12, 15, 18, 21, 24, 27])
```

In [43]:
```
# Linerly spaced arrays
h = np.linspace(0,10,num=5) #gives use 5 nums
h
```

Out[43]:
```
array([ 0. ,  2.5,  5. ,  7.5, 10. ])
```

In [44]:
```
# Specific data types in array
i = np.ones(5, dtype=np.int8)
i
```

Out[44]:
```
array([1, 1, 1, 1, 1], dtype=int8)
```

In [54]:
```
j = np.ones(4, dtype=np.int16)
j
```

Out[54]:
```
array([1, 1, 1, 1], dtype=int16)
```

In [53]:
```
k = np.ones(5, dtype=np.float32)
k
```

Out[53]:
```
array([1., 1., 1., 1., 1.], dtype=float32)
```

How to create an array

- ## 2-D Array ## Examples

In [56]:
```
import numpy as np
x = np.zeros((3,5))
x
```

Out[56]:
```
array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.]])
```

In [68]:
```
y = np.ones((6,4))
y
```

Out[68]:
```
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.]])
```

How to create an array

- ## 3-D Array ## Examples

In [69]:
```
# Making and reshaping a 3D array
l = np.arange(24). reshape(2,3,4)
l
```

Out[69]:
```
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]]])
```

In [ ]:

In [ ]: