Numpy Libarray

There are Three types of arrays in numpy

- 1D Array
- 2D Array
- 3D Array or higher

1D Array

- one dimensional array is know as 1D array
- It is also called a vector
- There is no difference between rows and columns

In [1]:
```
# 1D Array through numpy

# Import libarray
import numpy as np

# array difing in numpy
a = np.array([1,2,3,4,5,6,])
a
```

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

In [2]:
```
type(a)
```

Out[2]:
```
numpy.ndarray
```

In [3]:
```
len(a)
```

Out[3]:
```
6
```

In [4]:
```
a[0]
```

Out[4]:
```
1
```

In [5]:
```
a[5]
```

Out[5]:
```
6
```

In [6]:
```
a[0:]
```

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

In [7]:
```
# 2nd method of creating an array
b = np.zeros(3)
b
```

Out[7]:
```
array([0., 0., 0.])
```

In [8]:
```
# 3rd method of creating an array
c = np.ones(3)
c
```

Out[8]:
```
array([1., 1., 1.])
```

In [9]:
```
# 4th method of creating an array
d = np.empty(3)
d
```

Out[9]:
```
array([1., 1., 1.])
```

In [10]:
```
# other methods in 1D arrays
# Making arrays with given range
e = np.arange(20)
e
# last element is exclusive in arrays
```

Out[10]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19])
```

In [11]:
```
# creating arrays with specific ranges
f = np.arange(5,30)
f
```

Out[11]:
```
array([ 5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21,
       22, 23, 24, 25, 26, 27, 28, 29])
```

In [12]:
```
# Creating arrays with specific intervals
g = np.arange(3,30,3)
g
```

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

In [13]:
```
# Linearly placed elements
i = np.linspace(1,30,num=6)
i
```

Out[13]:
```
array([ 1. ,  6.8, 12.6, 18.4, 24.2, 30. ])
```

In [14]:
```
# creating arrays with specific data types
j = np.ones(5,dtype=np.int8)
j
```

Out[14]:
```
array([1, 1, 1, 1, 1], dtype=int8)
```

In [15]:
```
k = np.ones(5,dtype=np.int16)
k
```

Out[15]:
```
array([1, 1, 1, 1, 1], dtype=int16)
```

In [16]:
```
l = np.ones(5,dtype=np.int32)
l
```

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

In [17]:
```
m = np.ones(5,dtype=np.int64)
m
```

Out[17]:
```
array([1, 1, 1, 1, 1], dtype=int64)
```

In [18]:
```
n = np.ones(5,dtype=np.float16)
n
```

Out[18]:
```
array([1., 1., 1., 1., 1.], dtype=float16)
```

In [19]:
```
o = np.ones(5,dtype=np.float32)
o
```

Out[19]:
```
array([1., 1., 1., 1., 1.], dtype=float32)
```

In [20]:
```
p = np.ones(5,dtype=np.float64)
p
```

Out[20]:
```
array([1., 1., 1., 1., 1.])
```

2D Array

- 2D array is known as two dimensional array
- it is also called matrix

In [21]:
```
# import libraray
import numpy as np

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

Out[21]:
```
array([[1, 2, 3],
       [2, 3, 4],
       [3, 4, 5]])
```

In [22]:
```
type(a)
```

Out[22]:
```
numpy.ndarray
```

In [23]:
```
len(a)
```

Out[23]:
```
3
```

In [24]:
```
a[0]
```

Out[24]:
```
array([1, 2, 3])
```

In [25]:
```
a[0][1]
```

Out[25]:
```
2
```

In [26]:
```
a[2][2]
```

Out[26]:
```
5
```

In [27]:
```
a[0][0:]
```

Out[27]:
```
array([1, 2, 3])
```

In [28]:
```
a[0:][1][1]
```

Out[28]:
```
3
```

In [29]:
```
a[0:]
```

Out[29]:
```
array([[1, 2, 3],
       [2, 3, 4],
       [3, 4, 5]])
```

In [30]:
```
q= np.zeros((2,4))
q
```

Out[30]:
```
array([[0., 0., 0., 0.],
       [0., 0., 0., 0.]])
```

In [31]:
```
r= np.ones((5,7))
r
```

Out[31]:
```
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., 1., 1., 1., 1., 1.]])
```

In [32]:
```
s= np.empty((3,4))
s
```

Out[32]:
```
array([[1.38175465e-311, 3.16202013e-322, 0.00000000e+000,
        0.00000000e+000],
       [1.11260619e-306, 2.23188620e+180, 3.79913445e+175,
        1.46142358e+185],
       [2.27285401e+184, 1.24561420e-047, 7.24776416e+169,
        1.83025367e-076]])
```

3D Arrays

In [33]:
```
a = np.arange(24).reshape(2,3,4)
a
```

Out[33]:
```
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 [34]:
```
type(a)
```

Out[34]:
```
numpy.ndarray
```