Functions to create ndarrays

Usefull for AI projects

np.zeros(shape)----------

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

In [5]:
```
X = np.zeros((3,4))
print(X)
print("x object type:",type(X))
print("x have data type:", X.dtype)
print("x has shape", X.shape)
```

```
[[0. 0. 0. 0.]
 [0. 0. 0. 0.]
 [0. 0. 0. 0.]]
x object type: <class 'numpy.ndarray'>
x have data type: float64
x has shape (3, 4)
```

np.ones(shape)----------

In [6]:
```
X = np.ones((3,4))
print(X)
print("x object type:",type(X))
print("x have data type:", X.dtype)
print("x has shape", X.shape)
```

```
[[1. 1. 1. 1.]
 [1. 1. 1. 1.]
 [1. 1. 1. 1.]]
x object type: <class 'numpy.ndarray'>
x have data type: float64
x has shape (3, 4)
```

np.full((shape),5)

In [7]:
```
X = np.full((3,4),5)
print(X)
```

```
[[5 5 5 5]
 [5 5 5 5]
 [5 5 5 5]]
```

Indentity Matrix .eye(5) ----Linear Algebra

In [8]:
```
X = np.eye(5)
print(X)
```

```
[[1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0.]
 [0. 0. 1. 0. 0.]
 [0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 1.]]
```

np.diag([10,20,30,40])----------

In [11]:
```
X = np.diag([10,20,30,40])
print(X)
```

```
[[10  0  0  0]
 [ 0 20  0  0]
```

```
[ 0  0 30  0]
[ 0  0  0 40]]
```

arrange(start,stop,step)

np.arange() ---------- one arguement

In [3]:
```
x = np.arange(10)
print(x)
```

```
[0 1 2 3 4 5 6 7 8 9]
```

np.arrange(start,stop) ---------- 2 arguement

In [15]:
```
x = np.arange(4,10)
print(x)
```

```
[4 5 6 7 8 9]
```

np.arrange(start,stop,steps) ---------- 3 arguement

In [18]:
```
x = np.arange(1,30,2)
print(x)
```

```
[ 1  3  5  7  9 11 13 15 17 19 21 23 25 27 29]
```

non integer steps np.linspace(start,stop,N<---not steps,endpoint = False)

In [4]:
```
x = np.linspace(0,25,10)
```

```
print('x = \n', x)
x = np.linspace(0,25,10, endpoint = False)
print('x = ', x)
```

```
x =
 [ 0.          2.77777778  5.55555556  8.33333333 11.11111111 13.888888
89
 16.66666667 19.44444444 22.22222222 25.        ]
x =  [ 0.   2.5  5.   7.5 10.  12.5 15.  17.5 20.  22.5]
```

.reshape()

In [2]:
```
Y = np.arange(20).reshape(-1, 5)
print(Y)
X = np.linspace(0,50,10, endpoint=False).reshape(5,2)
print(X)
```

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]
[[ 0.  5.]
 [10. 15.]
 [20. 25.]
 [30. 35.]
 [40. 45.]]
```

.random.random(shape), np.random.randint(start, stop, size = shape)

In [3]:
```
X = np.random.random((3,3))
print('X = \n', X)
X = np.random.randint(4,15,size=(3,2))
print('X = \n', X)
```

```
X =
 [[0.65162111 0.07898379 0.40552787]
 [0.39490045 0.69297374 0.94276345]
 [0.59691184 0.41291959 0.79232599]]
X =
 [[ 8  4]
 [ 7 10]
 [14  4]]
```

.random.normal(mean, sd, size=(x,y))

In [20]:
```
X = np.random.normal(0, 0.1, size=(5,5))
print(X)
```

```
[[-0.13978359  0.07410743 -0.06034441 -0.03401122 -0.02604417]
 [-0.02058213 -0.13463805  0.05012651  0.00294888 -0.10147499]
 [ 0.03941023  0.0157028   0.00028625  0.0029958  -0.07734968]
 [ 0.04267798 -0.1556809   0.08867821 -0.09194726 -0.03287841]
 [-0.0656284  -0.11098516  0.06804602  0.02815427 -0.04598979]]
```