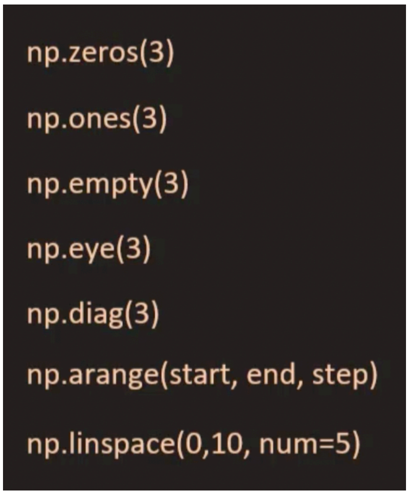
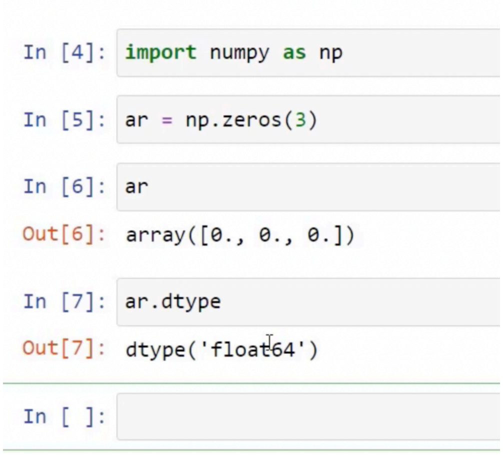
Creating an Array

- Functions using which we can create different type of arrays



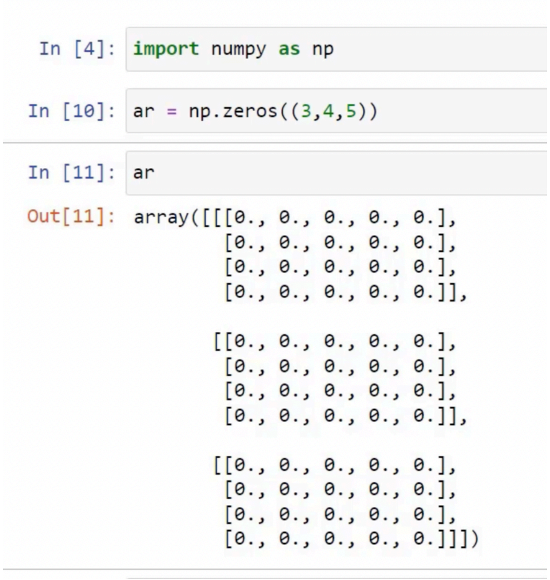
```
np.zeros(3)
np.ones(3)
np.empty(3)
np.eye(3)
np.diag(3)
np.arange(start, end, step)
np.linspace(0,10, num=5)
```

- Creating 1D array using **np.zeros( )** in Jupiter notebook



```
In [4]: import numpy as np

In [5]: ar = np.zeros(3)

In [6]: ar
Out[6]: array([0., 0., 0.])

In [7]: ar.dtype
Out[7]: dtype('float64')

In [ ]:
```

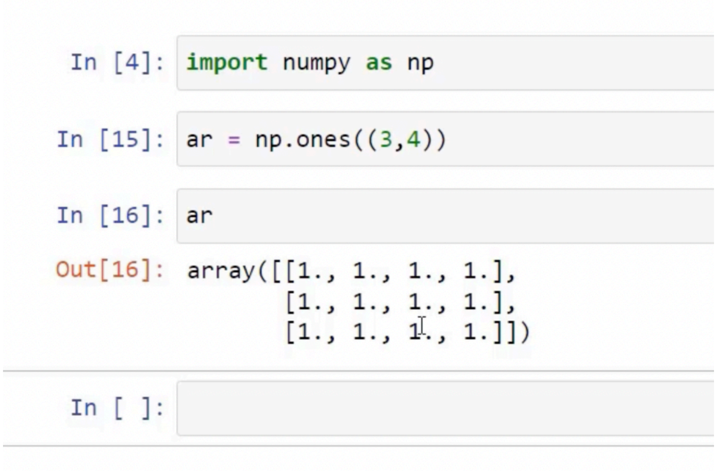
Creating 2D arrays using **np.zeros( )**



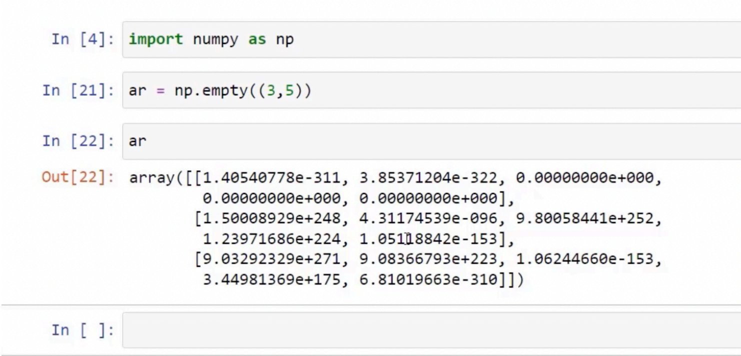
```
In [4]:  import numpy as np

In [10]: ar = np.zeros((3,4,5))

In [11]: ar

Out[11]: array([[[0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.]],

                [[0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.]],

                [[0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.],
                 [0., 0., 0., 0., 0.]]])
```

- Just like zeros we have **np.ones( )** where every element in an array are filled with ones

- **np.empty( )** - the elements inside the array can be anything its also called garbage values

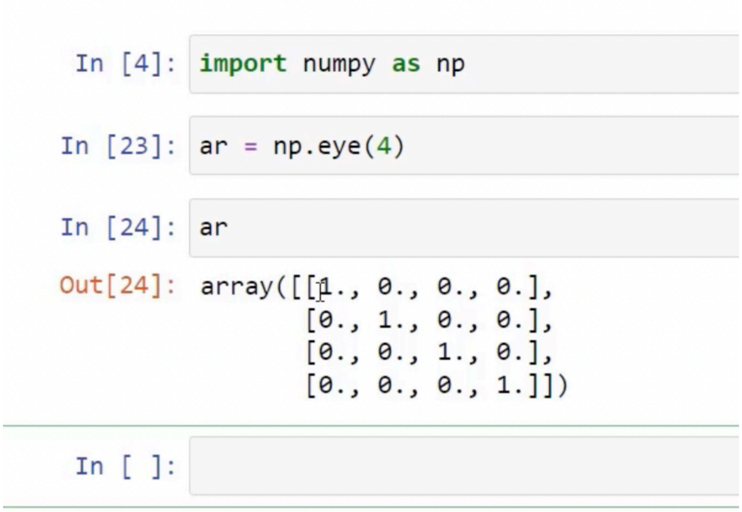


```
In [4]:  import numpy as np

In [15]: ar = np.ones((3,4))

In [16]: ar

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

In [ ]:
```



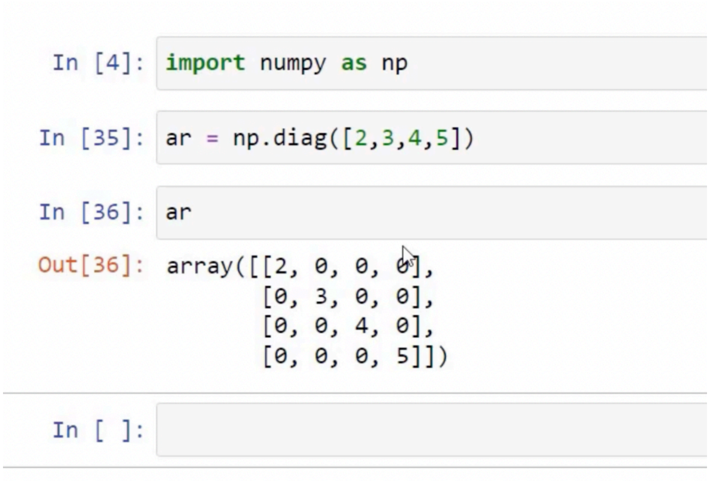
```
In [4]:  import numpy as np

In [21]: ar = np.empty((3,5))

In [22]: ar

Out[22]: array([[1.40540778e-311, 3.85371204e-322, 0.00000000e+000,
                 0.00000000e+000, 0.00000000e+000],
                [1.50008929e+248, 4.31174539e-096, 9.80058441e+252,
                 1.23971686e+224, 1.05118842e-153],
                [9.03292329e+271, 9.08366793e+223, 1.06244660e-153,
                 3.44981369e+175, 6.81019663e-310]])

In [ ]:
```

- **np.eye( )** - used for creating identity values where diagonal values in a matrix is 1 , only 1 parameter must be given in this method



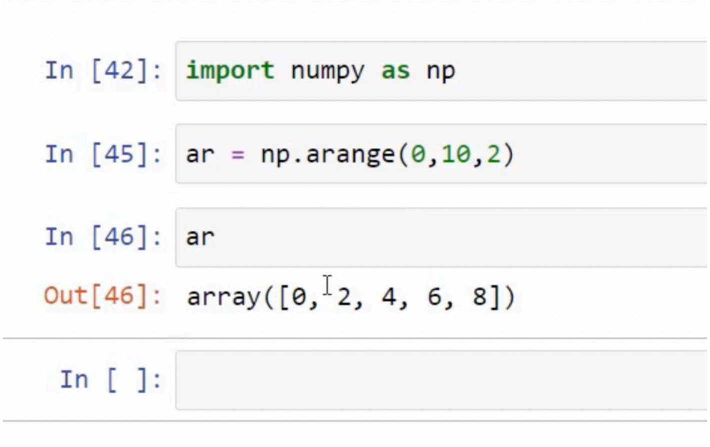
```
In [4]:  import numpy as np

In [23]: ar = np.eye(4)

In [24]: ar

Out[24]: array([[1., 0., 0., 0.],
                [0., 1., 0., 0.],
                [0., 0., 1., 0.],
                [0., 0., 0., 1.]])

In [ ]:
```

- **np.diag([2,3])** - it'll create a matrix of 2x2



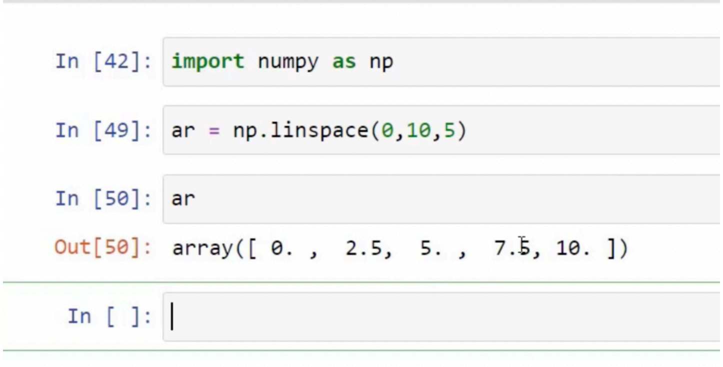
```
In [4]: import numpy as np

In [35]: ar = np.diag([2,3,4,5])

In [36]: ar
Out[36]: array([[2, 0, 0, 0],
               [0, 3, 0, 0],
               [0, 0, 4, 0],
               [0, 0, 0, 5]])

In [ ]:
```

- **np.arange(start : end : step)** - same as range function



```
In [42]: import numpy as np

In [45]: ar = np.arange(0,10,2)

In [46]: ar
Out[46]: array([0, 2, 4, 6, 8])

In [ ]:
```

- **np.linspace(0 ,10 , num = 5)** - fill elements between 0 to num = 5 in equal difference till 10



```
In [42]: import numpy as np

In [49]: ar = np.linspace(0,10,5)

In [50]: ar
Out[50]: array([ 0. ,  2.5,  5. ,  7.5, 10. ])

In [ ]:
```