Numpy Exercise

Import NumPy as np

In [2]:

```
import numpy as np
```

Create an array of 10 zeros

In [4]:

```
np.zeros(10)
```

Out[4]:

```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

Create an array of 10 ones

In [6]:

```
np.ones(10)
```

Out[6]:

```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

Create an array of 10 fives

In [12]:

```
np.linspace(5,5,10)
```

Out[12]:

```
array([5., 5., 5., 5., 5., 5., 5., 5., 5., 5.])
```

Create an array of the integers from 10 to 50

In [14]:

```
np.arange(10,51)
```

Out[14]:

```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26,
       27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43,
       44, 45, 46, 47, 48, 49, 50])
```

Create an array of all the even integers from 10 to 50

In [17]:

```
np.arange(10,52,2)
```

Out[17]:

```
array([10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 42,
       44, 46, 48, 50])
```

Create a 3x3 matrix with values ranging from 0 to 8

In [21]:

```
A=np.arange(0,9)
A.reshape(3,3)
```

Out[21]:

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

Create a 3x3 identity matrix

In [22]:

```
np.eye(3,3)
```

Out[22]:

```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

Use NumPy to generate a random number between 0 and 1

In [30]:

```
np.random.rand(1,1)
```

Out[30]:

```
array([[0.63119974]])
```

In [15]:

```

```

Out[15]:

```
array([ 0.42829726])
```

Use NumPy to generate an array of 25 random numbers sampled from a standard normal distribution

In [31]:

```
np.random.randn(5,5)
```

Out[31]:

```
array([[-0.62001405,  0.37733677, -1.77829401, -0.32372951, -0.39016995],
       [ 0.88860484, -0.11512877,  0.11142431, -1.36514653,  0.77395549],
       [-1.65108094, -1.44842979, -1.16231313,  1.30193128,  0.59889972],
       [ 0.6443392 ,  1.16660549, -2.04397058,  0.61743326, -1.55992685],
       [-2.06102005, -0.81764901,  0.65583648,  1.64180421, -0.12455133]])
```

In [33]:

```
np.random.randn(1,25)
```

Out[33]:

```
array([[-1.08963311,  0.34723399,  1.12752067,  1.14672805, -1.35357761,
         0.77824012, -0.6048839 ,  1.49461315,  0.12396275, -0.02837432,
         0.93330373, -0.00922692,  1.06009981,  0.77693176, -0.61514033,
        -0.39122002, -0.27716127, -0.97001869,  1.51569025, -0.49635166,
         0.02620545, -1.44394347, -0.38838703,  0.23181131, -1.01061814]])
```

Create the following matrix:

In [39]:

```
Z=np.linspace(0.01,1,100)
Z.reshape(10,10)
```

Out[39]:

```
array([[0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 ],
       [0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 ],
       [0.21, 0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 ],
       [0.31, 0.32, 0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 ],
       [0.41, 0.42, 0.43, 0.44, 0.45, 0.46, 0.47, 0.48, 0.49, 0.5 ],
       [0.51, 0.52, 0.53, 0.54, 0.55, 0.56, 0.57, 0.58, 0.59, 0.6 ],
       [0.61, 0.62, 0.63, 0.64, 0.65, 0.66, 0.67, 0.68, 0.69, 0.7 ],
       [0.71, 0.72, 0.73, 0.74, 0.75, 0.76, 0.77, 0.78, 0.79, 0.8 ],
       [0.81, 0.82, 0.83, 0.84, 0.85, 0.86, 0.87, 0.88, 0.89, 0.9 ],
       [0.91, 0.92, 0.93, 0.94, 0.95, 0.96, 0.97, 0.98, 0.99, 1.  ]])
```

Create an array of 20 linearly spaced points between 0 and 1:

In [44]:

```
np.random.rand(4,5)
```

Out[44]:

```
array([[0.91635299, 0.33260941, 0.70520236, 0.64944823, 0.99852375],
       [0.48684355, 0.2249463 , 0.02196563, 0.29090065, 0.58315828],
       [0.40097379, 0.9858162 , 0.09855065, 0.3121439 , 0.19955377],
       [0.99931711, 0.44254847, 0.65728205, 0.98322314, 0.63184003]])
```

In [45]:

```
np.random.rand(1,20)
```

Out[45]:

```
array([[0.45561954, 0.71847214, 0.88789511, 0.77308773, 0.39598445,
        0.36238433, 0.04149921, 0.59024991, 0.04486927, 0.47805615,
        0.41933216, 0.76568844, 0.86534484, 0.20472473, 0.32148194,
        0.16002921, 0.49329103, 0.85519448, 0.35678598, 0.20035934]])
```

Numpy Indexing and Selection

Now you will be given a few matrices, and be asked to replicate the resulting matrix outputs:

In [83]:

```
B=np.arange(1,26).reshape(5,5)
B
```

Out[83]:

```
array([[ 1,  2,  3,  4,  5],
       [ 6,  7,  8,  9, 10],
       [11, 12, 13, 14, 15],
       [16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

In [39]:

```
# WRITE CODE HERE THAT REPRODUCES THE OUTPUT OF THE CELL BELOW
# BE CAREFUL NOT TO RUN THE CELL BELOW, OTHERWISE YOU WON'T
# BE ABLE TO SEE THE OUTPUT ANY MORE
```

In [55]:

```
B[2:,1:]
```

Out[55]:

```
array([[12, 13, 14, 15],
       [17, 18, 19, 20],
       [22, 23, 24, 25]])
```

In [40]:

```

```

Out[40]:

```
array([[12, 13, 14, 15],
       [17, 18, 19, 20],
       [22, 23, 24, 25]])
```

In [29]:

```
# WRITE CODE HERE THAT REPRODUCES THE OUTPUT OF THE CELL BELOW
# BE CAREFUL NOT TO RUN THE CELL BELOW, OTHERWISE YOU WON'T
# BE ABLE TO SEE THE OUTPUT ANY MORE
```

In [59]:

```
B[3,4]
```

Out[59]:

```
20
```

In [41]:

```

```

Out[41]:

```
20
```

In [30]:

```
# WRITE CODE HERE THAT REPRODUCES THE OUTPUT OF THE CELL BELOW
# BE CAREFUL NOT TO RUN THE CELL BELOW, OTHERWISE YOU WON'T
# BE ABLE TO SEE THE OUTPUT ANY MORE
```

In [86]:

```
a=B[:3,1]
a.reshape(3,1)
```

Out[86]:

```
array([[ 2],
       [ 7],
       [12]])
```

In [42]:

```

```

Out[42]:

```
array([[ 2],
       [ 7],
       [12]])
```

In [31]:

```
# WRITE CODE HERE THAT REPRODUCES THE OUTPUT OF THE CELL BELOW
# BE CAREFUL NOT TO RUN THE CELL BELOW, OTHERWISE YOU WON'T
# BE ABLE TO SEE THE OUTPUT ANY MORE
```

In [87]:

```
B[4]
```

Out[87]:

```
array([21, 22, 23, 24, 25])
```

In [46]:

```

```

Out[46]:

```
array([21, 22, 23, 24, 25])
```

In [32]:

```
# WRITE CODE HERE THAT REPRODUCES THE OUTPUT OF THE CELL BELOW
# BE CAREFUL NOT TO RUN THE CELL BELOW, OTHERWISE YOU WON'T
# BE ABLE TO SEE THE OUTPUT ANY MORE
```

In [89]:

```
B[3:]
```

Out[89]:

```
array([[16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

In [49]:

```

```

Out[49]:

```
array([[16, 17, 18, 19, 20],
       [21, 22, 23, 24, 25]])
```

Now do the following

Get the sum of all the values in mat

In [90]:

```
B.sum()
```

Out[90]:

```
325
```

In [50]:

```

```

Out[50]:

```
325
```

Get the standard deviation of the values in mat

In [91]:

```
B.std()
```

Out[91]:

```
7.211102550927978
```

In [51]:

```

```

Out[51]:

```
7.2111025509279782
```

Get the sum of all the columns in mat

In [96]:

```
B.sum(0)
```

Out[96]:

```
array([55, 60, 65, 70, 75])
```

In [53]:

```

```

Out[53]:

```
array([55, 60, 65, 70, 75])
```

Get the sum of all the rows in mat

In [98]:

```
B.sum(1)
```

Out[98]:

```
array([ 15,  40,  65,  90, 115])
```

In [ ]:

```

```