NumPy Exercises

Now that we've learned about NumPy let's test your knowledge. We'll start off with a few simple tasks, and then you'll be asked some more complicated questions.

Import NumPy as np

```
import numpy as np
```

Create an array of 10 zeros

```
np.zeros(shape=10)
```

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

Create an array of 10 ones

```
np.ones(shape=10)
```

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

Create an array of 10 fives

```
data=np.array([5,5,5,5,5,5,5,5,5,5])
data
```

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

Create an array of the integers from 10 to 50

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

```
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

```
np.arange(10,51,2)
```

```
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

```
number=np.matrix([[0,1,2],[3,4,5],[6,7,8]])
number
```

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

Create a 3x3 identity matrix

```
np.eye(N=3)
```

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

Use NumPy to generate a random number between 0 and 1

```
np.random.rand()
```

```
0.05964457068452089
```

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

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

```
array([-0.6945001 ,  0.01261385,  1.49698509, -1.04120497, -0.19256142,
       -0.40731924,  0.92792029,  0.07583119,  1.0479581 , -0.02039842,
        1.06387948, -1.09488633, -0.34737281,  1.05982548,  0.12238177,
        0.87078346, -0.80703977,  1.21654256,  0.80831637,  0.40094578,
       -0.01150907,  0.08423788, -1.79767958, -0.15942397, -1.77908978])
```

Create the following matrix:

```
np.arange(0.01,1.01,0.01)
```

```
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:

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

```
array([0.18632977, 0.05623977, 0.12082171, 0.46716668, 0.59017275,
       0.36095136, 0.98699068, 0.01362941, 0.2758801 , 0.39363261,
       0.49271713, 0.02212424, 0.46785922, 0.86637992, 0.38898455,
       0.97936984, 0.22634946, 0.10887895, 0.64734783, 0.43015628])
```

Numpy Indexing and Selection

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

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

```
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]])
```

```
num=np.arange(12,27).reshape(3,5)
np.delete(num,4,1)
num
```

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

```
num
```

```
array([[12, 13, 14, 15, 16],
       [17, 18, 19, 20, 21],
       [22, 23, 24, 25, 26]])
```

```
num[1,3]
```

```
20
```

```
num[1,3]
```

```
20
```

```
matrix=np.arange(1,16).reshape(3,5)
matrix[:,[1]]
```

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

```
matrix[:,[1]]
```

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

```
np.arange(21,26)
```

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

```
np.arange(21,26)
```

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

```
n=np.arange(16,26).reshape(2,5)
n
```

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

```
n
```

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

Now do the following

Get the sum of all the values in mat

```
mat.sum()
```

```
325
```

Get the standard deviation of the values in mat

```
mat.std()
```

```
7.2111025509279782
```

▼ Get the sum of all the columns in mat

```
mat.sum(axis=0)
```

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

Great Job!