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

```
arr_1 = np.zeros(dtype = float,shape = (1,10))
print(arr_1)
```

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

Create an array of 10 ones

```
arr_2 = np.ones(shape = (1,10),dtype = float)
print(arr_2)
```

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

Create an array of 10 fives

```
arr_3 = np.full((1,10),5,dtype = float)
print(arr_3)
```

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

Create an array of the integers from 10 to 50

```
arr_4 = np.arange(10,51)
print(arr_4)
```

```
[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

```
arr_5 = np.arange(10,51,2)
print(arr_5)
```

```
[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

```
arr_6 = np.array([[0,1,2],[3,4,5],[6,7,8]])
print(arr_6)
```

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

Create a 3x3 identity matrix

```
arr_7 = np.identity(3,dtype = float)
print(arr_7)
```

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

Use NumPy to generate a random number between 0 and 1

```
rand_num = np.random.normal(0,1,1)
print(rand_num)
```

```
[0.09775066]
```

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

```
std_num = np.random.normal(0,1,25)
print(std_num)
```

```
[ 1.94693926 -0.90049437  0.86552458  0.12448639 -0.39255627 -
1.39583907
  0.368249   -0.59952302  0.1221451  -0.01812651 -0.59001918
0.61066681
 -1.15785639 -0.60035939  0.71867982  1.86970412 -1.40948    -
0.62290185
 -0.88133983  0.25288967 -0.44892897 -0.35463382 -2.0761115
0.72117215
  0.29691084]
```

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

```
arr_8 = np.linspace(0,1,20)
print(arr_8)
```

```
[0.         0.05263158 0.10526316 0.15789474 0.21052632 0.26315789
 0.31578947 0.36842105 0.42105263 0.47368421 0.52631579 0.57894737
 0.63157895 0.68421053 0.73684211 0.78947368 0.84210526 0.89473684
 0.94736842 1.        ]
```

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]])
```

```
# 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
print(mat[2:,1:])
```

```
[[12 13 14 15]
 [17 18 19 20]
 [22 23 24 25]]
```

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

```
# 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
print(arr_9[1][3])
```

```
20
```

```
20
```

```
# 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
arr_9 = mat[0:3,1].reshape(3,1)
print(arr_9)
```

```
[[ 2]
 [ 7]
 [12]]
```

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

```
# 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
arr_10 = np.array([mat[4,:]])
print(arr_10)
```

```
[[21 22 23 24 25]]
```

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

```
# 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
arr_11 = np.array([mat[3:,:]])
print(arr_11)
```

```
[[[16 17 18 19 20]
  [21 22 23 24 25]]]
```

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

Now do the following

Get the sum of all the values in mat

```
print(mat.sum())
```

```
325
```

Get the standard deviation of the values in mat

```
print(mat.std())
```

```
7.211102550927978
```

Get the sum of all the columns in mat

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

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