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.

ASSIGNMENT(01/09/**2023**)

NAME:- T.Akshath Singh REGISTRATION NUMBER:- 21BCB7022

Import NumPy as np

```
import numpy as np
```

Create an array of 10 zeros

```
# Create an array of 10 zeros
array_zeros = np.zeros(10)
print("Array of 10 zeros:")
print(array_zeros)
```

```
Array of 10 zeros:
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
```

Create an array of 10 ones

```
# Create an array of 10 ones
array_ones = np.ones(10)
print("\nArray of 10 ones:")
print(array_ones)
```

```
Array of 10 ones:
[1. 1. 1. 1. 1. 1. 1. 1. 1. 1.]
```

Create an array of 10 fives

```
# Create an array of 10 fives
array_fives = np.ones(10) * 5
print("\nArray of 10 fives:")
print(array_fives)
```

```
Array of 10 fives:
[5. 5. 5. 5. 5. 5. 5. 5. 5. 5.]
```

Create an array of the integers from 10 to 50

```
# Create an array of integers from 10 to 50
array_integers = np.arange(10, 51)
print("\nArray of integers from 10 to 50:")
print(array_integers)
```

```
Array of integers from 10 to 50:
[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

```
# Create an array of all the even integers from 10 to 50
array_even = np.arange(10, 51, 2)
print("\nArray of even integers from 10 to 50:")
print(array_even)
```

```
Array of even integers from 10 to 50:
[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

```
# Create a 3x3 matrix with values ranging from 0 to 8
matrix = np.arange(9).reshape(3, 3)
print("\n3x3 Matrix with values from 0 to 8:")
print(matrix)
```

```
3x3 Matrix with values from 0 to 8:
[[0 1 2]
 [3 4 5]
 [6 7 8]]
```

Create a 3x3 identity matrix

```
# Create a 3x3 identity matrix
identity_matrix = np.eye(3)
print("\n3x3 Identity Matrix:")
print(identity_matrix)
```

```
3x3 Identity Matrix:
[[1. 0. 0.]
 [0. 1. 0.]
 [0. 0. 1.]]
```

Use NumPy to generate a random number between 0 and 1

```
random_number = np.random.rand()
print("\nRandom Number between 0 and 1:")
print(random_number)
```

```
Random Number between 0 and 1:
0.7092453760005708
```

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

```
random_numbers = np.random.randn(25)
print("\nArray of 25 random numbers from a standard normal distribution:")
print(random_numbers)
```

```
Array of 25 random numbers from a standard normal distribution:
[-1.11426856e+00 -5.84548390e-01  1.06576526e+00  1.71234286e-01
 -1.54258201e+00  5.60393465e-02 -2.80022483e+00 -1.22715826e-01
  2.48725024e+00 -2.09970140e-01  2.84198317e-01  1.34320329e+00
 -1.47773796e+00 -4.44594242e-01  8.22639578e-01  2.75727285e+00
  2.17929751e-04 -9.92810158e-01 -2.74903820e-01 -1.02251898e+00
 -1.21112897e+00 -1.30287269e-01  5.85375665e-01 -5.49024175e-01
  2.79106775e-01]
```

Create the following matrix:

```
custom_matrix = np.arange(0.01, 1.01, 0.01).reshape(10, 10)
print(custom_matrix)
```

```
[[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:

```
linearly_spaced_points = np.linspace(0, 1, 20)
print("\nArray of 20 linearly spaced points between 0 and 1:")
print(linearly_spaced_points)
```

```
Array of 20 linearly spaced points between 0 and 1:
[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
```

```
submatrix1 = mat[2:6, 1:6]
print(submatrix1)
```

```
[[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
```

```
element2 = mat[3:6, 4:6]
print(element2)
```

```
[[20]
 [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
```

```
column3 = mat[0:3, 1:2]
print( column3)
```

```
[[ 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
```

```
row4 = mat[4:6, 0:6]
print(row4)
```

```
[[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
```

```
submatrix5 = mat[3:6,0:6]
print( submatrix5)
```

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

Now do the following

Get the sum of all the values in mat

```
total_sum = np.sum(mat)
print("Sum of all values in mat:", total_sum)
```

```
Sum of all values in mat: 325
```

Get the standard deviation of the values in mat

```
mat = np.arange(1, 26).reshape(5, 5)
std_deviation = np.std(mat)
print("Standard Deviation of values in mat:", std_deviation)
```

```
Standard Deviation of values in mat: 7.211102550927978
```

Get the sum of all the columns in mat

```
column_sums = np.sum(mat, axis=0)
print("Sum of each column in mat:")
print(column_sums)
```

```
Sum of each column in mat:
[55 60 65 70 75]
```

Double-click (or enter) to edit