ASSIGNMENT 1

GUDLA NISHIKA (21BCE8562)

VIT-AP

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

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

Create an array of 10 zeros

In [ ]:
```
zeros_array = np.zeros(10)
zeros_array
```

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

Create an array of 10 ones

In [ ]:
```
ones_array = np.ones(10)
ones_array
```

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

Create an array of 10 fives

In [ ]:
```
ones_array1 = np.ones(10)*5
ones_array1
```

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

Create an array of the integers from 10 to 50

In [ ]:
```
integers_array = np.arange(10,51)
integers_array
```

Out[ ]:
```
array([10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 2
6,
       27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 4
3,
       44, 45, 46, 47, 48, 49, 50])
```

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

In [ ]:
```
integers_array1 = np.arange(10,51,2)
integers_array1
```

Out[ ]:
```
array([10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36, 38, 40, 4
2,
       44, 46, 48, 50])
```

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

In [ ]:
```
array_matrix = np.arange(9)
matrix_33 = array_matrix.reshape(3,3)
matrix_33
```

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

Create a 3x3 identity matrix

In [ ]:
```
identity_matrix = np.eye(3)
identity_matrix
```

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

Use NumPy to generate a random number between 0 and 1

In [ ]:
```
random_number = np.random.rand()
random_number
```

Out[ ]:
```
0.18617222289740742
```

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

In [ ]:
```
random_number = np.random.randn(25)
random_number
```

Out[ ]:
```
array([-1.13091346, -0.8080517 , -1.14623874,  0.96549251,  1.95026429,
        0.9329296 ,  1.61539689,  0.04596843, -0.62935243,  0.45881684,
        1.10574502, -0.8005973 , -0.70559908,  1.16304631, -0.47592185,
       -2.15877943, -1.38837564,  1.21796113, -0.90667442, -2.09437498,
       -1.13557517,  0.84879267, -1.32424904,  1.03595476, -1.84206134])
```

Create the following matrix:

In [ ]:
```
matrix = np.linspace(0.01,1.0,100)
nishi = matrix.reshape(10,10)
nishi
```

Out[ ]:
```
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 [ ]:
```
linearly_spaced_points = np.linspace(0,1,20)
linearly_spaced_points
```

Out[ ]:
```
array([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:

In [ ]:
```
mat = np.arange(1,26).reshape(5,5)
mat
```

Out[ ]:
```
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 [ ]:
```
# 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 [ ]:
```
mat[2:5,1:5]
```

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

In [ ]:
```
# 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 [ ]:
```
print(mat[3,4])
```

```
20
```

In [ ]:
```
# 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 [ ]:
```
mat[0:3,1:2]
```

Out[ ]:
```
array([[ 2],
       [ 7],
```

```
       [12]])
```

In [ ]:
```
# 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 [ ]:
```
mat[4:]
```

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

In [ ]:
```
# 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 [ ]:
```
mat[3:]
```

Out[ ]:
```
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 [ ]:
```
matrix_sum = np.sum(mat)
print(matrix_sum)
```

```
325
```

Get the standard deviation of the values in mat

In [ ]:
```
matrix_std = np.std(mat)
print(matrix_std)
```

```
7.211102550927978
```

Get the sum of all the columns in mat

In [ ]:
```
column_sum = np.sum(mat, axis=0)
column_sum
```

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

If we write print(func), it'll give output without writing "array" infront.