Name: Bylapudi Lahari

Email: lahari.21bce9969@vitapstudent.ac.in

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 [1]:
```
import numpy as np
```

Create an array of 10 zeros

In [4]:
```
arr_zeros = np.zeros(10)
arr_zeros
```

Out[4]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

Create an array of 10 ones

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

Out[5]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

Create an array of 10 fives

In [8]:
```
arr_fives = np.array([5.0] * 10)
arr_fives
```

Out[8]:
```
array([5., 5., 5., 5., 5., 5., 5., 5., 5., 5.])
```

Create an array of the integers from 10 to 50

In [9]:
```
arr_integers = np.arange(10, 51)
arr_integers
```

Out[9]:
```
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 a 3x3 matrix with values ranging from 0 to 8

```
In [10]: matrix = np.arange(9).reshape(3, 3)
         matrix
```

```
Out[10]: array([[0, 1, 2],
                [3, 4, 5],
                [6, 7, 8]])
```

Create a 3x3 identity matrix

```
In [11]: matrix_iden = np.eye(3)
         matrix_iden
```

```
Out[11]: array([[1., 0., 0.],
                [0., 1., 0.],
                [0., 0., 1.]])
```

Use NumPy to generate a random number between 0 and 1

```
In [15]: random_number = np.array([np.random.rand()])
         random_number
```

```
Out[15]: array([0.05931698])
```

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

```
In [19]: random_numbers = np.random.randn(25)
         random_numbers
```

```
Out[19]: array([-0.62544448, -2.15301665,  1.4405828 ,  0.35149619,  1.16367311,
                -0.18761184,  0.75435611,  1.26546578, -0.92423696,  0.23267393,
                -0.00747242,  0.07083847,  0.01906714, -0.15384276,  0.77826503,
                 1.32083839,  2.56865306,  0.521638  , -1.72773365, -0.02361155,
                 0.10239908,  0.04228238, -0.78780692,  0.36382259,  0.87209407])
```

Create the following matrix:

```
In [18]: matr = np.arange(0.01, 1.01, 0.01).reshape(10, 10)
         matr
```

```
Out[18]: 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.  ]])
```

```
In [20]: linear_points = np.linspace(0, 1, 20)
         linear_points
```

Out[20]:
```
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 [21]:
```
mat = np.arange(1,26).reshape(5,5)
mat
```

Out[21]:
```
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 [23]:
```
mat1 = mat[2:, 1:]
mat1
```

Out[23]:
```
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 [24]:
```
mat2 = mat[3, 4]
mat2
```

Out[24]:
```
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 [25]:
```
mat3 = mat[0:3, 1:2]
mat3
```

Out[25]:
```
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 [26]:
```
mat4 = mat[4, :]
mat4
```

Out[26]:
```
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 [27]:
```
mat5 = mat[3:5, :]
mat5
```

Out[27]:
```
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 [28]:
```
sum_of_values = np.sum(mat)
sum_of_values
```

Out[28]:
```
325
```

Get the standard deviation of the values in mat

In [29]:
```
std_deviation = np.std(mat)
std_deviation
```

Out[29]:
```
7.211102550927978
```

Get the sum of all the columns in mat

In [30]:
```
column_sum = np.sum(mat, axis=0)
column_sum
```

Out[30]:
```
array([55, 60, 65, 70, 75])
```

In [ ]:
```
```