Import NumPy as np

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

Create an array of 10 zeros

In [2]:
```
my_array = np.zeros(10)
my_array
```

Out[2]:
```
array([0., 0., 0., 0., 0., 0., 0., 0., 0., 0.])
```

Create an array of 10 ones

In [3]:
```
my_array2 = np.ones(10)
my_array2
```

Out[3]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

Create an array of 10 fives

In [4]:
```
my_array3 = np.full(10,5.)
my_array3
```

Out[4]:
```
array([5., 5., 5., 5., 5., 5., 5., 5., 5., 5.])
```

Create an array of the integers from 10 to 50

In [5]:
```
my_array4 = np.arange(10, 51)
my_array4
```

Out[5]:
```
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

In [6]:
```
even_integers = np.arange(10, 51, 2)

even_integers
```

Out[6]:
```
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

In [7]:
```
my_array5 = np.arange(9)
my_matrix = my_array5.reshape(3, 3)
my_matrix
```

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

Create a 3x3 identity matrix

In [8]:
```
identity_matrix = np.eye(3)
identity_matrix
```

Out[8]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

Use NumPy to generate a random number between 0 and 1

In [9]:
```
random_number = np.random.rand()
random_number
```

Out[9]:
```
0.6022765110828388
```

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

In [10]:
```
random_numbers = np.random.normal(0, 1, 25)
random_numbers
```

Out[10]:
```
array([-1.20294564e-01,  5.41645257e-01, -5.38681872e-01,  7.23054571e-01,
       -5.33528225e-01, -6.49324769e-01, -6.32013764e-01, -1.33513046e+00,
        1.28721964e+00, -2.58577777e+00,  3.13618252e-01, -3.32371286e-01,
       -5.49927102e-01, -1.09720379e-03,  5.78255690e-01,  3.52186375e-01,
        1.94809183e+00, -1.80707016e+00,  1.42858508e-01,  8.18445955e-01,
       -9.91635190e-01,  9.53778074e-01,  3.03264431e-01,  1.99687305e+00,
        8.91342464e-01])
```

Create the following matrix:

In [11]:
```
values = np.arange(0.01, 1.01, 0.01)
matrix = values.reshape(10, 10)
matrix
```

Out[11]:
```
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 [12]:
```
linear_space = np.linspace(0, 1, 20)
linear_space
```

Out[12]:
```
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 [32]:
```
mat = np.arange(1,26).reshape(5,5)
mat
```

Out[32]:
```
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 [14]:
```
# 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 [15]:
```
matrix = [
    [12, 13, 14, 15],
    [17, 18, 19, 20],
    [22, 23, 24, 25]
]

# Convert the nested list to a NumPy array if needed
matrix_array = np.array(matrix)
matrix_array
```

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

In [16]:
```
# 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 [17]:
```
20
```

Out[17]:
```
20
```

In [18]:
```
# 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 [19]:
```
matrix_1=[[2],[7],[12]]
matrix_array1 = np.array(matrix_1)
matrix_array1
```

Out[19]:
```
array([[ 2],
       [ 7],
       [12]])
```

In [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
```

In [27]:
```
my_list = [21, 22, 243, 24, 25]
my_list
```

Out[27]:
```
[21, 22, 243, 24, 25]
```

In [22]:
```
# 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 [28]:
```
lis = [[16, 17, 18, 19, 20],
       [21, 212, 23, 24, 25]]
matrix_array2 = np.array(lis)
matrix_array2
```

Out[28]:
```
array([[ 16,  17,  18,  19,  20],
       [ 21, 212,  23,  24,  25]])
```

Get the sum of all the values in mat

In [29]:
```
total_sum = np.sum(mat)
total_sum
```

Out[29]:
```
325
```

Get the standard deviation of the values in mat

In [33]:
```
std_deviation = np.std(mat)
std_deviation
```

Out[33]:
```
7.211102550927978
```

Get the sum of all the columns in mat

In [34]:
```
column_sums = np.sum(mat, axis=0)
column_sums
```

Out[34]:
```
array([55, 60, 65, 70, 75])
```

In [35]:
```
column_sums = np.sum(mat, axis=1)
column_sums
```

Out[35]:
```
array([ 15,  40,  65,  90, 115])
```

In [37]:
```
column_sums = np.sum(mat, axis=0)
column_sums
```

Out[37]:
```
array([55, 60, 65, 70, 75])
```

In [ ]:
```

```