In [1]:

```
# You can create a two-dimensional array of all ones by passing the number of
# rows and columns as the first and second parameters of the ones() method, as
# shown below:

import numpy as np
ones_array = np.ones((6,4))
print(ones_array)
```

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

In [2]:

```
# The zeros() method can be used to create a NumPy array of all zeros. Here is
# an example.

import numpy as np
zeros_array = np.zeros(6)
print(zeros_array)
```

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

In [4]:

```
# You can create a two-dimensional array of all zeros by passing the number of
# rows and columns as the first and second parameters of the zeros() method,
# as shown below:

import numpy as np
zeros_array = np.zeros((6,4))
print(zeros_array)
```

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

In [5]:

```
# The eye() method is used to create an identity matrix in the form of a twodimensional
# NumPy array. An identity matrix contains 1s along the diagonal,
# while the rest of the elements are 0 in the array.

import numpy as np
eyes_array = np.eye(5)
print(eyes_array)
```

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

In [6]:

```
# The random.rand() function from the NumPy module can be used to create
# a NumPy array with uniform distribution.

import numpy as np
uniform_random = np.random.rand(4, 5)
print(uniform_random)
```

```
[[0.35303551 0.44390219 0.57293219 0.23437116 0.48623852]
 [0.50085689 0.52910895 0.04483607 0.68992919 0.14194176]
 [0.6697423  0.57067137 0.8874449  0.02394839 0.40046026]
 [0.35088792 0.93548093 0.52310459 0.11438905 0.96774341]]
```

In [7]:

```
# The random.randn() function from the NumPy module can be used to create
# a NumPy array with normal distribution, as shown in the following example.

import numpy as np
normal_random = np.random.randn(4, 5)
print(uniform_random)
```

```
[[0.35303551 0.44390219 0.57293219 0.23437116 0.48623852]
 [0.50085689 0.52910895 0.04483607 0.68992919 0.14194176]
 [0.6697423  0.57067137 0.8874449  0.02394839 0.40046026]
 [0.35088792 0.93548093 0.52310459 0.11438905 0.96774341]]
```

In [8]:

```
# Finally, the random.randint() function from the NumPy module can be used
# to create a NumPy array with random integers between a certain range. The
# first parameter to the randint() function specifies the lower bound, the
# second parameter specifies the upper bound, and the last parameter specifies
# the number of random integers to generate between the range. The following
# example generates five random integers between 5 and 50.

import numpy as np
integer_random = np.random.randint(10, 50, 5)
print(integer_random)
```

```
[39 45 16 47 17]
```

In [9]:

```
# Depending on the dimensions, there are various ways to display the NumPy
# arrays. The simplest way to print a NumPy array is to pass the array to the print
# method, as you have already seen in the previous section. An example is
# given below:

import numpy as np
my_array = np.array([10,12,14,16,20,25])
print(my_array)
```

```
[10 12 14 16 20 25]
```

In [10]:

```
# You can also use loops to display items in a NumPy array. It is a good idea to
# know the dimensions of a NumPy array before printing the array on the
# console. To see the dimensions of a NumPy array, you can use the ndim
# attribute, which prints the number of dimensions for a NumPy array. To see
# the shape of your NumPy array, you can use the shape attribute.

import numpy as np
print(my_array.ndim)
print(my_array.shape)
```

```
1
(6,)
```

In [13]:

```
# To print items in a one-dimensional NumPy array, you can use a single
# for each loop, as shown below:

import numpy as np
for i in my_array:
    print(i)
```

```
10
12
14
16
20
25
```

In [ ]:

```

```