In [1]:

```
# The following script creates a two-dimensional NumPy array with four rows
# and five columns. The array contains random integers between 1 and 10. The
# array is then printed on the console.

import numpy as np
integer_random = np.random.randint(1,11, size=(4, 5))
print(integer_random)
```

```
[[ 6  5  2  5  1]
 [ 4 10  3  3 10]
 [ 6  7 10  2  2]
 [ 2  4  8  3  5]]
```

In [2]:

```
# Let’s now try to see the number of dimensions and shape of our NumPy
# array.

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

```
2
(4, 5)
```

In [5]:

```
# To traverse through items in a two-dimensional NumPy array, you need two
# for each loops: one for each row and the other for each column in the row.
# Let’s first use one for loop to print items in our two-dimensional NumPy
# array.

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

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

In [7]:

```
# To traverse through all the items in the two-dimensional array, you can use
# the nested foreach loop, as follows:

import numpy as np
for rows in integer_random:
    for column in rows:
        print(column)
```

```
6
5
2
5
1
4
10
3
3
10
6
7
10
2
2
2
4
8
3
5
```

In [8]:

```
# To add the items into a NumPy array, you can use the append() method from
# the NumPy module. First, you need to pass the original array and the item
# that you want to append to the array to the append() method. The append()
# method returns a new array that contains newly added items appended to the
# end of the original array. The following script adds a text item “Yellow”
# to an existing array with three items.

import numpy as np
my_array = np.array(["Red", "Green", "Orange"])
print(my_array)
extended = np.append(my_array, "Yellow")
print(extended)
```

```
['Red' 'Green' 'Orange']
['Red' 'Green' 'Orange' 'Yellow']
```

In [9]:

```
# In addition to adding one item at a time, you can also append an array of
# items to an existing array. The method remains similar to appending a single
# item. You just have to pass the existing array and the new array to the
# append() method, which returns a concatenated array where items from the
# new array are appended at the end of the original array.

import numpy as np
my_array = np.array(["Red", "Green", "Orange"])
print(my_array)
extended = np.append(my_array, ["Yellow", "Pink"])
print(extended)
```

```
['Red' 'Green' 'Orange']
['Red' 'Green' 'Orange' 'Yellow' 'Pink']
```

In [10]:

```
# To add items in a two-dimensional NumPy array, you have to specify
# whether you want to add the new item as a row or as a column. To do so, you
# can take the help of the axis attribute of the append method.
# Let’s first create a 3 x 3 array of all zeros.

import numpy as np
zeros_array = np.zeros((3,3))
print(zeros_array)
```

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

In [11]:

```
# To add a new row in the above 3 x 3 array, you need to pass the original array
# to the new array in the form of a row vector and the axis attribute to the
# append() method. To add a new array in the form of a row, you need to set 0
# as the value for the axis attribute. Here is an example script.

import numpy as np
zeros_array = np.zeros((3,3))
print(zeros_array)
print("Extended Array")
extended = np.append(zeros_array, [[1, 2, 3]], axis = 0)
print(extended)
```

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
Extended Array
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]
 [1. 2. 3.]]
```

In [12]:

```
# To append a new array as a column in the existing 2-D array, you need to set
# the value of the axis attribute to 1.

import numpy as np
zeros_array = np.zeros((3,3))
print(zeros_array)
print("Extended Array")
extended = np.append(zeros_array, [[1],[2],[3]], axis = 1)
print(extended)
```

```
[[0. 0. 0.]
 [0. 0. 0.]
 [0. 0. 0.]]
Extended Array
[[0. 0. 0. 1.]
 [0. 0. 0. 2.]
 [0. 0. 0. 3.]]
```

In [13]:

```
# To delete an item from an array, you may use the delete() method. You need
# to pass the existing array and the index of the item to be deleted to the
# delete() method. The following script deletes an item at index 1 (second item)
# from the my_array array.

import numpy as np
my_array = np.array(["Red", "Green", "Orange"])
print(my_array)
print("After deletion")
updated_array = np.delete(my_array, 1)
print(updated_array)
```

```
['Red' 'Green' 'Orange']
After deletion
['Red' 'Orange']
```