In [9]:

```
# To load a NumPy array in the “NPY” format, you can use the load() method,
# as shown in the following example.

import numpy as np

a = np.array(([i + j for i in range(5)
               for j in range(5)]))
# a is printed.
print("a is:")
print(a)

np.save('file', a)
print("the array is saved in the file.npy")

# the array is saved in the file.npy
b = np.load('file.npy')

# the array is loaded into b
print("b is:")
print(b)

# b is printed from file.npy
print("b is printed from file.npy")
```

```
a is:
[0 1 2 3 4 1 2 3 4 5 2 3 4 5 6 3 4 5 6 7 4 5 6 7 8]
the array is saved in the file.npy
b is:
[0 1 2 3 4 1 2 3 4 5 2 3 4 5 6 3 4 5 6 7 4 5 6 7 8]
b is printed from file.npy
```

In [10]:

```
# On the other hand, if your NumPy array is stored in the text form, you may
# use the loadtxt() method to load such a NumPy array.

import numpy as np
loaded_array = np.loadtxt("D:\internship 2\my_array.txt")
print(loaded_array)
```

```
[10. 12. 14. 16. 18. 20.]
```

In [11]:

```
# NumPy arrays contain various functionalities for performing statistical
# operations, such as finding the mean, median, and standard deviations of
# items in a NumPy array.
# To find the mean or average of all the items in a NumPy array, you need to
#pass the array to the mean() method of the NumPy module. Here is an
# example:

import numpy as np
my_array = np.array([2,4,8,10,12])
print(my_array)
print("mean:")
print(np.mean(my_array))
```

```
[ 2  4  8 10 12]
mean:
7.2
```

In [12]:

```
# You can also find the mean in a two-dimensional NumPy array across rows
# and columns. To find the mean across columns, you need to pass 1 as the
# value for the axis parameter of the mean method. Similarly, to find the mean
# across rows, you need to pass 0 as the parameter value.
# The following script finds the mean of a two-dimensional array containing
# two rows and three columns across rows and columns.

import numpy as np
my_array = np.random.randint(1,20, size = (2,3))
print(my_array)
print("mean:")
print(np.mean(my_array, axis = 1))
print(np.mean(my_array, axis = 0))
```

```
[[ 5 17 15]
 [13  9  4]]
mean:
[12.33333333  8.66666667]
[ 9.  13.   9.5]
```

In [1]:

```
# The median() method from the NumPy module is used to find the median
# value in a NumPy array. Here is an example.

import numpy as np
my_array = np.array([2,4,8,10,12])
print(my_array)
print("median:")
print(np.median(my_array))
```

```
[ 2  4  8 10 12]
median:
8.0
```

In [2]:

```
# Similarly, to find the median values across columns and rows in a twodimensional
# array, you need to pass 1 and 0, respectively, as the values for
# the axis attribute of the median method.
# The following script finds the median values across rows and columns for a
# two-dimensional array with three rows and five columns.

import numpy as np
my_array = np.random.randint(1,20, size = (3,5))
print(my_array)
print("median:")
print(np.median(my_array, axis = 1))
print(np.median(my_array, axis = 0))
```

```
[[ 4  4 11 14  6]
 [ 5 17  6  6 10]
 [19  7  7 12  5]]
median:
[6. 6. 7.]
[ 5.  7.  7. 12.  6.]
```

In [3]:

```
# The max() function returns the maximum value from the array, while the
# min() function returns the minimum value.
# The following script returns the minimum value in a NumPy array using the
# min() method.

import numpy as np
my_array = np.array([2,4,8,10,12])
print(my_array)
print("min value:")
print(np.amin(my_array))
```

```
[ 2  4  8 10 12]
min value:
2
```

In [4]:

```
# You can get the minimum values across all rows or columns in a twodimensional
# NumPy array by passing 0 or 1 as values for the axis attribute of
# the min() method. The value of 1 for the axis attribute returns the minimum
# values across all columns, whereas a value of 0 returns the minimum values
# across all rows.

import numpy as np
my_array = np.random.randint(1,20, size = (3,4))
print(my_array)
print("min:")
print(np.amin(my_array, axis = 1))
print(np.amin(my_array, axis = 0))
```

```
[[16 19  5  2]
 [11 17 13  1]
 [17  6 13  8]]
min:
[2 1 6]
[11  6  5  1]
```

In [5]:

```
# To get the maximum value from a NumPy array, you may use the max()
# method, as shown in the script below:

import numpy as np
my_array = np.array([2,4,8,10,12])
print(my_array)
print("max value:")
print(np.amax(my_array))
```

```
[ 2  4  8 10 12]
max value:
12
```

In [6]:

```
# Like the min() method, which returns the minimum value, you can get the
# maximum values across all rows or columns in a two-dimensional NumPy
# array by passing 0 or 1 as values for the axis attribute of the max() method.
# The value of 1 for the axis attribute returns the minimum values across all
# columns, whereas a value of 0 returns the minimum values across all rows.

import numpy as np
my_array = np.random.randint(1,20, size = (3,4))
print(my_array)
print("max:")
print(np.amax(my_array, axis = 1))
print(np.amax(my_array, axis = 0))
```

```
[[15 11 19  1]
 [17 15  8  2]
 [14 12 15  1]]
max:
[19 17 15]
[17 15 19  2]
```