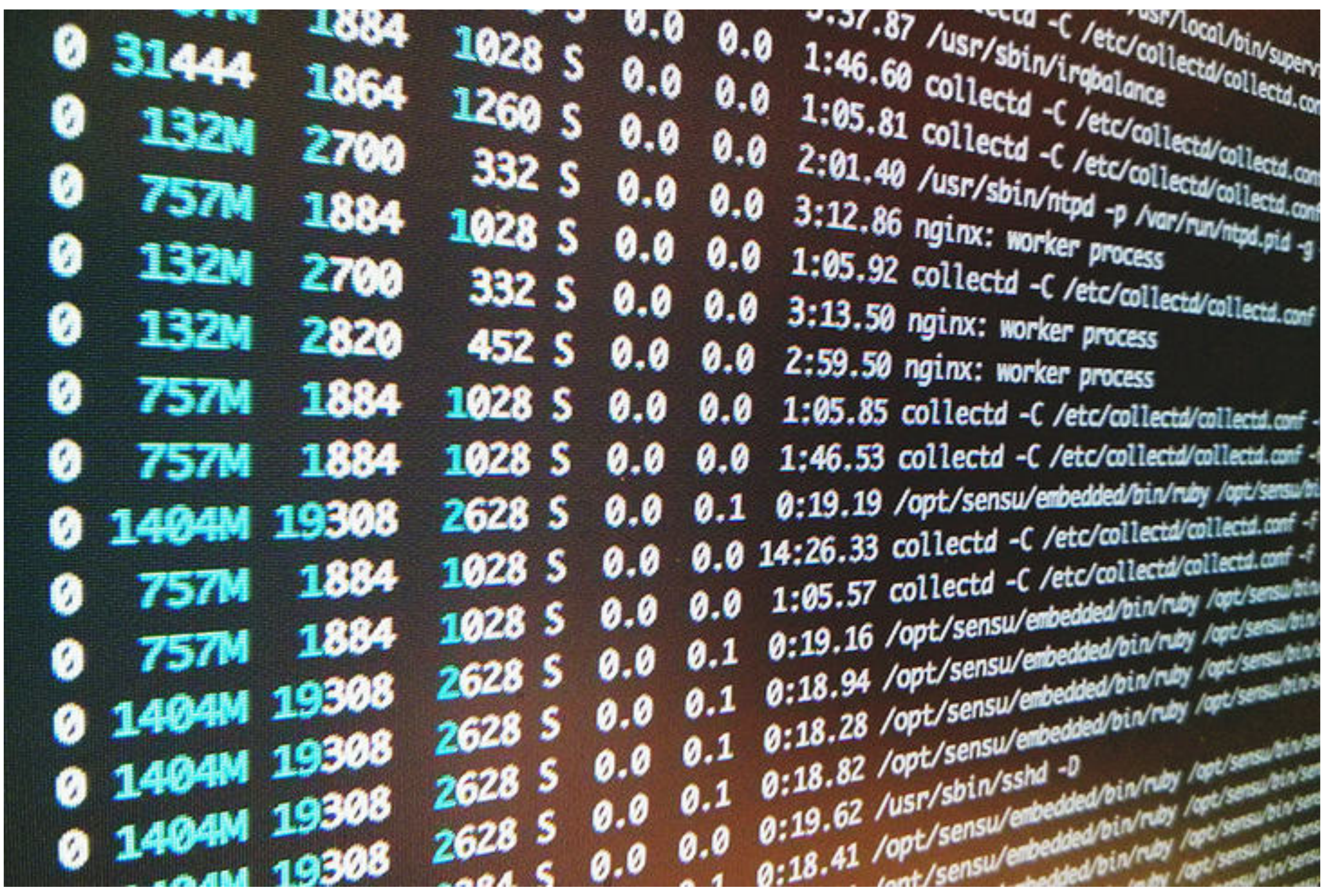

© Getty Images

Useful matrix manipulation code in Python

There will be times where you will need to access individual rows or columns from a matrix. There may be other times where you may need to do logical checks on array elements. The following code segments will be handy in such situations.

Slicing

Accessing elements of NumPy matrices and arrays.

Code example 1

This code grabs the first column of A:

```
print(A)
A[:,0]
```

As you can see index 0 refers to the first column. The output is given below:

```
[[0 1]
  [2 3]
  [4 5]]

  array([0, 2, 4])
```

Alternatively or, we could grab a particular element (in this case, the second column, last row):

Code example 2

You can extract a part of a list/string using the syntax [start:stop], which extracts characters between index start and stop.

|T| h| i| s| | i| s| | a| | s| t| r| i| n| g| .|
|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|-|
|0| 1| 2| 3| 4| 5| 6| 7| 8| 9| 10| 11| 12| 13| 14| 15| 16|

```
s = "This is a string."
# [start:stop:step]
print("------------------------")
print(s[2:10:1])
print("------------------------")
# These two are equal
print(s[0:10:2])
print("------------------------")
print(s[:10:2])
print("------------------------")
```

The output of this slicing string is as below:

```
------------------------
is is a
------------------------
Ti sa
------------------------
Ti sa
------------------------
```

Logical checks to extract values from matrices/arrays:

Consider the matrix:

$$A = \begin{bmatrix} 0 & 1 \\ 2 & 3 \\ 4 & 5 \end{bmatrix}$$

If we want to verify which elements in column two (2) are greater than 4, we can run the following code.

Code example 3

```
print("Boolean-->",A[:,1]>4) ## index
print("--------")
print("Index number-->",np.argmax(A[:,1]>4)) ## index
print("--------")
print("Values-->",A[A[:,1]>4]) ## value
```

This outputs the following. Again keep in mind that element indexes start from 0. The second column is referenced using index 1.

```
Boolean--> [False False  True]
--------
Index number--> 2
--------
Values--> [[4 5]]
```

For loops

Create a 12x12 matrix and print it out:

Code example 4

```
A = np.arange(24).reshape((12,2))
print(A)
print(A.shape)
```

The code contains two *print* commands. The first prints the matrix and the second prints the *shape* of the matrix (it has 12 rows and 2 columns).

```
[[ 0  1]
  [ 2  3]
  [ 4  5]
  [ 6  7]
  [ 8  9]
  [10 11]
  [12 13]
  [14 15]
  [16 17]
  [18 19]
  [20 21]
  [22 23]]
  (12, 2)
```

Code example 5

The following code is an example of how a *for-loop* can be used to access each row of a matrix. This is more efficient than using element references.

```
for rows in A:
print(rows)
```

The output is listed below:

```
[0 1]
  [2 3]
  [4 5]
  [6 7]
  [8 9]
  [10 11]
  [12 13]
  [14 15]
  [16 17]
  [18 19]
  [20 21]
  [22 23]
```

Share your response in the Student Discussion.

< Previous Next >