Indexing Arrays

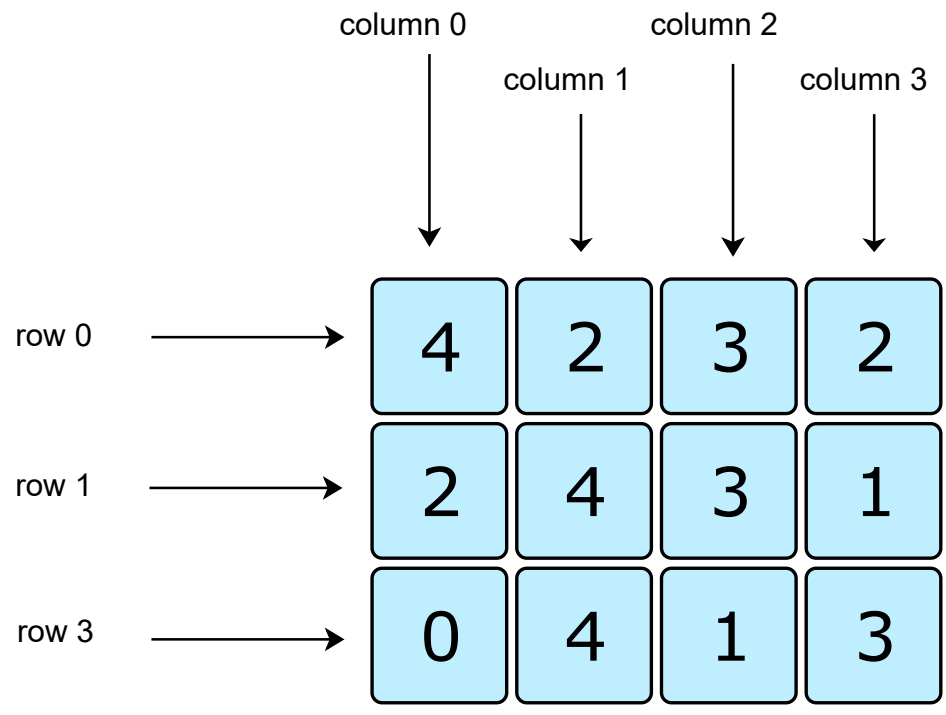
In this lesson, we will learn about indexing and slicing arrays.

WE'LL COVER THE FOLLOWING

Indexing

Similar to Python `lists()`, NumPy arrays can be indexed using square brackets.

> 💡 Indexing of arrays starts from **0**.



Elements of the one-dimensional arrays or vectors are accessed by specifying the index number of the array enclosed within square brackets.

```
import numpy as np

v = np.array([1, 5, 9])
print(v[1])
```

Elements of the two-dimensional arrays or matrices are accessed by specifying the row and column index of the array.

```
import numpy as np

M = np.array([[4, 2, 3, 2],
              [2, 4, 3, 1],
              [0, 4, 1, 3]])
print(M[1,2])
```

By similar logic, elements of n-dimensional arrays can be accessed using **n** indexes. In this course, most of our discussion will revolve around one-dimensional arrays, vectors, and two-dimensional arrays and matrices. We will not be exploring n-dimensional arrays in detail.

If we omit an index of a multidimensional array it returns the whole row. Let's see an example of this below:

```
import numpy as np

M = np.array([[4, 2, 3, 2],
              [2, 4, 3, 1],
              [0, 4, 1, 3]])
print(M[1])
```

We can achieve the same thing using the `:` index. This makes the code

readable and clearly states that you are accessing an entire row by choice and

not by accidentally missing the second index. The `:` index also provides an added functionality of accessing an entire column instead of a row.

```
import numpy as np

M = np.array([[4, 2, 3, 2],
              [2, 4, 3, 1],
              [0, 4, 1, 3]])
print(M[1,:]) # prints the entire 2nd row
print(M[:,1]) # prints the entire 2nd column
```

Array indexing is not just a read-only functionality. It allows you to both read and write. Let's look at an example:

```
import numpy as np

M = np.array([[4, 2, 3, 2],
              [2, 4, 3, 1],
              [0, 4, 1, 3]])
print(M)
M[2, 3] = 24 #changing element at 2nd row and 3rd column
print("--------")
print(M)
```

Index slicing

Index slicing is a technique to extract a part of an array. But why is this important? Sometimes while working on experiments, the raw data that we are getting from a machine or a sensor has some additional information that might not be useful for the calculation we are performing. So, in order to extract the information that is useful to our needs, we use index slicing.

The following syntax is used for slicing:

```
array[lower:upper:step]
```

In the output, `array` indices from 'lower' to 'upper - 1' in increments of size

‘step’ are included.

```
import numpy as np

v = np.array([1, 3, 5, 7, 9, 11])
print(v[2:4]) # 4th element will not be included
```

We can omit any of the three parameters and they will take their default values. The default value of

- `lower` is 0.
- `upper` is equal to the `size` property of the array.
- `step` is 1.

Let’s look at some examples of this below:

```
import numpy as np

v = np.array([1, 3, 5, 7, 9, 11])
print(v[2::])
print(v[:2:])
print(v[::2])
```

You can also try different variations of parameters `lower`, `upper` and `step` in the code above.

Slicing matrices

Index slicing works the same way for matrices:

```
import numpy as np
M = np.array([[(n + m * 20) for n in range(5)] for m in range(5)])

print(M)
print("------")
print(M[1:4, 1:3]) # slice rows 1, 2 and 3 and columns 1 and 2
```

In the next lesson, we will learn about array operations.