Multidimensional Arrays

In this lesson, we will learn about multidimensional arrays and their properties.

WE'LL COVER THE FOLLOWING

Creation

Arrays may have arbitrary dimensions as long as they fit in your computer's memory. *Multidimensional arrays*, commonly known as **matrices** in Python, can be created with any of the methods given in the previous lesson by specifying the number of rows and columns in the array.

Note that the number of rows and columns must be a tuple (so they need to be between parentheses), because the functions expect only one input argument for the shape of the array, which may be either one number or a tuple of multiple numbers.

```
import numpy as np

x = np.ones((3, 4)) # An array with 3 rows and 4 columns
print(x)
```

The code above results in the creation of a 3×4 matrix.

Arrays may also be defined by specifying all the values in the array. The `array`

function gets passed one list which consists of separate lists for each row of

the array. In the example below, the rows are entered on different lines. This may make it easier to enter the array, but it is not required.

```
import numpy as np

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

Meshgrid

A meshgrid can be created using the `meshgrid` function from the `numpy` module.

```
[X, Y] = np.meshgrid(x, y)
```

This returns 2-D grid coordinates based on the coordinates contained in vectors `x` and `y`. `X` is a matrix where each row is a copy of `x`, and `Y` is a matrix where each column is a copy of `y`.

The grid represented by the coordinates `X` and `Y` have `len(y)` rows and `len(x)` columns.

Hence, the dimensions of both `X` and `Y` are:

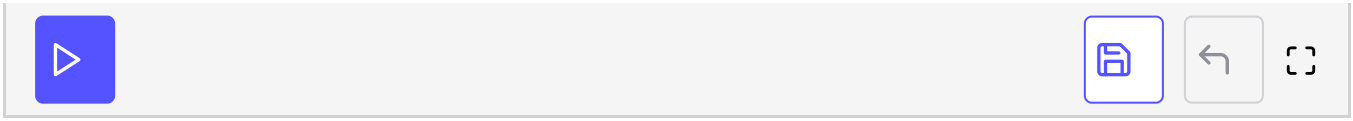
$$len(y) \times len(x)$$

```
import numpy as np

x = np.arange(0, 5)
y = np.arange(5, 11)

[X, Y] = np.meshgrid(x, y)

print(X)
print(Y)
```

Reshaping

You can change the shape of an array using the `reshape` function as long as the total number of entries doesn't change.

```
import numpy as np

x = np.array([[4, 2, 3, 2],
              [2, 4, 3, 1],
              [0, 4, 1, 3]])
print(x)
print()
print(np.reshape(x, (2, 6)))  # 2 rows, 6 columns
print()
print(np.reshape(x, (1, 12)))  # 1 row, 12 columns
```



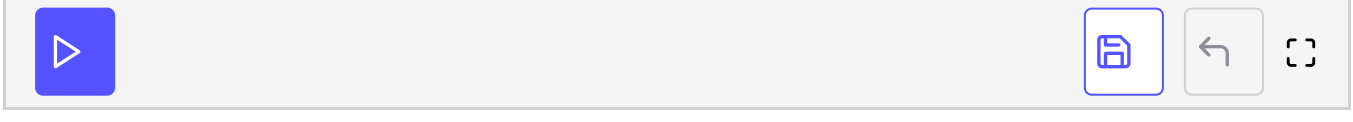
Properties of NumPy arrays

As discussed in the introductory lesson as well, NumPy arrays are of type `ndarray`.

This data structure has two very important properties:

1. `size` - has the number of elements in the `ndarray`
2. `shape` - is a tuple with the dimensions of the matrix stored in the form of `(rows, columns)`

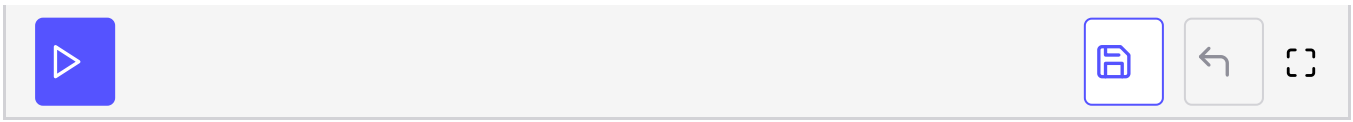
```
import numpy as np

v = np.array([1, 5, 9])
M = np.array([[4, 2, 3, 2],
              [2, 4, 3, 1],
              [0, 4, 1, 3]])

print("size of v is ", v.size)
print("shape of v is ", v.shape)

print("size of M is ", M.size)
print("shape of M is ", M.shape)
```



Other useful properties are:

3. `itemsize` - bytes per element of the `ndarray`
4. `nbytes` - total number of bytes of the `ndarray`
5. `ndim` - dimensions of the `ndarray`
6. `dtype` - data type of elements in `ndarray`

```
import numpy as np

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

print("Bytes per element of M is", M.itemsize)
print("Number of bytes of M are", M.nbytes)
print("Dimensions of M are", M.ndim)
print("Data type of elements in M is", M.dtype)
```



Let's test your knowledge of 1-D and multidimensional arrays with a quiz in the next lesson.