The Basics of NumPy Arrays

Data manipulation in Python is nearly synonymous with NumPy array manipulation: even newer tools like Pandas (Chapter 3) are built around the NumPy array. This section will present several examples using NumPy array manipulation to access data and subarrays, and to split, reshape, and join the arrays. While the types of operations shown here may seem a bit dry and pedantic, they comprise the building blocks of many other examples used throughout the book. Get to know them well!

We'll cover a few categories of basic array manipulations here:

Attributes of arrays
   Determining the size, shape, memory consumption, and data types of arrays

Indexing of arrays
   Getting and setting the value of individual array elements

Slicing of arrays
   Getting and setting smaller subarrays within a larger array

Reshaping of arrays
   Changing the shape of a given array

Joining and splitting of arrays
   Combining multiple arrays into one, and splitting one array into many

NumPy Array Attributes

First let's discuss some useful array attributes. We'll start by defining three random arrays: a one-dimensional, two-dimensional, and three-dimensional array. We'll use NumPy's random number generator, which we will *seed* with a set value in order to ensure that the same random arrays are generated each time this code is run:

```
In[1]: import numpy as np
       np.random.seed(0)  # seed for reproducibility

       x1 = np.random.randint(10, size=6)  # One-dimensional array
       x2 = np.random.randint(10, size=(3, 4))  # Two-dimensional array
       x3 = np.random.randint(10, size=(3, 4, 5))  # Three-dimensional array
```

Each array has attributes `ndim` (the number of dimensions), `shape` (the size of each dimension), and `size` (the total size of the array):

```
In[2]: print("x3 ndim: ", x3.ndim)
       print("x3 shape:", x3.shape)
       print("x3 size: ", x3.size)

x3 ndim:  3
x3 shape: (3, 4, 5)
x3 size:  60
```