NumPy Recipes

A practical introduction to NumPy

Martin McBride

NumPy Recipes

A practical introduction to NumPy

Martin McBride

This book is for sale at http://leanpub.com/numpy

This version was published on 2020-04-18

Contents

Foreword

This book aims to provide an introduction to the use and application of the Python NumPy library. NumPy is a large library, and this book introduces many of its most important features in the form of recipes for performing certain functions. This provides a practical grounding that can be used as a basis for understanding the extensive documentation available from the NumPy website.

Who is this book for?

This book is primarily aimed at developers who have at least a small amount of Python experience, who wish to use the NumPy library for data analysis, machine learning, image or sound processing, or any other mathematical or scientific application. It only requires a basic understanding of Python programming.

About the author

Martin McBride is a software developer, specialising in computer graphics, sound, and mathematical programming. He has been writing code since the 1980s in a wide variety of languages from assembler through to C++, Java and Python. He writes for PythonInformer.com and is the author of *Functional Programming in Python* available on leanpub.com. He is interested in generative art and works on the generativepy open source project.

Keep in touch

If you have any comments or questions you can get in touch by any of the following methods:

- Joining the Python Informer forum at http://pythoninformer.boards.net/[1]. Follow the NumPy[2] board in the Books section.
- The book page[3] on the Leanpub website.
- Signing up for the Python Informer newsletter at pythoninformer.com
- Following @pythoninformer on twitter.
- Contacting me directly by email (info@axlesoft.com).

[1]http://pythoninformer.boards.net/
[2]http://pythoninformer.boards.net/board/8/numpy
[3]https://leanpub.com/numpy

1 Introduction to NumPy

This is a brief overview of what NumPy is, and why makes it so useful. But first, we will look at how to install it.

1.1 Installing NumPy

The recommended way to install NumPy is the Anaconda platform.

This book covers NumPy version 1.17, but most example should work with any fairly recent version.

Anaconda can be installed on Windows, Linux or Mac OS X. It includes Python itself, NumPy, SciPy, Matplotlib and other packages. It also allows you to download many other packages using the Conda package manager.

This is particularly recommended for Windows, where installing some Python packages can be difficult.

Alternatively you can install NumPy, SciPy and matplotlib using pip:

```
python -m pip install --user NumPy scipy matplotlib
```

This should work for most systems. Using the `--user` flag installs the packages for the local user only, which can avoid certain install problems.

Finally, for Ubuntu and Debian Linux you can the Linux package manager:

```
sudo apt-get install python-NumPy python-scipy python-matplotlib
```

1.2 What is NumPy?

Numpy is a library that supports multidimensional arrays. It is optimised to handle very large arrays efficiently, making it ideal for large data sets.

Numpy provides a large number of optimised functions for performing various operations on arrays, from basic maths operations to Fourier transforms, linear algebra and statistics.

NumPy interfaces to many other Python libraries, and can be used to exchange raw data with libraries written in C and other languages.

1.3 NumPy vs Python lists

Numpy arrays and Python lists appear superficially quite similar, and they can be used interchangeably in some situations. However, lists can be very inefficient with large data sets.

A Python list is a general purpose dynamic list structure that holds Python objects. It automatically resizes if items are added or removed, and can hold a mixture of different Python types within the same list (lists are heterogeneous). Lists are 1-dimensional, although a list of lists can be used to simulate a multi-dimensional list.

A NumPy array is a statically sized array - once it has been created it cannot be resized. Arrays normally hold primitive data types (fixed size integers and floats typically) and all the elements in an array must be the same type (arrays are homogeneous). Arrays support multiple dimensions, and are *regular* (they have a fixed size in every dimension).

1.4 Advantages of NumPy

NumPy has several advantages when dealing with large data sets:

- Storage efficiency - each value is stored as a primitive data type. For example, in an array of 16 bit integers, each integer only occupies 2 byes. In a Python list, numbers are stored as objects occupying more memory (around 30 bytes is typical).
- Access efficiency - each value can be accessed directly from its memory location. In a list, a numerical value is stored within an object, which typically takes extra steps to access.
- Vectorisation - many operations in NumPy are *vectorised*. This means that a single Python function will apply the same operation to every element in the array. This avoids Python loops, and instead the loop is implemented in C, and is typically considerably faster.
- Compatibility with other libraries - many other libraries store data in a similar format to NumPy, allowing data to be exchanged directly with other libraries, simply by passing a memory pointer that references the data.

1.5 NumPy universal functions

As mentioned above, NumPy contains many vectorised functions (called *universal functions* or *ufuncs*) that can operate on an entire array in one call. It is also possible to reduce or accumulate values using a ufunc (for example calculating the sum of all values in an array, and the running total).

ufuncs can also operate on subsections of an array, using multi-dimensional slicing, or conditional application (*where* clauses). They can operate over arrays of different shapes using *broadcasting*. It is also possible to implement table look-up via *fancy indexing*.

1.6 Compatibility with other libraries

NumPy is compatible with many other libraries. To take a couple of example areas:

For **scientific computing** NumPy works with SciPy and Matplotlib to provide similar functionality to the MATLAB product.

For **image processing** NumPy works with packages such as PyCairo (vector drawing), Pillow (image processing) and OpenCV (computer vision).

2 Anatomy of a NumPy array

NumPy arrays come in various types, shapes and sizes. In this chapter will look at different array characteristics, and the terms used by NumPy to describe those characteristics.

2.1 NumPy arrays compared to lists

NumPy arrays are similar to lists in some ways. In the previous chapter we mentioned the performance differences, with NumPy arrays being more efficient for many uses. But they also differ in *capabilities*. Lists are generally more flexible, but less efficient. Here are the main differences.

Generally arrays are *homogeneous*, meaning every element of the array has the same type. For example, in a byte array all the elements are bytes. In a list, different elements if the same list can have different types.

Arrays are multi-dimensional, that is to say a single array can be 1-dimensional, or can have 2, 3 or more dimensions. Lists are always 1-dimensional, but you can have lists of lists, which simulate 2 dimensions (or lists of lists of lists, simulating 3 dimensions, and so on).

Arrays are also *regular*, so the shape of an n-dimensional array can be described by its size in each dimension. For example, a 2-dimensional might have a shape 3 by 5 meaning that it has 3 rows and each row has exactly 5 columns:

```
[[0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0.]]
```

This contrast with an *irregular* or *ragged* array where each row might have a different number of columns. An array like this **isn't possible** in NumPy:

```
[[0. 0. 0.]        # Not possible in NumPy!
 [0. 0. 0. 0. 0.]
 [0. 0.]]
```

The idea of regular arrays extends to any number of dimensions, and NumPy arrays are always regular.

NumPy arrays are usually numerical, containing either floats, ints or complex numbers of various types. They can also contain other types of data, but that is less common and many of the available NumPy functions only work with numerical data.

NumPy arrays have several primary characteristics that we will explore in this chapter:

- Rank, which means the number of dimensions.
- Shape, which defines the size in each dimension.
- Data type, which specifies the type and precision of the numerical type that is stored.
- Location, the physical memory location of the data array, which is very useful for sharing data with non-Python libraries.

2.2 Printing the characteristics of an array

In this recipe we will create a three simple arrays, and print their parameters.

Here is the code:

```
import NumPy as np

def print_array(name, a):
    print(name)
    print(a)
    print('rank =', a.ndim)
    print('shape =', a.shape)
    print('size =', a.size)
    print('data type =', a.dtype)
    print('element size =', a.itemsize)
    print('data location =', a.data)
    print()

# 1 dimensional array
a1 = np.array([1, 2, 3, 4])
print_array('1 dimensional array', a1)

# 2 dimensional array
a2 = np.array([[1.0, 1.0, 1.0, 1.0], [2.0, 2.0, 2.0, 2.0]])
print_array('2 dimensional array', a2)

# 3 dimensional array
a3 = np.array([[[1, 1, 1, 1], [2, 2, 2, 2]],
               [[3, 3, 3, 3], [4, 4, 4, 4]],
               [[5, 5, 5, 5], [6, 6, 6, 6]]])
print_array('3 dimensional array', a3)
```

This prints:

```
1 dimensional array
[1 2 3 4]
rank = 1
shape = (4,)
size = 4
data type = int32
element size = 4
data location = <memory at 0x0000016476EEF1C8>

2 dimensional array
[[1. 1. 1. 1.]
 [2. 2. 2. 2.]]
rank = 2
shape = (2, 4)
size = 8
data type = float64
element size = 8
data location = <memory at 0x0000016476F52668>

3 dimensional array
[[[1 1 1 1]
  [2 2 2 2]]

 [[3 3 3 3]
  [4 4 4 4]]

 [[5 5 5 5]
  [6 6 6 6]]]
rank = 3
shape = (3, 2, 4)
size = 24
data type = int32
element size = 4
data location = <memory at 0x0000016477242408>
```

We will look at these parameters one by one.

2.2.1 Rank

The **rank** of an array is simply the number of axes (or dimensions) it has.

The rank of an array can be found from its `ndim` attribute:

- `a1` array is 1-dimensional so its rank is 1.

- `a2` is 2-dimensional so its rank is 2.
- `a3` is 3-dimensional so it has a rank of 3.

Typical uses of different ranks are discussed later in this chapter.

2.2.2 Shape

The **shape** of an array gives the size of the array in each dimension, usually expressed as a tuple. The number of elements in the shape sequence is always equal to the rank.

The shape an be found from its `shape` attribute.

- `a1` is a rank 1 array, with a length of 4, its shape is a single element tuple `(4, )`.
- `a2` is a rank 2 array which has 2 rows of 4 columns, so its shape is `(2, 4)`.
- `a3` is a rank 3 array which has 3 planes, each with 2 rows of 4 columns, so its shape is `(3, 2, 4)`.

2.2.3 Size

The **size** of an array is simply the total number of elements it contains. It is found using the `size` attribute.

The `size` is equal to the product of all the elements in the `shape`:

- `a1` is just a 1-dimensional array of 4 elements, so it has a `size` of 4.
- `a2` is a 2-dimensional array of 2 by 4 elements, so it has a `size` of 8.
- `a3` is a 3-dimensional array of 3 by 2 by 4 elements, so it has a `size` of 24.

2.2.4 Data type

In Python, number types (int, float and complex) are stored as objects.

By contrast, NumPy arrays are made up of *primitive data types*. These are mainly numerical data types, that are stored directly in memory without any extra information.

The data type of an array can be found using the **dtype** attribute.

- `a1` and `a3` are integer arrays, because all the values are integers. They have a type of `int32` (a 4 byte signed integer format).
- `a2`, as you may have noticed, is formed with float values. Since NumPy arrays are homogeneous (it can't mix different types of element), if any of the inputs is a float then a float array is created. The type is `float64`, the same type of float that Python uses for its float types.

There are other data types, as discussed below, with more details in the *Reference* chapter.

2.2.5 Item size

We can find the size of an element in an array using the `itemsize` attribute. The element size is determined by the data type, so for example an `int32` array will have an item size of 4 (32 bits is 4 bytes). A `float64` array with have an item size of 8.

2.2.6 Data location

We can find the memory address of the actual array data, using the **data** attribute.

That isn't particularly useful if you are writing a pure Python program, because Python can't directly access memory locations. It can be useful if you are calling other low level libraries that have Python bindings and can accept memory pointers as parameters.

2.3 Array rank examples

Here are some typical uses for different array ranks.

Rank 1 arrays are 1-dimensional, and are typically used to store individual samples (for example the heights of a set of people), or sequences (for example, the value of the pound against the dollar every day for the past year), or time varying values such as sound data.

Rank 2 arrays are 2-dimensional. A rank 2 array is sometimes called a matrix. Rank 2 arrays are used to store row and column data, like a spreadsheet. Other uses include storing image data.

Rank 3 arrays are 3-dimensional. They can be used to store volumetric data, for example the air temperature at different places in a room. The can also store 2-dimensional vector fields, colour images (where each pixel is represented by a vector containing the red, green and blue values values), among other uses.

2.4 Data types

The main data types are integers, unsigned integers, floats, complex numbers and bools. Most array creation functions allow you to choose the data type using the `dtype` optional parameter, see the chapter *Creating arrays*.

2.4.1 Integers

Unlike Python integers, which have virtually unlimited range, NumPy integers have fixed sizes. We can choose a specific size using `np.int8`, `np.int16`, `np.int32` or `np.int64`.

For example, `np.int8` stores integers as a signed 8-bit quantity, with allowed values in the range -128 to +127. Each number occupies exactly 1 byte, so if you had a NumPy array shape `(100, 200)` the data would occupy exactly 20,000 bytes. `np.int16` stores integers as a signed 16-bit quantity, giving a range of -32768 to 32767. See the *Reference* chapter for more details.

There are also various type that have platform dependent sizes. For example, `np.short` corresponds to the definition of a C language `short` integer on the particular platform. On some platforms this might correspond to a 16 bit integer, on others it might correspond to a 32 bit integer, and on other platforms it might be larger or smaller.

Generally if you know that you need a particular size of integer, use the first form. For example if you wanted to store 16-bit sound data, choose `np.int16` then you will always get 16-bit storage. But if you are exchanging data with a C library that uses `short` integers, then choose `np.short` so that NumPy will use the correct size for however C defines shorts on the platform you are running on.

2.4.2 Unsigned integers

Unsigned integers are used to represent values that can never be negative. We can choose a specific size using `np.uint8`, `np.uint16`, `np.uint32` or `np.uint64`. We can also choose from platform dependent sizes such as `np.ushort`.

The advantage of unsigned ints is that they give a larger range of positive values. For example `np.uint8` has a range of 0 to 255 (compared to `np.int8` which has a range of -128 to 127). For data that is known to be always non-negative, this can be useful.

2.4.3 Floating point values

NumPy supports two types of floating point numbers. `np.float64` is equivalent to Python floating point numbers, and also equivalent to the C `double` type. It has a precision of approximately 16 decimal places.

`np.float32` has a lower range and precision, equivalent to the C `float` type. It would normally only be used where ultra-high precision is not required. It has a lower precision of approximately 7 decimal places, but occupies half the memory of a double float.

2.4.4 Complex number formats

A complex number consists of two floating point numbers, the real part and the imaginary part.

`np.complex128` uses two 64 bit floating point numbers (each equivalent to `np.float64`) stored side by side as a 16 byte (128 bit) block.

`np.complex64` uses two 32 bit floating point numbers (each equivalent to `np.float32`) stored side by side as a 8 byte (64 bit) block.

2.4.5 Boolean

A boolean value, `np.bool_` is a byte value similar to `np.uint8`. However its value is interpreted as true or false:

- A value of zero is false.
- Any non-zero value is true.

2.4.6 List of main data types

In summary, here is a list of the main data types:

- Signed integer - `np.int8`, `np.int16`, `np.int32`, `np.int64`
- Unsigned integer - `np.uint8`, `np.uint16`, `np.uint32`, `np.uint64`
- Float - `np.float32`, `np.float64`
- Complex - `np.complex64`, `np.complex128`
- Boolean - `np.bool_`

For more information about primitive data types, see the chapter *Reference*.

3 Creating arrays

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

3.1 Creating an array of zeroes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

3.2 Creating other fixed content arrays

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

3.3 Choosing the data type

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

3.4 Creating multi-dimensional arrays

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

3.5 Creating like arrays

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

3.6 Creating an array from a Python list

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

3.7 Controlling the type with the array function

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

3.8 array function anti-patterns

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

3.9 Creating a value series with arange

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

3.10 Rounding error problem with arange

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

3.11 Create a sequence of a specific length with linspace

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

3.12 Making linspace more like arange using the endpoint parameter

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

3.13 Obtaining the linspace step size

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

3.14 Other sequence generators

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

3.15 Creating an identity matrix

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

3.16 Creating an eye matrix

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

3.17 Using vectorisation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

4 Vectorisation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

4.1 Performing simple maths on an array

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

4.2 Vectorisation with other data types

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

4.3 Vectorisation with multi-dimensional arrays

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

4.4 Expressions using two arrays

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

4.5 Expressions using two multi-dimensional arrays

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

4.6 More complex expressions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

4.7 Using conditional operators

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

4.8 Combining conditional operators

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

5 Universal functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

5.1 Example universal function - sqrt

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

5.2 Example universal function of two arguments - power

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

5.3 Summary of ufuncs

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

5.3.1 Maths operations

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

5.3.2 Trigonometric functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

5.3.3 Bit manipulation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

5.3.4 Comparison functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

5.3.5 Logical functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

5.3.6 Min and max

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

5.3.7 Float functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

5.4 ufunc methods

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

5.4.1 Reduce

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

5.4.2 Accumulation

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

5.5 Optional keyword arguments for ufuncs

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

5.5.1 out

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

5.5.2 where

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6 Indexing, slicing and broadcasting

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6.1 Indexing an array

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6.1.1 Indexing in 1 dimension

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6.1.2 Indexing in 2 dimensions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6.1.3 Picking a row or column in 2-dimensions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6.1.4 Indexing in 3 dimensions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6.1.5 Picking a row or column in a 3D array

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6.1.6 Picking a matrix in a 3D array

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6.2 Slicing an array

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6.2.1 Slicing lists - a recap

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6.2.2 Slicing 1D NumPy arrays

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6.2.3 Slicing a 2D array

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6.2.4 Slicing a 3D array

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6.2.5 Full slices

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6.3 Slices vs indexing

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6.4 Views

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6.5 Broadcasting

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6.5.1 Broadcasting from 1 to 2 dimensions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6.5.2 Broadcasting 1 to 3 dimensions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6.5.3 Broadcasting 2 to 3 dimensions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6.6 Broadcasting rules

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6.7 Broadcasting a column vector

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6.8 Broadcasting a row vector and a column vector

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6.9 Broadcasting scalars

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6.10 Efficient broadcasting

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

6.11 Fancy indexing

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

7 Array manipulation functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

7.1 Copying an array

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

7.2 Changing the type of an array

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

7.3 Changing the shape of an array

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

7.4 Splitting arrays

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

7.4.1 Splitting along different axes

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

7.4.2 Unequal splits

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

7.4.3 Alternative functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

7.5 Stacking arrays

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

7.5.1 Stacking 2-dimensional arrays

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

8 File input and output

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

8.1 CSV format

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

8.2 Writing CSV data

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

8.2.1 Adding a header or footer

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

8.2.2 Changing the line separator

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

8.2.3 Compressing the output file

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

8.3 Reading CSV data

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

8.3.1 Skipping header or footer

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

8.3.2 Reading compressing file

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

9 Using Matplotlib with NumPy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

9.1 Installing Matplotlib

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

9.2 Plotting a histogram

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

9.3 Plotting functions

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

9.4 Plotting functions with NumPy

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

9.5 Creating a heatmap

This content is not available in the sample book. The book can be purchased on Leanpub at http://leanpub.com/numpy.

10 Reference

10.1 Data types

10.1.1 Unsigned integer sizes and ranges

10.1.2 Signed integer sizes and ranges

10.1.3 Integer wrap-around

10.1.4 Float characteristics

10.1.5 Complex characteristics