**Introduction to other file types**

Now that you have mastered the art of importing flat files in Python, it is time to check out a number of other file types that you will find yourself needing to work with as a data scientist.

**Other file types**

In this chapter, you will learn how to import Excel spreadsheets, which professionals from all disciplines use to store their data. You will also gain familiarity with importing MATLAB, SAS and Stata files, which are commonplace. You will also learn how to import HDF5 files and you'll actually import an HDF5 file containing data from the Laser Interferometer Gravitational-Wave Observatory project that provided empirical support for Einstein's Theory of Gravitational Waves in 2016. HDF5 files are becoming a more prevalent way to store large datasets, as demonstrated by the fact that the LIGO researchers use it to store their data.

**Pickled files**

Another file type you'll learn about in this Chapter is that of a 'pickled' file. This is a file type native to Python. The concept of pickling a file is motivated by the following: while it may be easy to save a numpy array or a pandas dataframe to a flat file, there are many other datatypes, such as dictionaries and lists, for which it isn't obvious how to store them. 'Pickle' to the rescue! If you want your files to be human readable, you may want to save them as text files in a clever manner (JSONs, which you will see in a later chapter, are appropriate for Python dictionaries). If, however, you merely want to be able to import them into Python, you can serialize them. All this means is converting the object into a sequence of bytes, or bytestream. As this is a course in Importing Data in Python,

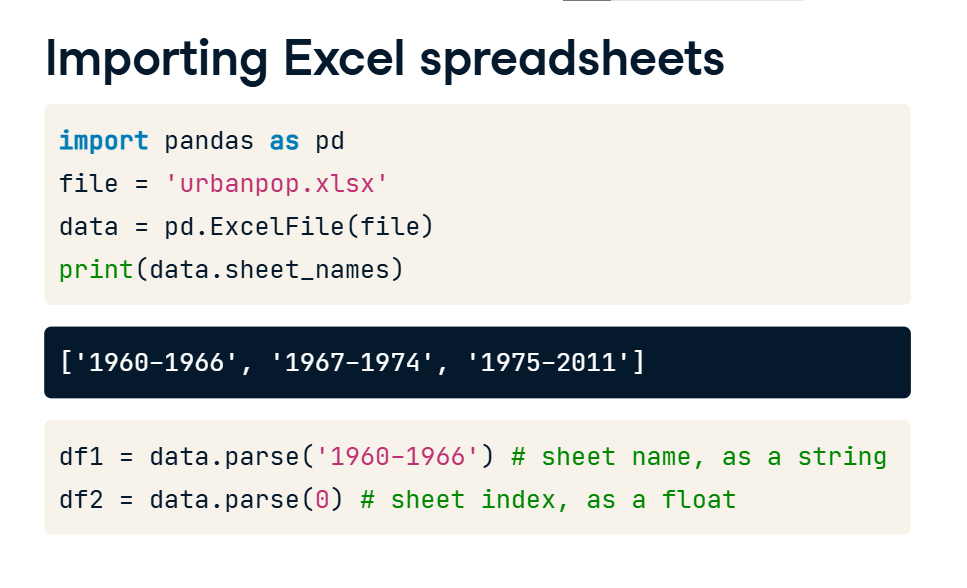
**Pickled files**

you'll learn how to import files that have already been pickled. As you have done before, when opening such a file, you'll want to specify that it is read only; you'll also want to specify that it is a binary file, meaning that it is computer-readable and not human-readable. To specify both read only and binary, you'll want pass the string 'rb' as the second argument of open.



**Importing Excel spreadsheets**

You'll then dive head-first into Excel spreadsheets, the use of which is so widespread that they need next to no introduction at all. An Excel file generally consists of a number of sheets. There are many ways to import Excel files and you'll use pandas to do so because it produces dataframes natively, which is great for your practice as a Data Scientist. As you can see in this example, you can use the functionExcelfile to assign an Excel file to a variable data. As an Excel file consists of sheets, the first thing to do is figure out what the sheets are. This is straightforward with the command 'data dot sheet\_names'. To then load a particular sheet as a dataframe, you need only apply the method parse to the object data with a single argument, which is either the name as a string or the index as a float of the sheet that you wish to load: pandas is clever enough to know if you're telling it the sheet name or the index!



You'll also learn how to customize your spreadsheet import in order to skip rows, import only certain columns and to change the column names. That's enough from me,

**Importing SAS/Stata files using pandas**

There are many statistical software packages out there and, although you may not need to do so all the time, it will be important for you, as a working Data Scientist, to be able to import these files into your Python environment.

**SAS and Stata files**

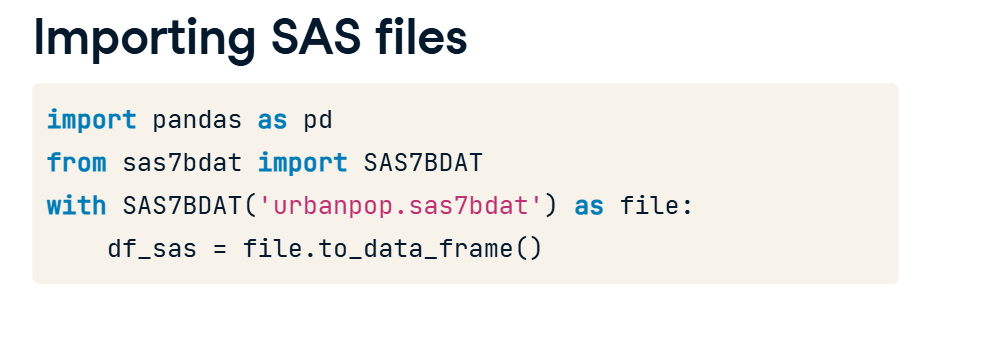
The most common examples are SAS, which is an acronym for 'Statistical Analysis System', and Stata, which is a contraction of 'Statistics' and 'Data'. The former is used a great deal in business analytics and biostatistics, while the latter is popular in academic social sciences research, such as economics and epidemiology.

**SAS files**

SAS files are important because SAS is a software suite that performs advanced analytics, multivariate analyses, business intelligence, data management, predictive analytics and is a standard for statisticians to do computational analysis.

**Importing SAS files**

The most common SAS files have the extension dot sas7bdat and dot sas7bcat, which are dataset files and catalog files respectively. You'll learn how to import the former as dataframes using the function SAS7BDAT (upper case) from the package sas7bdat (lower case). In this case, you can bind the variable file to a connection to the file 'urbanpop dot sas7bdat' in a context manager. Within this context, you can assign to a variable df\_sas the result of applying method to\_data\_frame to file.



**Importing Stata files**

Stata files have extension dot dta and we can import them using pandas. We don't even need to initialize a context manager in this case! We merely pass the filename to the function read\_stata and assign it to a variable, just like this. In the following exercises, you'll gain invaluable experience at importing these important file formats in Python as pandas dataframes and then seeing what was inside them.



**Importing HDF5 files**

According to the 2013 O'Reilly book Python and HDF5 by Andrew Collette,

**HDF5 files**

"In the Python world, consensus is rapidly converging on Hierarchical Data Format version 5, or 'HDF5,' as the standard mechanism for storing large quantities of numerical data." How large are we talking here? According to Collette, "It’s now relatively common to deal with datasets hundreds of gigabytes or even terabytes in size; HDF5 itself can scale up to exabytes." Let's explore with a concrete example from LIGO, the Laser Interferometer Gravitational-Wave Observatory project.

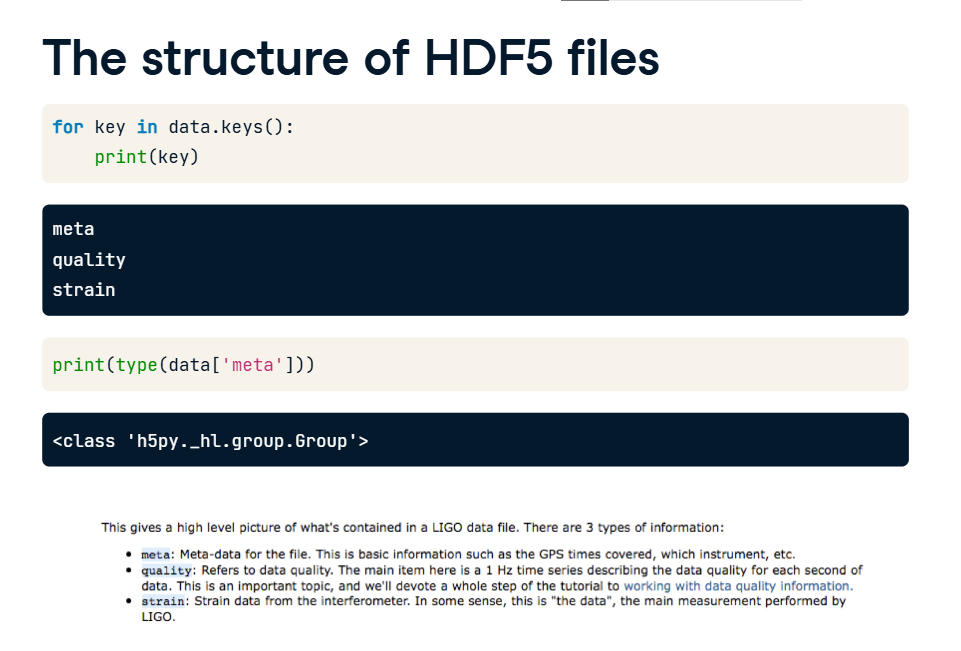
**Importing HDF5 files**

You first import the package h5py and then import the file using 'h5py dot File', remembering to use 'r' in order to specify read only. Printing the datatype to the shell reveals that we are dealing with an h5py file.



**The structure of HDF5 files**

But what is the structure of this file? You can explore it's hierarchical structure as you would that of a Python dictionary using the method keys. You see that there are three keys, meta, quality and strain. Each of these is an HDF group. You can think of these groups as directories. The LIGO documentation tells us that 'meta' contains meta-data for the file, 'quality' contains information about data quality and 'strain' contains 'strain data from the interferometer', the main measurement performed by LIGO, the data of interest. If you knew what data and metadata should be in each group, you could access it straightforwardly. However, if not, due to the hierarchical nature of the file structure, it is easy to explore. For example,



**The structure of HDF5 files**

let's say you wanted to find out what type of metadata there is, you could easily print out the keys. Now you know the keys, you can access any metadata of interest. If you're interested in 'Description' and 'Detector', you can pass these keys to the numpy-dot-array function to convert the values to a NumPy array. You see that the data in the file is 'Strain data time series from LIGO' and that the detector used was 'H1'. Next perhaps you would like to check out the actual data? Great idea and that's precisely what you're going to do in the upcoming exercises!



**The HDF Project**

Before you do so, it is also worth noting that the HDF project is actively maintained by the HDF group, based in Champaign, Illinois and formerly part of the University of Illinois Urbana-Champaign.

**Importing MATLAB files**

MATLAB, which is short

**MATLAB**

for Matrix Laboratory, is a numerical computing environment that is an industry standard in the disciplines of engineering and science. This is due in part to its powerful linear algebra and matrix capabilities, in part to its proprietary nature and in part to how difficult the academic world finds it to shake off old habits. Regardless of the reasons for its widespread use, the fact of the matter is that a lot of people use MATLAB and save their data as 'dot mat' files, the file format native to MATLAB. How can you import these into Python?

**. SciPy to the rescue!**

Luckily for us Python afficionados, the standard library scipy has functions loadmat and savemat, which allow us to read and write dot mat files, respectively.

**What is a .mat file?**

"What exactly is in a dot mat file?" you may ask. To answer this, lets look at the MATLAB IDE. In particular,

**What is a .mat file?**

check out the MATLAB workspace where all your variables are stored. This workspace can contain strings, floats, vectors and arrays, among many other objects. A dot mat file is simply a collection of such objects.

**Importing a .mat file**

Now this means when importing a dot mat file in Python, we should expect to see a number of different variables and objects. In this code, I first import scipy-dot-io and then load the file 'workspace dot mat'. Checking out what type of object results tells me that it's a dictionary. How this dictionary relates to a MATLAB workspace is straightforward: the keys of the Python dictionary are the MATLAB variable names and the values of the Python dictionary are the objects that are assigned to the variables. In the example above, mat['x'] is a numpy corresponding to the MATLAB array x in your MATLAB workspace. It's that easy.

