Essential Python Packages

Introduction to Python Programming

# Overview

In this module, we will review the Python standard library and further explore three of the packages in this library: the *datetime*, *os*, and *glob* packages. In addition, we’ll explore the use of file paths which are essential to effective use of *os* and *glob* packages.

# The Python Standard Library

As mentioned in one of our previous module readings (Module 4), the standard library is a library of Python modules that provides many additional functions, well beyond the functions that are included in the Python built-in functions. By way of review, there are many Python modules available, written by many different developers, and the standard library is a set of modules developed by Python developers that almost always comes with Python. These modules provide essential functions that many other modules are built upon. You can see a list of the modules included in the standard library [here](https://docs.python.org/3/library/).

For further review, please refer to the Module 4 reading in which we covered the standard library and also how to import packages from the standard library into our code.

# The Datetime Package

*Datetime* is a very useful package that allows us to efficiently work with dates and times. We can easily convert strings to datetime objects and extract the properties of the datetime. For example, given a datetime, we can easily get the year, month, day, hour, or second. We can assign and convert timezones. We can easily produce different string formats of the date, for example, ‘2022-11-01’ or ‘Nov 11, 21’. We can also easily calculate the length of time between datetimes. Although we can write our own functions to do all of these things, that would be very time-consuming. Utilizing the datetime package is much more efficient.

# The OS Package

The *os* package allows us to easily interact with the operating system. We can move and copy files, check if files exist, build file paths, and more. The os package can be very useful in basic scripting when you need to systematically download data and store it in a specific location, or you need to move files from one location to another.

# The Glob Package

The *glob* package allows users to easily discover files with a name or location that matches a specific pattern. For example, let’s say there is a directory of data. The directory contains .csv (comma-separated values) files and .txt (text) files. The .csv files contain the data, while the .txt files are just miscellaneous notes. We want to process all of the .csv files with a script, so we need all of the file paths for those .csv files. The glob package allows us to quickly find those file paths because we can ask it to find all files that are in the data directory that end with the file extensions ‘.csv’. Like the os package, the glob package can be very helpful when writing scripts.

# Understanding File Paths

To use the os and glob packages effectively, we need to understand *file paths*. A file path is simply a way to write out the location of a file on our computer. File paths are a fundamental part of programming, though it is still possible to use a computer every day without ever having to use them. This is because modern operating systems represent everything graphically, so instead of typing in file paths, we can easily double click on directories to get to the file we are looking for. Those of you who are already knowledgeable about file paths might just skim through this section.

File paths look different depending on which system you are using. For example, on OS X (Apple) or Linux, a file path will look like this:

/Users/williamhenry/Documents/notes.txt

On Windows, a file path will look like this:

C:\Documents\Newsletters\Summer2018.pdf

In either case, the paths can be read in a very similar way.

## Apple or OS X File Path

Let’s break down the path from the OS X (Apple) computer noted in the first example above.

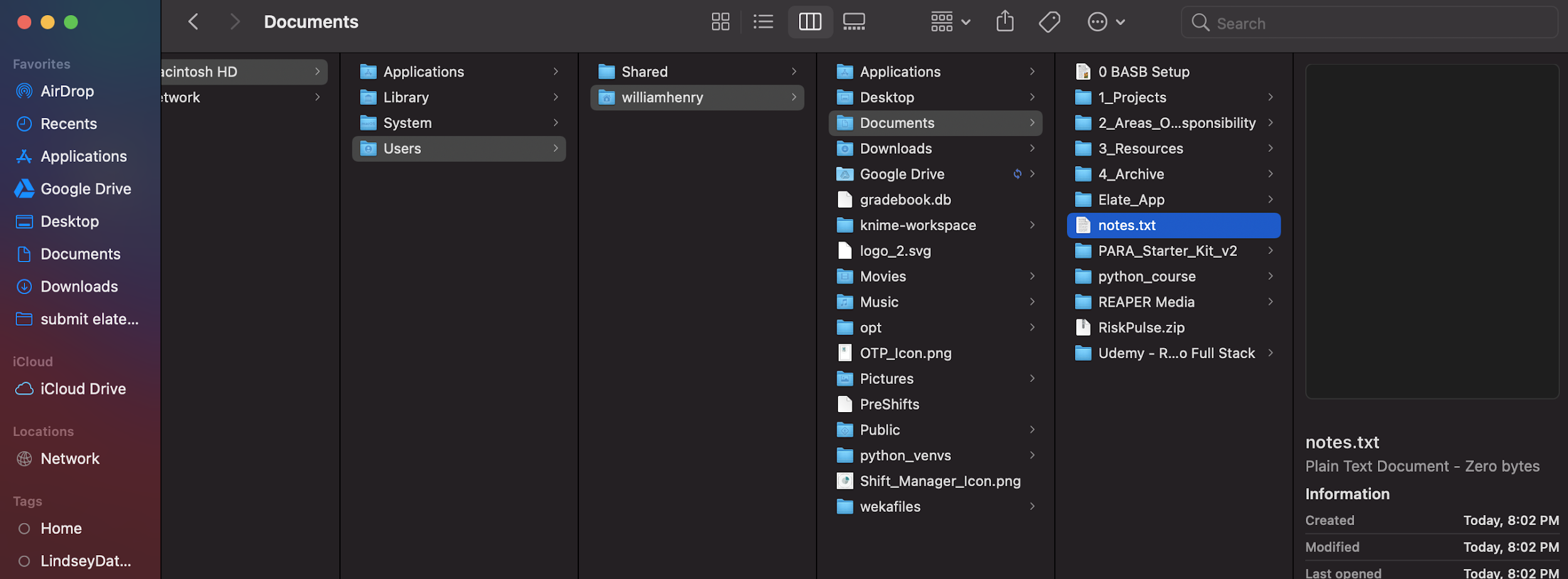
File paths are read left to right, so the first part of this file path is the ‘/’ symbol. This indicates that we are starting from the root directory of our computer. This is the directory that everything else in the computer is inside of. Imagine that your computer is an office building, and inside the office building are a bunch of file cabinets with files. You could think of the office building itself as one large folder or directory. This is like the root directory of your computer; it is the directory that everything else is inside of. A ‘full file’ path, also called an ‘absolute’ file path, always starts from the root. This is an example of a full file or absolute path.

/Users/williamhenry/Documents/notes.txt

The next part of the file path is ‘Users’, and this is the first directory that our file is in. We then read another ‘/’ symbol which also serves as the separator between directories. The next directory is ‘williamhenry’, which is inside of the ‘Users’ directory. Then, the next portion of the file path is the ‘Documents’ directory which is inside the ‘williamhenry’ directory. Finally, we see the last part of our file path, which is ‘notes.txt’, which is the name of the file that the file path is for.

If we were to explain this file pathway in simple terms, we might say that “the notes.txt file is inside the Documents directory, which is inside the ‘williamhenry’ directory, which is inside the ‘Users’ directory, which is in the root directory.” Another way we could say it is, “Start at the root directory, then go in the ‘Users’ directory, then go inside the ‘williamhenry’ directory, then go inside the Documents directory, and there will be the notes.txt file.”

Image 1 below is how the location of this file would look on an Apple computer.



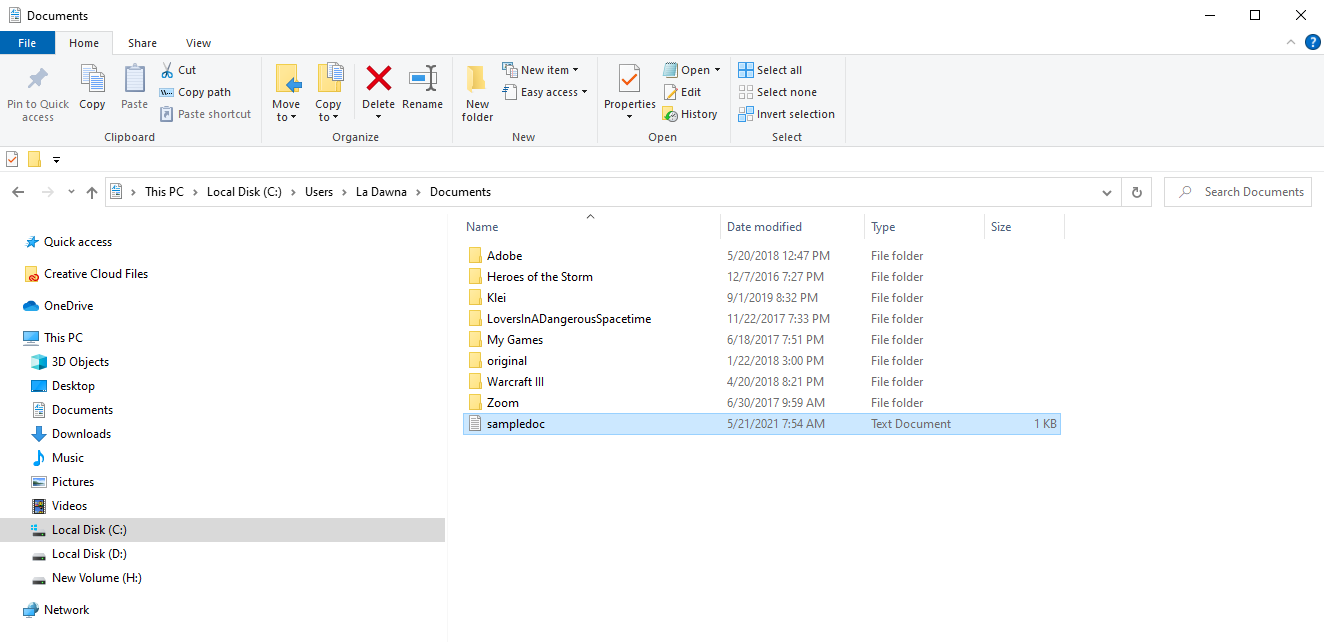
*Image 1: Screenshot of a sample file pathway on an Apple (Mac) computer*

## Windows File Path

Let’s now look at the Windows file path:

C:\Users\LaDawna\Documents\sampledoc.txt

Notice, it is very similar. However, instead of forward slashes (‘/’) used as separators, backward slashes (‘\’) are used as separators. The beginning is also different. It starts with ‘C:\’ instead of just ‘/’. The ‘C:\’ indicates that we are in the root directory of the ‘C’ drive.



*Image 2: Screenshot of a sample file pathway on a Windows computer*

The above file paths for the Apple OS X or Windows systems are called *absolute* file paths because they begin at the root of the computer (or root of the C drive, in the Windows example). File paths can also be *relative*, meaning that they do not begin at the root but begin relative to another location. We will look at relative paths in the next module.

# Conclusion

Being able to utilize the Python standard library is a way for programmers to tap into the many additional functions which almost always come with Python. Part of effectively using two of these functions (the os and glob packages) means that you’ll need to have knowledge of and the ability to use file paths. We’ll continue our discussion of file paths in the next module.