בס"ד

A Little Bit about Paths:

A file’s path is its address in the computer. In Windows, the paths is made up of the drive letter, directories, file name, and extension.

For example, consider the following path:

C:\Python27\python.exe

This tells the operating system (and you) that the desired file is in the C drive (most of your files will be. However, if you want to access a file from a USB drive, for example, it will receive a different drive letter). It is in the Python27 folder, its name is python.exe.

File Extensions

In Windows, each file has an extension, which indicates to the operating system what type of file it is, and what type of program should be used to open it. For example, consider the following extensions:

.txt – the file is a text file

.docx – the file is a Microsoft Word document

.exe – the file is an executable program

.cs – the file is a C# source file

.py – the file is a Python source file

This indicates the type of data that the file contains. Extensions are not mandatory – try creating a file without an extension (Python is useful for this – open a file, for example, called r’C:\file’. ). On attempts to open it, Windows will say it does not know which program should be used. You could right click and choose the desired program. This can be done for any type of file.

Try opening ‘python.exe’ with Notepad++, for example. Or – try creating a file with your own extension (a.testingfile), or with multiple extensions (a.txt.txt.exe). What happens?

Relative Paths

Above, we looked at a complete path. But what happens if you omit parts of the path? Try, for example, opening the following files:  
‘temp1.txt’,

r‘temp\_dir\temp2.txt’,

r’\temp\_dir2\temp3.txt’

and search for these files – where were they created?

Each process has a ‘home’ directory, called the ‘current working directory’. The current working directory starts out as the directory the process started in, but the process can decide to change it.

In IPython, you can do this with:

os.getcwd() # getcwd==get current working directory

os.chdir(r’c:\new\working\directory’)

Back to files: If you fail to provide a complete path for the file, the ‘last’ option is to put the file in the current working directory. This will also work when searching for a file – try playing around with: os.path.exists(), giving a relative path or a full path.

Also, notice that searching for ‘\f.txt’, and ‘f.txt’ is not the same: the first will search for the file in the current drive (\f.txt -> c:\f.txt) and the second for f.txt in the working directory.

Relative paths in Python packages

This also has a lot of implications for Python packages – when installing a package, or importing libraries, Python checks for the package according to a certain order, and part of the order is searching in the working directory. This means that the same section of code might work from a certain place and not work from another.

Try creating a directory with two Python files: foo.py and bar.py. The contents of the foo.py file should be:

import bar

Open command line window in the working directory of your folder (alt+d -> type cmd -> enter, command line window should be opened. Type ‘python foo.py’, it should work

Now, copy foo.py to another directory, and do the same. What happens?

This is because in the first example the bar.py module was found, because it was in the working directory.

However, some libraries, like the os module, for example, are available from any working directory. This is because part of installing the package is adding its path to the default paths that are searched.

This can be the source of many problems – for example, what happens if you create another library in the current working directory with the same name as an installed package?

Backslashes in Python

In Windows, the path separator is the ‘\’ character. As you know, this character is also used to escape characters. For example, ‘\n’ is not a string with two characters, but a single character that marks a newline. This can cause problems – for example, try printing the string:

‘c:\path\newpath.txt’

There are two ways to get around this: the first is escaping the escape character. This means you would write the path as:

‘c:\\path\\newpath.txt’

Another option is using the raw option for python strings. Python offers a marking for a string that automatically escapes backslashes. For example, try printing the string:

r’c:\path\newpath.txt’