Working with the file system

The programming techniques we will look at today all require the following at the top of the program:

```
import os
```

Many of the following tools need a path – a way to get to a file. There are two kinds of paths.

An absolute path starts with a disk and works down to a file. For example,

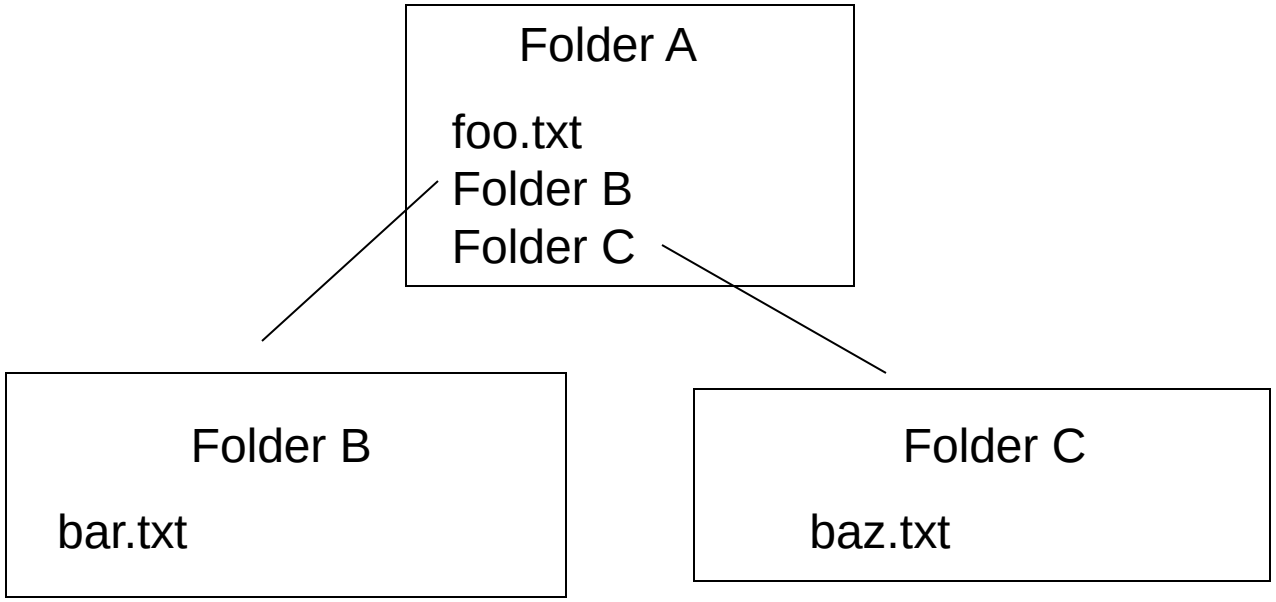
“C:\Users\bob\Documents\Classes\cs150-fall15\Class Examples and Notes\December\December 4”

A relative path starts in the current folder and works its way to a file. There are two handy abbreviations for this:

"." indicates the current folder

".." indicates the folder containing the current folder.

For example, in the following picture Folder A contains file “foo.txt” and also folders B and C, which contain “bar.txt” and “baz.txt”



If you are in Folder A, the relative path to file baz.txt is

“Folder C\baz.txt”, but if you are in Folder B the relative path is “..\Folder C\baz.txt”

There is a small issue in Windows (not on Macs) because Windows uses “\” to separate between folders and Macs and Linux both use “/”. The “\” character when followed by some letters has a special meaning, such as “\n” to indicate a new line or line break. Accordingly, when we specify a path as a string for Windows we use “\\”, as in “Folder A:\\baz.txt”

A few of the most useful tools for working with file systems are

os.getcwd( ) -- returns the absolute path to the current folder

os.listdir(path) -- returns a list of all of the files and folders in the folder at the end of this path

os.path.isdir(path) – returns True if the object at the end of the path is a folder (which some call a directory).

Many functions that work with these tools take as an argument a path, and are naturally recursive: they work with the files at the end of the path and recurse on the directories at the end of the path.