FILE PATHS

A close up of text on a white background

Description automatically generated

**Problem:** File not found error in Python, even if the file is right there in front of you! What gives?

**Answer:** Computers use “paths” in order to find files, which are like sets of directions. If you get a file not found error, that may mean that the path the computer is following isn’t leading to the file. Here’s a bit of info about paths to help you understand them/ troubleshoot this error.

Computers are like trees, where the bottom of the trunk is the root directory, each branch off of that is a folder, and the leaves are your files. The path tells the computer which branches to travel down in order to get the necessary leaf. Each slash is like “cd” in terminal/ Git Bash, telling the computer to move to a different folder (or branch); each word is the folder to go into; each “..” means leave the current folder to the one that contains it (just like in terminal/ Git Bash). There are two kinds of paths: Absolute Paths and Relative Paths.

Absolute Paths are directions to the file from the base of the trunk. No matter where the python file is running from, the computer will always follow the given path from the base of the trunk. As long as the path is valid, the computer will find the file. You can tell that a path is absolute if it starts with “C://” (or “/User/” for mac). That lets the computer know it should ignore where the python file is and start its search from the bottom of the tree. You use the logic of absolute paths any time you go into your class or homework repository from your root directory.

Example: to get to the budget\_data csv, my path will be "C:\Users\Sarah Popelka\Documents\GitHub\GWARL201906DATA1\02-Homework\03-Python\Instructions\PyBank\Resources\budget\_data.csv", no matter where I am running the python file. This translates as “Start at the base of the tree, go inside the users folder, go inside the Sarah Popelka folder, go inside the Documents folder, go inside the GitHub folder, go inside the GWARL201906DATA1 folder, go inside the 02-Homework folder, go inside the 03-Python folder, go inside the Instructions folder, go inside the PyBank folder, go inside the Resource folder, and find the csv called ‘budget\_data.’”

Relative Paths are directions to the file from where the python file is running. These are trickier, because they will differ depending on where the file is running from. If you move the python file, you need to change the any relative paths as well. Relative paths will start with a “/” (no “C:” in front for windows, and no “/User/” for mac). You use the logic of relative paths any time you switch between folders within your class repository in terminal/ Git Bash.

Example: to get to the budget\_data csv, while running my python file from the 03-Python folder, my path will be “\Instructions\PyBank\Resources\budget\_data.csv”. This translates as “start where I’m running my python file, go inside of the instructions folder, go inside of the PyBank folder, go inside of the Resources folder, then find the csv called ‘budget\_data.’” HOWEVER, to get to the budget\_data csv, while running my python file from the PyPoll folder, my path would be “\..\PyBank\Resources\budget\_data.csv”. This translates as “start where I’m running my python file, leave that folder to the folder that contains it, go into the PyBank folder, go into the Resources folder, then find the csv called ‘budget\_data.’”

Using Absolute vs Relative: These each have different purposes, depending on the use case. If you want to be able to share your code easily without having the person change the file path, it’s helpful to use a relative path, because you can include the data and python file in one folder and it doesn’t matter where that folder is saved- the code will still be able to find the data, because the path doesn’t care about anything else outside of the folder that both are contained in. This is why the class repository uses relative paths- no matter where the content is cloned, as long as the python file isn’t moved, it will be able to find the associated data. If you want to point to a specific file whose location isn’t going to change (i.e. you’re not going to be sharing the code/data with people and it’s just going to be running on your machine), or if your file is far away from your python file, then you can just use the absolute path and not worry about whether the computer will find the file, because as long as your absolute path is correct, the computer will be able to find the file from anywhere, even if you move the location of the python file on your computer.

Okay, so I get the difference between Absolute paths and Relative paths, and I think my path should be correct, but I’m still getting an error. What gives?

* Check the spelling of all your folders and the file (don’t forget to check the extension, too!) and make sure that all the names and the extension are valid.
* Use either os.path.join() or put an r in front of your path string, to ensure that your path is formatted correctly. Python assumes that slashes are division symbols, so os.path.join() and r each tell Python that the slashes are actually part of the file path and not division symbols.
* If you are not using os.path.join() make sure that the direction of slashes is the appropriate direction for your operating system (each operating system has a different way of formatting paths, and os.path.join() will automatically format the path based on the operating system, but you need to be conscious of that if you’re not using os.path.join().
* Start at the start of the path you’ve written and follow the instructions laid out in your path, as the computer would (from where your python file is stored). If you can’t get to the file FOLLOWING ONLY WHAT THE PATH SAYS TO DO, then the computer can’t either.

QUICK & EASY WAYS TO GET ABSOLUTE PATHS

PC: navigate to the file in file explorer and shift+right click on the file and select “copy as path.”

MAC: Navigate to the file in Finder and either select it and type command+i and copy the path listed under “where” in inspector, OR drag the file into terminal (and terminal will print the path).

BOTH: In VS Code, you can also right click on the filename in the left-hand panel and click “copy path”