Homework 1

Problem 1: Phone-number forensics (**hw1pr1.ipynb**)

10,000 files? No problem! For Problem 1 this week your challenge is to read and analyze a large phonebook of files (around 10,000). The files are available in the zip archive directory (linked at top of this page). Your task is to ask and answer a few questions of ours and several questions of your own design -- about the phone numbers + names in this large set of files. Then, you'll ask-and-answer three questions about your own local files!

[[Part A]] Phone-number files

The notebook guides you to programmatically access files and folders. Specifically, **`os.walk(path)`** returns a list - once it's converted to a list - of all of the subfolders under **path** (including path itself). Read over and run the examples -- they will be your building blocks for problem 1!

Then, look through the larger folder, named "." which has a full name that specific to your system (for my system, it's `'/Users/zacharydodds/Desktop/sci50/week1_spr24/hw1pr1'` yours will be different.)

Within that large folder are many subfolders, with sub subfolders, etc. And, there are LOTS of phone-number files! Each phone-number file has a phone number on its first line and a name on its second line. Both phone numbers and names come in several formats. (It is the challenge of this problem to write scripts to handle those different formats sensibly and capably!)

Your task is, first, to write functions that find the answer these questions:

- How many **.txt files** are in the entire folder?
    - Hint: the starting file, above, helps a lot on this one! It's many thousands of them...
- What is the **maximum depth** of directories in the entire folder (in other words, what's the maximum number of times that it's possible to move deeper into a subdirectory, overall)?
    - Hint: count the number of forward-slashes and backward-slashes!

    - How to count? Here is an example: 'geese'.count('e') returns 3. 'a/b/c'.count('/') returns 2.
    - Try small examples in a custom cell (it's tricky with the '\\' character, so you'll want to pre-debug this!
    - Key: it's discovery -- don't try to solve it, rather explore + discover your way, zig-zagging towards useful stuff!
- Across all of the files, how many of the phone numbers **contain exactly 10 digits**? 7 digits? 11 digits?
    - Hint: Remember the function from week0 where you extracted *only the* digits? That will be helpful!
    - Here, you might want a function with an *extra input to indicate the number of digits to search for!*
- Of the exactly-ten-digit phone numbers, how many are in the area code 909 (the area code will be the first three digits of a ten-digit number).
    - Hint: cleaning, slicing, Python - yay!

Numbers, Names, and Commas... For each contact-info file, there is a phone number on the top line (there may be distractor characters other than digits that need to be cleaned) and a name on the second line (there may be non-alphabetic characters there that need to be removed). There are two format-types:

That is, if a ***comma*** appears on the second line means that the text on the left of the comma (without any non-alphabetical characters) is a last name and the text on the right is a first name. If there is *no comma*, the text on the left of a space is a first name and on the right of that space is a last name. One could imagine this rule having exceptions in a real set of contact-data, but it would require context *outside the files* to distinguish them. So, for us, we'll take them as the syntax provides.

- How many people have your **last** name?
    - Hint#1: use the presence of a comma to determine whether the last name is 1st or 2nd!
    - Hint#2: watch out for newline characters, '\n' Consider s.split('\n'), s.strip(), s.startswith, ...
    - Choose *another first name present*: How many people have that last name?
- How many people have your **first** name?
    - Choose *another first name present*: How many people have that first name?
- Are there any phone numbers that have *more than* 10 digits?
- How many people have three "i"'s somewhere in their name (not necessarily consecutiiive!)

    - s.count("i") is a friend!   (This can be case sensitive/insensitive: that is up to you.)

As with the file's examples, be sure to have a short "**Results:**" section for each of these answers. In particular, the "what more might be done?" part of that results section may help with asking-and-answering your own questions. That's next!

Three of your own questions for Part A ...

Then, create **three *more* questions** -- of your own design -- that you can ask about the data in these files. For full credit, your questions should expand the *kinds* of questions that can be answered. That is, don't limit yourself to simply asking about a different last name.

Here are a few examples in this build-your-own-question spirit -- feel free to use at most *one* of these, but the other two should be of your own algorithmic or process-design...

- Area codes beginning with "2" are in the northeast; those beginning with "9" are in the southwest. Are there more NE or SW phone numbers across the whole dataset (meaning, area codes beginning with "2" or "9" respectively)?
- How many different last names (or first names) are present across the entire dataset? (Use a dictionary!)
- How many of the phone numbers contain the substring "42" somewhere within them?
- Or, how many have digits that add up to 42 ? 47? Are prime?
- How many files have three "i"s in them?

When you submit, be sure your file has used markdown (or other annotation/context) in order to

(a) share the three questions you wanted to investigate
(b) share what the answers were to your questions, and
(c) do make sure your include your answers (don't delete the results!)

[[Part B]]   **[Exploring your own files!]**

For the second half of this problem, *adapt these scripts and cells and apply them to a directory somewhere in your own filesystem!* You may have ***lots*** of files (I do!) or you may have only a few. Be sure it's somewhere you have at least 42 files, so there are enough to ask some questions.

- That is, choose a *different* top-level folder and then adapt your prior investigations in order to answer these questions:

    - how many files are there total? (Choose somewhere with *at least* 42 files.)
    - how deep was the deepest path present?
    -
- **Three of your own questions for Part B:**
- That is, for this part -- ask and answer three more questions of your own design, about your files. They can be serious, e.g., number of .py vs .java files. (Hoping for more of the former!)   Or, they can be whimsical, e.g., how many contain the string '42' somewhere in their contents in ascii? in binary?! How many files with "Screen Shot" in their file name? What was the most common file-extension present in your file-set, e.g., .png, .txt, .py, etc. And so on...   *These are just seed-ideas. Feel free to use some of these - and create at least one of your own.*

As before, you'll submit your answers - and short reflections on them - as part of your notebook.

Again, ***do not*** submit your files themselves! Feel free to clear-contents of any outputs not needed (especially any really large ones). Do be sure include the questions and answers you found.

Creative and unusual custom-questions (with answers) are always welcome.

Good luck with homework #1, everyone!