Word Counting 1.0

Assignment instructions

For this assignment, you are going to create a small program that lets a user count specific words in a text file.

However, I'm not going to walk you through step-by-step. Instead, I'm going to give you a list of "user requirements", where I list things that the (hypothetical) users of this program want it to do, and then it's up to you to figure out a way to do it.

I will give you a series of "hints", though, to help you get started.

User requirements

These statements are from the user's perspective:

1. I want to be able to run this program at the command line.
2. I want the program to tell me which directory I am currently in (aka my working directory), and I want it to list all of the files and folders in that directory.
3. I want the program to ask me what file to search and what word to search for.
4. I want the program to count all the times it finds that word in that file.
5. I want to be able to search texts that are potentially in different folders from where I'm running the program.
6. I want the program to tell me how many times that word occurs in that file, but in a prettier way than just spitting out a number. I'd like it to say something like: "The word <WORD> was found <NUMBER> times in this file: <FILENAME>"

Submission

In order to complete this assignment, download the whole `word_count_program.zip` file from ELMS. When you unzip this file, you should see a file called `word_count.py` and a folder called `monty_python_sketches` that contains three text files. A copy of this instructions file is also included. Edit the `word_count.py` file and submit ONLY THAT FILE for the Word Counting 1.0 assignment.

Grading

If you complete the user requirements, you will be able to score points in Core Python Competencies, which are the primary source of points in the course. But you will only score points if the code actually works and contributes to the program. Extraneous code that shows off something for no reason will not score you points.

In this assignment, you will be able to score points on Expressions & Variables, Functions & Operators, Modules & Methods, and Input/Output. If you complete all the requirements appropriately, you may score at least 8 points.

Note that points are **scored** for good coding, not **deducted** for incomplete code or mistakes, so do as much as you can!

Hints

1. Before you start writing Python code, try making a step-by-step outline of what you want the program to do, using comments (which start with the # symbol).
2. Note that the user requirements are a list of things they want it to do, **not necessarily** a list of steps in the order you should do them. You need to figure out the steps yourself for filling all the requirements.
3. When I say "run at the command line" (requirement #1), I mean that the user wants to use the Terminal (or Anaconda Prompt) **outside of VS Code** to run the `word_count.py` file. **You** can (and should) still use VS Code the normal way when working on your code, but when you want to test your final program, you should run it on your Terminal program (if on Mac) or Anaconda Prompt (if on Windows), to make sure it works like the user wants it to.
4. The user requirements may describe things in a vague way. This is intentional, because being a good programmer often means figuring out what you think the user really wants. Not many customers tell you exactly how to accomplish what they want, that's *your* job!
5. Because of this, there are different ways you can choose to fulfill the requirements as described. Choose a way that makes sense to you that you think would work well if you were the user of this program.
6. Remember that you can use the function `listdir()` from the `os` module to get a list of all the files and folders in a directory.
7. Remember that uppercase and lowercase are different, and you should think about how to handle that when searching for a word.
8. If you are having trouble, try to troubleshoot one step at a time, to make sure each part of your program works, and to help isolate what is going wrong.