CS 1301 Homework 9

Homework – Functional Programming and File Navigation!

Due: Friday, April 19th before 11:55pm

THIS IS AN INDIVIDUAL ASSIGNMENT!

You should work individually on this assignment. You may collaborate with other students in this class. Collaboration means talking through problems, assisting with debugging, explaining a concept, etc. Students may collaborate with only fellow students currently taking CS 1301, the TAs, and the lecturer. You should not exchange code or write code for others. For individual assignments, each student must turn in a unique program. Your submission must not be substantially similar to another student's submission. Collaboration at a reasonable level will not result in substantially similar code.

Scored out of 100 points

Files to Submit:

hw9.py ***(make sure to complete all 3 parts!)***

If you need help, we have several resources to help you successfully complete this assignment:

- The TA Helpdesk – Schedule posted on class website.

- Email the TAs

- Jay's office hours

Notes:

• Don't forget to include the required comments and collaboration statement (as outlined on the course syllabus).

• Do not wait until the last minute to do this assignment in case you run into problems.

• If you find a significant error in the homework assignment, please let a TA know immediately.

Part 1- averageRGB

For the first problem, you will write a function called *averageRGB*, which will use reduce to help calculate the average red, blue, and green values of pixels in a picture. The function does not need to take in any parameters. First, your robot will take a picture. You do not need to show or save this picture, but you may if you want to for testing purposes.

Your function should go through each pixel and put its color values in a separate list, meaning once you're done going through the picture, you'll have three lists, one of red values, one of green values, and one of blue values.

You should then **use reduce** to help compute the average (use it to sum up all the individual values in each list). You may use a helper function or a lambda function with reduce. You must use reduce somewhere in your function. Once you have your three averages, your function should **return** those three averages in a tuple, with the red one first, the green one second, and the blue one third. Because these are color values, you should round each average **to the nearest integer**.

Sample function call:

```
>>> averageRGB()

(34, 209, 154)

>>>
```

Nothing should be output to the shell until the final return statement. You can use pictures you have saved on your computer to test this program without having the robot take a picture, but your final version should use a picture taken by the robot. You may create helper functions to assist your main function.

Part 2- findFiles

For the second part, you will write a function called *findFiles*. This function should take in an absolute file path as a parameter and return two lists, one that has the names of all Python files and another that has the names of all text files. However, these lists need to contain just the filenames and not any of the file extensions, meaning that *example.txt* would be returned as *example*.

You can assume that the only two extensions you'll be looking for are **.py** and **.txt**.

Your function should take in an absolute file path as a parameter. It should then use listdir (from the os module) to get alist of all of the files in the directory. Use **filter** to go through the list and store all filenames of python and text files into one or two lists. Whether or not you separate them into two lists when you first store them is up to you.

In order to navigate through filenames, you'll need to use the OS module. The Python webpage for the OS module will be very useful, as will functions such as *os.listdir*.

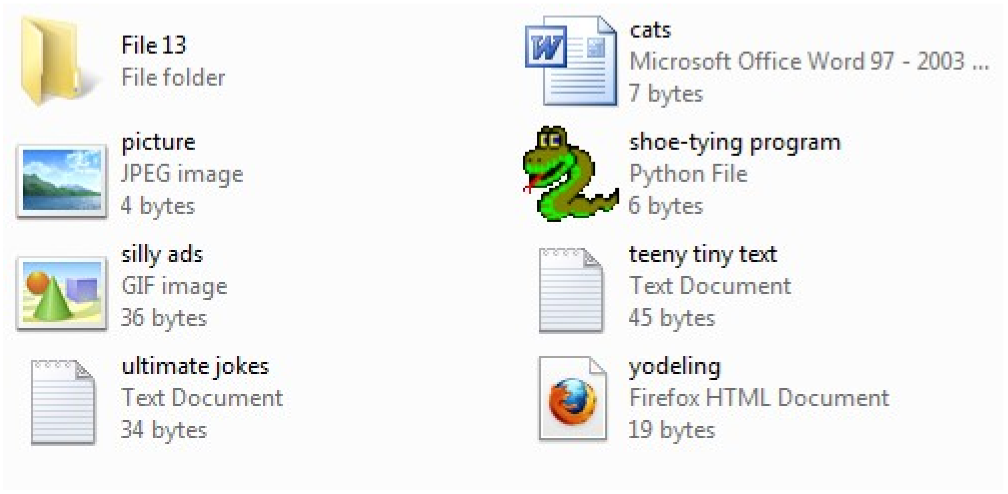
Once you have all your filenames, your function should **use map** to go through your list(s) and take the extensions off the filenames. This should not modify the actual files, just the names you stored in your list(s).

You should then return a **tuple of length two** in which each item in the tuple is a list, the first with all the python filenames and the second one with all the text filenames.

Sample function call:

Assume this absolute file path exists: **C:\Users\Me\Desktop\Folder**

…and its contents look like so:



File 13
File folder

cats
Microsoft Office Word 97 - 2003 ...
7 bytes

picture
JPEG image
4 bytes

shoe-tying program
Python File
6 bytes

silly ads
GIF image
36 bytes

teeny tiny text
Text Document
45 bytes

ultimate jokes
Text Document
34 bytes

yodeling
Firefox HTML Document
19 bytes

Running *findFiles* on this folder would look like this:

```
>>> findFiles("C:\Users\Me\Desktop\Folder")
(['shoe-tying program'], ['teeny tiny text', 'ultimate jokes'])
>>>
```

Again, you may create helper functions to help you with this function.

Part 3- findAllFiles

The last part of this homework is an extension of the second part. You're going to write a function called *findAllFiles* that will do the same as *findFiles*—taking in an absolute file path as a parameter and going through and finding all python and text files—except that you should check **all** folders inside the initial directory and any folders inside those folders, and so on.

You are encouraged to use the function you wrote for part two to help with part three. You may use recursion to solve this problem.

Once again, you'll need to use the OS module, particularly functions on this page. Pay special attention to the os.path.isdir() function.

Sample function call:

If you wanted to run *findAllFiles* on **C:\Users\Me\Desktop\Folder**, where the folder called File 13 contains a python file called *robot dance.py* and a text file called *grocery list.txt*, then running *findAllFiles* would look like this:

```
>>> findAllFiles("C:\Users\Me\Desktop\Folder")
(['shoe-tying program', ‘robot dance’], ['teeny tiny text', 'ultimate jokes', ‘grocery list’])
>>>
```

If there were any folders inside File 13, they would need to be checked for python and text documents as well. In order to handle sub-directories, you may need to use recursion.

Grading Rubric

Part 1- averageRGB- 20 points Total

Robot takes picture	2pts
Collects and separates color values	3pts
Uses reduce	5pts
Correctly computes averages	5pts
Returns tuple that contains the three averages	3pts
Averages are rounded to the nearest whole number	2pts

Part 2- findFiles- 40 points

Takes in absolute file path	3 pts
Gets filenames from absolute file path	2 pts
Uses filter correctly	10pts
Removes extensions from filenames	5pts
Uses map correctly	10pts
Returns tuple that contains correct lists	10pts

Part 3- findAllFiles- 40 points

Takes in absolute file path	5pts
Goes through all sub-folders (and sub-sub folders, etc) to find files	25pts
Returns tuple that contains correct lists	10 pts