Lab 3: Ins and outs of files

Overview

We'll work on how to find, read, and write basic file types, and use that goal to pick up another few key concepts: dictionaries, tuples, try-except blocks, and iterators.

Prerequisites

Make sure you've gone through the next section of Codecademy through "Exam Statistics" exercise. It will also be useful as you work through this week's material to reference the Python.org site on dictionaries, tuples, iterators, try-except blocks, the "os" module, and the "csv" module.

Reminders

Journals: Keep a journal for this lab, separate from the prior lab. Use the naming convention from before: <yourlastname>_<yourfirstname>_lab3_journal.docx.

Checklist of deliverables

Lab is due at the beginning of the next lab (2pm) on Thursday January 28. When you complete the lab, use Zip or 7Zip to bundle these files and place the zipped file in the Geo578/Drop/Lab3 folder:

Lab 3 Journal: <yourlastname>_<yourfirstname>_lab3_journal.docx or .pdf
Section 2: <yourinitials>_lab3_functions.py
<yourinitials>_l3s.py

Section 1:

Find and work through the Codebook from Week 3.

Steps

1. Files:
    a. We'll be working from the code book. Make a new directory in your student directory called "week3", and **copy the codebook files** from Classes\Geo578\Data\Codebooks\week3\ to your directory
    b. NOTE: There are both the code files and some files we'll be reading and writing to. Get all of 'em.
2. NOTE: I've changed the file naming convention a bit to make it simpler to spot bugs in one vs the other.
    a. The functions file is called: codebook3_functions.py
    b. The script file is called: cbk3.py
3. Save these to your own folder.

Section 1: Deliverables

None.

Section 2:

Make your own functions!

This week, we move to an approach that gives people more flexibility as their skills allow. Here's how it works:

1. We'll all work on the "standard fare" programs. Those are worth 15 points total.
2. Then, you will choose from a second group menu of programs to reach 15 points.

Section 2 Deliverables

1. In your lab3_journal.docx file, document your trials and travails (if any) and the logic of what you're trying to do.
2. Make your own files. Again, note the slightly different naming, which I hope will make it easier to not get confused about which file errors are showing up in:
    a. <yourinitials>_lab3_functions.py
    b. <yourinitials>_l3s.py
3. As before: Define the functions in the "functions.py" file, and develop variables in the "l3s.py" file and call the functions.
4. Use the rules on good commenting as in prior labs: Comment each logical section within your code and introduce each function well. Then, within each function, comment the lines where anything new happens.
5. In your "functions.py" file, include all of the functions to meet your chosen functions below. In the comments for each function, make it clear which directive that function is responding to from the list below.
6. If you use a text file as a tester for a program, please include that so we can also test it.

Standard fare programs: Do all of these

Program 1: (5pts). Adapt the "is_numeric" function to acknowledge that float and int are not the ONLY numeric types! Check on Python.org to find at least two other numeric types, and change the "is_numeric" function to correctly recognize those as numeric types. In your scripts file, show that it works by passing at least one of the two other types.

Program 2: (5pts). Adapt the "assign_shape_params" function so it will test whether dim1 and dim2 are **numeric** and **both positive** numbers, returning to the user with a complaint if either one is not.

Program 3: (5pts). To the function "calc_shape_area," add calculations of area to the other shapes that are defined in the "assign_shape_params" function.

Pick a combination to reach 15 points. If you want up to five bonus points, you can pick programs that add up to 20. Note: In each of these point totals, we calculate that

20% of the score is for good commenting of your code, 50% for reasonable code, and 30% for results (i.e. working code). For all of these, ensure that potential hazards are insulated in try/except blocks that handle exceptions gracefully.

Program 4: (5pts). To the "headfootfile" function, add code to write the line number of the file at the front of each line, starting at 1 and numbering every line. Then do one more manipulation of your choice to change the input and output. *Hint: A counter variable can be incremented by 1 using: counter += 1.*

Program 5: (10pts). Develop a function that scans a directory for files of a certain type (i.e. ".txt" or ".py" or ".shp", etc.), sorts them according to a file characteristic (e.g. time stamp of creation, size of file in bytes, etc.) and writes a new text file with the names of all of the files that meet some criterion of that file characteristic (e.g. only new files, only files of a certain size, etc.).

Program 6: (5pts). Develop a function that reads a ".py" file, converts all commented lines to all-capitalized letters, and writes out a new copy of the file with "allcaps" inserted into the original filename.

Program 7: (10pts). Develop a function that writes a text file with the filenames of all files in a given directory, and for each file listed in the text file, shows a descriptive name for the file type. For example, an MS Word document that ends in either '.docx' or '.doc' would be listed in the output file as "An MS Word Document", a '.txt' file would be listed as "A text file", etc. *Hint: A dictionary object is a good way to do lookup tables!*

Program 8: (5pts). Develop a function that reads in a text file and writes out a copy where every numeric value in the file has been manipulated in some way (e.g. multiplied, etc.).

Program 9: (10pts). Develop a function that develops summary metrics on each '.py' file in a given directory, and writes out the list of files with associated summary metrics to a new text file. Summary metrics for each file could be: Proportion of lines of code commented vs. not, number of try/except blocks used, number of if/else statements used, number of occurrences of some string regular expression, etc. Test it on one of the Python distribution directories (i.e. a directory with a lot of '.py' files!), but **be careful not to overwrite any files**!!!

Program 10: (5pts). Develop a function to produce a text file with a list of filenames in a given directory sorted by the length of the filename in characters.

Program 11: (10pts). Develop function to read a '.csv' file with X, Y coordinates, and then use it to develop a program that will find the closest X, Y coordinate in a text file to a given target X, Y. coordinate. You can use any tool you'd like to read the '.csv' file.