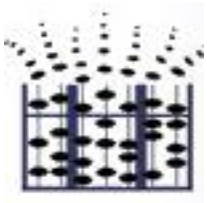

CIS 211
Spring 2014

Project 8: Scripts

Due Friday June 6
Upload via Blackboard by 11:00 P.M.

Reading

Lecture notes for May 30 to June 4 (available on the class web sites).

Warning

This week's projects uses Python library functions that interact with your operating system, including functions that delete files. We strongly recommend following the incremental development style for these projects. While you are debugging your program, have it print a message like "deleting file X" instead of calling the function that deletes the file. When you are sure the program is doing the right thing add the statement that calls the function that deletes the file.

Test Data

The projects require you to write a set of "scripts" that work with files and directories. To test the scripts, download a file named `FederalistPapers.tgz`. When you unpack the file you will have new folder named `FederalistPapers`.

This folder has a set of files created by LaTeX, a document formatting system widely used in math and science. Files with names that end with `.tex` are the document files; these are plain text files containing markup commands that tell LaTeX how to format the document. When LaTeX runs it generates lots of temporary files with names that end `.aux`, `.log`, etc.

A Note for Windows Users

If you use Microsoft Windows you will need to download and install a package named Cygwin. This package includes `wc`, `grep`, and several other useful Unix utilities.

Programming Projects

1. Write a program named `cleanup.py` that deletes all the temporary files from a LaTeX project. The command line argument should be the name of the top level directory containing the main `.tex` file. The program should delete all the temporary files in that directory and in any subdirectories.

   When your program runs it should print the names of the files it is deleting. This is what you should see when you run the program with the `FederalistPapers` directory:

   ```
   % python3 cleanup.py FederalistPapers
   FederalistPapers/TheFederalistPapers.aux
   FederalistPapers/TheFederalistPapers.log
   FederalistPapers/TheFederalistPapers.pdf
   FederalistPapers/TheFederalistPapers.toc
   FederalistPapers/Legalese/license.aux
   ```

```
FederalistPapers/Number1/federalist1.aux
FederalistPapers/Number2/federalist2.aux
FederalistPapers/Number3/federalist3.aux
```

For this project a temporary file is any file that has a name that ends with one of these extensions:

```
.aux  .idx  .ilg  .ind  .log  .pdf  .toc
```

2. Write a program named `backup.py` that copies the files from a LaTeX project to a backup directory. Name your backup directory `versions`. The first time the program runs it should create the `versions` folder and then copy the project to a folder named `1` inside `versions`. If the program is run again it should create another new folder inside `versions`. The second backup will be saved in `versions/2`, the third in `versions/3`, and so on.

3. Write a program named `pstats.py` that will compute some basic statistics about a LaTeX project. The output will use the same format as the Unix utility named `wc` (word count). This is what you should see when you run your program with the `FederalistPapers` directory:

```
% python3 pstats.py FederalistPapers
       4       8     915    7546  FederalistPapers
```

This output says the `FederalistPapers` project has 4 chapters and 8 sections, and that there are a total of 915 lines and 7546 words in all the `.tex` files.

To count the number of chapters your program should run the Unix utility named `grep` with this command (where `file.tex` is the name of one of the LaTeX source files):

```
% grep '\\chapter' file.tex
```

To count sections use the same command, but with `section` instead of `chapter`. To count the number of lines and words in a file your program should run the `wc` command and capture the results.

Hint: use Python's "raw string" format when you create a string containing a Unix command. For example, this is the best way to make the command that searches for lines containing the LaTeX `\chapter` command:

```
cmnd = r"grep '\\chapter' {}".format(filename)
```

See the lecture notes for more information about raw strings and why they're useful.

Extra Credit Ideas

- Have your backup.py script ignore temporary files (files with names ending `.aux`, `.idx`, etc).
- Instead of deleting files copy them to a trash folder. Either create a trash folder in the project directory or figure out how to copy to your computer's trash folder.
- Write a script that recovers deleted files by removing them from the trash and restoring them to the project directory.

What to Turn In

Documentation Write a short description (two or three paragraphs total) of what you did for this project. The documentation should be in a file named `writeup` with an extension that identifies the file format (.doc for Microsoft Word, .pdf for Adobe PDF, .txt for plain text, .rtf for rich text format).

If you did anything extra for this project describe it in a separate paragraph at the end of your writeup.

Create a package (tar or zip format) that includes your writeup and your three Python scripts and upload the package via Blackboard.