Programming Project F

Directories, Databases, and Zip Files

Background

Folders, or directories, are the universal way of organizing files on general-purpose computers. Databases are the preferred method of storing data for subsequent retrieval. Files are often compressed to save space, especially when transferring them from one place to another. In this project you will use process directories and store their contents in Zip files and a relational database table. We will use the **sqlite3** module that comes with Python for the SQL database portion. (Note: we will do no joins or complicated queries; we'll just process a single table.)

Requirements

This project is divided into 3 parts, and you will write a separate, short Python script for each part.

Part 1

In this part you will write a script named *readfiles.py* that takes a directory name as a single command-line argument. The directory, *Stuff* contains over 1,000 files of various types occupying 56mb of disk space in itself and many subdirectories. Download the zip file Stuff.zip from https://chuckallison.com/cs3270 and unzip it into a directory tree onto your system. The execution

```
$ python3 readfiles.py Stuff
filesdb created
```

discovers all files in the associated directory subtree. It stores the information in a Sqlite3 database file named **filesdb**, in a table named **files** containing 3 fields: (**ext**, **path**, **fname**), which represent the file extension (*.cpp*, *.docx*, py, etc.), the full path name of the *directory* (only, not including the file name) the file resides in, and the name of the file (e.g., *foo.cpp*). Here is a sample of some of the entries in the **files** table:

```
('doc', '/Users/chuck/UVU/3270/Assignment8/Stuff', 'OOPSLA-Decimals.doc')
('doc', '/Users/chuck/UVU/3270/Assignment8/Stuff', 'OOPSLA-Python.doc')
('ppt', '/Users/chuck/UVU/3270/Assignment8/Stuff', 'XP.ppt')
('ppt', '/Users/chuck/UVU/3270/Assignment8/Stuff/ACCU', 'ACCU-Templates.ppt')
('ppt', '/Users/chuck/UVU/3270/Assignment8/Stuff/ACCU', 'CodeQuality.ppt')
('cpp', '/Users/chuck/UVU/3270/Assignment8/Stuff/ACCU', 'Counted.cpp')
...
```

If a file does not have an extension, use **None** for that field—Sqlite will then store that as a NULL in the database table. Output the result of the query "*select * from files*" to a text file named *files-part1.txt*. You will submit this file along with your source code.

Part 2

In this part you will write a script named *gatherfiles.py* that receives the name of your database file and one or more file extensions, for example

```
$ python3 gatherfiles.py filesdb cpp py
```

This script opens the database file and then does the following for each file extension appearing on the command-line:

1. Retrieves the file name information for all files with that extension, and forms the full pathname of that file
2. Adds the contents of that file by adding the file to a zip file using the full pathname as its name, so that the directory structure is preserved. The name of the zip file is the name of the extension followed by ".zip", for example, *cpp.zip*.
3. Prints to standard output the number of files added to the zip file.

After executing *gatherfiles.py* with the command-line arguments shown above, the files *cpp.zip* and *py.zip* would be created in the current directory, and the following would be written to the console:

```
658 cpp files gathered
19 py files gathered
```

You should be able to unzip the output zip file and verify that the original directory structure is reproduced.

Part 3

The third script, extractfiles.py, extract from a zip file all files whose basenames i.e., (file names without any path prefix) match a regular expression, for example

```
$ python3 extractfiles.py cpp.zip ^[Cs].*
81 files extracted
```

The command above will extract all files from *cpp.zip* that begin with 'C' or 's'. Note that **zipfile.extract** uses *full path* information, placing extracted files in their associated directories, so you may have to execute this script in or extract the files to a directory other than the one where the original directory you're extracting exists.

Implementation Notes

In Part 1, ignore file names that start with '.' or '_'.

Submit your three scripts, output text file from Part 1, zipped output directory from Part 2, and zipped output directory from the **extractfiles** execution shown above together in a zip file (so you will have two zip files inside the outer zip file that you submit).