Practical 09 - Automation, using OS

Today we will explore Python as an automation tool to help us automate repetitive tasks with files and the **os** module.

# Walkthrough Example - Renaming Files

Download the files from <https://github.com/CP1404/Practicals2016/tree/master/Prac10>

* Extract the **Lyrics.zip** file into the project directory so it’s in a subdirectory called Lyrics.
* Open the directory Lyrics/Christmas so you can see the files listed in your file browser or PyCharm   
  (but note that PyCharm doesn’t always refresh as often as you might like).
* Open the file, **cleanupFiles.py** and run it.

**Notice:**

* It imports the **os** and **shutil** modules for working with the operating system and files
* Comments explain what parts of it do. There are two commented-out options. Try each, one at a time:
  + **rename** files by replacing spaces with \_ and “.TXT” with “.txt” (in same directory)
  + **move** files to a subdirectory with the new name

**Note:** renaming files changes their names (amazing!), so to re-run your code with the files in their original state, you can copy them from the Lyrics.zip file provided.

The files we’re working with today are from a **real-world example**. Lindsay uses words projection software at church that takes these files. The modifications are real needs, so this is another example of using (Python) programming to automate tasks in your (or someone else’s) everyday life! Nice :)

### Modifications:

1. Notice that the file naming has been done inconsistently, e.g. some are PascalCase like “SilentNight.txt” and some have spaces like “Away In A Manger.txt” or are not in Title Case like   
   “O little town of bethlehem.TXT”  
   Write code to make them consistently use the format like “Away\_In\_A\_Manger.txt”, “Silent\_Night.txt” and “O\_Little\_Town\_Of\_Bethlehem.txt” respectively:

|  |  |
| --- | --- |
| **Existing Filename (inconsistent format)** | **Desired Filename (consistent)** |
| Away In A Manger.txt | Away\_In\_A\_Manger.txt |
| SilentNight.txt | Silent\_Night.txt |
| O little town of bethlehem.TXT | O\_Little\_Town\_Of\_Bethlehem.txt |
| ItIsWell (oh my soul).txt | It\_Is\_Well\_(Oh\_My\_Soul).txt |

**Important:**

Do NOT try and solve all of these cases at once. Rather, work up to them, testing a function like **get\_fixed\_filename(name)** that returns a fixed filename. Test just printing the names before renaming the files. When it works for one case, add to it to handle another one… iterative development!

1. Currently the program runs only in one directory.  
   Make it work for all directories. You can do this a number of ways, including using **os.walk()**, which loops through all subdirectories, giving you (each time): directory name, subdirectory list, file list.  
   Uncomment the os.walk() example at the bottom of the cleanupFiles.py example file to see how it works.

## From Scratch

* Extract the **FilesToSort.zip** file which contains files with various names and extensions.
* Write code to sort these files into subdirectories for each extension.

### Version 1:

Use **os.mkdir()** to create a directory with for each new extension that you see and use **shutil.move()** to move files into these.   
E.g. move all files ending in “.txt” to a directory you create called “txt”, and all “.doc” files to a “doc” directory.

Do not try and create directories you’ve already made.

**Tip:** You might like to add the extensions to a list as you process the files.

|  |  |
| --- | --- |
| **Before:** | **After:** |
| Screen Shot 2015-10-20 at 1.29.53 pm.png |  |

### Version 2:

Let the user categorise different extensions as the program encounters these, then move them all into those subdirectories.   
E.g.

* one user might want a category “docs” containing all .doc, .docx, .rtf, .txt and “images” containing .jgp, .gif, .png.
* another user might want a category “office” containing .doc, .docx, .xls, but put the .txt files in a “text” category directory.

**Tip:** You might like to add the extensions to a dictionary and make a list of the categories as you process the files.

**Note:** there are two parts to this - **categorising the extensions** and **moving the files**. You should approach them as separate steps.

For one example run with these files (user input in **green**):

What category would you like to sort doc files into? **Docs**

What category would you like to sort docx files into? **Docs**

What category would you like to sort png files into? **Images**

What category would you like to sort gif files into? **Images**

What category would you like to sort txt files into? **Docs**

What category would you like to sort xls files into? **Spreadsheets**

What category would you like to sort xlsx files into? **Spreadsheets**

What category would you like to sort jpg files into? **Images**

|  |  |
| --- | --- |
| **Before:** | **After:** |
| Screen Shot 2015-10-20 at 1.29.53 pm.png |  |

# Extension & Practice Work

### Check files for missing data:

The song lyric text files should all have copyright information in them on a line that starts with **.i** like:

.i © 2011 Thankyou Music (Admin. by Crossroad Distributors Pty. Ltd.)

Write a program that reports the names and locations of all of the files that are missing this line.

**Version 2**

Automatically look up the copyright information from the Internet based on the song title and author, then add the data to the file… Good luck with that ;)