ANSWER THE RIDDLE

HOMEWORK 3, DUE THURSDAY, MARCH 27, 2014

Objective

To write a program to find the names of all birds mentioned in Henry David Thoreau's *Walden* by cross-referencing the text of *Walden* against the NCBI taxonomy database of all known common bird names.

Files

- `biof309_hw3_directions.pdf` - This file.
- `biof309_hw3_template.py` - A partially completed program where you fill in the blanks.
- `biof309_hw3_files.tar.gz` - (21MB) - zipfile containing raw data files including:
  - `guess_my_gfs_name.png` - Contains my girlfriend's riddle
  - `taxonomy_files_readme.txt` - Tells you what columns are what in the taxonomy files - provided only for your interest, not necessary to complete assignment.
  - `walden.txt` (0.5MB) - The entire text of Henry David Thoreau's *Walden*
  - `nodes.dmp` (66MB) - Complete NCBI taxonomy database structure, containing taxon id (integer), parent taxon id (integer), taxon rank (kingdom, class, genus, order, species), etc.
  - `names.dmp` (82MB) - Scientific and common names for each corresponding taxon in `nodes.dmp`.

Background

When my girlfriend first introduced herself to me, she gave me a riddle to guess her name: that it was the name of a bird that was mentioned in passing in the text of Henry David Thoreau's *Walden*. So I used NCBI taxonomy data to obtain the name of every known species of bird (class *Aves*), and I cross-referenced that against *Walden*. In this homework you'll retrace my steps!

Assignment Overview

Most of the program has been written already. You must complete the function `ReturnSetOfWordsInFile`. You are allowed to add or change the main part of the program at the bottom of the template to do the final cross-referencing, but it is not allowed to change any other part of the program to make your code work.

In the template file, I have included three functions:

(1) `BuildTaxonNamesDict` parses the `names.dmp` file and returns a `dict` mapping taxon node ids to taxon names. **Important:** if the taxon has multiple common names the function will concatenate them with a `'/'` delimiter, see Tips section below for more info.

(2) `BuildTaxonomyTree` parses the `nodes.dmp` file and returns two `dict`s, the first mapping node ids to a list of child nodes, the second mapping node ids to node taxonomy rank (kingdom, class, genus, order, species, etc..)

(3) `TaxonomyWalker` is a recursive function that takes the `dict`s generated above as well as a starting taxon and returns a `set` of names for all child taxa that are species or subspecies.

Additionally in the `__main__` part of the program at the bottom of the template I have called the functions for the class *Aves* (node# 8782) so that you already have all the bird names.

It is your task to complete the last step by writing a function that parses a file containing the text of *Walden* and returns a `set` of all the words used. Then you find the intersection of the two sets, and my girlfriend's name will be one of the words in the intersection.

Directions

(1) Download the files. Run the template file inside a command prompt to make sure that it is parsing the NCBI files correctly. The program should output an indented list of all orders families species and subspecies with their names.

(2) Rename the template file to follow the naming convention: `yourlastname_firstinitial_hw2.py`.

(3) Fill in the blanks.

(4) Send me an email with your program as an attachment with the subject *Yourlastname HW2 Submission,* ***along with your guess as to what my girlfriend's name is.***

Concepts used

The concepts used in this program may include, but is not limited to, tree traversal, the built-in Python type `set`, regular expressions, string operations, calling functions, etc...

Python Packages to be Used

`re` - for regular expressions

Tips

- **There are multiple common names for various species of birds!** My function `BuildTaxonNamesDict` builds a dictionary where the key is the taxon id number, and the value is the common name, except that when there are multiple common names, the functions puts them into one string delimited by a `/`, *e.g.*, `Carrier Pigeon/Domestic Pigeon/Rock Dove/Rock Pigeon`. Don't forget to split these strings into individual common names before performing your intersection.
- Don't take capitalization into account, use all lowercase or uppercase words.
- You can create a simple regular expression that returns all words in a string by splitting on any character or sequence of characters that are **non-alphanumeric**.
- It might be helpful for you to write the two word lists out to separate files, perhaps sorted using the `sorted` function, to make sure you're properly splitting, capitalizing, etc.
- In the past students have tended to split multi-word common names into individual words for the purposes of doing the intersection, *e.g.*, `Carrier Pidgeon` becomes two words `Carrier` and `Pidgeon` ... **you don't need to do that**. You should end up with a intersection of 8 words and my girlfriend's name should be among them.