SI 601 Fall 2015 Homework 3 (100 points)

Due Date: Tuesday Feb 3 5:00pm

Part 1 (25 points)

Part 1 is about XML parsing. The provided file Rom.xml was downloaded from http://www.folgerdigitaltexts.org/download.html. It is the Shakespeare play Romeo and Juliet in XML format. The format is explained in http://www.folgerdigitaltexts.org/FDT_Documentation.pdf

Your task is to parse Rom.xml and generate a mapping from the Scene/Act/Line Number in Act to FTLN line numbers. For example, the first line in the si601_hw3_part1_desired_output.txt file is
1.1.1 ftln-0015

This means that Scene 1, Act 1, Line 1 (denoted by 1.1.1) is mapped to ftln-0015.

Your output file should be the same as si601_hw3_part1_desired_output.txt. You should use the xml.etree.ElementTree module I talked about in class to parse the XML file **WITHOUT USING ANY REGULAR EXPRESSIONS**.

Part 2 (75 points)

IMDB is a well-known movie ratings website. The IMDB top movies list is particularly interesting. Another popular movie info web site is http://www.themoviedb.org/. The question is: are the ratings from IMDB and themoviedb.org correlated? We'll find out in part 2 of this homework.

You'll be using these modules for this homework: `urllib2`, `BeautifulSoup`, `json`, and `time`. Instructions for installing BeautifulSoup are in Lecture 3 slides.

Step 1. (10 points)

Fetch the IMDB top 250 movies using this URL:

http://www.imdb.com/chart/top

and save it in a HTML file named `step1.html`. The saved HTML file should look similar to `step1_desired_output.html`. Also note that a few movies have titles or actors with, e.g. accented characters. **Make sure you use the `utf8` encoding to write out the HTML to use Unicode and preserve any non-English characters.**

Step 2. (20 points)

Parse the HTML page above with BeautifulSoup, extract movie information for top 100 movies as described below, and save the result in a tab-delimited file named step2.txt. Note that the HTML page contains top 250 movies, but we only need the top 100 movies. Your `step2.txt` file should have 3 columns and 100 rows. The 3 columns should be:

`IMDB_ID`, `Title`, `IMDB Rating`

The IMDB_ID is the part that sits between last two slashes in the movie URL in the table. For example, the first movie has `href="/title/tt0111161/?ref_=chttp_tt_1",` so the IMDB ID is `tt0111161`

Your tab-delimited `step2.txt` file should look like `step2_desired_output.txt`. Here's a sample of the first four lines:

tt0111161	The Shawshank Redemption	9.2
tt0068646	The Godfather	9.2
tt0071562	The Godfather: Part II	9.0
tt0468569	The Dark Knight	8.9

Step 3. (20 points)

Use the Web service http://www.themoviedb.org/documentation/api to get themoviedb.org rating for each of the top 100 movies using the IMDB ID you collected in Step 2. To use this API, you will need to register for an (free) account and get your API Key, which is a long string. For example, if your API key is sfhdskfhdskfhsakfd, this URL fetches a JSON response for the movie "The Shawshank Redemption", which has IMDB ID tt0111161:

http://api.themoviedb.org/3/find/tt0111161?api_key= sfhdskfhdskfhsakfd&external_source=imdb_id

You should see this JSON response:


```
{"movie_results":[{"adult":false,"backdrop_path":"/xBKGJQsAIeweesB79KC8
9FpBrVr.jpg","id":278,"original_title":"The Shawshank
Redemption","release_date":"1994-09-
14","poster_path":"/9O7gLzmreU0nGkIB6K3BsJbzvNv.jpg","popularity":4.537
44570017064,"title":"The Shawshank
Redemption","video":false,"vote_average":8.1,"vote_count":3588}],"perso
n_results":[],"tv_results":[],"tv_episode_results":[],"tv_season_result
s":[]}
```


The "vote_average":8.1 number is the themoviedb.org rating we would like to compare to IMDB ratings.

IMPORTANT! You MUST pause 10 seconds between EVERY HTTP request to themoviedb.org. If you don't do this, and send requests omdbapi.com continuously in a loop with no delay, the server may reject your request.

HINT: Use the `sleep(x)` function in the `time` module after each HTTP request to pause *x* seconds.

Save your results in a text file named step3.txt that contains the IMDB ID and the JSON string (separated by a tab) for each movie on each line. The file should look like step3_desired_output.txt

Step 4. (20 points)

After you verify that your step 3 output is correct, **you can comment out your URL fetching code for step 3 to avoid running that time-consuming step from now on**. Now open the file you saved in step 3, load the JSON string on each line into a variable, extract just the 'vote_average' numbers, and then join it with the IMDB data based on the IMDB IDs. Save your results in a CSV file named step4.csv, which should have the following columns:

'IMDB ID', 'title', 'IMDB rating', 'themoviedb rating'

The file should look like `step4_desired_output.csv`

Step 5. (5 points)

Load your `step4_desired_output.csv` into Excel or Google Doc and make a scatter plot that looks like si601_w15_part2_scatter_plot.pdf.

What to submit:

A zip file named si601_hw3_***youruniquename***.zip that contains:

1. Your python source code files for part 1 and part 2

2. All of your output files for part 1 and part 2: