Chapter 9

Student Exercise

For this exercise, you will put your new web scraping skills to work to scrape archived data on the "proactive disclosure" site run by the Department of Canadian Heritage.

The department hands out a lot of money to community groups and organizations, meaning it could potentially be subject to pressure from members of parliament to direct money to their ridings (districts).

The department makes the data from April 1, 2015 available as CSV files, but prior to that, it is only available via HTML files. Because you want at least five years of data, you need to scrape the prior years data. You can find the landing page at http://www.pch.gc.ca/trans-trans/eng/1360356760939/1402667613878

This is a simple scrape and can be accomplished using urllib2, Beautiful Soup, and Unicodecsv as described in Chapter 9 and its accompanying tutorials.

Complete the following tasks:

1. Using developer tools, examine the HTML of each page that you will navigate and/or scrape.
2. Conceptualize your scrape as discussed in Chapter 9. Figure out what path your scraper will need to follow as it navigates from the landing page to each of the pages below it in the hierarchy. If it helps, draw a diagram of how the code will have to work.
3. Write the code to scrape the data from the innermost page.
4. Write the code to handle the page that contains the summary detail of all the grants and contributions for one quarter, including a loop to then scrape the

detail page. Add the code you developed in step 3 to the loop you write in this step.

5. Write the code to handle the landing page and loop through each of the summary detail pages. Add the code you have written in steps 3 and 4 to the loop you write in this step.
6. Make sure your code includes elements to make requests to the government web server, parse the returned data, and write the data into CSV files.
7. Don't forget to include pauses in your script using time.sleep()so you don't hit the server too often. A delay of at least a second is advisable between each call to the web server. Longer is better.

Some tips:

It is good practice to test each line of code as you go, using print statements to print to the screen any content retrieved from the web or manipulated by your code.

If you save a copy of the HTML for one example of each page you need to scrape, particularly for the pages lower in the hierarchy, you can then use Python's open statement, combined with the read() method, to open and read the HTML file while designing your scraper. This saves having to hit the server over and over again.