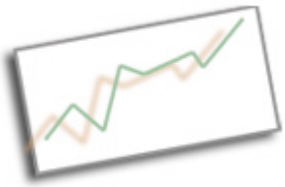

CodeActually
Dr. Cindy Royal
Texas State University
School of Journalism and Mass Communication

Programming a Scraper Using Python

The previous scraping techniques we have covered are very powerful and may be all that you need. But let's take a look at how you might program your own scraper. Learning these techniques will help you understand the mechanics of scraping tools. And there are other features that may allow you to have more control of the data you scrape. You can use any Web programming language to program a scraper, like Python, Ruby or PHP. We are going to look at an example using Python. You can often find a scraper and modify it for your own purposes.

This exercise the Python language and the BeautifulSoup Python library for pulling data out of HTML pages. We'll write code in a Python file in TextWrangler, and then we'll run the script using the Terminal.

It is helpful to have knowledge of html and css concepts before you learn to scrape a web page. These are basic concepts that all students who have taken the first few modules of a Web Design class should know.

- table – html code for introducing a table on an html page
- tr – html code for table row
- td – html code for table data
- div – a section of a Web page. It can be identified further with classes or ids.
- Basic understanding of how html is styled with attributes and inline and external css.

You will need to be able to look at html code and identify selectors that will help the script find the data.

Check your python version. Make sure you have python installed and are using at least version 2.7.6

```
$ python –version
```

Also, make sure you have the pip program installed. This allows you to install Python libraries that you will use during this lesson. If you don't have pip, you will have to install it (see https://pip.pypa.io/en/latest/installing/).

pip --version

If you don't have pip, run:

$ sudo easy_install pip

Install the following now with pip:

$ sudo pip install BeautifulSoup
$ sudo pip install Requests

Basic Scraper

We are going to scrape the contents of one page. Go to the Oscare winners on Wikipedia that we hace used before. https://en.wikipedia.org/wiki/List_of_Academy_Award-winning_films . The script below finds data in a table and scrapes it from the site.

1. With your Terminal open and a new Text Wrangler document ready, use the Terminal to cd to a folder on your computer. Let's just mkdir from your root directory to make a folder called "scraper". Then cd into "scraper".

   $ mkdir scraper
   $ cd scraper
2. Now in Text Wrangler save the page in that folder. Save it as scrape.py. You will now be in the appropriate directory to run the file in the Terminal.
   $ python scrape.py
3. Here is some basic code for getting the data from this page.

```
import csv
import requests
from BeautifulSoup import BeautifulSoup

url =
'https://en.wikipedia.org/wiki/List_of_Academy_Award-
winning_films'
response = requests.get(url)
html = response.content

soup = BeautifulSoup(html)
```

```
table = soup.find('div', attrs={'class': 'mw-content-
ltr'})

list_of_rows = []
for row in table.findAll('tr'):
    list_of_cells = []
    for cell in row.findAll('td'):
        text = cell.text
        list_of_cells.append(text.encode('utf-8'))
    list_of_rows.append(list_of_cells)

outfile = open("./music.csv", "wb")
writer = csv.writer(outfile)
writer.writerows(list_of_rows)
```

A few things about Python:

- The program is space sensitive, so tabbing matters.
- If you want to comment any lines of the code, precede each line with a # sign. Commenting allows you to add helpful instructions or descriptions to code that does not affect its ability to run. And, it can allow you to remove certain lines of code for testing.
- If you get any error messages in the console, read them. They will help you troubleshoot any problems.

4. Let's take a look at what is happening. First, we import several libraries that we will need. import csv allows us to create a csv at the end of the script. import requests allows us to handle url requests and BeautifulSoup lets us select html.
5. Next, we establish the url of the site we are scraping. We put it in a variable named "url", so we can use it in the script. The next two lines create variables for the response and html. We will also use html in the script.
6. The next two lines use BeautifulSoup to access the html (DOM) and make a selection.
7. The find line is the key to this exercise. You have to go into the code and find a CSS selector that identifies where in the code you want to start scraping. The data is usually stored in a table (although it could be in a list). Usually a table will have a class or id. In this case, there was a parent div that had a class "wrapper". I included that information as an attr.

8. Next we create an empty array called list_of_rows. This will contain all our rows of data until we are ready to write it to a file.
9. Next there are nested for loops that go through each row of the table, extract the text from each cell and append it to the list_of_cells array. That is then appended to the list_of_rows array. That happens for every row of the table.
10. The last few lines simply write the data from the final list_of_rows array to a file, in this case, we named it music.csv. After you run the script, you will find it in the same folder as your scripts.
11. From the terminal, run:
    $ python scrape.py

    After a few seconds, the prompt should reappear. Check your folder for the csv and open it in Excel.

    You have created your first scraper!

Exercise: Modify the scraper above to scrape the contents of this page, a list of Academy Award-nominated films.
http://247wallst.com/media/2017/01/24/americas-100-largest-newspapers/

Hint: You have to find a containing selector for the table you wish to scrape and modify the "find" statement. And, you have to change the url.

Multi-page Scraper

Now that we have mastered the art of scraping a page, we can move on to automating the ability to scrape an entire site. Go to Texas Music Office's Musician Registry site. The site is now an app with several pages listing more than 5000 musicians. See the scraper code below that creates a loop to run through each page, scrape it for the h2 and li items in the html and append it to the csv. The output uses the "ab" code (append binary), instead of the "wb" code in the previous example (write binary). The output file is initially written with wb before the loop, but then it is appended each time through the loop.

Put the code on the following page into a new python file. You can name it scrape_all.py.

The code is commented using the # to describe each section.

```
#handles encoding problems
import sys
reload(sys)
sys.setdefaultencoding('utf-8')

#import libraries

import csv
import re
import requests
from BeautifulSoup import BeautifulSoup

#url of the site
site =
'https://gov.texas.gov/apps/music/directory/talent/all/genre/all
/p'

#opens the file with a new write command
outfile = open("./music2.csv", "wb")

#loops through numbers in range for url suffix
for num in range(1,295):
    num = str(num)
    url = site + num

    #sends request for html content
    response = requests.get(url)
    html = response.content
    soup = BeautifulSoup(html)

    #finds location in DOM to repeat
    div = soup.findAll('div', id=re.compile("ContentPlace"))

    list_of_rows = []
    #finds each div
    for i in div:
        list_of_cells = []
        #gets the heading
        for k in i.findAll('h2'):
            text = k.text
            list_of_cells.append(text)
        #gets the list items
        for j in i.findAll('li'):
            text = j.text
            list_of_cells.append(text)
        #appends to list of rows
        list_of_rows.append(list_of_cells)
```

```
#appends to the file, rather than write a new one
outfile = open("./music2.csv", "ab")
writer = csv.writer(outfile)
writer.writerows(list_of_rows)
```

This exercise was modified from the tutorial at http://first-web-scraper.readthedocs.io/en/latest/.