A beginner’s guide to data scraping in Python

By Garrett Lay

1. **Introduction**

New to the python language I spent countless hours googling tutorials on how to use python but never really came across a true beginners guide for data scraping. Most tutorials expected you to be familiar with certain aspects of data mining or html and some were easy to copy and emulate but didn’t really give you an explanation of what was going on. In this paper I set out to change all of that by creating a quick and easy guide for those who are new to Python and looking to learn how to successfully scrape data from a website. The last part of the paper will look into a new type of data scraping by using an extension for Google’s Chrome web browser.

Like most computer languages there are many ways to do the same task, Python is no exception to this. This guide is just one of many ways you can scrape basic data from a website and should only be used as a base in which you should start from as you begin to learn the python language.

Let’s start off with a few basic terms that we’ll need to understand before moving forward in this guide.

HTML Tables – An HTML table is divided into rows (with the <tr> tag), and each row is divided into data cells (with the <td> tag). td stands for "table data," and holds the content of a data cell. A <td> tag can contain text, links, images, lists, forms, and other tables (W3 Schools)

Note: HTML tables are structured just like tables in excel and by using python we can easily scrape data from tables found on a website and save the data in an excel file on a local drive.

Python Library – A library is a collection of standard programs and subroutines that are stored and available for immediate use ( Python Software Foundation)

Browser Extension - A computer program that extends the functionality of a web browser in some way ( Python Software Foundation)

1. **Setting up a Python Environment**

First we need to install a Python Environment, which will allow us to run code written in the python language. For this guide we will use the Enthought Canopy distribution, a free download at the website: <https://www.enthought.com/products/canopy>

More detailed instructions can also be found at: http://quant-econ.net/getting\_started.html

Note: you do not have to use the Canopy distribution but it is highly recommended for first time users. The Canopy distribution comes with most common libraries packaged in but in case you choose not to use Canopy this guide will cover the basic steps to install all necessary libraries (John Stachurski).

We will use the following libraries to perform our data scraping;

Beautiful Soup – A library designed for screen-scraping HTML and XML in Python

lxml – A library for processing XML and HTML in Python

To install these libraries using the MAC OSX operating system open a terminal win and type in the following commands, one at a time:

sudo easy\_install pip

pip install BeautifulSoup4

pip install lxml

Windows 7 & 8 users; make sure you have python environment installed then open the command prompt and navigate to your root C:/ directory and type in the following commands, one at a time:

easy\_install BeautifulSoup4

easy\_install lxml

Our libraries are now installed and it is time to start writing our data scraping code.

**3. Running Python**

Before you begin data scraping you’re going to need an objective. For this example our objective will be to scrape the current BCS college football rankings. First thing we need to do is use a web browser to navigate to the website that contains this data.

Note: Google’s Chrome web browser has a nifty built in element inspection function that allows us to quickly analyze html code and is recommended for all beginners.

Using Chrome, navigate to: <http://www.bcsfootball.org/>

Identify the table on the left hand side of the webpage, right click anywhere on it, and then select ***inspect element*** from the dropdown menu (Stack Overflow).

This will cause a window to pop-up on the bottom or side of your screen that displays the website’s html code. The rankings appear in a table, so scan through the html data until you find the line of code that highlights the table on the webpage.

Note: Chrome’s Inspect Element function is really neat in that it highlights the part of the webpage corresponding to its html code allowing you to quickly find what the information you are looking for.

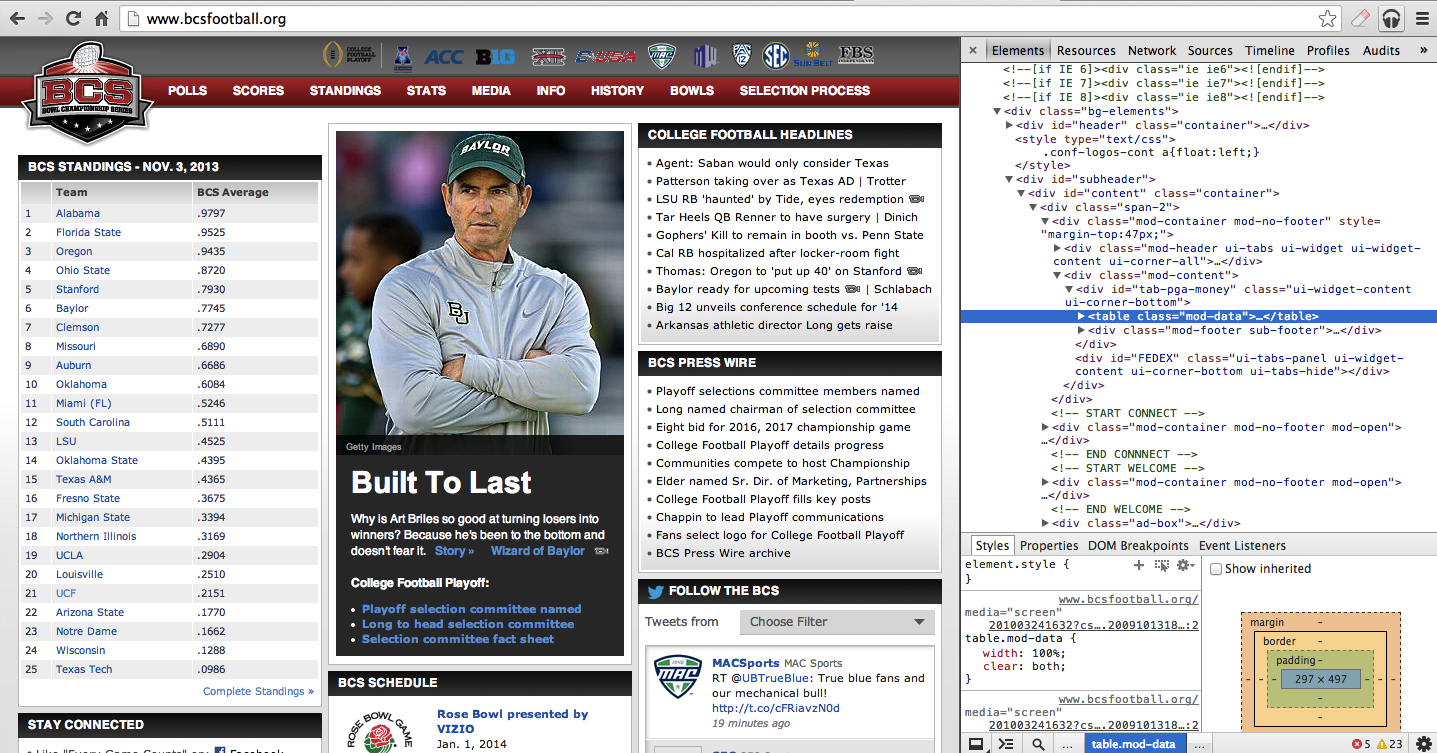


Figure 1. Snapshot of [www.bcsfootball.org](http://www.bcsfootball.org) in Chrome with Inspect Element window opened

Locate the line that reads < table class = "mod-data" > and verify it highlights the BCS rankings on the left hand side of the webpage. This html table contains the data we look to get, now lets use python to extract that data.

Note: Html code will vary from website to website but most follow the same structure. Keep in mind as you follow this guide that you may have to enter in different code corresponding to the website you’re looking to scrape. Instead of ‘tr’ a website may use something else like ‘p’ or ‘h1’, ‘h3’, etc.

In Canopy, open a new file and type in the following;

1] import urllib2

2] from bs4 import BeautifulSoup

Line 1 imports the urllib4 module in Python allowing access to url links while line 2 calls the Beautiful Soup library which is used to scrape data from the web page (John Stachurski).

Next type in;

3]

4] soup = BeautifulSoup(urllib2.urlopen('http://www.bcsfootball.org’).read())

5]

Notice line 3 is intentionally left blank to keep our code clean and organized so that it can be read easily by others. Line 4 uses both the beautiful soup library and the urllib2 module to access our target webpage (Stack Overflow).

Now type in;

6] for row in soup('table', {'class': 'mod-data'})[0].tbody('tr'):

7] tds = row('td')

8] print tds[0].string, tds[1].string

Here is where the code gets a little tricky. Line 6 and 7 uses a function in the Beautiful Soup library to find the table with class = “mod-data” and extracts the information contained in the body(tbody) and tags(tr) in the table (Stack Overflow). Line 8 then prints the first two elements, or strings in this case, and displays it in the output code.

After the code has been typed in, double-check it for any typos.

It should read as follows;

1] import urllib2

2] from bs4 import BeautifulSoup

3]

4] soup = BeautifulSoup(urllib2.urlopen('http://www.bcsfootball.org’).read())

5]

6] for row in soup('table', {'class': 'standings has-team-logos’})[0].tbody('tr'):

7] tds = row('td')

8] print tds[0].string, tds[1].string

Notice lines 7 & 8 are both indented as they are part of the for loop, not having them indented breaks them from the loop.

Note: As mentioned before keep in mind you may have to use different code outside the give ‘tr’ found in this guide. Feel free to experiment with different variables and increase or decrease the print out of tds[x].string, you can do as many printed strings as you like depending on how much data you wish to display, you can also skip numbers.

You can now run the code and view the results displayed in the output display window in Canopy.

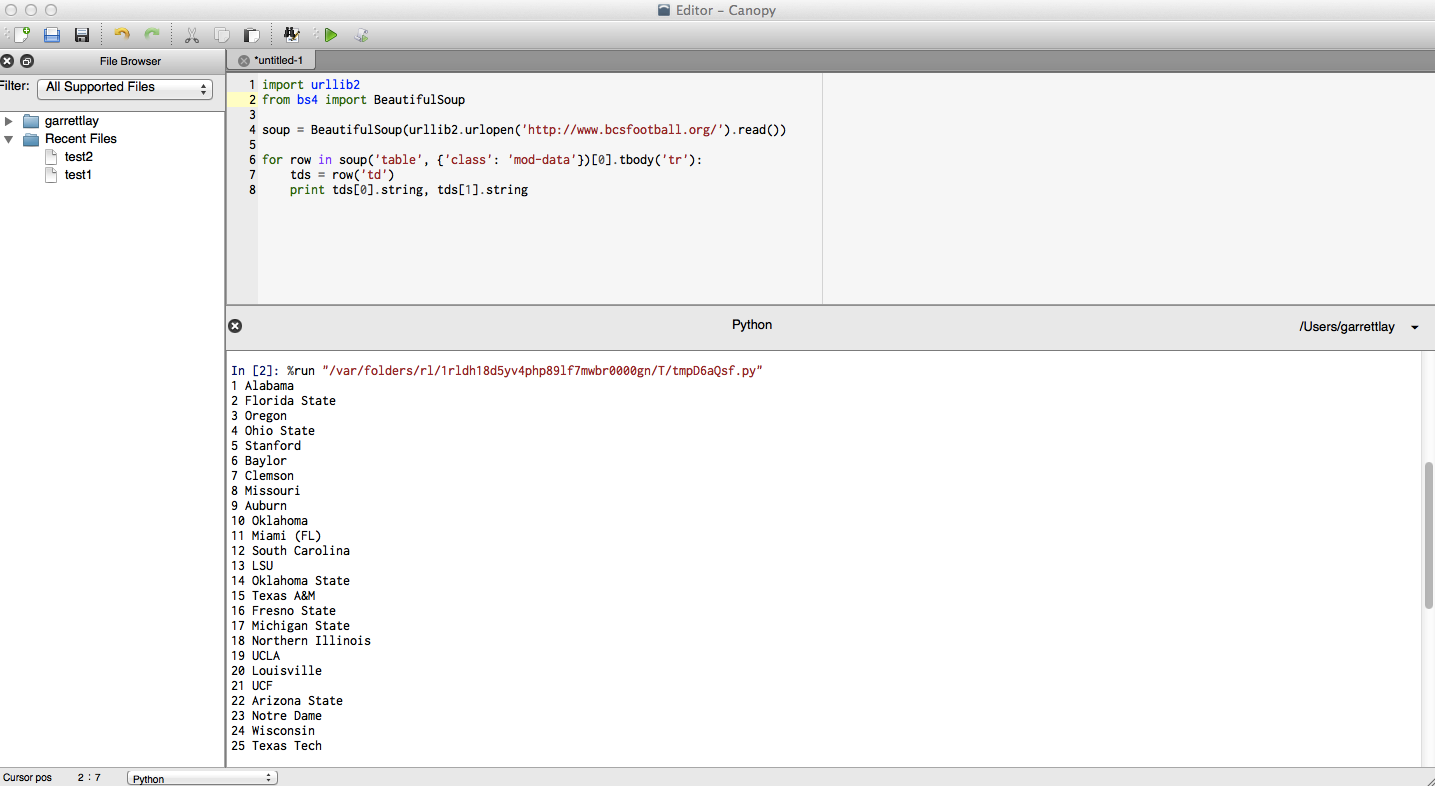


Figure 2. Code and output in Canopy from data scraping bcsfootball.org

This is about as simple as it gets but is a good starting point to get a basic understanding of how you can use python. The next section will give a brief tutorial on how to scrape data directly from an uploaded database using Google Chrome.

**4. Scraping with Chrome**

Google’s Chrome web browser utilizes python code very similar to the code seen in the previous sections. We are now going to look at a very powerful tool that can be used with Chrome that allows scraping basic data in an easier and much friendlier fashion when compared to using a python environment like canopy.

First you need to open your Chrome browser and go into the settings menu. Once in settings click on the extensions tab and a list will populate with your current installed extensions, scroll to the bottom of the list and click on the link that says ***get more extensions.*** This will load the Chrome appstore in your browser, do a search for “scraper” and locate an extension called “scraper 1.x” and click on the +Free/install button next to it, a pop-up will appear and you will need to click Ok to it in order to add the extension to Chrome. It should only take a few seconds to install the extension.

Once the extension is installed lets navigate back to the BCSfootball.org webpage to test out our new tool on the BCS rankings table. Put the mouse cursor over the list and right click, a new option should be in the pop up menu called “Scrape similar …”.

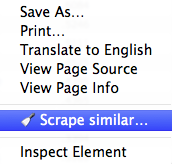


Figure 3. Scraper extension added to the right click menu in Chrome

Once you have verified the extension has been successfully installed in Chrome and appears in your right click menu, we can use it to scrape all the data from the BCS rankings table. To do this we need to highlight the first couple of rows, then leaving it highlighted, right click on it and select the new command “Scrape similar ...” as seen on the next page.

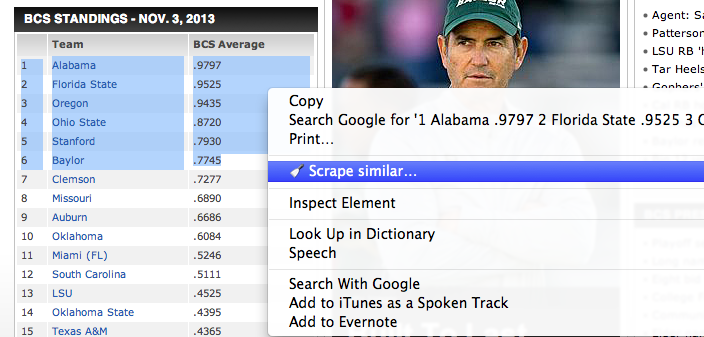


Figure 4. Highlighting a portion of an html table to use the scraping extension

By clicking on the Scrape similar selection Chrome runs a python script in the background that scrapes all the data from the table you select and presents it in a nice pop-up window.

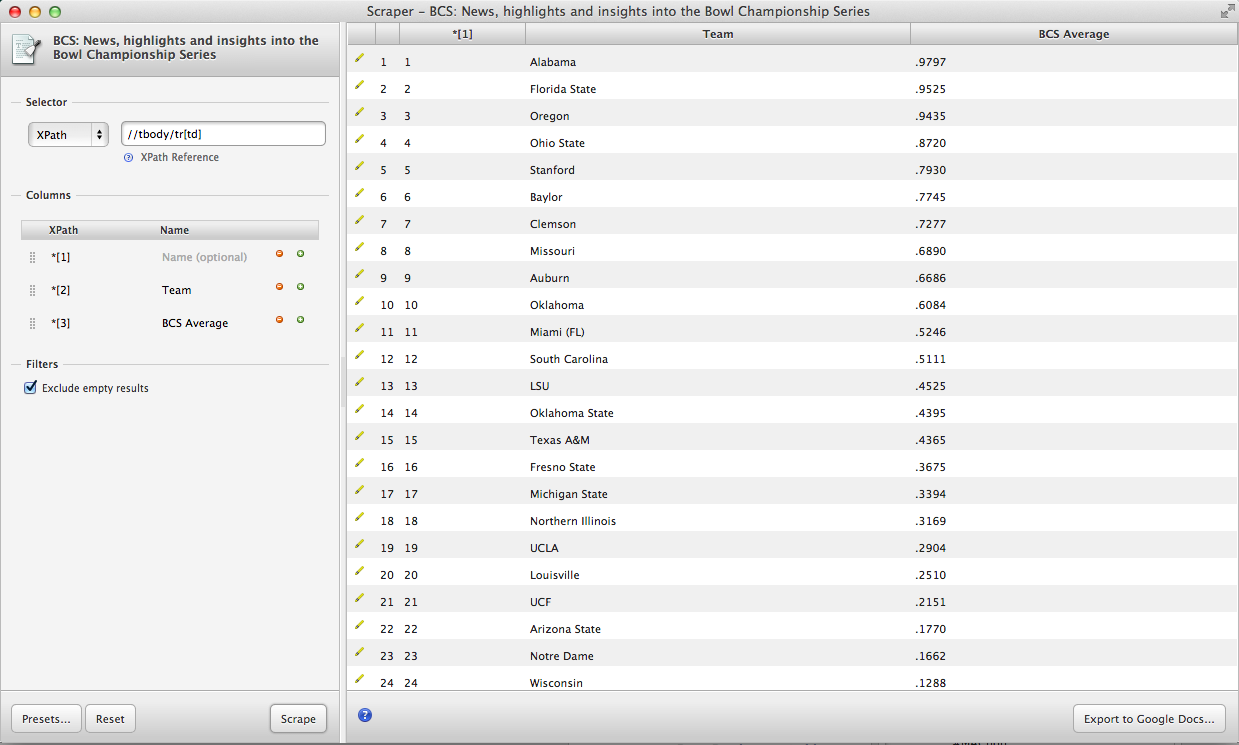


Figure 5. Pop-up window displaying data scraped by the Chrome Scraper extension

Chrome even gives you the option to export the data to Google Docs by click on the button in the bottom right hand portion of the screen. Clicking on the export button will load the file in Google docs as seen below where it can be altered and/or saved as an Excel, CSV, or odd file.

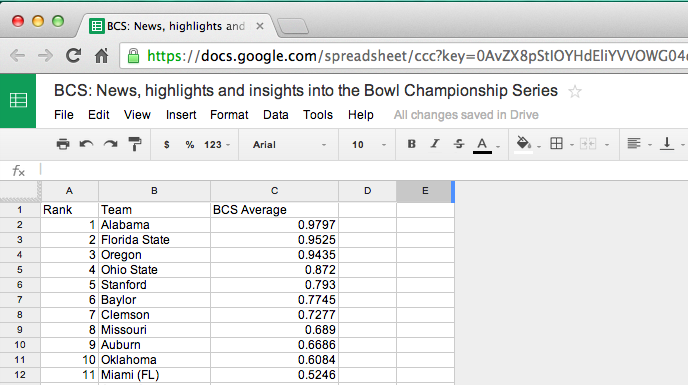


Figure 6. BCS Football Ranking data exported to Google Docs

This quick and simple to use tool is very powerful for newcomers with no experience in data scraping but unfortunately is very limited in its capabilities when compared to python. For basic data like a BCS rankings list the tool has proven to be much handier than using python but for large tasks like analyzing millions of tweets to track trends in social media it unfortunately won’t be able to do anything for you and you’ll have to load your python environment to accomplish the task. It is however, a very neat tool and shows the direction in which the ease of data gathering is headed.

**5. Conclusion**

As a first time python user, I struggled for hours and days to learn the basics but now that I have the small stuff figured out I am beginning to take strides in learning the capabilities of this powerful computer language. I hope my tutorial has helped grow your understanding of python and the basics of data scraping tables from html code. I’ve only been using python for a few days and have already learned so much outside of this tutorial and I am finding the language to be very easy and forgiving to the user, so hang in there, it will all start to make sense soon enough. Be on the lookout for my next tutorial on how to use python “spiders” to track trends in social media. Happy coding.

# Works Cited

Python Software Foundation. (n.d.). *The Python Standard Library*. Retrieved 2013, from http://docs.python.org/2/library/urllib2.html

John Stachurski, T. J. (n.d.). *Quantitative Economics*. Retrieved 2013, from http://quant-econ.net/

*Stack Overflow*. (n.d.). Retrieved 2013, from http://stackoverflow.com/

*W3 Schools*. (n.d.). Retrieved 2013, from http://www.w3schools.com/sql/sql\_intro.asp