**Scraping The Weather Channel**

**Introduction**

Now that we’ve been introduced to Python and IPython, let’s put it to use by scraping some data from the Internet in real time. For the purposes of this exercise, we will scrape The Weather Channel’s website for the current temperature at our zip code and surrounding areas. To do so, we will use a combination of Python libraries including urllib, re, time, and datetime just to name a few.

**Purpose**

The purpose of this exercise is to further introduce you to Python and some of its powerful capabilities.

**Before We Begin…**

Please, if you all run into any trouble during the course of this exercise, do not hesitate to seek help from me. I want to make sure you all get off on the right foot. I’m here for you!!! ☺

**Procedure**

**Part 1: Inspecting The Weather Channel’s Website**

First off, we need to find the place where the current temperature for a zip code is stored within The Weather Channel’s HTML code. This will allow us to extract the data by using a pattern when we go to write the script.

1. Open up your favorite web browser (I will be using and recommend Google Chrome).
2. In Google Chrome, navigate to <http://www.weather.com>. Right click anywhere on the page and click Inspect Element. A window should appear at the bottom of the browser displaying the page’s HTML code.
3. In the Search Zip, City, or Place box in the top right corner of the page, type in 22802 and press enter. This will take us to Harrisonburg’s weather page.
4. The Inspect Element window contains a magnifying glass at the top left corner of the window. Click on it and navigate to the current temperature (located under the words Right Now and the condition image). Click on the current temperature. In the Inspect Element window, you should see a block of HTML code highlighted. This block of code will be vitally important here in a minute.

**Part 2: Building the Script**

For this exercise, we will be writing the script in a text editor and then executing it using IPython in Terminal/Command Prompt. With that being said, let’s get to it! Remember, there are many ways to display the information we are about to obtain. If you want to try something different, try it and see if it works!

A Quick Side Note: I would like to acknowledge that I obtained inspiration and basis for this code via Chris Reeves on YouTube.

1. Open up your favorite text editor (I will be using and recommend Sublime Text 2).
2. First off, let’s import some libraries that we will be using. In Sublime, type the following:

import urllib

import re

import datetime

import time

1. The library urllib allows us to fetch data from the Internet by opening a specified URL, which in turn allows us to “read” its contents. The library re lets us take the HTML code block we found using the inspector and put it into a form that IPython can use to search for the said pattern in the specified URL. Lastly, datetime and time will allow us to provide a date and time stamp to show when this data was scraped.
2. Let’s go ahead and setup the date and time so we can use it later in an output statement. Skip a line underneath the last import statement and type the following:

today = datetime.date.today()

seconds = time.time()

time = datetime.datetime.fromtimestamp(seconds).strftime('%I:%M:%S %p')

1. Remember the block of code that was highlighted on Google Chrome’s Inspect Element window? Let’s go ahead and put that to use. Go ahead and copy that highlighted portion of code, skip a line on your script, and then type the following:

html\_block = ‘*paset\_your\_block\_of\_code\_here*'

html\_pattern = re.compile(html\_block)

1. Your block of HTML code should look like this:

<span itemprop="temperature-fahrenheit">61</span>

It just so happened to be 61 degrees outside when I wrote this tutorial. Now we want to retrieve whatever is between the span tags at any given time because it’s not always going to be 61 degrees out. This being said, adjust your block of HTML code so it looks like this:

<span itemprop="temperature-fahrenheit">(.+?)</span>

When we go to compile the block of HTML code into a regular expression, the (.+?) will represent the “get whatever value is in here”.

1. Next, we need to go ahead and open up the URL we’ve been using to get Harrisonburg’s weather in IPython and then “read” its contents (the HTML code that is). Skip a line underneath you last piece of script and type the following:

html\_file = urllib.urlopen(‘http://www.weather.com/weather/today/22802:4:US’)

html\_text = html\_file.read()

1. If you want to see that IPython has indeed “read” the HTML code, skip a line and type the following:

print html\_text

1. Now, let’s find the pattern that we stored in html\_file and then compiled to html\_pattern in html\_text. Right underneath where we declared html\_text, type the following:

current\_temp = re.findall(html\_pattern,html\_text)

1. By all rights, the current temperature for Harrisonburg, VA should be stored in current\_temp. Let’s find out! Right underneath print html\_text, type the following:

print current\_temp

1. After this, let’s go ahead an print it out in a statement. To do so, we need to format current\_temp as a string object instead of a list object (re.compile returns a list object) by using the join command. Also, we will add in a degree symbol by using the Unicode notation u”\u00b0” as well as use the today and time variables that we set up earlier. So, without further ado, let’s skip a line and put the finishing touches on this script by typing the following:

print ("Current temperature in Harrisonburg, VA on " + today.strftime('%m-%d-%y') + " at " + time + " is " + '\n'.join(current\_temp) +u"\u00b0"+"F")

1. Finally, make sure to save the file with a .py extension. You can name it whatever you want but make sure it is short, sweet, and relevant.

**Part 3: Running the Script**

Now that we have built our script and saved accordingly, lets go ahead and run it in Terminal/Command Prompt.

1. Open a Terminal/Command Prompt
2. Type ipython *the\_name\_of\_your\_program*.py and press enter.
3. If everything works out, you should see the HTML code of the Weather Channel URL, the temperature stored in current\_temp, and the final statement that we wrote.

**Wrap-Up**

Congratulations! You have just successfully scraped The Weather Channel’s website! Just imagine what else you can scrape. The possibilities are endless.