Python Notes - Data Analysis in Python

Somenath Sinha

April 2018

Contents

Part I

Python Web Scraping

Chapter 1

Scraping with Selenium

1.1 Web Scraping Basics

1.1.1 When to Web Scrape

Simply put, when data sets need to be analysed and it's more efficient to automate data gathering than manually collect data, but the data isn't presented in any standardized arranged format, web scraping becomes an option. Further, it's useful when updated data needs to be retrieved.

1.1.2 Website structure

Primarily a web page consists of three types of components:

- **HTML** - For content and structure of data
- **CSS** - For arranging data visually, i.e., style
- **JavaScript** - For interactivity

In case of HTML tags, the syntax is: `<tag attribute="value">Content</tag>`. CSS tags can be used to specify which data to select for styling - but in the case of web-scraping, we can use these *classes* and *IDs* to specify which element to work on/scrape. Classes represent multiple tags/elements while IDs identify a single element.

1.2 Selecting elements for scraping

1.2.1 Interacting with a web site

We can use a browser's *Developer Tools* to inspect all of the HTML, CSS and JavaScript content of a web site. Let us consider a web site, the commit log on Github for Microsoft's TypeScript. It can be found at https://github.com/Microsoft/TypeScript/commits/master. This data set can provide us with insights regarding who is making changes, what changes were made and when these changes are made more frequently!

To inspect a specific commit message (which are arranged here inside s), all we need to do is right click on the element of choice and then select *Inspect (Chrome)* or *Inspect Element (Firefox)*. For example, if we're interested in the author of these commits, we can *right click* on the author name for any commit and inspect it. Since each author name follows the same specific style, we can guess they use the same CSS class, which we can confirm using the developer console. The author name element has the structure:

```
<a href="/Microsoft/TypeScript/commits?author=andy-ms" class="commit-author
↪  tooltipped tooltipped-s user-mention" aria-label="View all commits by
↪  andy-ms">andy-ms</a>
```

After the analysis of other target elements (if needed), we can now be sure that each of the authors are within <a> tags with a class of `commit-author`, with the name of the au*absolute* thor as the content of the tag.

Similarly, each of the commit messages have the structure:

```
<a title="Update outdated comments (#23320)" class="message" data-pjax="true"
↪  href="/Microsoft/TypeScript/commit/a004571d3ebb7da6a7d0915465829dee9874e495">Update
↪  outdated comments (</a>
```

Thus, we now know that the commit messages have the class of `message`.

1.2.2 Verifying Data Selection using JavaScript Console

We can ensure that the correct elements are being targeted by viewing the output of javascript selectors. For example, to see element with the class of `message`, we use:

```
$$(".message")
```

The above is an example of some of the helper functions that browsers provide. A complete list of all helper functions can be found here. Some of them are:

Terms	Description	Example
$()	Returns the first element matching the provided CSS selector	`$(".className")`
$$()	Returns an array of all the elements matching the provided CSS selector	`$$(".className")`
$0	Returns the Currently inspected element	
$_	The value of the last calculated command	
$x()	Returns nodes matching the XPath expression. Use when *all else fail!* XPath can be obtained by: `R.click -> Copy XPath`	
keys()	Returns a list of keys/properties on that object.	
values()	Returns a list of values for keys on that object.	
inspect()	Opens the object inspector for a given object	
pprint()	Formats the contents of object/array for printing on console.	

If we carefully look at the output, we can see that in commits with messages like: , there are multiple messages per commit, as a previous commit number is referenced. For such cases, we select parent elements:

```
<p class="commit-title">
    <a class="message"  href="/Microsoft/TypeScript/commit/01b22">Merge pull request</a>
    <a class="issue-link"
     ↪  href="https://github.com/Microsoft/TypeScript/pull/23331>#23331</a>
    <a class="message"  href="/Microsoft/TypeScript/commit/01b22">from
 ↪  aboveyou00/master</a>
</p>
```

So, we'd directly select the `.commit-title` element instead of its children. Note that the use of *XPath*s are not encouraged since XPaths are the location of an element in the DOM hierarchy such as `/html/body/div[4]/div/p/a[2]` and even a small change like adding/removing an element may break the query!

Chapter 2

Using the Selenium Module

We have to install a **WebDriver** and the **Selenium** module to control the WebDriver to scrape the web using our python scripts. The most common use of a WebDriver is to automate the testing of web apps. Instead of clicking every button on a web-app, we write code that simulates the events and data entry that we need. This is extremely useful for us, since it allows us to automate the browser using our python code.

We can use either the *ChromeDriver* (uses Google Chrome) or the *FirefoxDriver*. First we need to download the **chromedriver** or **geckodriver** (for firefox) binary executables for our selenium module to be able to control it.

2.1 Usage of Selenium WebDriver

To open a web page using the WebDriver, we first import `webdriver` from the **selenium** package, and then use create an instance of the *webdriver* and assign it to an object that represents the browser:

```
from selenium import webdriver
browser = webdriver.Chrome("venv/chromedriver-Linux64")
browser.get("https://www.google.com")
```

The above code opens the *google.com* homepage in a new Google Chrome browser window. To use Firefox instead, we can use:

```
browser = webdriver.Firefox(executable_path="venv/geckodriver")
```

We can now perform a bunch of actions on this browser using the object:

```
# Shows the current URL in the browser
print("Now visiting: ", browser.current_url)

sleep(2)
# Goes to a new URL in the same browser window
browser.get("https://github.com")
```

This script shows the present URL being visited, then waits 2 seconds and goes to *github.com*. To navigate back and close the browser:

```
browser.get("https://www.google.com")
# Shows the current URL in the browser
print("Now visiting: ", browser.current_url)
sleep(2)

# Goes to a new URL in the same browser window
browser.get("https://www.facebook.com")
print("Now visiting: ", browser.current_url)
sleep(1.5)

# Go back to google, and then close the browser
browser.back()
sleep(1)
browser.close()
```

This generates the *console* output:

```
Now visiting:  https://www.google.com/
Now visiting:  https://www.facebook.com/
```

We can even execute custom JavaScript code using:

```
# Execute JavaScript
browser.execute_script("alert('Hello')")
```

This produces an alert window with the content *Hello*. To take a screenshot of the present window, save it to a file and then quit the browser all together, we use:

```
# Take a screenshot of the current page
browser.save_screenshot("sc1.png")
browser.quit()
```

2.2 Manipulating DOM elements

There are several methods of finding specific elements, but most useful among them is the `find_element_by_css_selector()` method. To do an automated google search, we do first identify the relevant elements. For us, it's the search-bar, having an id of *lst-ib* and the search submit button, with the name of *btnK*. We can then do:

```
from selenium import webdriver
browser = webdriver.Firefox(executable_path="venv/geckodriver")
browser.get("https://www.google.com")

# The search bar has the id: #lst-ib
searchBar = browser.find_element_by_css_selector("#lst-ib")
# Type custom statement into search bar
searchBar.send_keys("Selenium WebDriver")

okButton = browser.find_element_by_name("btnK") # Google button has the name "btnK"
#Click the submit button
okButton.click()
```

On the results page, we can clear the existing text in the search-box and enter a new query and search it using:

```
# Obtaining the NEW search bar on the results page
searchBar = browser.find_element_by_css_selector("#lst-ib")
searchBar.clear()
searchBar.send_keys("Next query!")
okButton = browser.find_element_by_css_selector("#mKlEF")  # The search button has the
↪  id: #mKlEF
okButton.click()
```

2.3 Getting a list of data from a website

To get a list of all the post titles from Reddit's front page, the code is:

```
from selenium import webdriver
b = webdriver.Firefox(executable_path="venv/geckodriver")
b.get("https://reddit.com")
titles = b.find_elements_by_css_selector("a.title.may-blank")
for i, post_title in enumerate(titles):
    print(i+1, "-", post_title.text)
b.quit()
```

This gives us the names of the posts in the format:

```
1 - Rare picture of Gandhari using VR headset to live stream the War.
2 - Bandipura Karnataka [NP]
3 - Lord Shree Krishna showing off his latest fidget spinner purchased from Flipkart
4 - Forgotten Proof of First Open Heart Surgery That Ever Happened
5 - Adarsh Balika
6 - Seriously
7 - Ancient Indian Torrent Technology
8 - Deadpool 2: The Final Trailer
9 - MRW my parents are grilling my brother about his gf
10 - Lack of vitamins
11 - Deepika Singh, Lawyer For Kathua Victim's Family, Sends Legal Notice To Zee Hindi
↪  News For Alleged False News Vilifying Her
12 - Rottweiler is not a fan of vegetables.
13 - just news. no exaggerations,no personal opinions,no Debates. just plain news.
14 - 'Cannot distinguish new currency notes': Visually impaired people protest in Kerala
15 - New Paytm phishing scam alert
16 - Big DATA and Hadoop training session during Dwaparayug
17 - [NP] Found this sculpture of Lord Ganesha with a laptop in my hostel
18 - Match Thread: Kings XI Punjab vs Sunrisers Hyderabad at Punjab Cricket Association
↪  IS Bindra Stadium, Mohali, Chandigarh
19 - Japan's highest bridge's height is compared to Godzilla
20 - PsBattle: Ant and a Wasp
21 - Bangalore Medical College invites an anti-abortion homophobic creationist to the
↪  college every year. Please help.
22 - Looking down on a thunderstorm from above the clouds.
23 - My coworker had this as her desktop background. Turns out these are her grandparents
↪  in Yellowstone in the 1940s. Thought it belonged here.
24 - Russian guy locks his head in a cage in an attempt to give up smoking. His wife has
↪  the only key and only opens it for meals.
25 - "Number" is shortened as "No." but doesn't actually contain an "o".
```

To get all posts on the first 4 pages of Reddit, we can do:

```
from selenium import webdriver
b = webdriver.Firefox(executable_path="venv/geckodriver")
b.get("https://reddit.com")
i = pg = 1
while pg <= 4 :
    titles = b.find_elements_by_css_selector("a.title.may-blank")
    for i, post_title in enumerate(titles, i):
        print(i, "-", post_title.text)
    nxtBtn = b.find_element_by_css_selector(".next-button a")
    nxtBtn.click()
    pg += 1
b.quit()
```

The above code clicks the next button multiple times to go to the next page.