Module 11 Notes

##Overview

(As captured from Data Bootcamp Module 11)

Web scraping automates the tedious tasks of extracting online data for analysis. Instead of a person manually visiting each website, copying the data, and then pasting that data into a file, a web-scraping script automatically performs all those actions and, if necessary, organizes the data for analysis.

The web scraping process consists of opening a browser (like Google Chrome), visiting a webpage, and then interacting with that page. The interactions might include logging in to the site or using a search bar to search for text or an item. One of the key ways to making this process efficient is to automate it. With this automation, we don’t need to manually scan dozens of websites and repeat the interactions on each site.

In this module, as assigned Data Analyst, I will automate a web browser to scrape (Web Scrapping), or extract, data then visualize and analyze the data.

By the end of this module, as assigned Data Analyst, I will:

* Describe basic HyperText Markup Language (HTML) elements.
* Explain how websites use HTML to structure a webpage.
* Explain how Cascading Style Sheets (CSS) uses the class and id attributes to style HTML elements and identify the components of a webpage for scraping.
* Use Beautiful Soup and Splinter to both automate a web browser and scrape data.
* Visualize and analyze scraped data by using Python tools.
* Describe basic HyperText Markup Language (HTML) elements.
* Explain how websites use HTML to structure a webpage.
* Explain how Cascading Style Sheets (CSS) uses the class and id attributes to style HTML elements and identify the components of a webpage for scraping.
* Use Beautiful Soup and Splinter to both automate a web browser and scrape data.
* Visualize and analyze scraped data by using Python tools.

##Purpose

(This module is built around a project that mirrors a real-world scenario that would require data analysis and visualization)

SpaceForward is an ambitious aerospace company that’s doing research about resource extraction from nearby planets. Robin, a data analyst at SpaceForward has been tasked with gathering information about the climate of Mars. In addition, Robin has been asked to collect news items about Mars missions.

As a Data Analyst, I am assisting the Client Robin scrape, organize, analyze, and visualize the data.

##Deliverable 1: Scrape Titles and Preview Text from Mars News (40 points)

Open the Jupyter Notebook in the starter code folder named part\_1\_mars\_news.ipynb.

1. Use automated browsing to visit the [Mars NASA news siteLinks to an external site.](https://redplanetscience.com/). Inspect the page to identify which elements to scrape.
2. Create a Beautiful Soup object and use it to extract text elements from the website.

Graphical user interface, text

Description automatically generated

1. Extract the titles and preview text of the news articles that you scraped. Store the scraping results in Python data structures as follows:
   * Store each title-and-preview pair in a Python dictionary. And, give each dictionary two keys: title and preview.
   * Store all the dictionaries in a Python list.
   * Print the list in your notebook.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text

Description automatically generated

##(Optional) Step 4: Export the Data

Optionally, store the scraped data in a file or database (to ease sharing the data with others). To do so, export the scraped data to either a JSON file or a MongoDB database.

Graphical user interface, text, application, email

Description automatically generated

##Deliverable 2: Scrape and Analyze Mars Weather Data (60 points)

Open the Jupyter Notebook in the starter code folder named part\_2\_mars\_weather.ipynb. You will work in this code as you follow the steps below to scrape and analyze Mars weather data.

1. Use automated browsing to visit the [Mars Temperature Data SiteLinks to an external site.](https://data-class-mars-challenge.s3.amazonaws.com/Mars/index.html). Inspect the page to identify which elements to scrape. Note that the URL is https://data-class-mars-challenge.s3.amazonaws.com/Mars/index.html.
2. Create a Beautiful Soup object and use it to scrape the data in the HTML table. Note that this can also be achieved by using the Pandas read\_html function. However, use Beautiful Soup here to continue sharpening your web scraping skills.

Graphical user interface, text, application, email

Description automatically generated

1. Assemble the scraped data into a Pandas DataFrame. The columns should have the same headings as the table on the website. Here’s an explanation of the column headings:
   * id: the identification number of a single transmission from the Curiosity rover
   * terrestrial\_date: the date on Earth
   * sol: the number of elapsed sols (Martian days) since Curiosity landed on Mars
   * ls: the solar longitude
   * month: the Martian month
   * min\_temp: the minimum temperature, in Celsius, of a single Martian day (sol)
   * pressure: The atmospheric pressure at Curiosity's location

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generated

1. Examine the data types that are currently associated with each column. If necessary, cast (or convert) the data to the appropriate datetime, int, or float data types.

Graphical user interface, text, application, email

Description automatically generated

1. Analyze your dataset by using Pandas functions to answer the following questions:
   1. How many months exist on Mars?



* 1. How many Martian (and not Earth) days worth of data exist in the scraped dataset?

Scatter chart

Description automatically generated with medium confidence

* 1. What are the coldest and the warmest months on Mars (at the location of Curiosity)? To answer this question:
     + Find the average the minimum daily temperature for all of the months.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

* + - Plot the results as a bar chart.

Graphical user interface, application, Excel

Description automatically generated

* 1. Which months have the lowest and the highest atmospheric pressure on Mars? To answer this question:
     + Find the average the daily atmospheric pressure of all the months.

Graphical user interface, text, application

Description automatically generated

* + - Plot the results as a bar chart.

Graphical user interface, chart, application, bar chart

Description automatically generated

* 1. About how many terrestrial (Earth) days exist in a Martian year? To answer this question:
     + Consider how many days elapse on Earth in the time that Mars circles the Sun once.

Graphical user interface, text, application, email

Description automatically generated

* + - Visually estimate the result by plotting the daily minimum temperature.

Graphical user interface, chart, application, Word

Description automatically generated

1. Export the DataFrame to a CSV file.

Graphical user interface, text, application

Description automatically generated

**Robin** was recently hired as a data analyst at SpaceForward (a fictitious company). SpaceForward is an ambitious aerospace company that’s doing research about resource extraction from nearby planets. Robin has been tasked with gathering information about the climate of Mars. She’s also been asked to collect news items about Mars missions. Although some of this information isn’t readily available in either CSV or JavaScript Object Notation (JSON) files or through APIs, it can be found on public websites.

Fortunately, Robin knows of a way to collect and organize web data into a usable database: web scraping.

**The Week Ahead**

In this module, you'll first automate a web browser to **scrape**, or extract, data about the National Aeronautics and Space Administration (NASA) Mars exploration. You’ll then visualize and analyze the data.

With all the information that the internet makes available, people and businesses have access to an overwhelming amount of data. But, that data isn’t always formatted in a way that makes it easy to analyze.

Enter web scraping. Web scraping automates the tedious tasks of extracting online data for analysis. Instead of a person manually visiting each website, copying the data, and then pasting that data into a file, a web-scraping script automatically performs all those actions and, if necessary, organizes the data for analysis. That is, instead of us manually collecting and formatting the data, web scraping automates the process for us.

Web scraping has a wide range of applications. Worldwide, large companies use web scraping to assess their reputations or to track the online presence of their competitors. People can also use web scraping on a smaller scale for personal projects. For example, web scraping can simplify the process of collecting the current news about a particular subject.

**What You Will Learn**

By the end of this module, you’ll be able to:

* Describe basic HyperText Markup Language (HTML) elements.
* Explain how websites use HTML to structure a webpage.
* Explain how Cascading Style Sheets (CSS) uses the class and id attributes to style HTML elements and identify the components of a webpage for scraping.
* Use Beautiful Soup and Splinter to both automate a web browser and scrape data.
* Visualize and analyze scraped data by using Python tools.

Splinter :The web scraping process consists of opening a browser (like Google Chrome), visiting a webpage, and then interacting with that page. The interactions might include logging in to the site or using a search bar to search for text or an item. One of the key ways to making this process efficient is to automate it. With this automation, we don’t need to manually scan dozens of websites and repeat the interactions on each site.

So, we'll use a tool named Splinter, which automates our web browser actions.

Beautiful Soup : Most webpages are crowded with many elements. But when performing a web scrape, you likely want to scrape only a subset of the elements from each page. And, you can do so by using Beautiful Soup! Beautiful Soup is a Python library that allows you to pull out and parse specific information from a webpage. You first use Splinter to visit a webpage and then use Beautiful Soup to extract the information that you want.

Recall that HTML is code that’s used to specify elements and their roles on a webpage. These elements include titles, headers, footers, headings, lists, paragraphs, links, and more.

Chrome Developer Tools (DevTools). With DevTools, developers can review the structure of any webpage. And not only that, but it also has a search function. This will help make sense of the tags and elements holding the data that Robin is seeking.

use CSS selectors to efficiently find elements in Chrome DevTools.

Recall that a class can contain multiple elements but that an id must be unique.

Beautiful Soup can search for text in many ways. But, the syntax typically remains the same: first find a tag and then find an attribute. We can search for elements by using only a tag, such as <span /> or <h1 />. But, a class or id attribute makes the search more specific.

Recall that a CSS class attribute gets represented with a period (.). And, an id attribute gets represented with a hashtag (#). So, a div element that has a class attribute of list\_text gets notated as div.list\_text. Likewise, a button element that has an id attribute of more gets notated as button#more. And, remember that an id attribute value must be unique on a webpage.

working thru online example and not getting the expected result as per online lesson

as part of one of the exercises I was instructed to use the following code via the online lesson…

Here is a link to the online module <https://courses.bootcampspot.com/courses/2421/pages/11-dot-5-1-performing-an-automated-web-scrape?module_item_id=817415>

along with screen shot

Graphical user interface, text, application

Description automatically generated

the error message I get

Graphical user interface, text, application

Description automatically generated