Web Scraping - Lab

Introduction

Now that you've seen a more extensive example of developing a web scraping script, it's time to further practice and formalize that knowledge by writing functions to parse specific pieces of information from the web page and then synthesizing these into a larger loop that will iterate over successive web pages in order to build a complete dataset.

Objectives

You will be able to:

- Navigate HTML documents using Beautiful Soup's children and sibling relations
- Select specific elements from HTML using Beautiful Soup
- Use regular expressions to extract items with a certain pattern within Beautiful Soup
- Determine the pagination scheme of a website and scrape multiple pages

Lab Overview

This lab will build upon the previous lesson. In the end, you'll look to write a script that will iterate over all of the pages for the demo site and extract the title, price, star rating and availability of each book listed. Building up to that, you'll formalize the concepts from the lesson by writing functions that will extract a list of each of these features for each web page. You'll then combine these functions into the full script which will look something like this:

```
df = pd.DataFrame()
for i in range(2,51):
    url = "http://books.toscrape.com/catalogue/page-{}.html".format(i)
    soup = BeautifulSoup(html_page.content, 'html.parser')
    new_titles = retrieve_titles(soup)
    new_star_ratings = retrieve_ratings(soup)
    new_prices = retrieve_prices(soup)
    new_avails = retrieve_avails(soup)
    ...
```

Retrieving Titles

To start, write a function that extracts the titles of the books on a given page. The input for the function should be the `soup` for the HTML of the page.

```
def retrieve_titles(soup):
    #Your code here
    warning = soup.find('div', class_="alert alert-warning")
    book_container = warning.nextSibling.nextSibling
    titles = [h3.find('a').attrs['title'] for h3 in book_container.findAll('h3')]
    return titles
```

Retrieve Ratings

Next, write a similar function to retrieve the star ratings on a given page. Again, the function should take in the `soup` from the given HTML page and return a list of the star ratings for the books. These star ratings should be formatted as integers.

```
def retrieve_ratings(soup):
    #Your code here
    warning = soup.find('div', class_="alert alert-warning")
    book_container = warning.nextSibling.nextSibling
    star_dict = {'One': 1, 'Two': 2, 'Three':3, 'Four': 4, 'Five':5}
    star_ratings = []
    regex = re.compile("star-rating (.*)")
    for p in book_container.findAll('p', {"class" : regex}):
        star_ratings.append(p.attrs['class'][-1])
    star_ratings = [star_dict[s] for s in star_ratings]
    return star_ratings
```

Retrieve Prices

Now write a function to retrieve the prices on a given page. The function should take in the `soup` from the given page and return a list of prices formatted as floats.

```
def retrieve_prices(soup):
    #Your code here
    warning = soup.find('div', class_="alert alert-warning")
    book_container = warning.nextSibling.nextSibling
    prices = [p.text for p in book_container.findAll('p', class_="price_color")]
    prices = [float(p[1:]) for p in prices] #Removing the pound sign and converting
    return prices
```

Retrieve Availability

Write a function to retrieve whether each book is available or not. The function should take in the `soup` from a given html page and return a list of the availability for each book.

```
def retrieve_availabilities(soup):
    #Your code here
    warning = soup.find('div', class_="alert alert-warning")
    book_container = warning.nextSibling.nextSibling
    avails = [a.text.strip() for a in book_container.findAll('p', class_="instock av
    return avails
```

Create a Script to Retrieve All the Books From All 50 Pages

Finally, write a script to retrieve all of the information from all 50 pages of the books.toscrape.com website.

```
import re
import requests
from bs4 import BeautifulSoup
import pandas as pd



#Your code here
titles = []
star_ratings = []
prices = []
avails = []
for i in range(1,51):
    if i == 1:
        url = 'http://books.toscrape.com/'
    else:
            url = "http://books.toscrape.com/catalogue/page-{}.html".format(i)
    html_page = requests.get(url)
    soup = BeautifulSoup(html_page.content, 'html.parser')
    titles += retrieve_titles(soup)
    star_ratings += retrieve_ratings(soup)
    prices += retrieve_prices(soup)
    avails += retrieve_availabilities(soup)
df = pd.DataFrame([titles, star_ratings, prices, avails]).transpose()
df.columns = ['Title', 'Star_Rating', 'Price_(pounds)', 'Availability']
print(len(df))
df.head()
```

```
1000
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	Title	Star_Rating	Price_(pounds)	Availability
0	A Light in the Attic	3	51.77	In stock

	Title	Star_Rating	Price_(pounds)	Availability
1	Tipping the Velvet	1	53.74	In stock
2	Soumission	1	50.1	In stock
3	Sharp Objects	4	47.82	In stock
4	Sapiens: A Brief History of Humankind	5	54.23	In stock

Level-Up: Write a new version of the script you just wrote.

If you used URL hacking to generate each successive page URL, instead write a function that retrieves the link from the `"next"` button at the bottom of the page. Conversely, if you already used this approach above, use URL-hacking (arguably the easier of the two methods in this case).

```
#Your code here
def get_next_page(soup):
    next_button = soup.find("li", class_="next") #May return none if on final page
    if next_button:
        return next_button.find('a').attrs['href']
    else:
        return None

def parse_url(url, titles=[], star_ratings=[], prices=[], avails=[]):
    html_page = requests.get(url)
    soup = BeautifulSoup(html_page.content, 'html.parser')
    titles += retrieve_titles(soup)
    star_ratings += retrieve_ratings(soup)
    prices += retrieve_prices(soup)
    avails += retrieve_availabilities(soup)
    next_url_ext = get_next_page(soup)
    if next_url_ext:
        next_url = '/'.join(url.split('/')[:-1])+'/' + next_url_ext
        return parse_url(next_url, titles, star_ratings, prices, avails)
    else:
        return titles, star_ratings, prices, avails


url = 'http://books.toscrape.com/'
titles, star_ratings, prices, avails = parse_url(url, titles, star_ratings, prices,

df = pd.DataFrame([titles, star_ratings, prices, avails]).transpose()
df.columns = ['Title', 'Star_Rating', 'Price_(pounds)', 'Availability']
```

```
print(len(df))
df.head()
```

```
2000
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	Title	Star_Rating	Price_(pounds)	Availability
0	A Light in the Attic	3	51.77	In stock
1	Tipping the Velvet	1	53.74	In stock
2	Soumission	1	50.1	In stock
3	Sharp Objects	4	47.82	In stock
4	Sapiens: A Brief History of Humankind	5	54.23	In stock

```
# As you can see, this method actually returned messier data
# with a slew of repeats:
```

```
len(df[df.duplicated()])
```

```
1000
```

```
len(df[~df.duplicated()])
```

```
1000
```

Summary

Well done! You just completed your first full web scraping project! You're ready to start harnessing the power of the web!

Releases

No releases published

Packages

No packages published

Contributors 2

mathymitchell

mas16 matt

Languages

● **Jupyter Notebook** 100.0%