Lab Assignment 5: Web Scraping

DS 6001: Practice and Application of Data Science

Instructions

Please answer the following questions as completely as possible using text, code, and the results of code as needed. Format your answers in a Jupyter notebook. To receive full credit, make sure you address every part of the problem, and make sure your document is formatted in a clean and professional way.

For the following problems, you will be scraping http://books.toscrape.com/ (http://books.toscrape.com/). This website is a fake book retailer, designed to mimic the design of many retail websites. It exists solely to help students practice web-scraping, so there aren't going to be any ethical concerns with this particular exercise, and there shouldn't be any issues with rate limits or other gates that could prevent web-scraping. Take a moment and look at this website, so that you know what you will be working with.

Your goal is to generate a dataframe with four columns: one for the title, one for the price, one for the star-rating, and one or the book cover JPEG's URL. The dataframe will also 1000 rows, one for each of the 1000 books listed on the 50 pages of this website.

Problem 0

Import the following libraries:

```
In [1]: import numpy as np
        import pandas as pd
        import requests
        from bs4 import BeautifulSoup
        import sys
        sys.tracebacklimit = 0 # turn off the error tracebacks
```

Problem 1

Pull the HTML code from http://books.toscrape.com/ (http://books.toscrape.com/). Make sure you provide a user agent string. Then parse this HTML code and save the parsed code as a separate Python variable. [3 points]

```
In [60]: # Target URL of the book store
         url = "http://books.toscrape.com/"
```

```
In [61]: # Get response from the server
         response = requests.get(url, headers={'User-agent':'Dima Mikhalov'})

         # Check status, 200 expected
         response

Out[61]: <Response [200]>

In [62]: # View HTML as text string
         response.text[:1000]

Out[62]: '<!DOCTYPE html>\n<!--[if lt IE 7]>      <html lang="en-us" class="no-j
         s lt-ie9 lt-ie8 lt-ie7"> <![endif]-->\n<!--[if IE 7]>         <html lan
         g="en-us" class="no-js lt-ie9 lt-ie8"> <![endif]-->\n<!--[if IE 8]>
         <html lang="en-us" class="no-js lt-ie9"> <![endif]-->\n<!--[if gt IE 8]
         ><!--> <html lang="en-us" class="no-js"> <!--<![endif]-->\n    <head>\n
         <title>\n    All products | Books to Scrape - Sandbox\n</title>\n\n
         <meta http-equiv="content-type" content="text/html; charset=UTF-8" />\n
         <meta name="created" content="24th Jun 2016 09:29" />\n        <meta na
         me="description" content="" />\n        <meta name="viewport" content
         ="width=device-width" />\n        <meta name="robots" content="NOARCHIV
         E,NOCACHE" />\n\n        <!-- Le HTML5 shim, for IE6-8 support of HTML
         elements -->\n        <!--[if lt IE 9]>\n        <script src="//html5sh
         im.googlecode.com/svn/trunk/html5.js"></script>\n        <![endif]-->\n
         \n         \n            <link rel="shortcut icon" href="static/oscar/fa
         vicon.'

In [63]: # Parse HTML tags
         books = BeautifulSoup(response.text)
```

Problem 2

Extract all 20 of the book titles and save them in a list. [2 points]

```
In [64]: # In a for loop check all `h3` tags and store .string
         titles = [i.string for i in books.find_all('h3')]
         titles

Out[64]: ['A Light in the ...',
          'Tipping the Velvet',
          'Soumission',
          'Sharp Objects',
          'Sapiens: A Brief History ...',
          'The Requiem Red',
          'The Dirty Little Secrets ...',
          'The Coming Woman: A ...',
          'The Boys in the ...',
          'The Black Maria',
          'Starving Hearts (Triangular Trade ...',
          "Shakespeare's Sonnets",
          'Set Me Free',
          "Scott Pilgrim's Precious Little ...",
          'Rip it Up and ...',
          'Our Band Could Be ...',
          'Olio',
          'Mesaerion: The Best Science ...',
          'Libertarianism for Beginners',
          "It's Only the Himalayas"]
```

```
In [65]: # Check how many elements were added to the lits
         len(titles)

Out[65]: 20
```

Problem 3

Extract the price of each of the 20 books and save these prices in a list. (The prices are listed in British pounds, and include the £ symbol. Remove the £ symbols: if you've saved the prices in a list named `prices` , then the following code should work: `prices = [s.replace('Â£', '') for s in prices]` .) [2 points]

In [66]:
```
# Build a list of corrected prices
prices = [i.string.replace('Â£', '') for i in books.find_all('p')[1::3]]
# [start:stop:step]
prices
```

Out[66]:
```
['51.77',
 '53.74',
 '50.10',
 '47.82',
 '54.23',
 '22.65',
 '33.34',
 '17.93',
 '22.60',
 '52.15',
 '13.99',
 '20.66',
 '17.46',
 '52.29',
 '35.02',
 '57.25',
 '23.88',
 '37.59',
 '51.33',
 '45.17']
```

In [67]:
```
# Check how many elements were added to the lits
len(prices)
```

Out[67]:
```
20
```

Problem 4

Extract the star level ratings for the 20 books. [Hint: for tags such as `<p class="star-rating One">` in which the class has a space, the class is actually a list in which the first item in the list is `"star-rating"` and the second item in the list is `"One"` . It's possible to search on either item in this list.] [3 points]

```
In [68]: # Build a list of # stars:
         ratings = [i.attrs.get('class')[1] for i in books.find_all('p', 'star-ra
         ting')]
         ratings

Out[68]: ['Three',
          'One',
          'One',
          'Four',
          'Five',
          'One',
          'Four',
          'Three',
          'Four',
          'One',
          'Two',
          'Four',
          'Five',
          'Five',
          'Five',
          'Three',
          'One',
          'One',
          'Two',
          'Two']
```

```
In [69]: # Check how many elements were added to the lits
         len(ratings)

Out[69]: 20
```

Problem 5

Extract the URLs for the JPEG thumbnail images that show the covers of the 20 books. (Maybe we want to mine the images to build models that predict the star level, literally judging books by their covers.) [2 points]

```
In [70]: # Build and check the list of URLS to thumbnail images:
         images = [i.attrs.get('src') for i in books.find_all("img" )]
         images
```

```
Out[70]: ['media/cache/2c/da/2cdad67c44b002e7ead0cc35693c0e8b.jpg',
          'media/cache/26/0c/260c6ae16bce31c8f8c95daddd9f4a1c.jpg',
          'media/cache/3e/ef/3eef99c9d9adef34639f510662022830.jpg',
          'media/cache/32/51/3251cf3a3412f53f339e42cac2134093.jpg',
          'media/cache/be/a5/bea5697f2534a2f86a3ef27b5a8c12a6.jpg',
          'media/cache/68/33/68339b4c9bc034267e1da611ab3b34f8.jpg',
          'media/cache/92/27/92274a95b7c251fea59a2b8a78275ab4.jpg',
          'media/cache/3d/54/3d54940e57e662c4dd1f3ff00c78cc64.jpg',
          'media/cache/66/88/66883b91f6804b2323c8369331cb7dd1.jpg',
          'media/cache/58/46/5846057e28022268153beff6d352b06c.jpg',
          'media/cache/be/f4/bef44da28c98f905a3ebec0b87be8530.jpg',
          'media/cache/10/48/1048f63d3b5061cd2f424d20b3f9b666.jpg',
          'media/cache/5b/88/5b88c52633f53cacf162c15f4f823153.jpg',
          'media/cache/94/b1/94b1b8b244bce9677c2f29ccc890d4d2.jpg',
          'media/cache/81/c4/81c4a973364e17d01f217e1188253d5e.jpg',
          'media/cache/54/60/54607fe8945897cdcced0044103b10b6.jpg',
          'media/cache/55/33/553310a7162dfbc2c6d19a84da0df9e1.jpg',
          'media/cache/09/a3/09a3aef48557576e1a85ba7efea8ecb7.jpg',
          'media/cache/0b/bc/0bbcd0a6f4bcd81ccb1049a52736406e.jpg',
          'media/cache/27/a5/27a53d0bb95bdd88288eaf66c9230d7e.jpg']
```

```
In [71]: # Check how many elements were added to the lits
         len(images)
```

```
Out[71]: 20
```

Problem 6

Create a dataframe with one row for each of the 20 books, and the book titles, prices, star ratings, and cover JPEG URLs as the four columns. [2 points]

```
In [72]: # First create a dictionary to define the structure:
         my_dict = {'Title': titles,
                    'Price': prices,
                    'Rating':ratings,
                    'Covers':images,
                   }

         # Convert dictionary to a DataFrame for presentation and analysis
         pd.DataFrame(my_dict)
```

Out[72]:

	Title	Price	Rating	Covers
0	A Light in the ...	51.77	Three	media/cache/2c/da/2cdad67c44b002e7ead0cc35693c...
1	Tipping the Velvet	53.74	One	media/cache/26/0c/260c6ae16bce31c8f8c95daddd9f...
2	Soumission	50.10	One	media/cache/3e/ef/3eef99c9d9adef34639f51066202...
3	Sharp Objects	47.82	Four	media/cache/32/51/3251cf3a3412f53f339e42cac213...
4	Sapiens: A Brief History ...	54.23	Five	media/cache/be/a5/bea5697f2534a2f86a3ef27b5a8c...
5	The Requiem Red	22.65	One	media/cache/68/33/68339b4c9bc034267e1da611ab3b...
6	The Dirty Little Secrets ...	33.34	Four	media/cache/92/27/92274a95b7c251fea59a2b8a7827...
7	The Coming Woman: A ...	17.93	Three	media/cache/3d/54/3d54940e57e662c4dd1f3ff00c78...
8	The Boys in the ...	22.60	Four	media/cache/66/88/66883b91f6804b2323c8369331cb...
9	The Black Maria	52.15	One	media/cache/58/46/5846057e28022268153beff6d352...
10	Starving Hearts (Triangular Trade ...	13.99	Two	media/cache/be/f4/bef44da28c98f905a3ebec0b87be...
11	Shakespeare's Sonnets	20.66	Four	media/cache/10/48/1048f63d3b5061cd2f424d20b3f9...
12	Set Me Free	17.46	Five	media/cache/5b/88/5b88c52633f53cacf162c15f4f82...
13	Scott Pilgrim's Precious Little ...	52.29	Five	media/cache/94/b1/94b1b8b244bce9677c2f29ccc890...
14	Rip it Up and ...	35.02	Five	media/cache/81/c4/81c4a973364e17d01f217e118825...
15	Our Band Could Be ...	57.25	Three	media/cache/54/60/54607fe8945897cdcced0044103b...
16	Olio	23.88	One	media/cache/55/33/553310a7162dfbc2c6d19a84da0d...
17	Mesaerion: The Best Science ...	37.59	One	media/cache/09/a3/09a3aef48557576e1a85ba7efea8...
18	Libertarianism for Beginners	51.33	Two	media/cache/0b/bc/0bbcd0a6f4bcd81ccb1049a52736...
19	It's Only the Himalayas	45.17	Two	media/cache/27/a5/27a53d0bb95bdd88288eaf66c923...

Problem 7

Create a function that takes the URL of the webpage to scrape as an input, applies the code you wrote for questions 1 through 6, and generates the dataframe from question 6 as the output. [3 points]

In [105]:
```
# Defining the function, note - user-name is also required as per #1 req
uirement
def books_scraper(tagret_url, user_agent):
    response = requests.get(tagret_url, headers={'User-agent': user_agen
t})
    books = BeautifulSoup(response.text)
    titles = [i.string for i in books.find_all('h3')]
    prices = [i.string.replace('Â£', '') for i in books.find_all('p')[1
::3]]
    ratings = [i.attrs.get('class')[1] for i in books.find_all('p', 'sta
r-rating')]
    images = [i.attrs.get('src') for i in books.find_all("img" )]
    my_dict = {'Title': titles, 'Price': prices, 'Rating':ratings, 'Cove
rs':images,}
    results = pd.DataFrame(my_dict)
    return results
```

Problem 8

Notice that there are many pages to http://books.toscrape.com/ (http://books.toscrape.com/). When you click on "Next" in the bottom-right corner of the screen, it takes you to http://books.toscrape.com/catalogue/page-2.html (http://books.toscrape.com/catalogue/page-2.html). The front page is the same as http://books.toscrape.com/catalogue/page-1.html (http://books.toscrape.com/catalogue/page-1.html), and there are 50 total pages.

Write a loop that uses the function you wrote in question 7 to scrape each of the 50 pages, and append each of these data frames together. If you write this loop correctly, your dataframe will have 1000 rows (20 books on each of the 50 pages).

Some hints:

- Typing `new_df = pd.DataFrame()` with nothing in the parentheses will create an empty data frame on which new data can be appended.
- There are many loops you can use, but the most straightforward one is a for-values loop that counts from 1 to 50. In Python, you can initialize such a loop with for i in range(1, 51):, and indenting every line below it that belongs inside the loop. Inside the loop, the letter i is now a stand-in for the number currently being considered.
- You will need to figure out how to replace the number in URLs like http://books.toscrape.com/catalogue/page-2.html (http://books.toscrape.com/catalogue/page-2.html) with the number currently under consideration in the loop. You might need the `str()` function, which turns numeric values into strings.

[3 points]

```
In [125]: # Placeholder for the results and root URL
new_df = pd.DataFrame()

# For loop to scrape multile pages:
for i in range(1, 51):
    next_url = 'http://books.toscrape.com/catalogue/page-' + str(i) + '.
html'
    print("Step", i, "--checking next url:", next_url)
    r = requests.get(next_url)

    # Check if response is okay, if not then break
    if r.status_code != 200:
        break

    # If is okay, use books_scraper function and append the results
    else:
        new_df = new_df.append(books_scraper(next_url, "Dima"), ignore_i
ndex = True)
        # Check next page
        i += 1

# Report meta data for the resulst
new_df.info()
```

```
Step 1 --checking next url: http://books.toscrape.com/catalogue/page-1.
html

<ipython-input-125-fdc3dc565707>:16: FutureWarning: The frame.append me
thod is deprecated and will be removed from pandas in a future version.
Use pandas.concat instead.
  new_df = new_df.append(books_scraper(next_url, "Dima"), ignore_index
= True)
```

```
Step 2 --checking next url: http://books.toscrape.com/catalogue/page-2.
html
Step 3 --checking next url: http://books.toscrape.com/catalogue/page-3.
html
Step 4 --checking next url: http://books.toscrape.com/catalogue/page-4.
html
Step 5 --checking next url: http://books.toscrape.com/catalogue/page-5.
html
Step 6 --checking next url: http://books.toscrape.com/catalogue/page-6.
html
Step 7 --checking next url: http://books.toscrape.com/catalogue/page-7.
html
Step 8 --checking next url: http://books.toscrape.com/catalogue/page-8.
html
Step 9 --checking next url: http://books.toscrape.com/catalogue/page-9.
html
Step 10 --checking next url: http://books.toscrape.com/catalogue/page-1
0.html
Step 11 --checking next url: http://books.toscrape.com/catalogue/page-1
1.html
Step 12 --checking next url: http://books.toscrape.com/catalogue/page-1
2.html
Step 13 --checking next url: http://books.toscrape.com/catalogue/page-1
3.html
Step 14 --checking next url: http://books.toscrape.com/catalogue/page-1
4.html
Step 15 --checking next url: http://books.toscrape.com/catalogue/page-1
5.html
Step 16 --checking next url: http://books.toscrape.com/catalogue/page-1
6.html
Step 17 --checking next url: http://books.toscrape.com/catalogue/page-1
7.html
Step 18 --checking next url: http://books.toscrape.com/catalogue/page-1
8.html
Step 19 --checking next url: http://books.toscrape.com/catalogue/page-1
9.html
Step 20 --checking next url: http://books.toscrape.com/catalogue/page-2
0.html
Step 21 --checking next url: http://books.toscrape.com/catalogue/page-2
1.html
Step 22 --checking next url: http://books.toscrape.com/catalogue/page-2
2.html
Step 23 --checking next url: http://books.toscrape.com/catalogue/page-2
3.html
Step 24 --checking next url: http://books.toscrape.com/catalogue/page-2
4.html
Step 25 --checking next url: http://books.toscrape.com/catalogue/page-2
5.html
Step 26 --checking next url: http://books.toscrape.com/catalogue/page-2
6.html
Step 27 --checking next url: http://books.toscrape.com/catalogue/page-2
7.html
Step 28 --checking next url: http://books.toscrape.com/catalogue/page-2
8.html
Step 29 --checking next url: http://books.toscrape.com/catalogue/page-2
9.html
Step 30 --checking next url: http://books.toscrape.com/catalogue/page-3
```

```
0.html
Step 31 --checking next url: http://books.toscrape.com/catalogue/page-3
1.html
Step 32 --checking next url: http://books.toscrape.com/catalogue/page-3
2.html
Step 33 --checking next url: http://books.toscrape.com/catalogue/page-3
3.html
Step 34 --checking next url: http://books.toscrape.com/catalogue/page-3
4.html
Step 35 --checking next url: http://books.toscrape.com/catalogue/page-3
5.html
Step 36 --checking next url: http://books.toscrape.com/catalogue/page-3
6.html
Step 37 --checking next url: http://books.toscrape.com/catalogue/page-3
7.html
Step 38 --checking next url: http://books.toscrape.com/catalogue/page-3
8.html
Step 39 --checking next url: http://books.toscrape.com/catalogue/page-3
9.html
Step 40 --checking next url: http://books.toscrape.com/catalogue/page-4
0.html
Step 41 --checking next url: http://books.toscrape.com/catalogue/page-4
1.html
Step 42 --checking next url: http://books.toscrape.com/catalogue/page-4
2.html
Step 43 --checking next url: http://books.toscrape.com/catalogue/page-4
3.html
Step 44 --checking next url: http://books.toscrape.com/catalogue/page-4
4.html
Step 45 --checking next url: http://books.toscrape.com/catalogue/page-4
5.html
Step 46 --checking next url: http://books.toscrape.com/catalogue/page-4
6.html
Step 47 --checking next url: http://books.toscrape.com/catalogue/page-4
7.html
Step 48 --checking next url: http://books.toscrape.com/catalogue/page-4
8.html
Step 49 --checking next url: http://books.toscrape.com/catalogue/page-4
9.html
Step 50 --checking next url: http://books.toscrape.com/catalogue/page-5
0.html
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   Title   1000 non-null   object
 1   Price   1000 non-null   object
 2   Rating  1000 non-null   object
 3   Covers  1000 non-null   object
dtypes: object(4)
memory usage: 31.4+ KB
```

```
In [ ]:
```