SI206 W23 Project 2

Introduction:

In this project, you will work with BeautifulSoup and html files with data from Airbnb.com. Airbnb's public facing website has lots of data from their database of vacation rentals, however, this database is not publically accessible. We can use web scraping to gather data for our own analysis. *Web scraping* is the process of taking messy data from the web, processing it, cleaning it, and turning it into something useful for analysis.

Web Scraping & Accountability:

Accountability is important for any democracy. Watchdogs and investigative journalists use web scraping from pages publicly available on the internet to hold corporations and governments socially accountable for their products and decisions.

For example, The Center for Investigative Reporting used web scraping to gather members of extremist Facebook hate groups and police Facebook groups. They then cross-referenced the two lists and confirmed hundreds of users that were both currently employed in law enforcement and a member of an online hate group. More than 50 police departments took action or launched internal investigations after the journalists called them with their findings.

Context for this project:

Starting in the 1990s, the San Francisco Bay Area has had an affordable housing shortage. The introduction of Airbnb.com caused serious concerns about the platform further exacerbating this crisis by taking potential rental units off the housing market. As of 2015, lawmakers have attempted to regulate this by requiring potential listers to receive a business license from the local government before offering short-term rentals (rentals of less than 30 days). This allows the government to manage the number of short-term rental units in the city. Additionally, San Francisco Law requires platforms that provide the service of connecting listers and renters (such as airbnb, vrbo, etc) to reasonably verify that listers are registered with the government. However, this system is held accountable through complaints to the local government. **The government will investigate and hold listers and Airbnb accountable only after receiving a complaint.**

In this project:

This project consists of 7 questions and an extra point question. We will be investigating if Airbnb is complying with these local laws in order to strengthen this system of accountability. Airbnb does not make their database of vacation rentals publicly available. We must use web-scraping to gather this information. Web scraping was used by the group "Inside Airbnb" to add data to the debate of Airbnb's impact on residential communities. You can explore their historical data here!

Note that Airbnb.com explicitly forbids the use of web-scraping in their terms of service:

> *"Do not use bots, crawlers, scrapers, or other automated means to access or collect data or other content from or otherwise interact with the Airbnb Platform."*

However, the enforceability of this term is still not fully understood. There have been several high profile US Supreme Court cases debating the legality of web scraping. In order to not cause you to violate airbnb's terms of service, we've provided several static pages in the `html_files` folder for you to use that were obtained from Airbnb.com legally.

Specifically, we went to Airbnb.com on February 13th, 2023, searched for short term rentals in San Francisco's Mission District (the district with the highest density of Airbnbs), downloaded the first page of results given by the website's algorithm, as well as the page of each listing on the first page. Because we loaded the website and downloaded the pages by hand, this is not against the site's terms of service.

You will be scraping this data taken from Airbnb.com, cleaning it, and extracting information from it. You will need to use the BeautifulSoup library to parse through the HTML documents. We have provided static html documents for you to use.

NOTE: The static html documents may appear to be missing several elements or rearranged in an awkward or unexpected way. This is because the static documents do not contain extra CSS or JavaScript files that are necessary to make the site look how the designers wanted. If you'd like to see the page with html, css, and javascript elements, use this following link to see the search results page: Airbnb Search: Mission District, San Francisco, California and this link to see an example Airbnb listing page: Airbnb Listing ID: 1944564. You can then use the "Inspect element" feature of your browser to look at the html.

After you've implemented all of the required functions, you will need to write test cases for each one. We have provided some test cases and guidance for what to test for in the comments, but it will be up to you to implement the logic in the code.

If you choose to do the extra credit, you'll be using similar methods on the reviews page of a listing. Again, the static documents provided may appear weird or unreadable, use the following link to see a reviews page: Listing ID 1944564 Review Page.

IMPORTANT NOTE ABOUT OPENING FILES:

If you are getting "encoding errors" while trying to open, read, or write from a file, add the following statement to any of your open() functions:

encoding="utf-8-sig"

An example of that within the function would be:

open("filename", "r", encoding="utf-8-sig")

There are a few special characters present from Airbnb's pages that aren't defined in standard UTF-8 (which is what Python runs by default). This is beyond the scope of what you have learned so far in this class, so we have provided this for you just in case this happens to you. Good luck!

Assignment:

You will need to write several functions and their test cases. Start from the starter code provided, which looks like the following:

1. **`get_listings_from_search_results(html_file) -> list`**
    a. This function will load the file data from the variable `html_file` into a BeautifulSoup object. You will need to look through the `search_results.html` file to find how you can gather three pieces of information from the 18 Airbnb listings on the web page.
    b. Then return a list of tuples. Each tuple includes the *listing title*, *the number of reviews* the listing has, and the *listing id*. The listing id is found in the url of a listing, for example, https://www.airbnb.com/rooms/1944564 has the listing id "1944564". You must convert the number of reviews to an `integer`.
    c. Expected output:

```
[('Loft in Mission District', 422, '1944564'), ('Home in Mission District', 67, '49043049'), …]
```

2. **`get_listing_information(listing_id) -> tuple`**
    a. This function will take in a `string` containing the *listing id* of an Airbnb listing. NOTE: Use the static files in the `html_files` folder, do NOT send requests to the actual website. You will need to look through the html files of listings to find how you can gather the *policy number*, the *place type*, and *the number of bedrooms* of the listing. Look below for more information about these three items.
    b. The function will then return these three items as a tuple.
        i. *Policy number* (data type: `str`) - either a string of the policy number, `"Pending"`, or `"Exempt"`.
            1. This field can be found in the section about the host.
            2. Note that this is a text field the lister enters, this could be a policy number, or the word `"Pending"` or `"Exempt"` or many others. Look at the raw data, decide how to categorize them into the three categories.
        ii. *Place type* (data type: `str`) - either `"Entire Room"`, `"Private Room"`, or `"Shared Room"`
            1. Note that this data field is not explicitly given from this page. Use the following to categorize the data into these three fields.
                a. `"Private Room"`: the listing subtitle has the word "private" in it
                b. `"Shared Room"`: the listing subtitle has the word "shared" in it
                c. `"Entire Room"`: the listing subtitle has neither the word "private" nor "shared" in it
        iii. *Nightly rate of listing* (data type: `int`)
    c. Example output for listing id “1944564”:

```
('2022-004088STR', 'Entire Room', 181)
```

3. **`get_detailed_listing_database(html_file) -> list`**
    a. Write a function that calls the above two functions in order to return the complete listing information using the functions you’ve created. This function takes in a variable representing the path of the `search_results.html` file.
    b. The return value should be in this format:

```
[(Listing Title 1,Number of Reviews 1,Listing ID 1,Policy
Number 1,Place Type 1,Nightly Rate 1),(Listing Title 2,Number
```

```
of Reviews 2,Listing ID 2,Policy Number 2,Place Type 2,Nightly
Rate 2), ... ]
```

c. Expected output:

```
[('Loft in Mission District', 422, '1944564', '2022-004088STR',
'Entire Room', 181), ('Home in Mission District', 67,
'49043049', 'Pending', 'Entire Room', 147),...]
```

4. write_csv(data, filename)

a. Write a function that takes in a list of tuples called `data`, (i.e. the one that is returned by `get_detailed_listing_database()`), sorts the tuples in ascending order by *cost*, writes the data to a csv file, and saves it to the passed `filename`. The first row of the csv should contain `"Listing Title", "Number of Reviews", "Listing ID", "Policy Number", "Place Type", "Nightly Rate"`, respectively as column headers. For each tuple in the data, write a new row to the csv, placing each element of the tuple in the correct column. The data should be written in the csv in ascending order from *the least costly to the most costly*.

b. Expected Outcome in csv file:

```
Listing Title,Number of Reviews,Listing ID,Policy Number,Place Type,Nightly Rate
Private room in Mission District,198,23672181,STR-0002892,Private Room,109
Guesthouse in San Francisco,79,49591060,STR-0000253,Entire Room,110
…
```

5. check_policy_numbers(data) -> list

a. Write a function that takes in a list of tuples called `data`, (i.e. the one that is returned by `get_detailed_listing_database()`), and parses through the *policy number* of each, validating that the *policy number* matches the policy number format. Ignore any pending or exempt listings.

b. Return a list of the *listing id*s with respective policy numbers that do not match the correct format.

   i. *Policy numbers* are a reference to the business license San Francisco requires to operate a short-term rental. These come in two forms below. # means any digit 0-9.

      1. 20##-00####STR
      2. STR-000####

6. Questions:

Once you've completed the coding portion of this assignment, answer the following questions using the following information.

We know we want to keep Airbnb accountable by checking if an Airbnb listing does not have a *policy number* (a reference to the business license San Francisco requires to operate a short-term rental). Every entry in our database has a *policy number*, is pending a *policy number*, or is exempt from having one (hotels are exempt from this law). This is because Airbnb requires listers to enter this information in a text box before allowing their listing to go live.

However, looking through our database, there is a *policy number* that doesn't look like the other policy numbers. The listing id "16204265" has an unusual *policy number*. Using images of the exterior of the house posted on Airbnb, we can pinpoint which apartment building this rental unit is located in, and check the San Francisco Planning Office to find out if this Airbnb listing does not have a *policy number* and entered random numbers, or if the lister had a typo.

Through this process, we found that this lister does NOT have a short-term rental business license! This is an illegal rental unit that is taking a housing unit away from the local population. We can now file a complaint with the planning office to start an investigation!

Note that the "Property Information Map" of the San Francisco Planning Office may not work on eduroam or MWireless.

Turn in your answers to these questions as well as your code.

a. Throughout this project, we acted as investigators to uphold the system of accountability created by the San Francisco lawmakers: listers must register with the city's planning office and put the business license's number on Airbnb's website, Airbnb must display some effort in validating these policy numbers, and third parties can register a complaint of illegal short-term rentals with the city planning office. We used web-scraping to do the latter using several hours of our personal time.

   Imagine you're a software developer at either the San Francisco Planning Office (SFPO) or Airbnb.com. As a developer at one of these organizations, propose a different system or policy to your organization that verifies if the business license is valid for short term rentals in San Francisco. List at least two arguments you might hear at your organization

(either SFPO or Airbnb.com) against adopting your system/policy and how you would respond to these arguments.

b. The database we've created through web-scraping is a great data source of information for data scientists in order to answer and explore research questions. Skim through the Housing Insecurity in the US Wikipedia page and describe at least one research question that you could answer or explore using the dataset you just created if you were a data scientist working with a housing activist organization to fight against housing insecurity.

c. As discussed in the introduction, the legality of web scraping is still uncertain in the US. Skim through the Legal Issues section of Web Scraping in the US on Wikipedia and this article about the legal issues with the Computer Fraud and Abuse Act, and describe at least one factor you believe is important to consider when discussing the legality of web scraping and why.

d. Scraping public data does not always lead to positive results for society. For example, look to Facebook–Cambridge Analytica data scandal - Wikipedia or Clearview AI - Wikipedia. While web scraping is important for accountability and open access of information, we must also consider issues of privacy as well. Many argue that using someone's personal data without their consent (even if publicly provided) is unethical. Web scraping requires thoughtful intervention. Create at least two guidelines that we must consider when deciding to scrape or not to scrape public data.

7. EXTRA POINT QUESTION

`google_scholar_searcher(query) -> list`

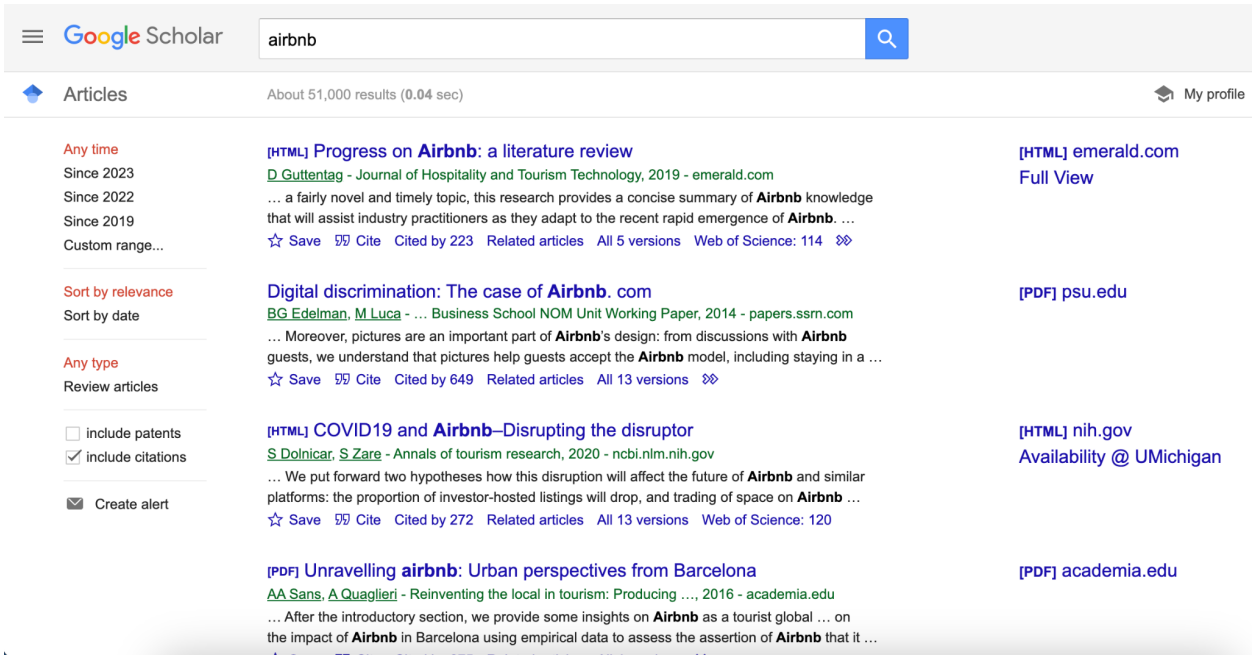
Although we worked with static html files due to legalities and Airbnb's terms of service, typically when working with web scraping, you make requests to a web server to retrieve data. For this extra credit question, you will use a Python module, `requests`, to send a request to the HTTP server and retrieve data from the server.

Parameter: query (`str`)
Return: a list of titles in the first page (`list`)

Write a function that imports the `requests` library of Python and sends a request to google scholar with the passed query. Using BeautifulSoup, find all titles and return the list of titles you see on page 1. (that means, you do not need to scrape results on other pages)

NOTE: You do not need to write test cases for the extra point question.

For example, for a search query “airbnb,”



the function returns:

```
['Progress on Airbnb: a literature review',
 'Digital discrimination: The case of Airbnb. com',
 'COVID19 and Airbnb–Disrupting the disruptor',
 'Unravelling airbnb: Urban perspectives from Barcelona',
 'Poster child and guinea pig–insights from a structured
literature review on Airbnb',
 'A first look at online reputation on Airbnb, where every stay
is above average',
 'A Lefebvrian analysis of Airbnb space',
 'Airbnb: the future of networked hospitality businesses',
 'Who benefits from the" sharing" economy of Airbnb?',
 'Why tourists choose Airbnb: A motivation-based segmentation
study']
```

Grading

Function	***points***
`def get_listings_from_search_results(filepath)`	30
`def get_listing_information(listing_id)`	30
`def get_detailed_listing_database()`	20
`def write_csv(data, filename)`	10
`def check_policy_numbers(data)`	20
`TestCases (10 points for each)`	50
Reflection that shows critical thought	40
Total	**200**
`def google_scholar_searcher(query):`	20 pts extra credit

We will be checking to make sure that you've implemented each function correctly. You will need to make sure that you are returning data in the specified format to get full credit. You will also need to make sure that you are calling the other functions when directed to do so.