SI201 F25 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.

The 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 6 questions and an extra point question. We will be investigating if Airbnb is complying with these local laws to strengthen this system of accountability. Airbnb does not make its 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 on 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. While you can open HTML files in VSCode, it will often be easier to find what you're looking for by opening the files **in your web browse**r and using the inspector tools (**see the screenshot below**). In Chrome and Firefox, you can access the inspector tools by right-clicking. In Safari, you will need to make sure you enable features for web developers (see here for more information).

Based on the example above, we can use our inspector tools to find out that if we're looking for the number of amenities, we might search for tags with the class "b9672i7 dir dir-ltr" or <button> tags with the class "l1j9v1wn b65jmrv v7aged4 dir dir-ltr"

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.

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 on 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**.
The test cases you need to complete are listed in the starter code (project_2.py). Please begin your work there.

Note: Make sure you thoroughly test your functions, as your solution will be run against instructor test cases to ensure correctness of your solution.

1. ***`load_listing_results(html_file) -> list[tuple]`***
    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* 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".
    c. Expected output:

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

2. ***`get_listing_details(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 out how you can gather the information below. Look below for more information about these six items.
    b. The function will then return these six 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"`, `"Exempt"` or many others. Look at the raw data, and decide how to categorize them into the three categories.
        ii. Host type (data type: `str`)
            1. If "Superhost" appears, then it is "`Superhost`"
            2. If no "Superhost", then set it as "`regular`"

iii. *Host name(s)* (data type: `str`) - either the hostname or "`missing`". Beware some listings have multiple hosts – please make sure to capture both names (e.g. "`Seth And Alexa`")

iv. *Place type* (data type: `str`) - either `"Entire Room"`, `"Private Room"`, or `"Shared Room"`

1. Note that this data field is not explicitly given on 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

v. *Number of Reviews* (num_review - data type: `int`) - the listing's number of reviews. Return 0 if a listing has no reviews yet.

vi. *Nightly rate of listing* (data type: `int`)

c. Example output for listing id "1944564":

```
('2022-004088STR', 'Superhost', 'Brian', 'Entire Room', 422, 181)
```

3. ***create_listing_database(html_file) -> list[tuple]***

a. Write a function that calls the above two functions 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,Listing ID 1,Policy Number 1,Host Level 1,Host
Name(s) 1, Place Type 1,Review Number 1,Nightly Rate 1),(Listing
Title 2,Listing ID 2,Policy Number 2,Host Level 2,Host Name(s)
2,Place Type 2,Review Number 2,Nightly Rate 2), ... ]
```

c. Expected output:

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

4. ***output_csv(data, filename)-> None***
    a. Write a function that takes in a list of tuples called `data`, (i.e. the one that is returned by `get_listing_details()`). First, sort the tuples in descending order by *`num_review`*. Write the sorted data to a CSV file and save it to the passed `filename`. The first row of the csv should contain `"Listing Title", "Listing ID", "Policy Number", "Host Level", "Host Name(s)", "Place Type", "Review number", "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 descending order from *the highest num_review to the lowest num_review*.
    b. Expected Structure in csv file:

```
Listing Title,Listing ID,Policy Number,Host Level,Host Name(s),Place Type,Review
Number,Nightly Rate
…
```

5. ***validate_policy_numbers(data) -> list[tuple]***
    a. Write a function and its header that takes in a list of tuples called `data`, (i.e. the one that is returned by **`create_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 and *host names* 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####
    c. Note: Make sure to uncomment lines of code in the **def test_validate_policy_numbers(self)** after writing the function header in your file.

6. Reflective 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. Dutch scholar Mark Bovens describes accountability, in essence, as the following: when someone is given responsibility or task, they *transparently* explain and justify their conduct to a *forum* that judges it against *standards*. *Consequences*, positive or negative, may follow based on the judgment.

   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.

   Do you think this current system provides adequate accountability around short-term rentals in San Francisco? In your response, describe why you believe the system is adequate or inadequate, using Bovens definition. If you believe the system is inadequate, discuss a possible change in actions required of at least one stakeholder (the San Francisco Planning

Office (SFPO), Airbnb hosts/listers, or the developers at Airbnb.com) to improve accountability in this system.

b. So far we've discussed three key stakeholders that play a role in regulating Airbnb listings in San Francisco: the San Francisco Planning Office (SFPO), Airbnb hosts/listers, and Airbnb developers.

   For this question, identify one additional stakeholder that is impacted by or could help improve the regulation of short-term Airbnb rentals in San Francisco. Using Bovens' description of accountability, explain what specific actions or changes in behavior you think this group could implement to improve the accountability and regulation of Airbnb listings in the city.

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. Many argue that using someone's personal data without their consent (even if publicly provided) is unethical. Web scraping requires thoughtful intervention. Develop at least two guidelines you think are important to consider when deciding whether to do web scraping. According to Bovens' definition of accountability, describe how these guidelines should be enforced and by whom.

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 on 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. Your searching results might not be the same as the example research results shown here.

For example, for a search query “airbnb”

the function returns:

```
['Progress on Airbnb: a literature review',
 'Digital discrimination: The case of Airbnb. com',
 'A first look at online reputation on Airbnb, where every stay is above average',
 'Why tourists choose Airbnb: A motivation-based segmentation study',
 'Who benefits from the" sharing" economy of Airbnb?',
 'Airbnb: the future of networked hospitality businesses',
 'The seven lives of Airbnb. The role of accommodation types',
 'Current state and development of Airbnb accommodation offer in 167 countries',
 'Airbnb: disruptive innovation and the rise of an informal tourism accommodation sector',
```

```
 'A socio-economic analysis of Airbnb in New York City']
```

Grading

Function	***points***
`def load_listing_results(html_file)`	20
`def get_listing_details(listing_id)`	40
`def create_listing_database()`	20
`def output_csv(data, filename)`	10
`def validate_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.