Web Scraping with Beautiful Soup

Basic HTML Structure

Basic HTML Tags:

- html, head, body
- div, a, p, h1, ul

Viewing HTML Source:

- Right-click on a webpage
- Select "View Page Source"

Beautiful Soup

https://pypi.org/project/beautifulsoup4/

In [ ]:
```
from bs4 import BeautifulSoup
```

Requests

https://pypi.org/project/requests/

In [ ]:
```
import requests
```

Scrap Yelp

https://www.yelp.com/biz/stone-tower-brews-morgantown

In [ ]:
```
from bs4 import BeautifulSoup
import requests

url = 'https://www.yelp.com/biz/stone-tower-brews-morgantown'
response = requests.get(url)
soup = BeautifulSoup(response.text, 'html.parser')

print(soup.title.text)
```

Extracting an Element

In [ ]:
```
# extract review elements on the page

reviews = soup.find_all('li', class_='y-css-1jp2syp')
```

```
for review in reviews:
    print(review.text)
```

Extracting Elements of the Review

Elements:

- Reviewer
- Rating
- Text
- Date

In [ ]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd

# URLs of the pages containing reviews
reviewer_urls = [
    "https://www.yelp.com/biz/stone-tower-brews-morgantown?start=10",
    "https://www.yelp.com/biz/stone-tower-brews-morgantown"
]

# Initialize lists to store extracted data
usernames = []
ratings = []
reviews = []
dates = []

for url in reviewer_urls:
    # Fetch the webpage content
    response = requests.get(url)
    soup = BeautifulSoup(response.text, 'html.parser')

    # Find all review containers
    review_containers = soup.find_all('li', class_='y-css-1jp2syp')

    # Extracting the date
    for review_container in review_containers:
        # find the element containing the date
        date_element = review_container.find('span', class_='y-css-wfbtsu')
        if date_element:
            date = date_element.text.strip()
        else:
            date = "NA"
        dates.append(date)

    # Iterate over each review container
    for review_container in review_containers:
        # Find the element containing the username
        user_element = review_container.find('span', class_='y-css-w3ea6v')
        if user_element:
            username = user_element.text.strip()
        else:
            username = "NA"
        usernames.append(username)
```

```
        # Find the element containing the rating within the review container
        rating_element = review_container.find('div', role='img')

        # Extract the rating from the aria-label attribute if it exists
        if rating_element and 'aria-label' in rating_element.attrs:
            rating = rating_element['aria-label'].split()[0]
        else:
            rating = "NA"
        ratings.append(rating)

        # Find the element containing the review
        review_element = review_container.find('p', class_='comment__09f24__D0cxf y-css-h9c2fl')
        if review_element:
            review = review_element.text.strip()
        else:
            review = "NA"
        reviews.append(review)

# Create a DataFrame from the extracted data
data = {'Date': dates, 'Username': usernames, 'Rating': ratings, 'Review': reviews}
df = pd.DataFrame(data)

# Print the DataFrame
print(df)

# Export the DataFrame
df.to_csv("stone_tower.csv", index=False)
```

In [ ]:

```
# create a function for scraping

def yelp_scrap():

    # Initialize lists to store extracted data
    usernames = []
    ratings = []
    reviews = []
    dates = []

    for url in reviewer_urls:
        # Fetch the webpage content
        response = requests.get(url)
        soup = BeautifulSoup(response.text, 'html.parser')

        # Find all review containers
        review_containers = soup.find_all('li', class_='y-css-1jp2syp')

        # Extracting the date
        for review_container in review_containers:
            # find the element containing the date
            date_element = review_container.find('span', class_='y-css-wfbtsu')
            if date_element:
                date = date_element.text.strip()
            else:
                date = "NA"
            dates.append(date)

        # Iterate over each review container
        for review_container in review_containers:
```

```
            # Find the element containing the username
            user_element = review_container.find('span', class_='y-css-w3ea6v')
            username = user_element.text.strip()
            usernames.append(username)

            # Find the element containing the rating within the review container
            rating_element = review_container.find('div', role='img')

            # Extract the rating from the aria-label attribute if it exists
            if rating_element and 'aria-label' in rating_element.attrs:
                rating = rating_element['aria-label'].split()[0]
            else:
                rating = "NA"
            ratings.append(rating)

            # Find the element containing the review
            review_element = review_container.find('p', class_='comment__09f24__D0cxf y-css-h9c2
            review = review_element.text.strip()
            reviews.append(review)

    # Create a DataFrame from the extracted data
    data = {'Date': dates, 'Username': usernames, 'Rating': ratings, 'Review': reviews}
    df = pd.DataFrame(data)

    # Print the DataFrame
    print(df)
```

In [ ]:
```
from bs4 import BeautifulSoup
import pandas as pd

# URLs of the pages containing reviews
reviewer_urls = [
    "https://www.yelp.com/biz/stray-cat-chimmi-shack-morgantown"
]


yelp_scrap()
```

In [ ]:
```
#Export the DataFrame

df.to_csv("stray_cat.csv")
```

In [ ]:
```
import requests
from bs4 import BeautifulSoup
import pandas as pd

def yelp_scrap(base_url, num_pages, csv_filename=None):
    # Initialize lists to store extracted data
    usernames = []
    ratings = []
    reviews = []
    dates = []

    for page_num in range(num_pages):
        # Generate the URL for the current page
        url = f"{base_url}?start={page_num * 10}"
```

```
        # Fetch the webpage content
        response = requests.get(url)
        soup = BeautifulSoup(response.text, 'html.parser')

        # Find all review containers
        review_containers = soup.find_all('li', class_='y-css-1jp2syp')

        # Iterate over each review container
        for review_container in review_containers:
            # Find the element containing the username
            user_element = review_container.find('span', class_='y-css-w3ea6v')
            username = user_element.text.strip() if user_element else 'N/A'
            usernames.append(username)

            # Find the element containing the rating within the review container
            rating_element = review_container.find('div', role='img')

            # Extract the rating from the aria-label attribute if it exists
            if rating_element and 'aria-label' in rating_element.attrs:
                rating = rating_element['aria-label'].split()[0]
            else:
                rating = "NA"
            ratings.append(rating)

            # Find the element containing the review
            review_element = review_container.find('p', class_='comment__09f24__D0cxf')
            review = review_element.text.strip() if review_element else 'N/A'
            reviews.append(review)

            # Find the element containing the date
            date_element = review_container.find('span', class_='y-css-wfbtsu')
            date = date_element.text.strip() if date_element else 'N/A'
            dates.append(date)

    # Create a DataFrame from the extracted data
    data = {'Date': dates, 'Username': usernames, 'Rating': ratings, 'Review': reviews}
    df = pd.DataFrame(data)
    df = df.dropna()

    # Print the DataFrame
    print(df)

    # Export to CSV if filename is provided
    if csv_filename:
        df.to_csv(csv_filename, index=False, header = True)
        print(f"Data exported to {csv_filename}")
```

In [ ]:

```
# add your own yelp page here

# Base URL of the restaurant's Yelp page
base_url = "https://www.yelp.com/biz/bartini-prime-morgantown"

# Number of pages to scrape (adjust as needed)
num_pages = 11

# CSV filename
csv_filename = "bartini_reviews.csv"
```

```
# Call the function with export option
yelp_scrap(base_url, num_pages, csv_filename)
```