Name: Santhosh Sunkara

Assignment

Web Scraper to extract information from Grab food-Singapore

Problem Statement:

Develop the scraper to scrape Grab Food Delivery

Website: Grab food- Singapore

Website Link: <https://food.grab.com/sg/en>

Location: PT Singapore - Choa Chu Kang North 6, Singapore, 689577

Result Format: gzip of ndjson

Solution approach

1. Import the python libraries needed
2. Load the food.grab.com page and add the given location as a params.
3. Scrape the data such as the name of the restaurants, cuisine, rating and so on..
4. Load the food.grab.com page for each restaurant in turn, scape the geo-location data and add their latitude and longitude to the dataframe
5. Save the data in the format of gzip of ndjosn

Installing Python Libraries

import requests

from bs4 import BeautifulSoup

import json

import gzip

import re

Use requests get the grab website url

def get\_restaurant\_data(url):

  headers = {

    'User-Agent' : 'Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.0.0 Mobile Safari/537.36'

  }

  restaurants = []

  try:

    params = {'location': 'choa+chu+kang'}  # Use 'location' parameter

    response = requests.get(url, headers=headers, params=params)

    response.raise\_for\_status()  # Raise exception for unsuccessful requests

Use Beautiful Soap to parse html and find the html tags to extract the data

# parse the html code using Beautiful Soap

    soup = BeautifulSoup(response.content, 'html.parser')

    # Find restaurant containers based on your website's structure (replace selectors if needed)

    restaurant\_containers = soup.find\_all('div', class\_='ant-row-flex ant-row-flex-start ant-row-flex-top asList\_\_\_1ZNTr')

Extract all the restaurant details

for container in restaurant\_containers:

      restaurant\_info = {}

      # Extract restaurant name

      name\_element = container.find('p', class\_='name\_\_\_2epcT')

      restaurant\_info['name'] = name\_element.text.strip() if name\_element else None

      # Extract cuisine(s)

      cuisine\_element = container.find('div', class\_='cuisine\_\_\_T2tCh')

      restaurant\_info['cuisine'] = cuisine\_element.text.strip() if cuisine\_element else None

Extract rating, estimated time and distance

# Extract rating, estimate time and distance

      sub\_details\_element = container.find('div', class\_='basicInfoRow\_\_\_UZM8d numbers\_\_\_2xZGn')

      if sub\_details\_element:

        sub\_details\_text = sub\_details\_element.text.strip()

        #Use regular expression to extract rating, estimate time and distance

        rating\_match = re.search(r"^(\d+**\.**\d)", sub\_details\_text)  # Match rating with decimal

        estimate\_time\_match = re.search(r"(\d{2}+) mins", sub\_details\_text)  # Match digits followed by "mins"

        distance\_match = re.search(r"(\d+**\.**?\d\*) km$", sub\_details\_text)  # Match digits followed by "km" at the end

        if rating\_match:

            restaurant\_info['rating'] = rating\_match.group(1)

        else:

            restaurant\_info['rating'] = None  # Indicate rating not found

        if estimate\_time\_match:

            restaurant\_info['estimate\_time'] = estimate\_time\_match.group(1) + ' mins'

        else:

            restaurant\_info['estimate\_time'] = None  # Indicate estimate time not found

        if distance\_match:

            restaurant\_info['distance'] = distance\_match.group(1) + 'km'

        else:

            restaurant\_info['distance'] = None  # Indicate distance not found

Extract promo check, promotional offers and image link

#Extract promotional offers

      offers\_element = container.find('div', class\_='basicInfoRow\_\_\_UZM8d discount\_\_\_3h-0m')

      restaurant\_info['promotional\_offers'] = offers\_element.text.strip() if offers\_element else None

      #Extract promo check

      promo\_element = container.find('div', class\_='promoTagHead\_\_\_1bjRG')

      restaurant\_info['is\_promo\_available'] = "true" if promo\_element else "false"

      #Extract Image Link

      image\_element = container.find('img', class\_='realImage\_\_\_2TyNE show\_\_\_3oA6B')  # Assuming image container class

      restaurant\_info['image\_url'] = image\_element['src'] if image\_element and 'src' in image\_element.attrs else None

      # Append restaurant info in the restaurants array ans print to cross check in the console

      restaurants.append(restaurant\_info)

      print(restaurant\_info)

Fetch the longitude, latitude and restaurant id

script\_element = soup.find('script', id='\_\_NEXT\_DATA\_\_')

      if script\_element:

        script\_data = script\_element.text.strip()

        restaurant\_data = json.loads(script\_data)  # Assuming valid JSON format

        print(f"Restaurant ID: {restaurant\_data.get('geolocation', {}).get('id')}")

        print(f"Latitude: {restaurant\_data.get('geolocation', {}).get('latitude')}")

        print(f"Longitude: {restaurant\_data.get('geolocation', {}).get('longitude')}")

except requests.exceptions.RequestException as e:

    print(f"Error occurred while fetching data: {e}")

  except Exception as e:

    print(f"Unexpected error: {e}")

  return restaurants

# Specific location URL for Grab restaurants in Singapore

grab\_restaurants\_url = "https://food.grab.com/sg/en/restaurants"

# Scrape data from the first page (adjust for pagination if needed)

restaurant\_data = get\_restaurant\_data(grab\_restaurants\_url)

Conversion of data to gzip of ndjson format

# This function converts the data to gzip of ndjson

# It then create the file with given file name

def convert\_to\_ndjson\_gzip(data, filename):

  with gzip.open(filename, 'wb') as f:

    for item in data:

      json\_string = json.dumps(item) + '\n'  # Add newline for NDJSON

      f.write(json\_string.encode('utf-8'))  # Encode as UTF-8

# Convert and compress data

convert\_to\_ndjson\_gzip(restaurant\_data, "restaurants.ndjson.gz")

print("Restaurant data converted to NDJSON.gz format (restaurants.ndjson.gz)")

The complete code:

import requests

from bs4 import BeautifulSoup

import json

import gzip

import re

def get\_restaurant\_data(url):

  headers = {

    'User-Agent' : 'Mozilla/5.0 (Linux; Android 6.0; Nexus 5 Build/MRA58N) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/124.0.0.0 Mobile Safari/537.36'

  }

  restaurants = []

  try:

    params = {'location': 'choa+chu+kang'}  # Use 'location' parameter

    response = requests.get(url, headers=headers, params=params)

    response.raise\_for\_status()  # Raise exception for unsuccessful requests

    # parse the html code using Beautiful Soap

    soup = BeautifulSoup(response.content, 'html.parser')

    # Find restaurant containers based on your website's structure (replace selectors if needed)

    restaurant\_containers = soup.find\_all('div', class\_='ant-row-flex ant-row-flex-start ant-row-flex-top asList\_\_\_1ZNTr')

    for container in restaurant\_containers:

      restaurant\_info = {}

      # Extract restaurant name

      name\_element = container.find('p', class\_='name\_\_\_2epcT')

      restaurant\_info['name'] = name\_element.text.strip() if name\_element else None

      # Extract cuisine(s)

      cuisine\_element = container.find('div', class\_='cuisine\_\_\_T2tCh')

      restaurant\_info['cuisine'] = cuisine\_element.text.strip() if cuisine\_element else None

      # Extract rating, estimate time and distance

      sub\_details\_element = container.find('div', class\_='basicInfoRow\_\_\_UZM8d numbers\_\_\_2xZGn')

      if sub\_details\_element:

        sub\_details\_text = sub\_details\_element.text.strip()

        #Use regular expression to extract rating, estimate time and distance

        rating\_match = re.search(r"^(\d+**\.**\d)", sub\_details\_text)  # Match rating with decimal

        estimate\_time\_match = re.search(r"(\d{2}+) mins", sub\_details\_text)  # Match digits followed by "mins"

        distance\_match = re.search(r"(\d+**\.**?\d\*) km$", sub\_details\_text)  # Match digits followed by "km" at the end

        if rating\_match:

            restaurant\_info['rating'] = rating\_match.group(1)

        else:

            restaurant\_info['rating'] = None  # Indicate rating not found

        if estimate\_time\_match:

            restaurant\_info['estimate\_time'] = estimate\_time\_match.group(1) + ' mins'

        else:

            restaurant\_info['estimate\_time'] = None  # Indicate estimate time not found

        if distance\_match:

            restaurant\_info['distance'] = distance\_match.group(1) + 'km'

        else:

            restaurant\_info['distance'] = None  # Indicate distance not found

      #Extract promotional offers

      offers\_element = container.find('div', class\_='basicInfoRow\_\_\_UZM8d discount\_\_\_3h-0m')

      restaurant\_info['promotional\_offers'] = offers\_element.text.strip() if offers\_element else None

      #Extract promo check

      promo\_element = container.find('div', class\_='promoTagHead\_\_\_1bjRG')

      restaurant\_info['is\_promo\_available'] = "true" if promo\_element else "false"

      #Extract Image Link

      image\_element = container.find('img', class\_='realImage\_\_\_2TyNE show\_\_\_3oA6B')  # Assuming image container class

      restaurant\_info['image\_url'] = image\_element['src'] if image\_element and 'src' in image\_element.attrs else None

      # Append restaurant info in the restaurants array ans print to cross check in the console

      restaurants.append(restaurant\_info)

      print(restaurant\_info)

      script\_element = soup.find('script', id='\_\_NEXT\_DATA\_\_')

      if script\_element:

        script\_data = script\_element.text.strip()

        restaurant\_data = json.loads(script\_data)  # Assuming valid JSON format

        print(f"Restaurant ID: {restaurant\_data.get('geolocation', {}).get('id')}")

        print(f"Latitude: {restaurant\_data.get('geolocation', {}).get('latitude')}")

        print(f"Longitude: {restaurant\_data.get('geolocation', {}).get('longitude')}")

  except requests.exceptions.RequestException as e:

    print(f"Error occurred while fetching data: {e}")

  except Exception as e:

    print(f"Unexpected error: {e}")

  return restaurants

# Specific location URL for Grab restaurants in Singapore

grab\_restaurants\_url = "https://food.grab.com/sg/en/restaurants"

# Scrape data from the first page (adjust for pagination if needed)

restaurant\_data = get\_restaurant\_data(grab\_restaurants\_url)

# This function converts the data to gzip of ndjson

# It then create the file with given file name

def convert\_to\_ndjson\_gzip(data, filename):

  with gzip.open(filename, 'wb') as f:

    for item in data:

      json\_string = json.dumps(item) + '\n'  # Add newline for NDJSON

      f.write(json\_string.encode('utf-8'))  # Encode as UTF-8

# Convert and compress data

convert\_to\_ndjson\_gzip(restaurant\_data, "restaurants.ndjson.gz")

print("Restaurant data converted to NDJSON.gz format (restaurants.ndjson.gz)")

Hence we were able to scrape the data of Grap food required details using Python, Beautiful soap, re with a format of gzip of ndjson.