Google Search Scraper Documentation

The goal of this task is to scrape as much information as possible on the competitor *WebRezPro* and which hotels they are working with. As there are no public databases with this information, we will have to scrape the information from a Google search - 'site:https://secure.webrez.com'

Requirements

External libraries used:

- **beautifulsoup4**
- **requests**
- **pandas**

How to install and run, for example:

- **Run the command 'pip install beautifulsoup4'**

How to use the web scraping .py file

Run the following code in your command terminal:

- **'python WebRezPro_scrape.py'**

Script Overview
The script I have created essentially works in two parts.
The first part being a function named 'scrape_data' which I have created to perform the basic scraping when it is given a url.
The second part is the code that has been tailored to work with Google Search as it appears today, ensuring that we see 100 results on the page and then looping over multiple times as the url is slightly adjusted to find new results.

All in all this script will output 5 elements:

- **Link e.g. https://secure.webrez.com/hotel/3574/**
- **Title e.g. MTN House by Basecamp - WebRezPro: Sign In**
- **Description i.e. The information visible on the Google search page**
- **Details_1 and Details_2: If we were to click on the link, this is the information below the hotel name and is most usually an address and contact details**
- **Features & Amenities: If the hotel on WebRezPro lists any, they will appear**

In [ ]:

```
### CODE BREAKDOWN

from bs4 import BeautifulSoup #pulls data out of HTML files
import requests #Interact with web APIs and/or download content by sending HTTP requests
import pandas as pd #Data manipulation


### CREATE THE FUNCTION ###

def scrape_data(url):
    try:
        # Step 1: Send an HTTP request to the specified URL
        html = requests.get(url)

        # Step 2: Check the request was successful, if not an error will be raised
        html.raise_for_status()

        # Step 3: Parse the HTML content using BeautifulSoup
```

```
        soup = BeautifulSoup(html.text, 'html.parser')

        # Step 4: Extract details below hotel name from the first two div elements with c
lass 'p-t-3'
        div_elements = soup.find_all('div', class_='p-t-3')
        details_1 = div_elements[0].text.strip() if div_elements else 'No details found'
        details_2 = div_elements[1].text.strip() if len(div_elements) > 1 else 'No detai
ls found'

        # Step 5: Extract features & amenities from label elements with class 'checkConta
iner m-b-0'
        label_elements = soup.find_all('label', class_='checkContainer m-b-0')
        features_amenities = [label.get_text(strip=True) for label in label_elements]

        # Step 6: Return the extracted details and features/amenities
        return details_1, details_2, features_amenities

    # Step 7: Handle exceptions (e.g., HTTP request error)
    except requests.RequestException as e:
        print(f"Error fetching data from {url}: {e}")
        return 'Error', 'Error', []


### SCRAPE AND EXPORT ###

headers = {
        "User-Agent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.3
6 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36"
}

# Set your parameters for the search
search_query = 'site:https://secure.webrez.com'
results_per_page = 100 #Any number between 10-100 - maximum is 100
total_results = 300 #In theory this can be larger than 300 but Google doesn't seem to sho
w any more results past 300
current_results = 0
all_data = []

# Construct the base URL for the Google search
base_url = f'https://www.google.com/search?q={search_query}&num={results_per_page}'

# Loop until the desired number of results is reached
while current_results < total_results:
    # Adjust the URL for the current page of search results
    url = f"{base_url}&start={current_results}"

    # Send an HTTP request to the Google search page
    html = requests.get(url, headers=headers)
    html.raise_for_status()

    # Parse the HTML content using BeautifulSoup
    soup = BeautifulSoup(html.text, 'html.parser')

    # Extract all div elements with class 'g' (each representing a search result)
    allData = soup.find_all("div", {"class": "g"})

    # Ensure not to go beyond the available search results
    for i in range(0, min(len(allData), results_per_page, total_results - current_result
s)):
        # Extract the link from the search result
        link = allData[i].find('a').get('href')

        # Check if the link is valid (starts with 'http' or 'https' and is not an adverti
sement link)
        if link is not None and (link.find('https') != -1 and link.find('http') == 0 and
link.find('aclk') == -1):
            # Call the 'scrape_data' function to get details from the linked page
            details_1, details_2, features_amenities = scrape_data(link)

            # Create an entry for the collected data
            entry = {
                "link": link,
```

```
                "title": allData[i].find('h3', {"class": "DKV0Md"}).text if allData[i].f
ind('h3', {"class": "DKV0Md"}) else None,
                "description": allData[i].find("div", {"class": "Hdw6tb"}).text if allDa
ta[i].find("div", {"class": "Hdw6tb"}) else None,
                "details_1": details_1,
                "details_2": details_2,
                "features_amenities": features_amenities
            }

            # Append the entry to the 'all_data' list
            all_data.append(entry)

    # Move to the next page of search results
    current_results += results_per_page

# Create a DataFrame from all the collected data
df = pd.DataFrame(all_data)

# Save the DataFrame to an Excel file
df.to_excel("google_scrape.xlsx", index=False)
```

Example Output

link	title	description	details_1	details_2	ures_amenities
https://se	MTN Hous	Descriptio	No details	No details	[]
https://se	Booking R	HI Charlot	No details	No details	[]
https://se	Siesta Suit	Tel: 011-5	Calle Emili	Tel: 011-5	['Balcony', 'Fridge', 'Garden view', 'Kitchenette', 'Microwave', 'Pets not allowed', 'WiFi Internet']

Conclusion

I was successfully able to write a script that scrapes Google for the names of the hotels that are using the competitor product. In order to improve this, consider paying for a 3rd party Search Engine Results Pages (SERP) tool that will improve scalability, reliability and quality.

Limitations:

- **Google search only showing 300 results even when there are more**
- **Google doesn't like when you make to many requests to their API and will block your IP, solution is to pay.**
- **Besides the name of the hotel, there is not much useful information that you can scrape just from the google search that doesn't already exist in your Hotel_data file.**
- **WebRezPro have not been consistent with their HTML code when creating each webpage, this makes it very difficult to scrape the correct information as for one page the program will pull the hotel name, the same line of code for another page will get their address**