**Documentation:**

1. **scrape\_restaurants Module:**
   * **scrape\_restaurants**: Function to scrape restaurant data from Grab's website.
     + Parameters: None
     + Returns: None
2. **process\_json Module:**
   * **process\_json**: Function to process JSON data obtained from Grab's website and combine recommended merchants and restaurant lists into a single dictionary.
     + Parameters:
       - **json\_file\_path**: Path to the JSON file containing Grab's website data.
       - **output\_file\_path**: Path to the output file where the combined list will be stored.
     + Returns: None
3. **ListMaking Module:**
   * **main**: Main function to process the combined JSON file and perform additional tasks.
     + Parameters: None
     + Returns: None
4. **GrabDetails Module:**
   * **main**: Main function to fetch data from Grab's website for each merchant ID asynchronously using ThreadPoolExecutor.
     + Parameters: None
     + Returns: None
5. **Set Up Environment:**
   * Ensure you have Python installed on your system.
6. **Install Dependencies:**
   * If there's a **requirements.txt** file containing dependencies, install necessary Python packages using **pip install -r requirements.txt**.
7. **Prepare Proxies List:**
   * Create a file named **proxies\_list.txt** containing a list of proxies to rotate during data fetching.
8. **Execute scrape\_main.py:**
   * Run the **scrape\_main.py** script. This script will internally execute the other modules in the specified order (**scrape\_restaurants**, **process\_json**, **ListMaking**, **GrabDetails**), following the flow defined within it.
9. **Follow Output/Log Messages:**
   * Monitor the console output for any errors or status messages during script execution.

By executing **scrape\_main.py**, you'll trigger the entire scraping process as defined in the code, following the specified order of execution for each module.

1. **Problem: Blockage by Website**

**Solution: Using Rotating Proxies**

Explanation:

* + The code utilizes a list of proxies stored in a file named "proxies\_list.txt".
  + It loops through the list of merchant IDs obtained from the "RestaurantList.txt" file.
  + For each merchant ID, it fetches the response from the Grab website using a rotating proxy.
  + If the response status code is 200, indicating success, it saves the response to an individual JSON file.
  + If there's an error or the status code is not 200, it retries with a different proxy after a delay of 5 seconds.
  + After fetching responses for all merchant IDs, it extracts specific information from each response and saves it to a compressed ndjson file.

1. **Problem: Updated HTML Rendering**

**Solution: Using Selenium to Get Updated HTML**

Explanation:

* + This code snippet uses Selenium to scrape data from Grab's restaurant page.
  + It initializes a WebDriver with Chrome and navigates to the URL of Grab's restaurants page.
  + Then, it scrolls down to the bottom of the page, allowing time for the page to load dynamically.
  + It continuously scrolls until the page height remains constant, indicating that all content has been loaded.
  + Once the page is fully loaded, it extracts the HTML source using Selenium.
  + Next, it uses BeautifulSoup to parse the HTML source and find the script tag containing the data.
  + If the script tag is found, it extracts the content, writes it to a JSON file, and then calls the **main** function from the **RestaurantList** module to process the scraped data.
  + Finally, it closes the WebDriver instance to release resources.