SCREENER SCRAPERS: DOCUMENTATION

OVERVIEW

The purpose of the scrapers developed is to extract financial and business-related data from the "SCREENER" platform for each company and organize it in various formats suitable for visualization and analysis. Three distinct scrapers have been created, each employing a unique approach and data storage method. These scrapers are as follows:

1) **Screener_Report:** This scraper enables users to manually select individual companies and extract their data from the "SCREENER" platform. The extracted information is consolidated into a comprehensive report stored in an Excel (.xlsx) file. The report is saved in a designated folder called "Screener Reports."

2) **Screener_One_Excel_DB:** Similar to the previous scraper, this tool allows users to manually select individual companies and retrieve their data from "SCREENER." However, the data is stored in multiple Excel files, categorized based on the type of information, in a single folder named "Screener Individual Data." This approach facilitates efficient storage in a database system. Each time the scraper is run, the existing files are overwritten, making it specific to the company being searched.

3) **Screener_Appending_Excel_DB:** Similar to Screener_One_Excel_DB, this scraper also supports manual selection of companies from "SCREENER" and stores their data in multiple Excel files within a folder named "Screener Append Data." However, instead of overwriting the existing files, the scraper appends the new company details to the respective Excel files. This allows for simultaneous processing of data and an effective storage solution in a database, as the user can run the scraper multiple times to append data for different companies.

WORKING

The working of various scrapers are as follows:

1) Screener_Report

The scraper extracts data from a website called “Screener” and creates an Excel report containing various information about a company, including its details, quarterly results, profit and loss, balance sheet, cash flow, ratios, shareholding pattern, and top articles from Money Control website related to the company.

Here is a step-by-step explanation of how the scraper works:

1. The code imports the required libraries and sets up the web scraping environment.
2. An instance of the Microsoft Edge browser is set up using the EdgeOptions class from the msedge.selenium_tools library.
3. The browser is launched, and the Screener website is loaded using the get() method.
4. The program pauses and waits for user input. This is done to allow the user to search for a company and ensure that the desired company page has loaded before proceeding.
5. The HTML content of the loaded page is parsed using BeautifulSoup.
6. The company name is extracted from the parsed HTML and stored in the ‘company’ variable.
7. The company description (About Section) is extracted from the HTML using the XPath and stored in the ‘about_text’ variable.
8. The script then tries to extract the Company Details Section creating a dictionary of the data before making it a dataframe.
9. Similarly, the script extracts the Quarterly results, profit and loss, balance sheet, cash flow, ratios, and shareholding pattern, section of the company. Each section is scraped using specific HTML elements and converted into pandas DataFrames.
10. The code then defines a function called ‘get_top_moneycontrol_articles’ to scrape the top 20 articles related to the company from the Money Control website. The function takes a query parameter and the number of articles to scrape. It constructs the URL based on the query, sends a request to the website, and parses the HTML response to extract the article titles and links. The function returns a DataFrame containing the article number, title, and link.

11. The function is called to scrape the top articles related to the company using the company name as the query. If the data is unavailable or the code encounters an exception, a corresponding error message is printed.
12. After collecting the data for a particular company, an Excel file is created using the pandas ExcelWriter with XlsxWriter as the engine.
13. The code writes the collected data into one single sheet in the Excel file. Each DataFrame is written with appropriate formatting, including titles, headers, and data rows.

Finally, the Excel file is saved in the "Screener Reports" directory with the company name as the file name. This resulting Excel report provides a comprehensive overview of the extracted financial and business-related data for the specified company from the "SCREENER" platform as a complete report.

2) Screener_One_Excel_DB

The "Screener_One_Excel_DB" scraper is designed to gather data from the Screener website to retrieve information about a single company. It saves the scraped data into multiple Excel files based on different sections of the company information.

Here is a step-by-step explanation of how the scraper works:

1. The code imports the required libraries and sets up the web scraping environment.
2. An instance of the Microsoft Edge browser is set up using the EdgeOptions class from the msedge.selenium_tools library.
3. The browser is launched, and the Screener website is loaded using the get() method.
4. The program pauses and waits for user input. This is done to allow the user to search for a company and ensure that the desired company page has loaded before proceeding.
5. The HTML content of the loaded page is parsed using BeautifulSoup.
6. The company name is extracted from the parsed HTML and stored in the 'company' variable.
7. The company description (About Section) is extracted from the HTML using the XPath and stored in the 'about_text' variable.
8. The script then tries to extract the Company Details Section creating a dictionary of the data before making it a dataframe.

9. Similarly, the script extracts the Quarterly results, profit and loss, balance sheet, cash flow, ratios, and shareholding pattern, section of the company. Each section is scraped using specific HTML elements and converted into pandas DataFrames.
10. The code then defines a function called 'get_top_moneycontrol_articles' to scrape the top 20 articles related to the company from the Money Control website. The function takes a query parameter and the number of articles to scrape. It constructs the URL based on the query, sends a request to the website, and parses the HTML response to extract the article titles and links. The function returns a DataFrame containing the article number, title, and link.
11. The function is called to scrape the top articles related to the company using the company name as the query. If the data is unavailable or the code encounters an exception, a corresponding error message is printed.
12. The extracted data from each section is transformed and saved row-wise in Excel files. This format allows for easy insertion of the data into a database. The excel files are always created for an individual company and gets overwritten every time the scraper is run successfully.

Finally, these Excel files are saved in a directory named "Screener Individual Data" with the file names as respective section name. These files facilitates efficient storage in a database system for individual company insertion.

3) Screener_Appending_Excel_DB:

This specific scraper is designed to scrape data from the "screener.com" website to retrieve information about a single company using BeautifulSoup and Selenium libraries in Python to extract and store the data, and is similar to the previous scraper but differs in the way of storing the data.

Here's a step-by-step procedure of how the data is scraped and stored:

1. The code imports the required libraries and sets up the web scraping environment.
2. An instance of the Microsoft Edge browser is set up using the EdgeOptions class from the msedge.selenium_tools library.
3. The browser is launched, and the Screener website is loaded using the get() method.

4. The program pauses and waits for user input. This is done to allow the user to search for a company and ensure that the desired company page has loaded before proceeding.
5. The HTML content of the loaded page is parsed using BeautifulSoup.
6. The company name is extracted from the parsed HTML and stored in the 'company' variable.
7. The company description (About Section) is extracted from the HTML using the XPath and stored in the 'about_text' variable.
8. The script then tries to extract the Company Details Section creating a dictionary of the data before making it a dataframe.
9. Similarly, the script extracts the Quarterly results, profit and loss, balance sheet, cash flow, ratios, and shareholding pattern, section of the company. Each section is scraped using specific HTML elements and converted into pandas DataFrames.
10. The code then defines a function called 'get_top_moneycontrol_articles' to scrape the top 20 articles related to the company from the Money Control website. The function takes a query parameter and the number of articles to scrape. It constructs the URL based on the query, sends a request to the website, and parses the HTML response to extract the article titles and links. The function returns a DataFrame containing the article number, title, and link.
11. The function is called to scrape the top articles related to the company using the company name as the query. If the data is unavailable or the code encounters an exception, a corresponding error message is printed.
12. The extracted data from each section is transformed and saved row-wise in Excel files. This format allows for easy insertion of the data into a database. The excel files are created for the first searched company and gets updated with new data every time the scraper is run successfully.

Each section of data is stored in a separate Excel files and instead of overwriting the previously created files, it appends new data to existing Excel files within the "Screener Append Data" directory. The title of each section is used as the filename, with the extension ".xlsx". Overall, this scraper automates the process of extracting company data from the Screener website and top articles related to the company from the Money Control website. It organizes the extracted data into an Excel report, making it easier to analyze and work with the data.

DATA COLLECTION

All the scrapers scrape data from "screener" platform and for each company various sections of data are collected and stored in separate pandas DataFrames. Here's a breakdown of the sections and the variables that contain the corresponding DataFrames:

1. **Company Name and Description:** The company name is stored in the variable **'company'**, and the company description (about section) is stored in the variable **'about_text'**.
2. **Company Details:** The company's key details, such as financial ratios, are stored in the **'df_data_dict'** DataFrame.
3. **Quarterly Results:** The company's quarterly financial results are stored in the **'df_quarters'** DataFrame.
4. **Profit & Loss:** The company's profit and loss statement data is stored in the **'df_profit_loss'** DataFrame.
5. **Balance Sheet:** The company's balance sheet data is stored in the **'df_balance_sheet'** DataFrame.
6. **Cash Flow:** The company's cash flow statement data is stored in the **'df_cash_flow'** DataFrame.
7. **Ratios:** The company's financial ratios data is stored in the **'df_ratios'** DataFrame.
8. **Shareholding:** The company's shareholding pattern data is stored in the **'df_shareholding'** DataFrame.
9. **Top Articles:** The top 20 articles about the company from Money Control website are stored in the **'top_articles'** DataFrame.

These DataFrames are then used to store and organize the data before appending it to separate Excel files for the purpose of data visualisation or analysis.

REQUIREMENTS

The data collection process in the provided scraper for "saverisk.com" requires the following packages and their respective versions:

- **warnings:** Used to suppress warning messages during the execution of the scraper.
- **os:** Provides functionality for interacting with the operating system, such as creating directories and checking file existence.
- **time:** Used for introducing delays during the scraping process to ensure proper page loading.
- **pandas (version 2.0.0):** Used for data manipulation and storage in DataFrames.
- **numpy (version 1.23.5):** Required for numerical operations and array manipulation.
- **requests (version 2.28.2):** Enables sending HTTP requests to retrieve web pages' content.
- **xlsxwriter (version 3.1.2):** Used for writing data to Excel files.
- **beautifulsoup4 (version 4.12.2):** A library for web scraping, used for parsing HTML and extracting data from web pages.
- **openpyxl (version 3.1.2):** Required for working with Excel files in Python.
- **msedge-selenium-tools (version 3.141.4)**: Provides the necessary tools for web scraping using the Microsoft Edge browser.

Make sure to have these packages installed and their respective versions compatible with the scraper code for smooth execution.

NOTE: It also requires to download the Microsoft Edge WebDriver executable suitable for your operating system and browser version. The WebDriver allows Selenium to communicate with the Microsoft Edge browser. Make sure to download the appropriate version of the WebDriver and add its location to your system's PATH variable. This driver must be present in the same directory as where the jupyter notebook is open. The current version utilized in the scraper is: **Version: 114.0.1823.79**

Stable internet connection is must for the scraper.

KEY NOTE: Initial login to screener from the browser (Microsoft Edge) is must for scraper. Screener requires an account login to display the complete data.