**PROJECT STRUCTURE**

* downloads/
* logs/
  + downloads.log
  + failed\_downloads.txt
  + scraper.log
* logger.py
* scheduler.py
* scraper.py

**downloads/** is the directory where the downloaded files are saved. The files spanning from different dates are separated using folders with the date as its name (downloads/YYYY-MM-DD/).

**logs/downloads.log** separate logger is used for logs only about the downloads.

**logs/failed\_downloads.txt** used to store all the dates that have been unsuccessful in downloading files. If not empty, it should contain space separated dates. (YYYY-MM-DD YYYY-MM-DD …)

**logs/scraper.log** used to store all logs outputted by the project.

**logger.py** provides functions to set up logging configurations.

**scheduler.py** schedules the execution of the commands at a specific time, Tuesdays-Saturdays only.

**scraper.py** contains command line arguments parsing, file downloader, and exception handlers.

**USAGE**

**scraper.py** is run using command line and has 4 different types of arguments:

* + python scraper.py –latest

Downloads the latest files available from SGX Derivatives. Usually contains yesterday’s files if yesterday is a weekday, or last Friday’s files if yesterday is a weekend.

* + python scraper.py –date <YYYY-MM-DD> [<YYYY-MM-DD>]

Downloads files based on the given date/s. It can accept one or many space-separated dates. It removes future dates or dates that are weekends, it returns an error if no given date is valid.

* + python scraper.py –date-range <YYYY-MM-DD> <YYYY-MM-DD>

Downloads files based on the given range inclusively. The first/start date should occur before the second/end date or else it returns an error. It removes future dates or dates that are weekends, it returns an error if no given date is valid.

* + python scraper.py –retry-failed

Downloads files based on the given dates from the *logs/failed\_downloads.txt* file. Successful downloads remove the specific dates but retains failed ones. It removes future dates or dates that are weekends, it returns an error if no given date is valid.

**scheduler.py** is run using the command –time with a 24-hour time argument:

* + python scheduler.py –time <HH:MM>

Runs two commands at the given time, every day for Tuesday-Saturday only. (Monday-Friday files are only available the next day, hence the Tuesday-Saturday)

The commands executed by this file are:

* + python scraper.py –latest
  + python scraper.py –retry-failed

**ADDITIONAL INFORMATION**

Logging:

All steps are considered in the logging feature of the project. This helps with debugging possible mistakes in the coding process. One logger displays messages to both stdout and file (scraper.log) to keep track of every step used in this project. Another logger special for the downloading part of the project writes duplicate messages to a file (downloads.log). Keeping a separate log for this section ensures tracking of the delivery of the files.

Error and Exception Handling:

All possible errors and exceptions are caught and logged properly. From errors by the user input in the command line (future dates, start date occurs after end date, weekend dates). To handling errors from the website itself (no file found and unsuccessful HTTP requests)

Empty / Incomplete Folders:

The script also removes empty date folders caused by unsuccessful downloads. This prevents confusion if the script downloads the files on the date or not. Even if the folder is incomplete of the files needed, the script recognizes the date as a failed one and logs them in the failed\_downloads file. This rewrites the entire incomplete date folder to make it complete (if files are already available).

Automatic Daily Downloading and Redownloading of Missing Files:

Using the scheduler.py, the script schedule runs of the scraper.py during the given time. It runs with the argument of –latest and –retry-failed, making it capable to download daily available files and redownload missing ones.

**LIMITATIONS**

For historical files and today’s files, the script made use of the website where the download link redirects to [https://links.sgx.com/1.0.0/derivatives-historical/{index}/{file}](https://links.sgx.com/1.0.0/derivatives-historical/%7bindex%7d/%7bfile%7d). This is the only way to download historical files but there is one caveat into using this indexing method. The indices on the website are inconsistent and not all dates have all the files required. With thorough inspection of this website, I found out that some indices are even in reversed order and even skipping some dates. **The script is only consistent for historical files starting from the year 2021.**