Documentation

Installation

```
pip install requirements.txt
```

Files

`app.py` : executes scraper.
`document_scraper.py` : scraper class.

Execution

```
./app.py ticker sort
```

- String: `ticker` : company ticker to scrape.
- Boolean: `sort` : whether or not to write the records in descending order.

Result

`{ticker}_13F.tsv` : tsv file containing information from all of the ticker's 13F xml info tables.

Design Decisions

Dealing with Different Info Table Formats

By looking at the html of each info table, I noticed that they all shared the same field names. However, whether or not the corresponding xml element was in the the xml element tree depended on if the data existed in that cell. Therefore, by searching for each possible element name and saving its value if it exists or an empty string if it doesn't, the info table scrape method can remain DRY and used for all info tables.

Technologies - BeautifulSoup vs. Scrapy vs. Selenium.

The goal of this crawler is that given one ticker, save information from all of its 13F reports. Looking at the flow of the website, getting to the required information is as follows:

for all of the tickers' document list pages, find all 13F document links ➡ for each document link, go to document page ➡ find information table xml link ➡ go to xml link ➡ scrape information.

Because of the tree-like structure of this traversal, a serial approach would lead to poor performance. Using BeautifulSoup and the requests module, the program waits for the response of the first 13F document link on one document list page to request the second 13F document link. There is no need for the program to wait here, since the subsequent paths of each of the requests are independent of each other. Therefore, leveraging the asynchronicity from Scrapy's event driven networking yields greater performance.

Furthermore, there was no dynamic content on any of the pages visited which means there is no need to use Selenium here.

Sort Flag

Because the requests are issued concurrently, records written to the output tsv file can be non-sequential. Depending on the use case for running the spider, this may or may not be acceptable. If the sort flag is set to true, the records will be written in descending order.

Limitations

Dealing with Different Info Table Formats

Because I hardcoded the xml element names to search for inside of info tables, the scraper could be less modular.

Older Document Format

Older documents do not contain a .xml information table file. They contain a .txt "complete submission text file" instead. This would require a different scraping function to handle these cases.
