DAFA Winter Term 21/22 - 28.01.2022

Scrapy
Fast, simple and extensible web scraping

Erik Reifer - 6822273

Overview - Structure

Introduction to Scrapy

What is Scrapy?

- Scrapy is a robust and extensible web scraping tool
- Once set up, scrapy is very fast and

has some nice caveats (more on this later)

How does Scrapy work?

- Like a regular scraping tool that creates lists for

objects being parsed

- Asynchronous requests

Why use Scrapy?

- Reliably scraping simple websites
- Extracting large datasets

Scrapy

An open source and collaborative framework for extracting the data you need from websites. In a fast, simple, yet extensible way.

Maintained by Zyte (formerly Scrapinghub) and many other contributors

Comparison to other scraping libraries

Selenium:

- Designed for testing and interacting with webpages
- Can also be used for scraping but is very slow
- Requires some tweaking when required to run automatically on a server

Beautiful Soup (BS4):

- Cannot create requests to webservers
- Used only for extracting data from html / xml files
- Sparse use cases where solely BS4 is used to crawl websites

Scrapy:

- Designed for web scraping
- Can be used to extract data from APIs (JSON)
- Scrapy can be set up to work with BS4

Use Cases for Scrapy

Use Case 1:

- Extract a large set of data from an archive-like website, e.g. archive.org
- One-time only

Use Case 2:

- Scrape every nth-point in time
- Results in time series data
- Good when collecting news data

Use Case 3:

- Scrape user-generated content on portals like reddit
- Depending on analysis one or multiple crawls necessary

Project: A simple webspider

Goal: Create a webspider that crawls the news from our faculty at Goethe University and saves the data to json format

Basic Setup:

- Installation via common package managers PyPi or Conda:

```
conda install -c conda-forge scrapy
```

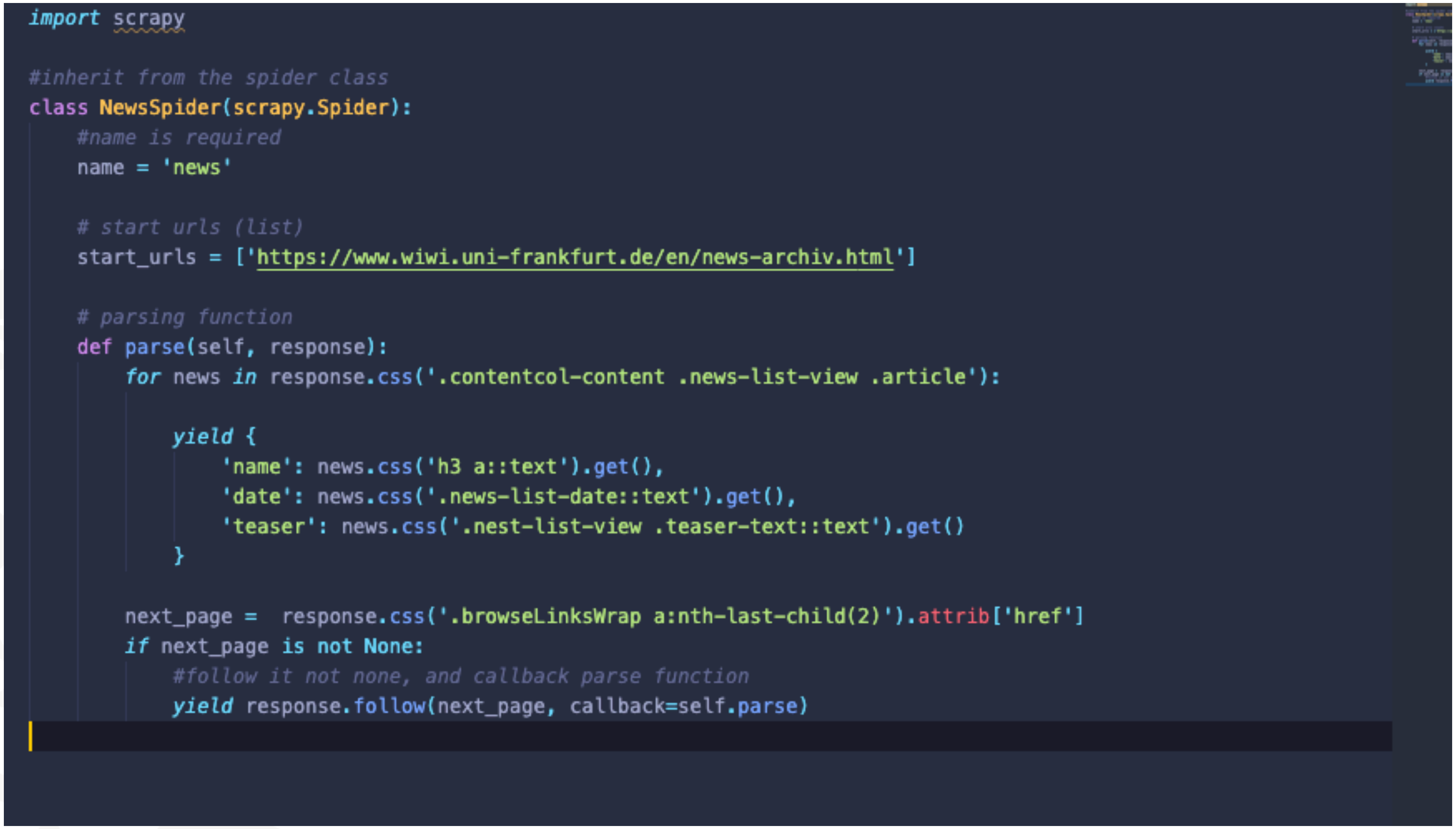
```
pip install Scrapy
```

- For mac users: Make sure xcode is installed or run: xcode-select –install
- It‘s good practice to use Scrapy within a virtual environment, e.g. use **venv**
- Create new Scrapy project: scrapy startproject <name>

```
scrapy startproject tutorial
```

Project: A simple webspider

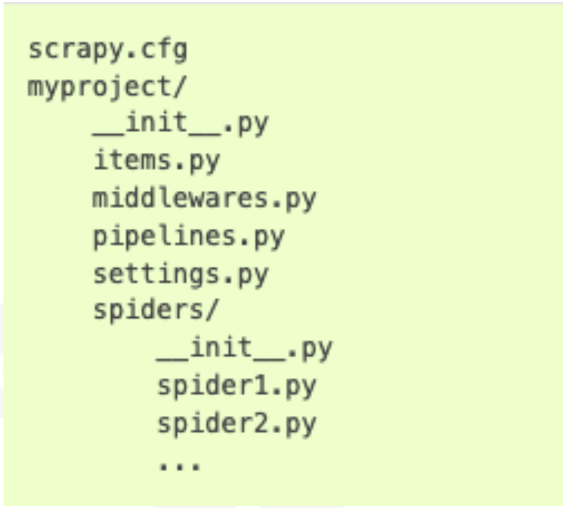
Open the jupyter notebook file on your computer or follow through with the html version



```python
import scrapy

#inherit from the spider class
class NewsSpider(scrapy.Spider):
    #name is required
    name = 'news'

    # start urls (list)
    start_urls = ['https://www.wiwi.uni-frankfurt.de/en/news-archiv.html']

    # parsing function
    def parse(self, response):
        for news in response.css('.contentcol-content .news-list-view .article'):

            yield {
                'name': news.css('h3 a::text').get(),
                'date': news.css('.news-list-date::text').get(),
                'teaser': news.css('.nest-list-view .teaser-text::text').get()
            }

        next_page =  response.css('.browseLinksWrap a:nth-last-child(2)').attrib['href']
        if next_page is not None:
            #follow it not none, and callback parse function
            yield response.follow(next_page, callback=self.parse)
```

Helpful Commands

Create a new scrapy project: **scrapy startproject <name>**



```
scrapy.cfg
myproject/
    __init__.py
    items.py
    middlewares.py
    pipelines.py
    settings.py
    spiders/
        __init__.py
        spider1.py
        spider2.py
        ...
```

Generate a new spider in the current folder or spider directory:
scrapy genspider <name> <doman>

Start the spider: **scrapy crawl <name>**

To start a spider in a self-contained python file use: **scrapy runspider <spider_file.py>**

Weaknesses of Scrapy

- **Downside of all webscraping:** Websites change, high maintenace required especially when using scrapers for a longer period
- Every website is different
- Cannot interact with the website (unlike selenium)
- Difficult HTML sites might be tricky to scrape in a large scale (use Xpath Selectors)
- Respects robots.txt files, similar to Selenium
- With more advanced websites in-depth knowledge of http requests and functionality of websites required to fully utilize scrapys framework

Final Notes & Further Resources

- Make a plan what data is needed and in what format
- Carefully inspect the webpage before coding the scraper
- Cookie Banners in European Sites can be challenging for Scrapy, e.g. the one on www.handelsblatt.com
- Documentation: https://docs.scrapy.org/en/latest/index.html
- Further Resources:

- Video Series Scrapy for beginners: https://www.youtube.com/watch?v=aIHTgF6polk&list=PLRzwgpycm-Fjvdf7RpmxnPMyJ80RecJjv
- Inspiration: https://coderslegacy.com/python/scrapy-project-examples/

Questions:

Feel free to leave me any questions in the forum or via E-Mail: erikreifer@gmail.com

Or pull the full source code from this presentation from git and try it out yourself:

https://github.com/e-reifer/scrapy_dafa