Data Extraction and Cleansing using Scrapy

Speakers: Mateen Ahmed
Ahmed Suffian Javed

Agenda

- Value of Data Extraction & Cleansing
- Explaining Scrapy framework
- Writing a Scrapy spider
- Data cleansing through pipelines
- Legality of crawling
- Deployment on Scrapinghub [bonus]

Applications

1. Data Science Process:
    - **Extraction**
    - **Cleansing**
    - Analysis
    - Visualization

2. Search Engines

Data Extraction

Few Data Sources:

- Databases
- Text files, CSV, JSON etc
- **Web Scraping**

Web Scraping - The analogy

Image Credits: Max Pixel
(http://maxpixel.freegreatpicture.com/Web-Wild-Spider-Wild-Nature-Spider-Web-Spider-2823306)

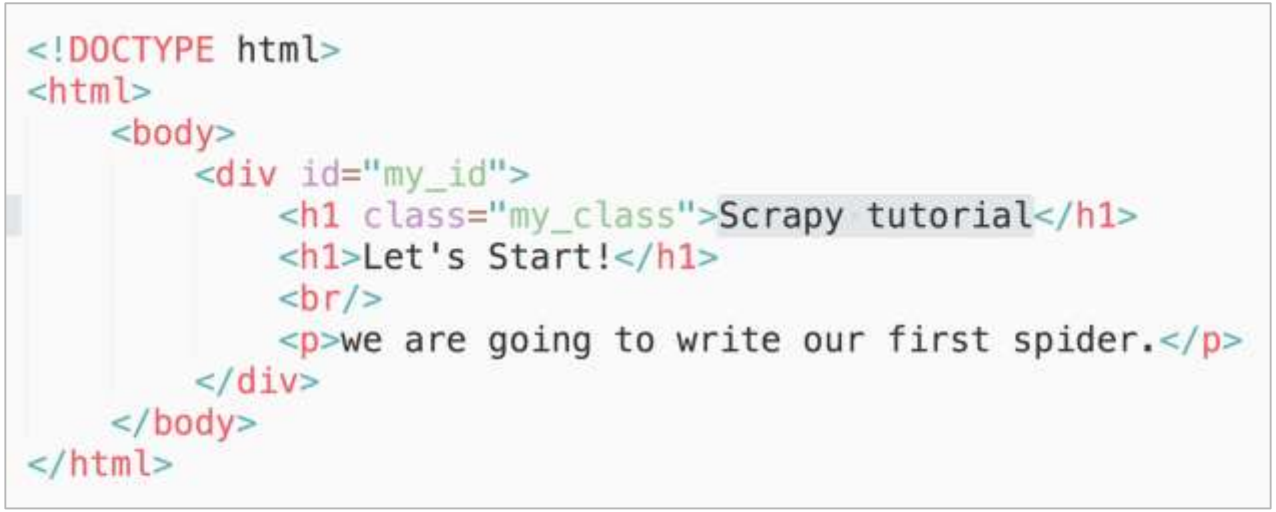
Why Scrapy?

- End to end tool for downloading, cleaning and saving data.
- Offers adequate post processing.
- Can handle websites behind login.
- Better error handling and resumable behaviour.
- Above all, **Asynchronous**.

Prerequisites for Scrapy

- Basics of Python
- Understanding of the Web:
  - HTML DOM
  - CSS/Xpath Selectors
- Scrapy installed on your machine.

Prerequisites explained



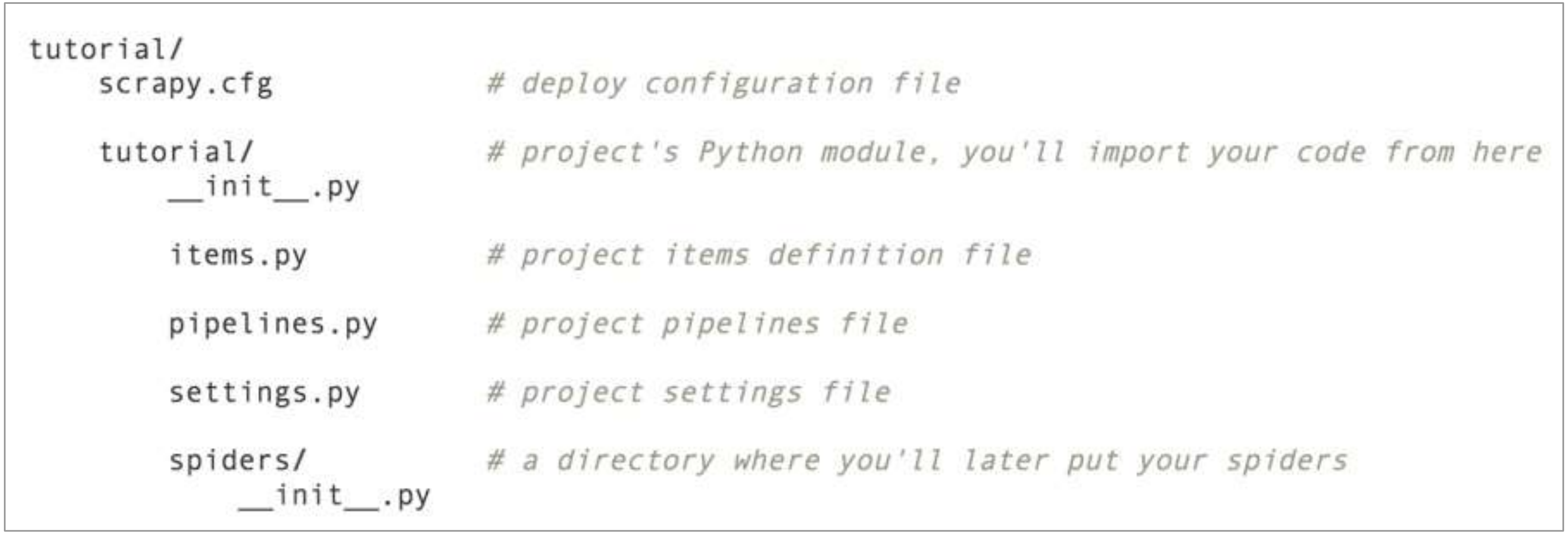
```
<!DOCTYPE html>
<html>
    <body>
        <div id="my_id">
            <h1 class="my_class">Scrapy tutorial</h1>
            <h1>Let's Start!</h1>
            <br/>
            <p>we are going to write our first spider.</p>
        </div>
    </body>
</html>
```

Xpath:

/html//div[@id="my_id"]/h1[@class="my_class"]/text()

CSS Selector: #my_id .my_class ::text

Python’s Scrapy Framework



```
tutorial/
    scrapy.cfg            # deploy configuration file

    tutorial/             # project's Python module, you'll import your code from here
        __init__.py

        items.py          # project items definition file

        pipelines.py      # project pipelines file

        settings.py       # project settings file

        spiders/          # a directory where you'll later put your spiders
            __init__.py
```

Spider Structure

Separation of concern:

- Crawler
- Parser

Crawling conveniently:

- Link Extractors
- Rules
- ItemLoaders

Let’s extract some data

Source Link: https://github.com/mateen91/scrapy-tutorial

Scrapy commands

Start New Project

```
scrapy startproject <project name>
```

Start New Spider

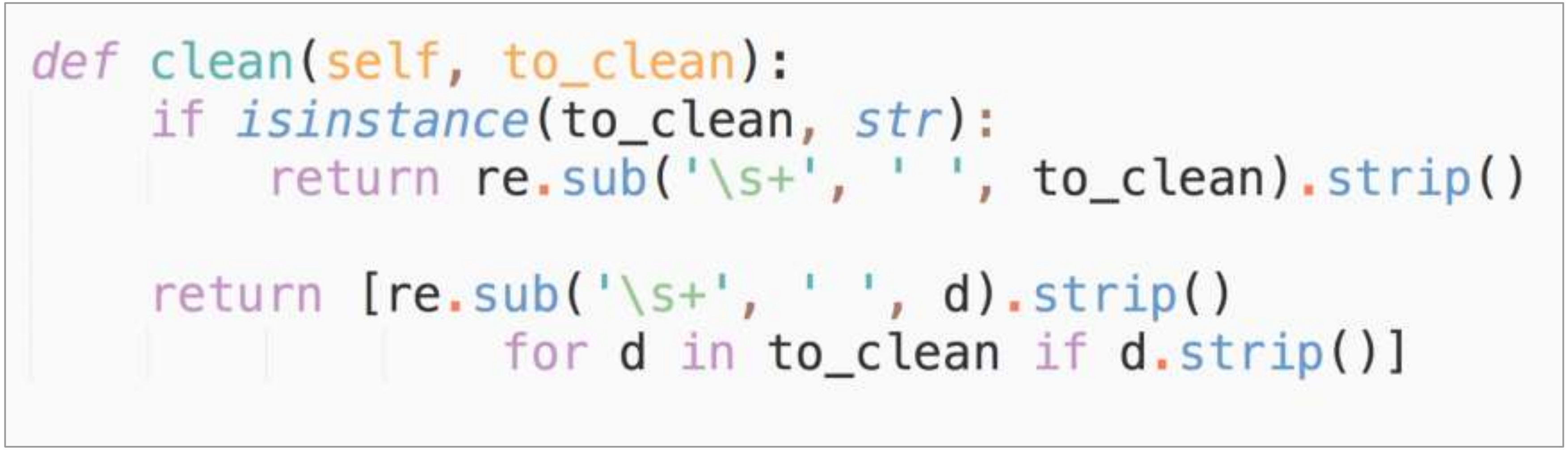
```
scrapy genspider <spider_name> <url>
```

Start a crawler

```
scrapy crawl <spider> -o <output_file>
```

Data Cleansing

Example: Remove extra whitespaces

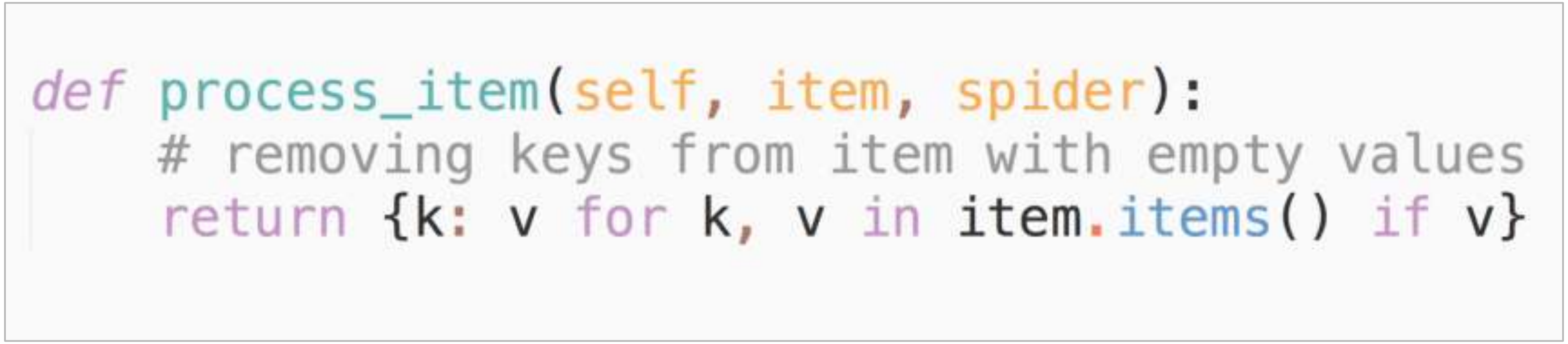


```python
def clean(self, to_clean):
    if isinstance(to_clean, str):
        return re.sub('\s+', ' ', to_clean).strip()

    return [re.sub('\s+', ' ', d).strip()
                for d in to_clean if d.strip()]
```

Scrapy Pipelines

Example: Remove empty fields from items



```python
def process_item(self, item, spider):
    # removing keys from item with empty values
    return {k: v for k, v in item.items() if v}
```

Robots.txt

- **Robots exclusion standard**
- Specifies which part can be crawled.

```
User-agent: *
Disallow: /
```

```
User-agent: *
Disallow: /cgi-bin/
Disallow: /tmp/
Disallow: /junk/
```

All robots stay out

All robots avoid these paths

Scrapy Settings

Some useful and commonly used settings:

- Download delay
- User-agent
- Enabling robots.txt
- Manging Throttling
- Enabling Cache

Anti blocking techniques

Rule of thumb: Don’t bombard the servers

- Increased Download delays
- Multiple User-agents
- Follow robots.txt
- Enable auto Throttle
- Use proxies - Not very ethical :)

Keeping it Legal and Ethical

- Public content but we should behave ethically.
- Follow robots.txt
- Always have some download delay.
- Do not crawl during peak hours.
- Write crawlers smartly - more data with less requests.

Questions