py4kids (https://github.com/wgong/py4kids)

Scrapy- Crawl & Scrape the Web

In this lesson we learn how to extract data from the web using powerful scrapy

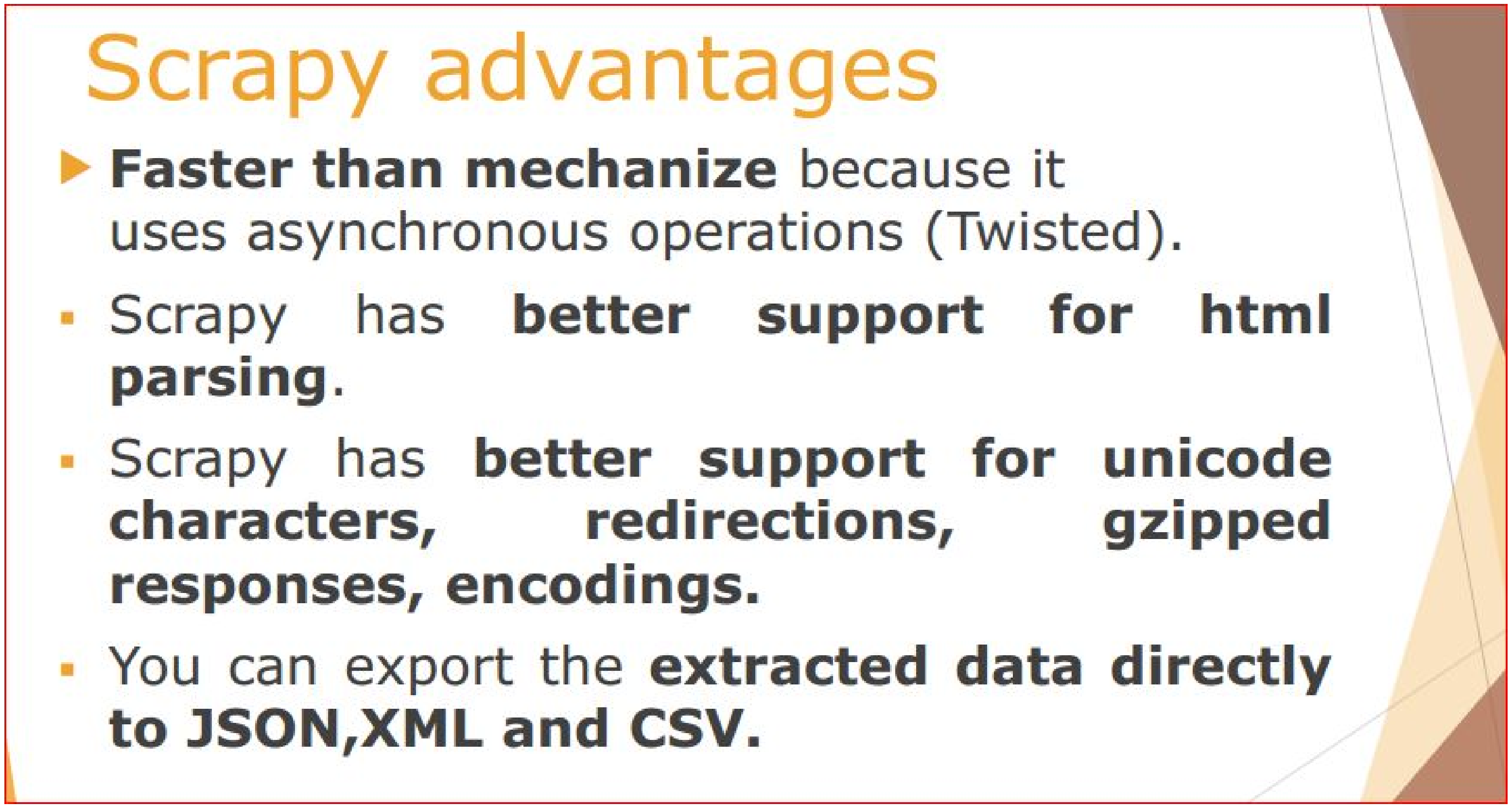
In [2]:
```
from jyquickhelper import add_notebook_menu
add_notebook_menu()
```

Out[2]:

Scrapy (https://scrapy.org/)

Overview

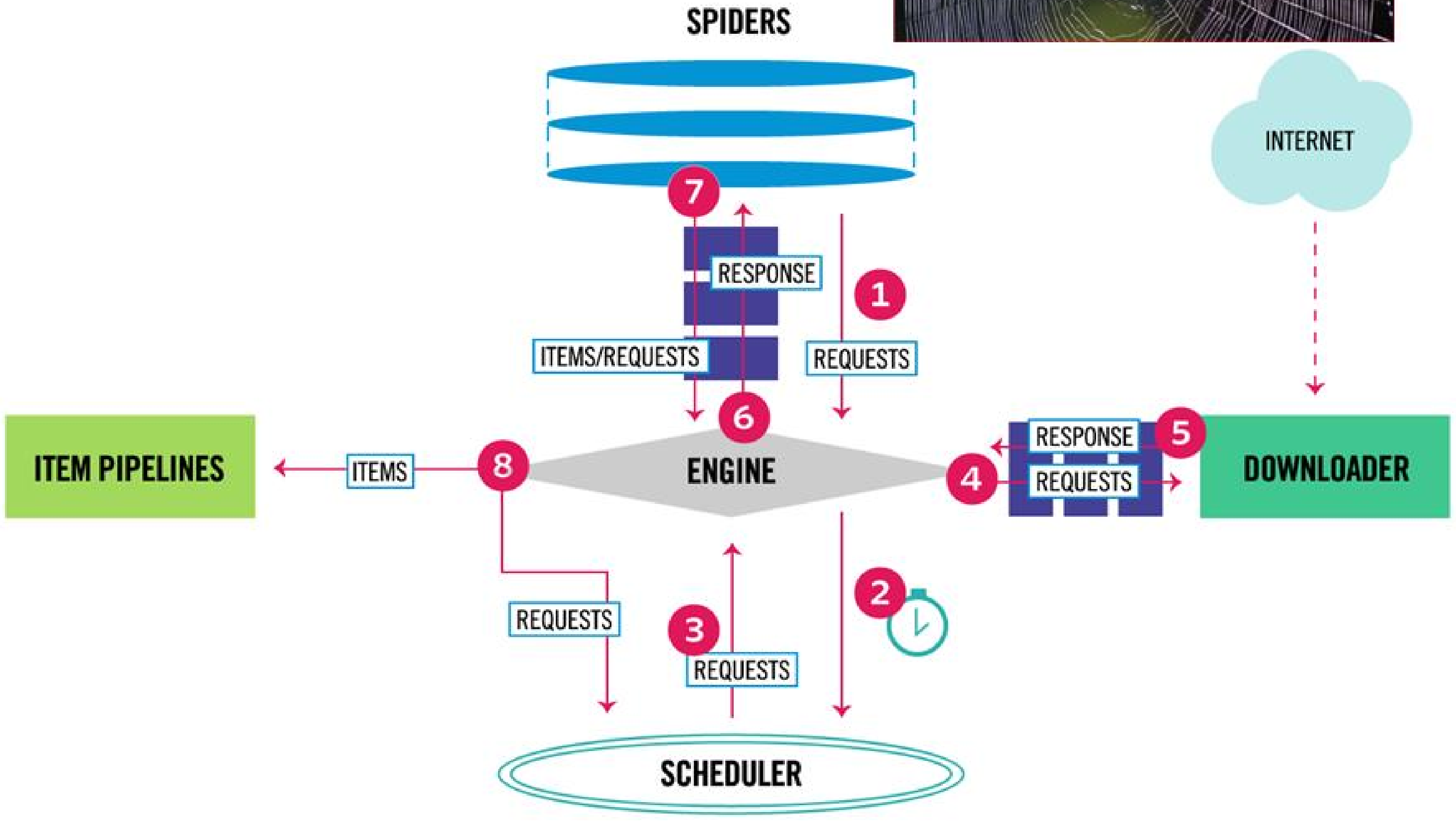
An open source and collaborative framework for extracting the data you need from websites. In a fast, simple, yet extensible way.



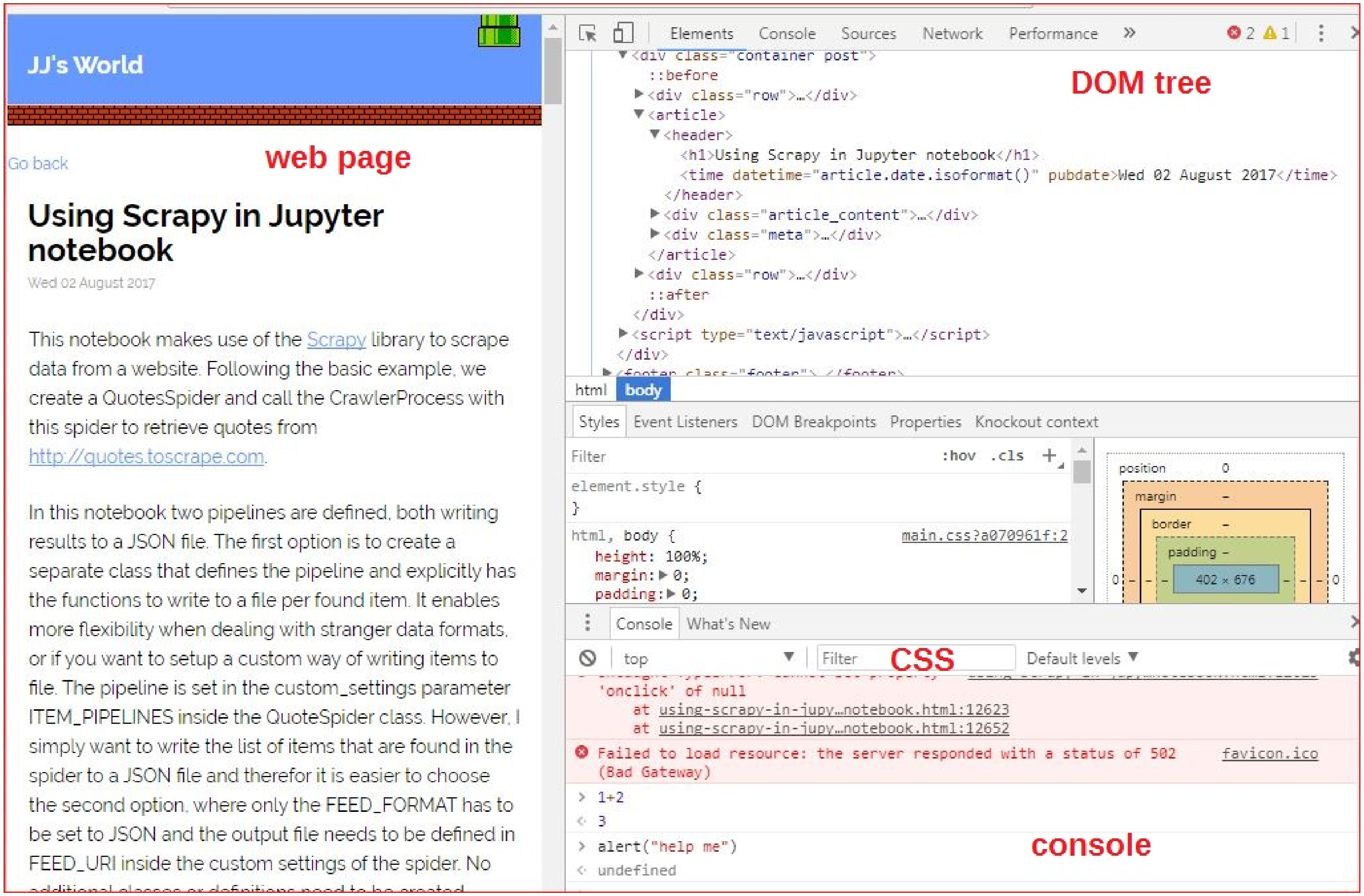
Scrapy advantages

- **Faster than mechanize** because it uses asynchronous operations (Twisted).
- Scrapy has **better support for html parsing**.
- Scrapy has **better support for unicode characters, redirections, gzipped responses, encodings.**
- You can export the **extracted data directly to JSON,XML and CSV.**

How does it work



DOM tree in browser



JJ's World

Go back

Using Scrapy in Jupyter notebook

Wed 02 August 2017

This notebook makes use of the Scrapy library to scrape data from a website. Following the basic example, we create a QuotesSpider and call the CrawlerProcess with this spider to retrieve quotes from http://quotes.toscrape.com.

In this notebook two pipelines are defined, both writing results to a JSON file. The first option is to create a separate class that defines the pipeline and explicitly has the functions to write to a file per found item. It enables more flexibility when dealing with stranger data formats, or if you want to setup a custom way of writing items to file. The pipeline is set in the custom_settings parameter ITEM_PIPELINES inside the QuoteSpider class. However, I simply want to write the list of items that are found in the spider to a JSON file and therefor it is easier to choose the second option, where only the FEED_FORMAT has to be set to JSON and the output file needs to be defined in FEED_URI inside the custom settings of the spider. No

```
▼<div class="container post">
   ::before
  ▶<div class="row">…</div>
  ▼<article>
    ▼<header>
       <h1>Using Scrapy in Jupyter notebook</h1>
       <time datetime="article.date.isoformat()" pubdate>Wed 02 August 2017</time>
     </header>
    ▶<div class="article_content">…</div>
    ▶<div class="meta">…</div>
   </article>
  ▶<div class="row">…</div>
   ::after
 </div>
▶<script type="text/javascript">…</script>
</div>
```

```
element.style {
}
html, body {                         main.css?a070961f:2
  height: 100%;
  margin:▶ 0;
  padding:▶ 0;
```

```
'onclick' of null
    at using-scrapy-in-jupy…notebook.html:12623
    at using-scrapy-in-jupy…notebook.html:12652
⊗ Failed to load resource: the server responded with a status of 502    favicon.ico
  (Bad Gateway)
> 1+2
< 3
> alert("help me")
< undefined
```

Install

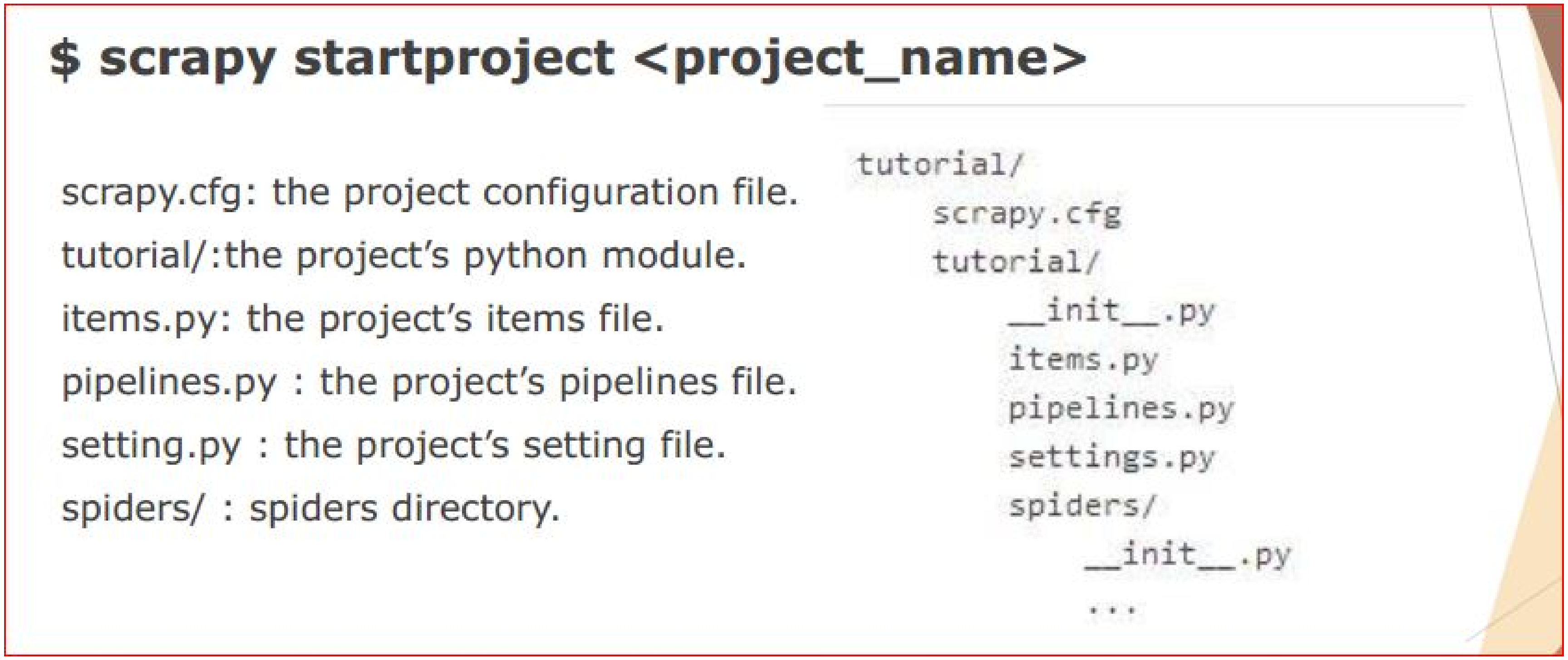
```
cmd> conda install scrapy
```

```
cmd> pip install scrapy
```

Create a project

- open a terminal window,
- create a working folder,
- type

```
cmd> scrapy startproject tutorial
```



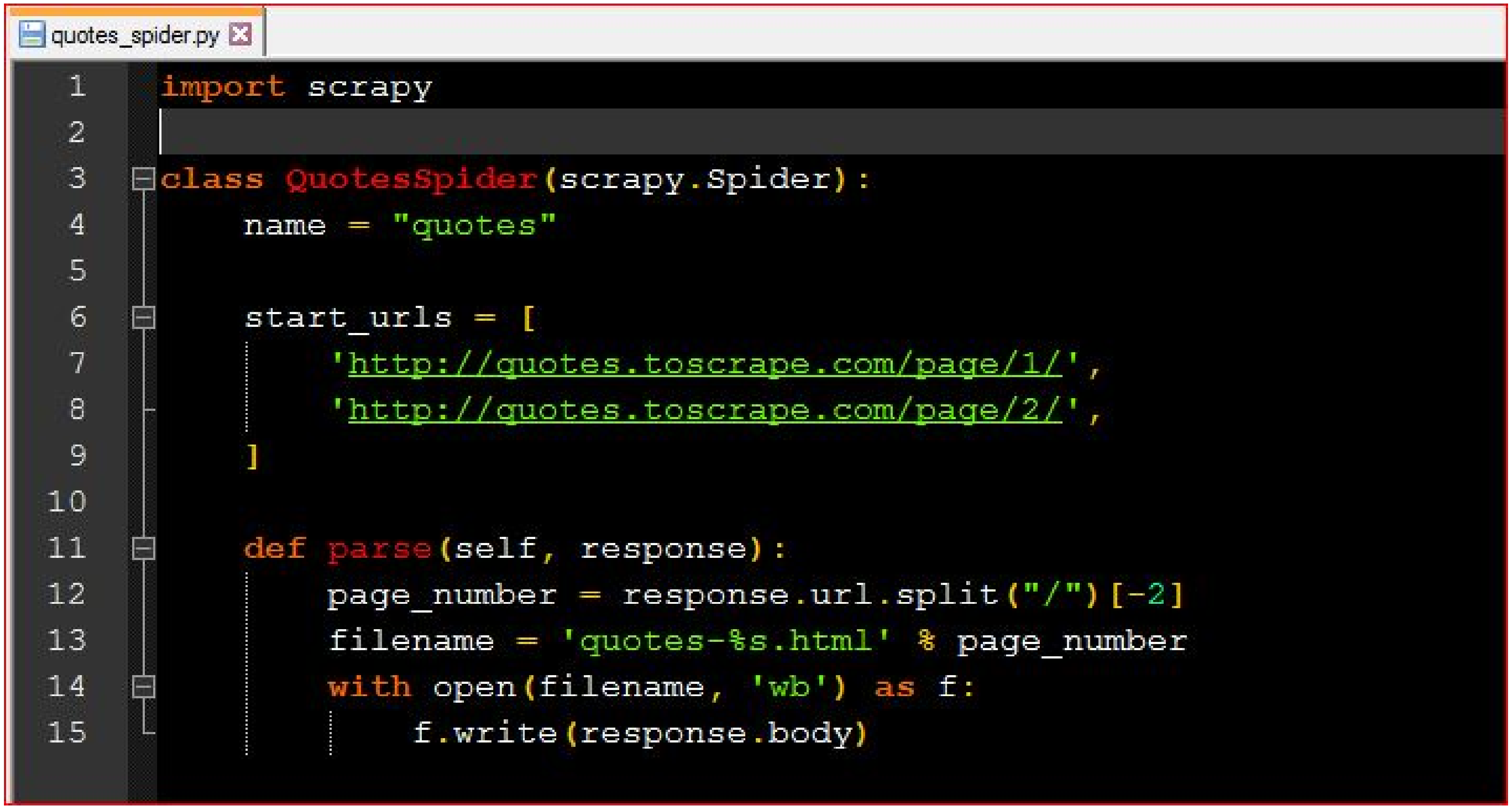
$ scrapy startproject <project_name>

scrapy.cfg: the project configuration file.
tutorial/:the project's python module.
items.py: the project's items file.
pipelines.py : the project's pipelines file.
setting.py : the project's setting file.
spiders/ : spiders directory.

```
tutorial/
    scrapy.cfg
    tutorial/
        __init__.py
        items.py
        pipelines.py
        settings.py
        spiders/
            __init__.py
            ...
```

Customize spider

- specify download web addresses (URL)
- define parser logic


quotes_spider.py

```
import scrapy

class QuotesSpider(scrapy.Spider):
    name = "quotes"

    start_urls = [
        'http://quotes.toscrape.com/page/1/',
        'http://quotes.toscrape.com/page/2/',
    ]

    def parse(self, response):
        page_number = response.url.split("/")[-2]
        filename = 'quotes-%s.html' % page_number
        with open(filename, 'wb') as f:
            f.write(response.body)
```

Launch spider

```
CMD> scrapy crawl quotes
```

Two files will be created in the current folder:

- quotes-1.html:
  - web copy (http://quotes.toscrape.com/page/1/)
  - local copy (./scrapy-tutorial/quotes-1.html)
- quotes-2.html
  - web copy (http://quotes.toscrape.com/page/2/)
  - local copy (./scrapy-tutorial/quotes-2.html)

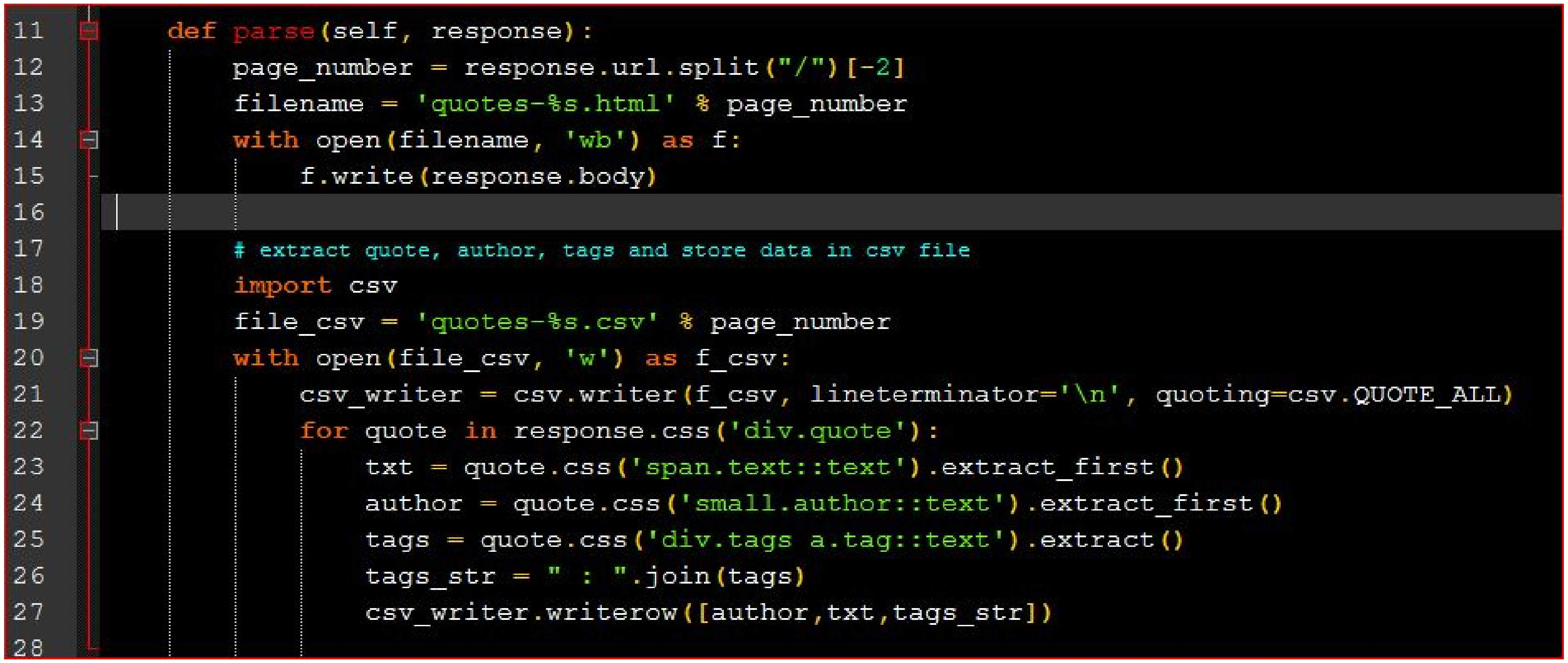
Extract data

```
CMD> scrapy crawl quotes
```

Two files will be created in the current folder:

- quotes-1.html:
  - web copy (http://quotes.toscrape.com/page/1/)
  - local copy (./scrapy-tutorial/quotes-1.html)
- quotes-2.html
  - web copy (http://quotes.toscrape.com/page/2/)
  - local copy (./scrapy-tutorial/quotes-2.html)

Extract data and save them into CSV file:



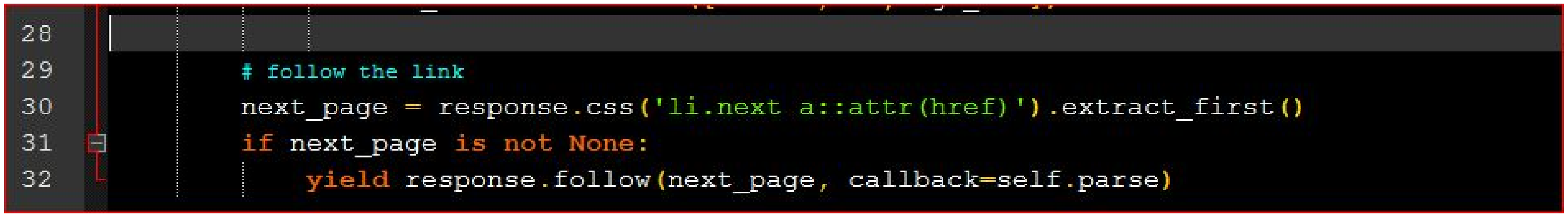
```
    def parse(self, response):
        page_number = response.url.split("/")[-2]
        filename = 'quotes-%s.html' % page_number
        with open(filename, 'wb') as f:
            f.write(response.body)

        # extract quote, author, tags and store data in csv file
        import csv
        file_csv = 'quotes-%s.csv' % page_number
        with open(file_csv, 'w') as f_csv:
            csv_writer = csv.writer(f_csv, lineterminator='\n', quoting=csv.QUOTE_ALL)
            for quote in response.css('div.quote'):
                txt = quote.css('span.text::text').extract_first()
                author = quote.css('small.author::text').extract_first()
                tags = quote.css('div.tags a.tag::text').extract()
                tags_str = " : ".join(tags)
                csv_writer.writerow([author,txt,tags_str])

```

Follow next link

To download all the quotes, follow "Next Page" line recursively:



```
        # follow the link
        next_page = response.css('li.next a::attr(href)').extract_first()
        if next_page is not None:
            yield response.follow(next_page, callback=self.parse)
```

References

- Basic Tutorial (https://doc.scrapy.org/en/latest/intro/tutorial.html)
- Using Scrapy in Jupyter notebook (http://www.jitsejan.nl/using-scrapy-in-jupyter-notebook.html)
- Build your 1st web crawler (https://medium.com/python-pandemonium/develop-your-first-web-crawler-in-python-scrapy-6b2ee4baf954)
- How To Crawl A Web Page with Scrapy and Python (https://www.digitalocean.com/community/tutorials/how-to-crawl-a-web-page-with-scrapy-and-python-3)
- Scrapy Tutorial: Web Scraping Craigslist (http://python.gotrained.com/scrapy-tutorial-web-scraping-craigslist/)

```
In [ ]:
```