Webscraping

- Harshal Chaudhari

What is scraping?

- Extract data from websites.
- Extract data from images/videos.
- Extract data from pdf files.

What is scraping?

- Extract data from websites.

Disclaimer

- Can be interpreted as a DoS attacks.
- Can affect quality of service provided by website (e.g. Uber API)
- Prohibited by Terms of Conditions of many websites.

Web administrators make it difficult to scrape websites, partly to protect what they view as intellectual property, and partly to save resources (bandwidth, server costs).

Ethics of scraping

The norms around appropriate behavior when collecting information from websites are still emerging, but a general principle is to **follow websites' terms of use** and, in particular, to follow the restrictions in the *robots.txt* file present on almost every website. For example, http://airbnb.com/robots.txt explicitly lists subdomains that are disallowed for automatic retrieval, and others (like the sitemap, as of March 20, 2018) that are fine.

Further reading

For some interesting discussions on the academic and journalistic ethics of scraping, see:
https://www.mailman.columbia.edu/research/population-health-methods/web-scraping

https://gijn.org/2015/08/12/on-the-ethics-of-web-scraping-and-data-journalism/

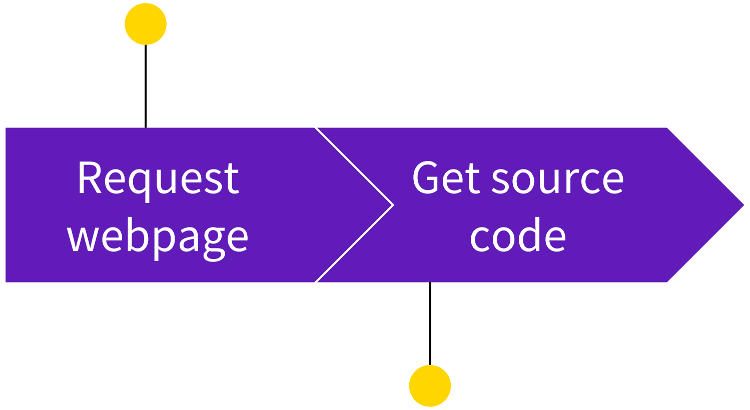
https://sites.stanford.edu/jumpstart-ituncon2015/ethics-legality-and-security-concerns-web-scrapingdata-collection-research

(This last site consists just of notes from a discussion at a workshop, but it is fairly intelligible.)

Procedure

Your browser requests a webpage using HTTP(s) protocol.

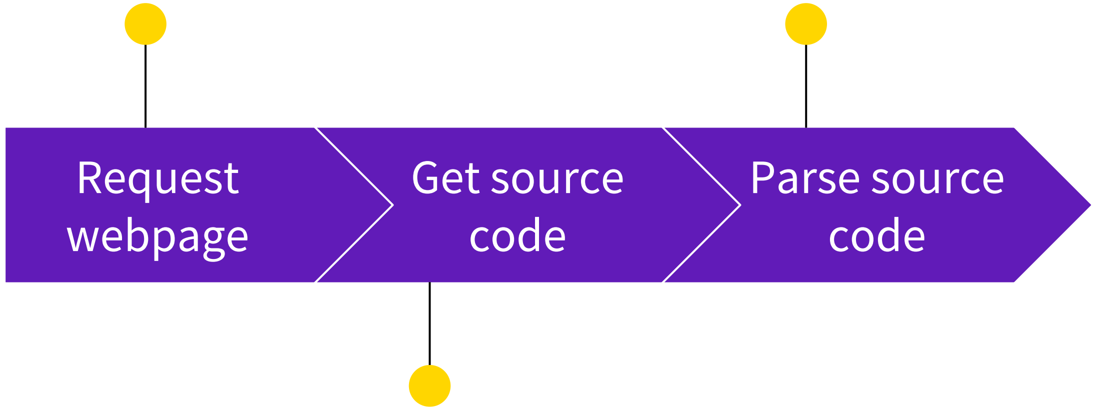
Your browser requests a webpage using HTTP(s) protocol.



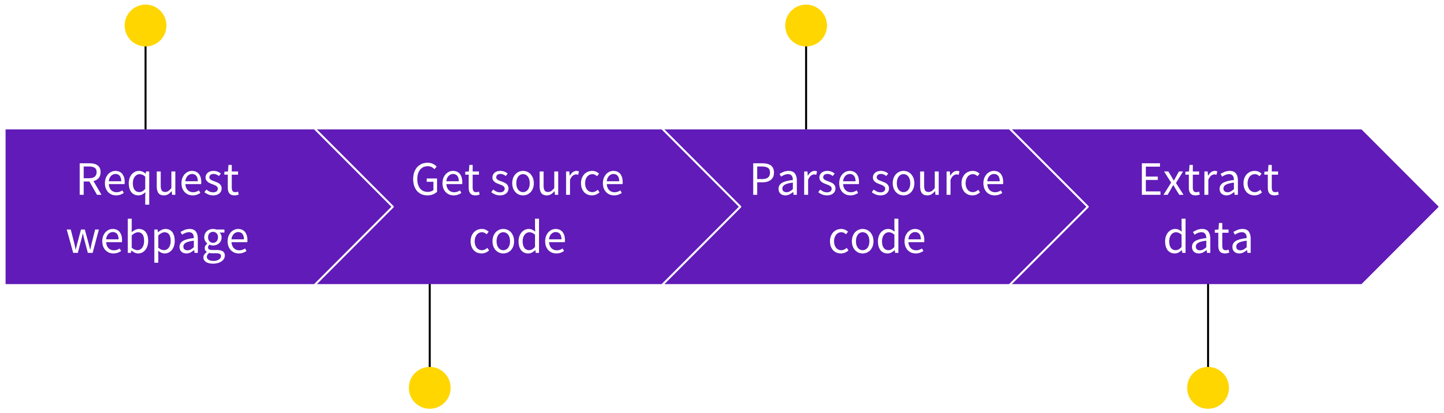
Webpage is typically coded in HTML language and its variants.

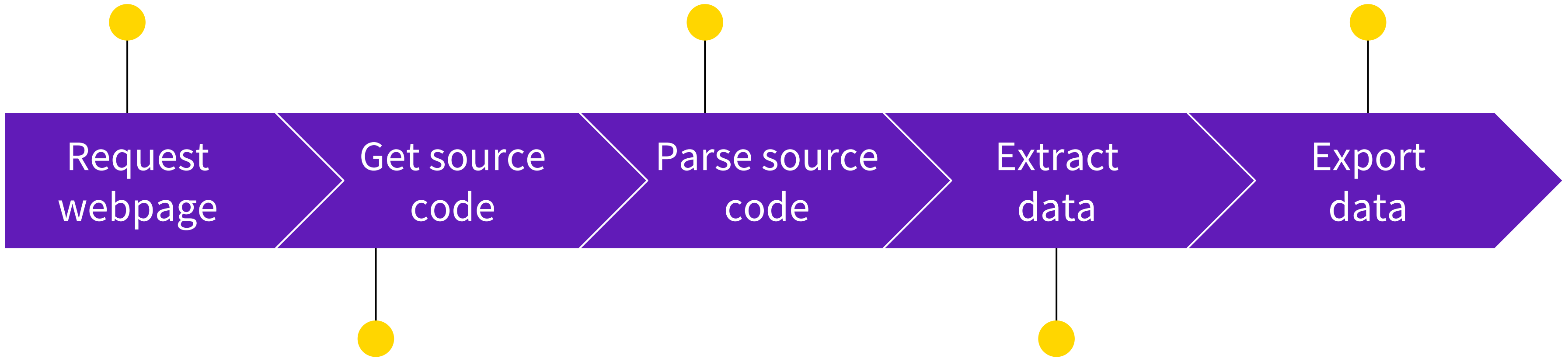
Your browser requests a webpage using HTTP(s) protocol.

Convert source code into DOM (Document Object Model Structure) parse tree



Webpage is typically coded in HTML language and its variants.


Your browser requests a webpage using HTTP(s) protocol.
Request webpage
Get source code
Webpage is typically coded in HTML language and its variants.
Convert source code into DOM (Document Object Model Structure) parse tree
Parse source code
Extract data
Navigate, search, modify parse tree to extract relevant data


Your browser requests a webpage using HTTP(s) protocol.
Request webpage
Get source code
Webpage is typically coded in HTML language and its variants.
Convert source code into DOM (Document Object Model Structure) parse tree
Parse source code
Extract data
Navigate, search, modify parse tree to extract relevant data
Export data tables to databases
Export data

Requesting Webpage

Command line tools

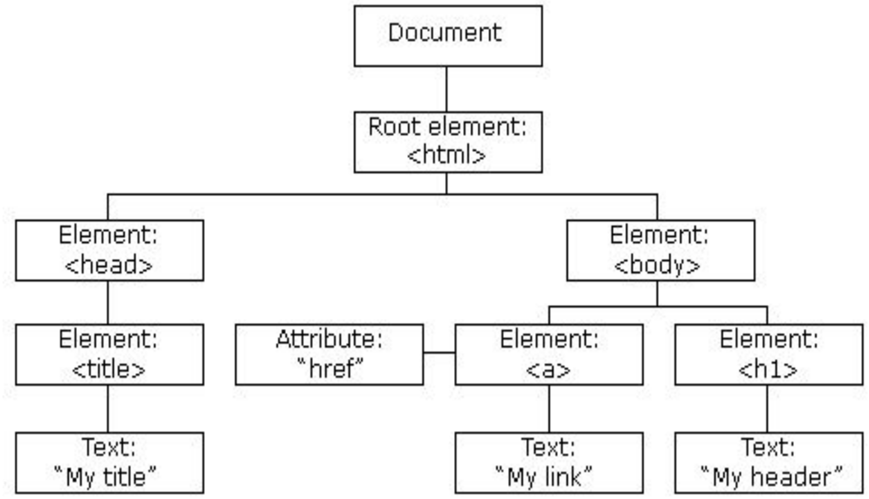
- ‘Wget’ or ‘curl’

Network access libraries

- Urllib, Request, pyCurl

Use “Inspect elements” option in browser to observe the requests in browser, mimic them as closely as possible.

Parsing Webpage



- Regular expressions
- Convert webpage source code into 'Document Object Model' (DOM).
- Traverse the DOM to extract relevant data.

BeautifulSoup

```
html_doc = """
<html><head><title>The Dormouse's story</title></head>
<body>
<p class="title"><b>The Dormouse's story</b></p>

<p class="story">Once upon a time there were three little sisters;
and their names were
<a href="http://example.com/elsie" class="sister"
id="link1">Elsie</a>,
<a href="http://example.com/lacie" class="sister"
id="link2">Lacie</a> and
<a href="http://example.com/tillie" class="sister"
id="link3">Tillie</a>;
and they lived at the bottom of a well.</p>

<p class="story">...</p>
"""
```

Documentation:
https://www.crummy.com/software/BeautifulSoup/bs4/doc/

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(html_doc, 'html.parser')
```

```
>>  soup.title
# <title>The Dormouse's story</title>
```

```
>>  soup.title.string
# u'The Dormouse's story'
```

```
>>  soup.find_all('a')
# [<a class="sister" href="http://example.com/elsie"
id="link1">Elsie</a>,
#  <a class="sister" href="http://example.com/lacie"
id="link2">Lacie</a>,
#  <a class="sister" href="http://example.com/tillie"
id="link3">Tillie</a>]
```

Storing & Exporting data

Exporting data

- SQLite database
- CSV files
- JSON files

Preferable to store original webpage source code, especially for webpages that frequently change.

Difficulties

- Websites identify automated access and block.
- HTML source codes change often, maintenance of ‘parsers’ is time-consuming activity.
- Modern websites rely on Javascript for dynamic loading.

Tricks to prevent being blocked

- Random ‘sleep’ time between consecutive requests.
- Randomize ‘User-Agent’ for each request.
- Use HTTP / SOCKS proxy lists.
- Multi-processing with proxies.

Sitemaps

- Major websites use ‘sitemap.xml’ to let search engines crawl their products / listings.
- Catalogue of listings.
- Robust to changes in source code.

Let’s look at sitemap of Airbnb.

AJAX Webpages

- Javascripts make requests to servers to fetch / display data.
- Source code obtained via code is not the same as the source code you see in browser.
- Example: Apple Jobs website https://jobs.apple.com/us/search

AJAX Webpages

- Extended HTTP Requests (XHR).
- Automated Browser Instance.
- Use JavaScript to write scraper - Node.js library.

Helpful Tips

- Write separate code modules for each step of the process.
- Only the ‘parser’ module changes for different websites.
- Use logging mechanism.

Be careful of unintended DOS attack!

See the Python notebook for some scraping examples

Thank You!