Web Scraping

Content

- Introduction
- HTML (and a bit XML)
- Beautiful Soup
- Apply

Content

- Introduction
- HTML (and a bit XML)
- Beautiful Soup
- Apply

Introduction

Web Scraping

Web scraping is a computer software technique of extracting information from websites

A data scientist should know how to scrape data from websites

Introduction

Web Pages

Web pages are build using **HTML**. Web scraping is all about HTML **tags**, therefore a basic understanding of HTML is necessary.

```html
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

This is a Heading

This is a paragraph.

Introduction

Don’t reinvent the wheel

Parsing a HTML document is nothing new and follows strict rules. Methods to convert a HTML document into a data structure are already available.

A possible Python module to parse a HTML document is **Beautiful Soup**

Content

- Introduction
- HTML (and a bit XML)
- Beautiful Soup
- Apply

HTML (and a bit XML)

Markup Languages

In computer text processing, a **markup language** is a system for annotating a document in a way that is syntactically distinguishable from the text.

As a simple example, the **start tag** indicates the start of a paragraph element and should be closed by the **end tag** , indicating the end of the element.

Some markup languages, such as the widely used **HTML**, have pre-defined presentation semantics—meaning that their specification prescribes how to present the structured data. Others, such as **XML**, do not have them and are general purpose.

HTML

```
Standard text and <b>bold text</b>
```

→ Standard text and **bold text**

LaTeX

```
Standard text and \bfseries{bold text}
```

→ Standard text and **bold text**

HTML (and a bit XML)

HyperText Markup Language (HTML)

- HTML describes the structure of web pages using markup
- HTML elements are the building blocks of HTML pages
- HTML elements are represented by tags
- HTML tags label pieces of content such as "heading", "paragraph", "table", and so on
- Browsers do not display the HTML tags, but use them to render the content of the page

tags are predefined

element

```
Standard text and <b>bold text</b>
```

tag tag

browser

Standard text and **bold text**

HTML (and a bit XML)

Basic syntax

element

```
<tag_name attribute_name="attribute_value">data</tag_name>
```

atttributes

start tag

data

end tag

HTML (and a bit XML)

Web Page

```
<html>

  <head>
    <title>The Dormouse's story</title>
  </head>

  <body>
    <h1 class="title"><b>The Dormouse's story</b></h1>
    <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>
  </body>

</html>
```

HTML (and a bit XML)

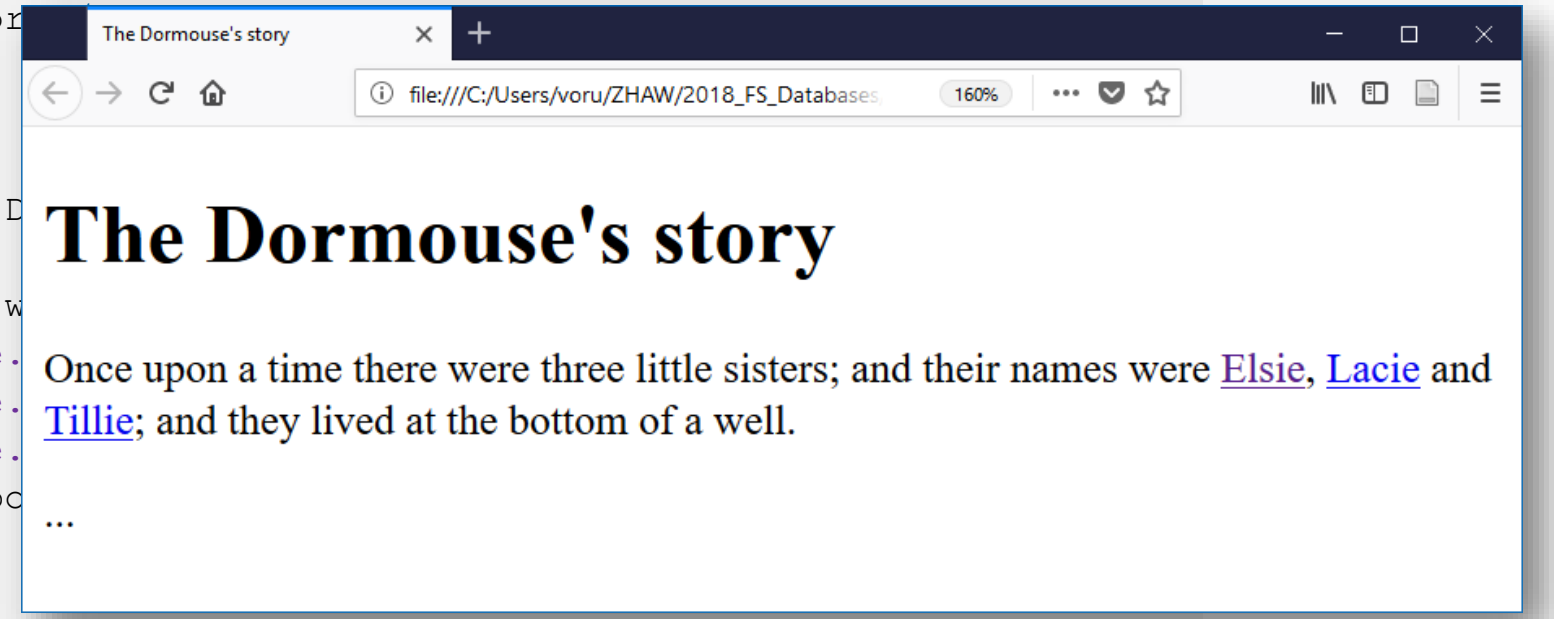
Web Page

```
<html>

  <head>
    <title>The Dormouse's stor
  </head>

  <body>
    <h1 class="title"><b>The D
    <p class="story">
      Once upon a time there w
      <a href="http://example.
      <a href="http://example.
      <a href="http://example.
      and they lived at the bo
    </p>
    <p class="story">...</p>
  </body>

</html>
```



HTML (and a bit XML)

eXtensible Markup Language (XML)

- XML is a markup language much like HTML
- XML was designed to store and transport data
- XML was designed to be self-descriptive
- XML is a W3C Recommendation

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <head>
    <title>Reminder</title>
    <date>14/02/2017</date>
  </head>
  <body lang='EN-US'>
    Theater this Sunday!
  </body>
</note>
```

XML and HTML were designed with different goals:

- XML was designed to carry data - with focus on what data is
- HTML was designed to display data - with focus on how data looks
- XML tags are not predefined like HTML tags are

XML does not do anything

HTML (and a bit XML)

Microsoft Office

Office

Word	*.docx
PowerPoint	*.pptx
Excel	*.xlsx
OneNote	
Outlook	

Microsoft uses XML to store data

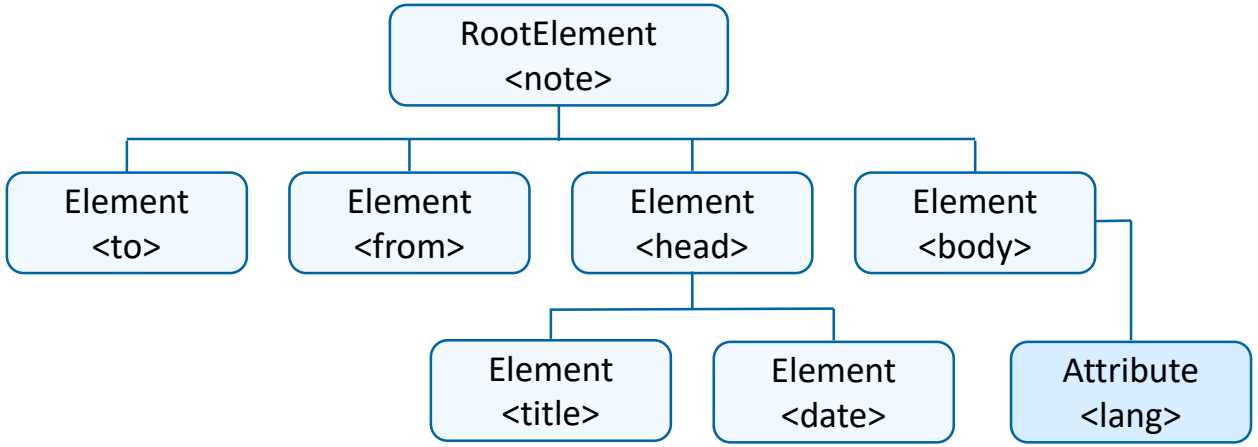
Example

HTML (and a bit XML)

Document Object Model (DOM)

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <head>
    <title>Reminder</title>
    <date>14/02/2017</date>
  </head>
  <body lang='EN-US'>
    Theater this Sunday!
  </body>
</note>
```

Document



DOM

HTML (and a bit XML)

Document Object Model (DOM)

```
<note>
  <to>Tove</to>
  <from>Jani</from>
  <head>
    <title>Reminder</title>
    <date>14/02/2017</date>
  </head>
  <body lang='EN-US'>
    Theater this Sunday!
  </body>
</note>
```

Document

parsing

```
// parse document
parser = new Parser()
dom = parser.parseFile(fileame)

// work with DOM
title = Dom.getElementsByTagName('title')
print(title[0].str())
```

```
>> parser.exe -filename 'myXMLFile'
Reminder
>>
```

pseudo-code

HTML (and a bit XML)

Parsing HTML/XML in Python

Python 3 is shipped with the module xml.

```
# import
from xml.dom import minidom

# parse file
dom = minidom.parse('note.xml')

# get title
title = dom.getElementsByTagName('title')
print(title[0].firstChild.data)
```

Content

- Introduction
- HTML (and a bit XML)
- Beautiful Soup
- Apply

Beautiful Soup (Python Module)

Installing Beautiful Soup

If you're using a recent version of Debian or Ubuntu Linux, you can install Beautiful Soup with the system package manager:

`$ apt-get install python-bs4` (for Python 2)

`$ apt-get install python3-bs4` (for Python 3)

Beautiful Soup 4 is published through PyPi, so if you can't install it with the system packager, you can install it with `easy_install` or `pip`. The package name is `beautifulsoup4`, and the same package works on Python 2 and Python 3. Make sure you use the right version of `pip` or `easy_install` for your Python version (these may be named `pip3` and `easy_install3` respectively if you're using Python 3).

```
$ easy_install beautifulsoup4
```

```
$ pip install beautifulsoup4
```

(The `BeautifulSoup` package is probably *not* what you want. That's the previous major release, Beautiful Soup 3. Lots of software uses BS3, so it's still available, but if you're writing new code you should install `beautifulsoup4`.)

If you don't have `easy_install` or `pip` installed, you can download the Beautiful Soup 4 source tarball and install it with `setup.py`.

```
$ python setup.py install
```

If all else fails, the license for Beautiful Soup allows you to package the entire library with your application. You can download the tarball, copy its `bs4` directory into your application's codebase, and use Beautiful Soup without installing it at all.

Beautiful Soup (Python Module)

Quick Start

```
# import beautiful soup
import bs4

# open the file and get the "soup"
with open('three_sisters.html') as html_doc:
        soup = bs4.BeautifulSoup(html_doc, 'html.parser')

# show
print(soup.prettify())
```

Beautiful Soup (Python Module)

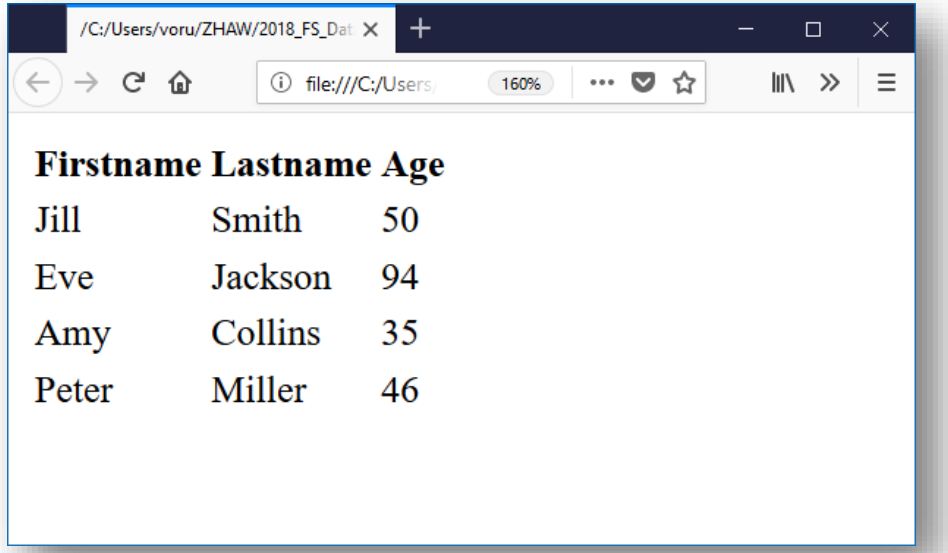
Navigate

```
# navigate
print(soup.title)                # title element
print(soup.title.name)           # element name
print(soup.title.string)         # element string
print(soup.title.parent.name)    # name of the parent element
print(soup.p)                    # first 'p' element
print(soup.p['class'])           # value of the attribute 'class'
print(soup.a)                    # first 'a' element
print(soup.find_all('a'))        # list of all 'a' elements
print(soup.find(id='link3'))     # element with id 'link3'
```

Beautiful Soup (Python Module)

Data is often presented in tables…

```
<table>
  <tr>
    <th>Firstname</th> <th>Lastname</th> <th>Age</th>
  </tr>
  <tr>
    <td>Jill</td> <td>Smith</td> <td>50</td>
  </tr>
  <tr>
    <td>Eve</td> <td>Jackson</td> <td>94</td>
  </tr>
  <tr>
    <td>Amy</td> <td>Collins</td> <td>35</td>
  </tr>
  <tr>
    <td>Peter</td> <td>Miller</td> <td>46</td>
  </tr>
</table>
```



Content

- Introduction
- HTML (and a bit XML)
- Beautiful Soup
- Apply

Apply

Requesting a web page

```
# import
import urllib.request

# get pointer to webpage
wp = urllib.request.urlopen('http://www.somepage.com')

# use it like a file pointer
[some code]

# close
wp.close()
```

Apply

Nobel Prize Winners