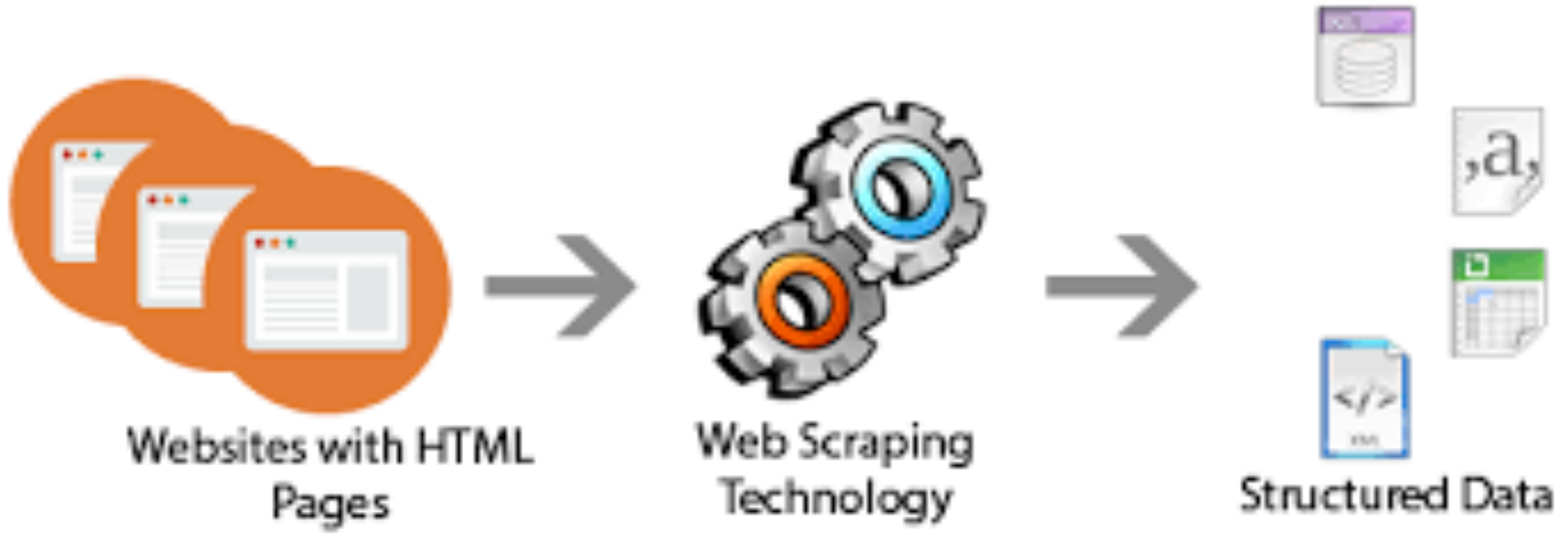
Webscraping basics

Week 6 day 4

Webscraping

- Web scraping is a technique of extracting information from websites. It focuses on transformation of unstructured data on the web, into structured data that can be stored and analyzed.



Ways to Webscrape

- 3rd Party Services (import.io)
- Write our own Python apps that pull HTML documents and parse them
  - Mechanize
  - Scrapy
  - Requests
  - libxml / XPath
  - Regular expressions

HTML

- The document itself is a document node.
- All HTML elements are element nodes.
    - <title>I am a title.</title> <p>I am a paragraph.</p> <strong>I am bold.</strong>
- All HTML attributes are attribute nodes.
- Text inside HTML elements are text nodes.
- Comments are comment nodes.

HTML – just another language

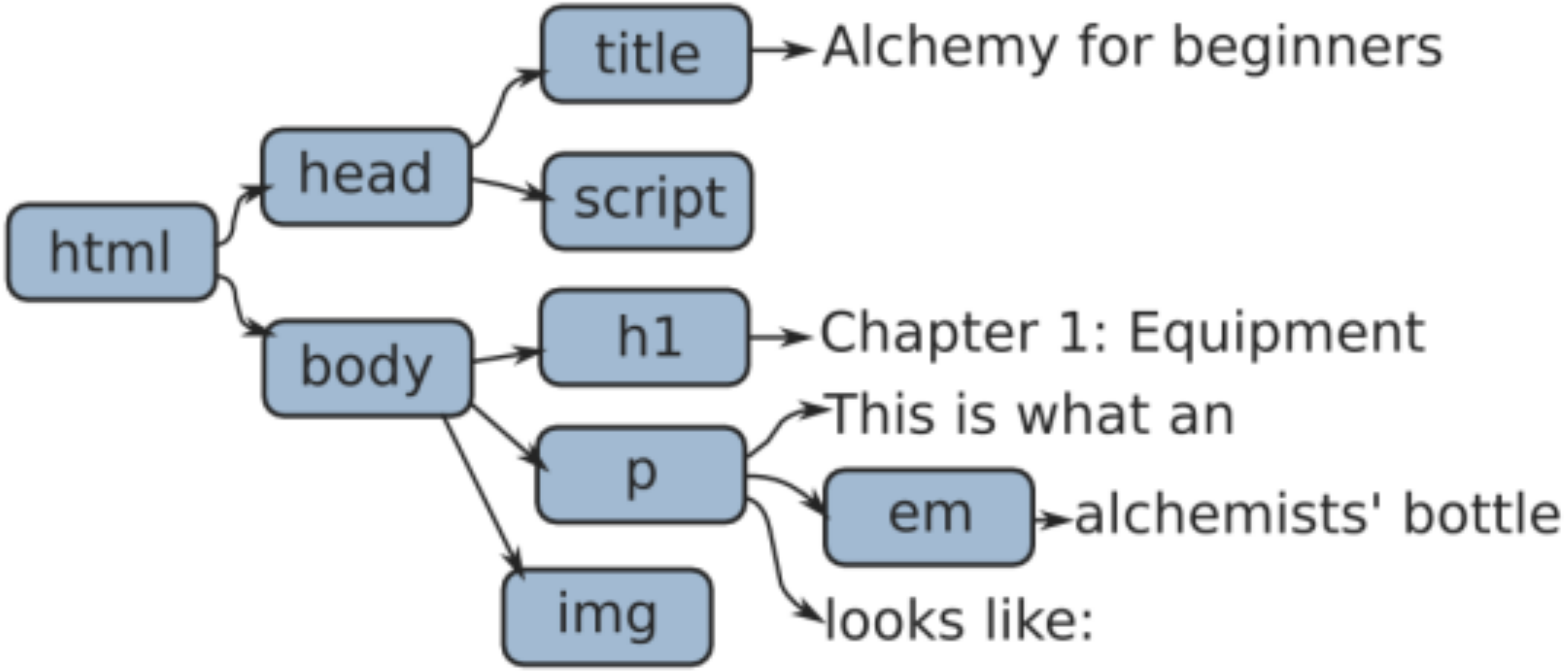
Elements can have parents and children:

```
<body>
<div>I am inside the parent element
        <div>I am inside a child element</div>
        <div>I am inside another child element</div>
        <div>I am inside yet another child element</div>
</div> </body>
```

HTML Elements

- Elements can also have attributes! Attributes are defined inside **element tags** and can contain data that may be useful to scrape.
- <a href="http://lmgtfy.com/?q=html+element+attributes" title="A title" id="web-link" name="hal">A Simple Link</a>

- HTML documents have what is called a hierarchical structure. Each element (or tag) except the top <html> tag is contained in another element, its parent. This element can in turn contain child elements. You can visualise this as a kind of family tree:



Xpath locates information inside HTML

- XPath uses path expressions to select nodes or node-sets
- XPath includes over 200 built-in functions.

Xpath expressions

Expression	Description
nodename	Selects all nodes with the name "*nodename*"
/	Selects from the root node
//	Selects nodes in the document from the current node that match the selection no matter where they are
.	Selects the current node
..	Selects the parent of the current node
@	Selects attributes

Xpath expressions: cont.

Path Expression	Result
bookstore	Selects all nodes with the name "bookstore"
/bookstore	Selects the root element bookstore **Note:** If the path starts with a slash (/) it always represents an absolute path to an element!
bookstore/book	Selects all book elements that are children of bookstore
//book	Selects all book elements no matter where they are in the document
bookstore//book	Selects all book elements that are descendant of the bookstore element, no matter where they are under the bookstore element
//@lang	Selects all attributes that are named lang

Xpath and HTML

To find the link in this page:
<html><body> <p>The fox jumped over the lazy brown <a href="dogs.html">dog</a>.</p> </body></html>

A raw XPath traverses the hierarchy from the root element of the document (page) to the desired element:
/html/body/p/a

XPath can find an element by ID like this:
//*[@id="element_id"]

XPath like this to find the first link that is a child of the element with ID=”fox”:
//∗[@id="fox"]/a

Xpath cheatsheet

http://ricostacruz.com/cheatsheets/xpath.html

A simple example: lxml and requests

```
from lxml import html
import requests
#Use requests.get to retrieve the web page with our data, parse it using the html module and save the results in tree

page =requests.get('http://econpy.pythonanywhere.com/ex/001.html')
tree = html.fromstring(page.content)

#tree now contains the html code
```

After a quick analysis, we see that in our page the data is contained in two elements - one is a div with title ‘buyer-name’ and the other is a span with class ‘item-price’:

```
<div title="buyer-name">Carson Busses</div>
<span class="item-price">$29.95</span>
```

Knowing this we can create the correct XPath query and use the lxml xpath function like this:

```
#This will create a list of buyers:
buyers = tree.xpath('//div[@title="buyer-name"]/text()')
#This will create a list of prices
prices = tree.xpath('//span[@class="item-price"]/text()')
```

```
print('Buyers: ', buyers)
print('Prices: ', prices)
```

```
Buyers: ['Carson Busses', 'Earl E. Byrd', 'Patty Cakes', 'Derri Anne
Connecticut', 'Moe Dess', 'Leda Doggslife', 'Dan Druff', 'Al Fresco', 'Ido
Hoe', 'Howie Kisses', 'Len Lease', 'Phil Meup', 'Ira Pent', 'Ben D. Rules',
'Ave Sectomy', 'Gary Shattire', 'Bobbi Soks', 'Sheila Takya', 'Rose Tattoo',
'Moe Tell']

Prices: ['$29.95', '$8.37', '$15.26', '$19.25', '$19.25', '$13.99', '$31.57',
'$8.49', '$14.47', '$15.86', '$11.11', '$15.98', '$16.27', '$7.50', '$50.85',
'$14.26', '$5.68', '$15.00', '$114.07', '$10.09']
```

Next

- from scrapy.selector import Selector
- from scrapy.http import HtmlResponse
- from bs4 import BeautifulSoup
- *beautiful soup:*
  - conda install bs4
  - conda install lxml
- *or if conda doesn't work*
  - pip install bs4
  - pip install lxml