Python library for web scraping

* Urllib2: Python building URL utility
* requests: HTTP library. Download files and web pages from the Internet. Can parse content. Make requests.
* Selenium: launch and control a web browser. Simulate navigating in a web browser. Contains a headless browser.
* mechanize: open webpages and parse info in webpages. Sort of Selenium + BeautifulSoup. But it can only handle static pages but not Javascript-generated pages.
* BeautifulSoup: A library designed for screen-scraping HTML and XML in Python
* lxml: A library for parsing XML and HTML in Python
* Scrapy: scraping framework
* Webscraping: Python library. Haven’t tried this yet.
* Pyquery: ???

HTML Tables – An HTML table is divided into rows (with the <tr> tag), and each row is divided into data cells (with the <td> tag). td stands for "table data," and holds the content of a data cell. A <td> tag can contain text, links, images, lists, forms, and other tables (W3 Schools)

Note: HTML tables are structured just like tables in excel and by using python we can easily scrape data from tables found on a website and save the data in an excel file on a local drive.

Good resources:

* <http://www.analyticsvidhya.com/blog/2015/10/beginner-guide-web-scraping-beautiful-soup-python/> Good intro to BeautifulSoup and HTML
* <http://jakeaustwick.me/python-web-scraping-resource/>
* Include overview of different methods: <http://rhodesmill.org/brandon/chapters/screen-scraping/>
* <https://automatetheboringstuff.com/chapter11/>
* Greg Reda:
  + <http://www.gregreda.com/2013/03/03/web-scraping-101-with-python/>
  + <http://www.gregreda.com/2013/04/29/more-web-scraping-with-python/>
  + <http://www.gregreda.com/2015/02/15/web-scraping-finding-the-api/> Something about client side scripting API
* <https://en.wikipedia.org/wiki/Dynamic_web_page>
  + Server-side scripting
  + Client-side scripting (e.g. JavaScript)
  + Ajax

Format of web content: XPATH, DOM, CSS

* <https://www.simple-talk.com/dotnet/.net-framework/xpath,-css,-dom-and-selenium-the-rosetta-stone/>
* <http://www.w3schools.com/xsl/xpath_syntax.asp>

Selenium:

* Good document: <http://selenium-python.readthedocs.io/>
* Good cheatsheet for selenium: <https://gist.github.com/huangzhichong/3284966>
* Use Chrome in Selenium.
  + Download standalone Chrome driver from into a folder <https://sites.google.com/a/chromium.org/chromedriver/home>
  + Specify the location of chromedriver.ext in br = webdriver.Chrome('folder\chromedriver ' )
* Get HTML of an element: element.get\_attribute('innerHTML')
* For loop with find\_element\_by\_xpath is much slower than one query find\_element**s**\_by\_xpath() since the former involves a lot of queries. The latter one returns a list of all elements.

Find div with span child that contains certain text. Refine search by its child, grandchild nodes.

driver.fine\_elements\_by\_xpath("//div[span[text()='hello world']]")

Logic combination

"//div[contains( text(), '.raw' ) and contains( text(), 'raw') ]"

Wildcard on ID, attributes, …

“//div[contains(@id, ‘substring’)]”

“//div[starts-with(@id, ‘substring’)]”