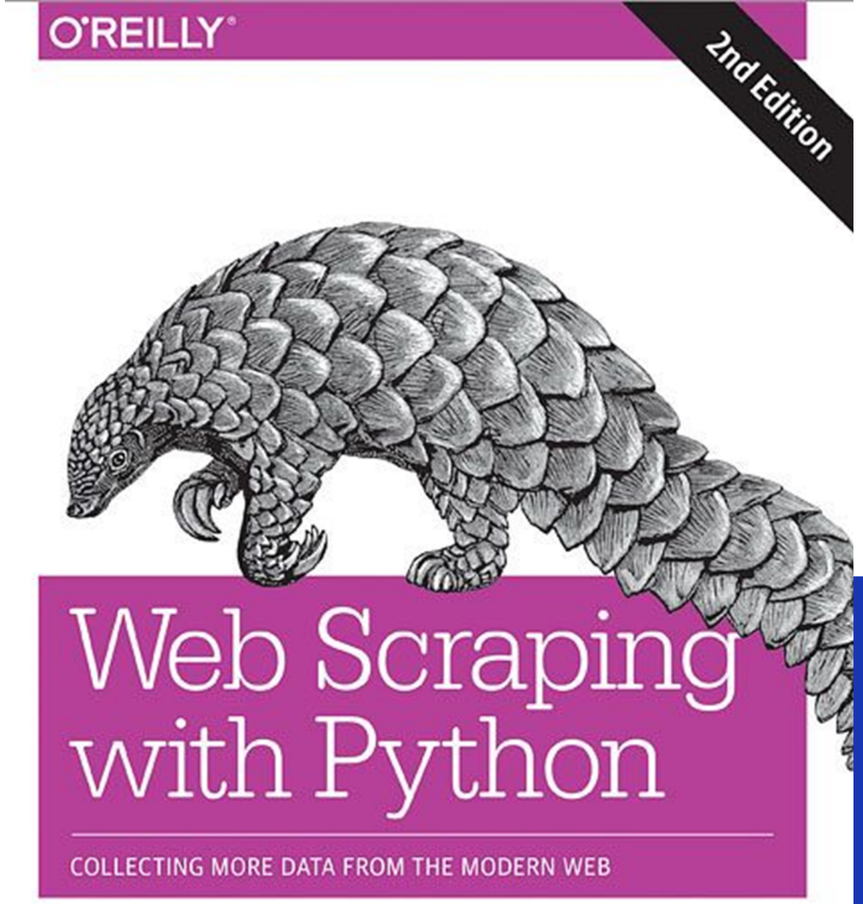
By Olha Kravets

WEB SCRAPING WITH PYTHON: COLLECTING MORE DATA FROM THE MODERN WEB



About the Author

Ryan Mitchell is a senior software engineer at HedgeServ in Boston, where she develops APIs and data analytics tools for hedge fund managers

Ryan Mitchell

About Web Scraping With Python

«The tools and examples included in the book allowed me to easily automate several repetitive tasks, freeing that time to solve more interesting problems. It is a results-oriented, quick read that is well rooted in real-world problems and solutions."

-Eric VanWyk
Electrical Computer Engineer,
Olin College of Engineering

Table of Contents

(Part 1)

Chapter 1. Your First Web Scraper

- Connecting
- An Introduction to BeautifulSoup
  - Installing BeautifulSoup
  - Running BeautifulSoup
  - Connecting Reliably and Handling Exceptions

```
from urllib.request import urlopen
from bs4 import BeautifulSoup

html = urlopen('http://www.pythonscraping.com/pages/page1.html')
bs = BeautifulSoup(html.read(), 'html.parser')
print(bs.h1)
```

The output is as follows:

```
<h1>An Interesting Title</h1>
```

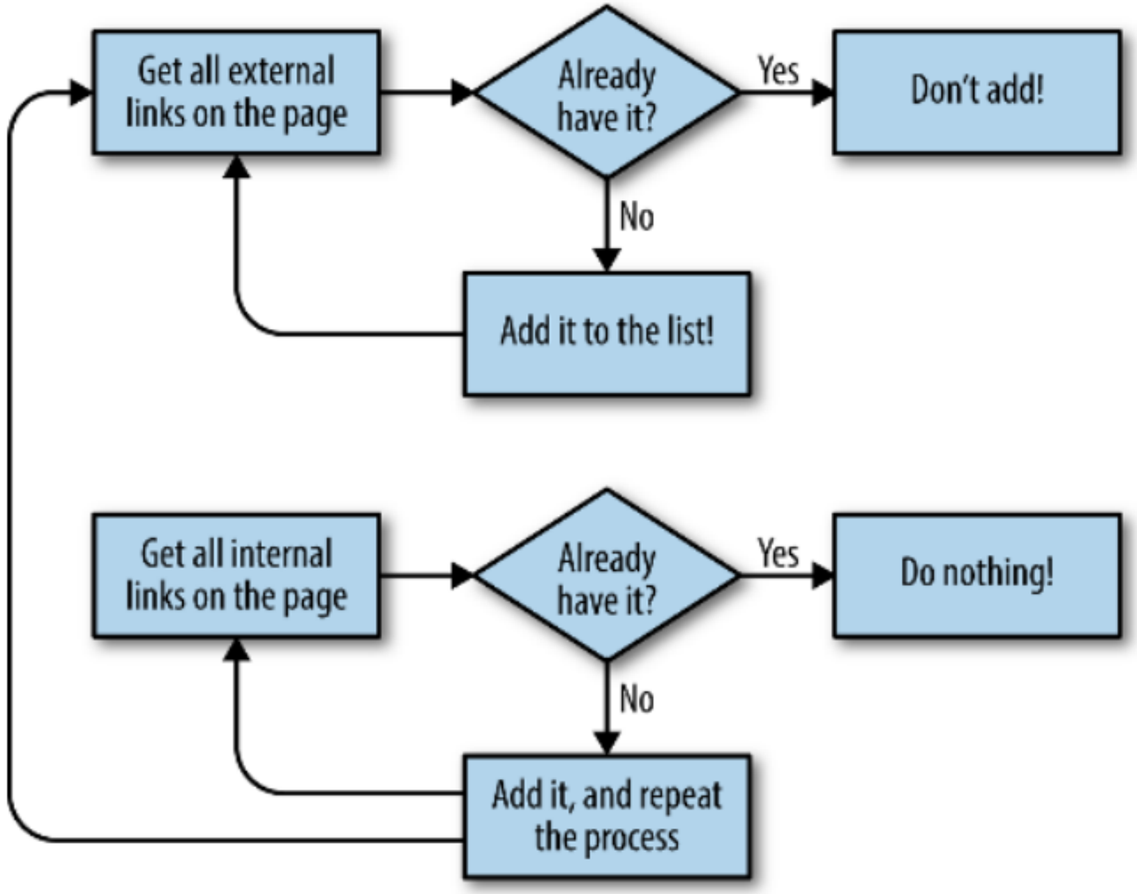
Chapter 2. Advanced HTML Parsing

```
from urllib.request import urlopen
from bs4 import BeautifulSoup
import re

html = urlopen('http://www.pythonscraping.com/pages/page3.html')
bs = BeautifulSoup(html, 'html.parser')
images = bs.find_all('img',
    {'src':re.compile('\.\.\/img\/gifts/img.*\.jpg')})
for image in images:
    print(image['src'])
```

Chapter 3. Writing Web Crawlers

- Traversing a Single Domain
- Crawling an Entire Site
  - Collecting Data Across an Entire Site
- Crawling Across the Internet



Flow diagram for the website crawler that collects all external links

Chapter 4. Web Crawling Models

```
import requests
from bs4 import BeautifulSoup

class Crawler:

    def getPage(self, url):
        try:
            req = requests.get(url)
        except requests.exceptions.RequestException:
            return None
        return BeautifulSoup(req.text, 'html.parser')

    def safeGet(self, pageObj, selector):
        """
        Utility function used to get a content string from a
        Beautiful Soup object and a selector. Returns an empty
        string if no object is found for the given selector
        """
        selectedElems = pageObj.select(selector)
        if selectedElems is not None and len(selectedElems) > 0:
            return '\n'.join(
            [elem.get_text() for elem in selectedElems])
        return ''

    def parse(self, site, url):
        """
        Extract content from a given page URL
        """
        bs = self.getPage(url)
        if bs is not None:
            title = self.safeGet(bs, site.titleTag)
            body = self.safeGet(bs, site.bodyTag)
            if title != '' and body != '':
```

```
crawler = Crawler()

siteData = [
    ['O\'Reilly Media', 'http://oreilly.com',
    'h1', 'section#product-description'],
    ['Reuters', 'http://reuters.com', 'h1',
    'div.StandardArticleBody_body_1gnLA'],
    ['Brookings', 'http://www.brookings.edu',
    'h1', 'div.post-body'],
    ['New York Times', 'http://nytimes.com',
    'h1', 'p.story-content']
]
websites = []
for row in siteData:
    websites.append(Website(row[0], row[1], row[2], row[3]))

crawler.parse(websites[0], 'http://shop.oreilly.com/product/'\
    '0636920028154.do')
crawler.parse(websites[1], 'http://www.reuters.com/article/'\
    'us-usa-epa-pruitt-idUSKBN19W2D0')
crawler.parse(websites[2], 'https://www.brookings.edu/blog/'\
    'techtank/2016/03/01/idea-to-retire-old-methods-of-policy-education/')
crawler.parse(websites[3], 'https://www.nytimes.com/2018/01/'\
    '28/business/energy-environment/oil-boom.html')
```

- Planning and Dealning Objects
- Dealing with Different Website Layouts
- Structuring Crawlers
  - Crawling Sites Through Search
  - Crawling Sites Through Links
  - Crawling Multiple Page Types
- Thinking About Web Crawler Models

Thank you for listening!