32. Web Scraping

Extracting information from web pages, *web scraping*, can be achieved in many ways using different Python libraries. A program that crawls over the web from link to link extracting information is called a *web crawler*. This chapter looks at two libraries for web scraping:

- urllib - a Python library for handling URLs. The urllib web site provides full documentation.
- Beautiful Soup - a library for extracting data from web sites. The latest version is Version 4

32.1 HTML

Most web pages use HTML (Hyper Text Markup Language) to form the structure of the web page. The following shows the structure of a simple web page.

```
<!DOCTYPE html>
<html>
<body>
  <h1>A level 1 heading</h1>
    <p>A paragraph</p>
  <h2>A smaller heading</h2>
    <p> another paragraph</p>
  <ol>
    <li> the first item </li>
    <li> the second item </li>
  </ol>
</body>
</html>
```

If you copy and paste the code above into a simple text editor, and save it with extension html, the file can be displayed as a web page by opening it with a web browser instead of a text editor.

Notice that the HTML tags are often in pairs, with a start tag < and and end tag </. The tags are not case sensitive but lowercase tags are typically used for html. A tag that starts with <! is a comment. The first line of code above tells web browsers that this is an html document, and is required. All of the html for the page must be within the opening `<html>` and closing `</html>` tags. The part of the page that is visible in a web browser is contained in the body tags.

Heading tags can be from h1 to h6, with each higher number giving a smaller heading. White space in html is ignored and just serves to make the code more readable. Paragraph tags, p, are used for text blocks. Ordered or unordered lists are made with `ol` and `ul` tags, with each item surrounded by `li` tags. Indents can make a long list more readable for coding. Indents do not affect how the lists are displayed.

32.1.1 Head

The body tags are necessary for a web page to display something interesting, but the `<head>` tags are not required. The most common use for head tags is to create a title that will be displayed in a browser tab.

```
<head>
  <title>An Interesting Title</title>
</head>
```

32.1.2 CSS

There are formatting attributes that can be used within tags to adjust colors, fonts, and more. However, the recommended approach is to separate content from formatting. HTML tags are used for the structure and content of web pages. Cascading style sheets, CSS, are used to control the appearance of html elements.

The CSS code is usually stored in another file and imported within the head tags. The main advantage of using CSS is that the pages of a web site can have a uniform look, without having to explicitly code each element in each page. Each element type is formatted once in the style sheet and the defined styles will apply to all pages that import the css.

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

Since CSS is more about web design than content, it is not relevant to web scraping and so CSS won't be described here. Many excellent tutorials for CSS and web design exist on the web.

32.1.3 Image tags

An image tag provides all the information to display the image within `<img ...>`, so there is no need for a closing tag. In the example below, the src argument gives the file source. Optional width and height arguments specify the size of the image. The optional alt argument gives text that will display if the image cannot be found, and also is used by html readers for the visually impaired.

```
<img src="myfolder/myimage.png" width="500" height="400"
                  alt="Description here">
```

32.1.4 Links

Links to other pages are enclosed in <a> tags. The link text is what is displayed to the user with formatting that indicates it is a hyperlink. Notice the forward slash at the end of the html reference address. Some web browsers will automatically add this but some will not which causes two requests to the server. To make a hyperlink open in a new page, add the target attribute to the tag: `<a href="..." target="_blank">`.

```
<a href="https://www.somepage.com/index.html/" > link text </a>
```

32.1.5 Tables

Tables can be created with table tags:

- table tags define the start and end of the table
- tr tags define table rows
- td tags define data in the row, essentially creating columns
- th tags define a heading row which will be bold

The table created with the simple table tags below displays as follows. Notice it is not formatted nicely. There are deprecated formatting attributes that could be used to control the display of the table elements. Instead, cascading style sheets, CSS, should be used.

ID Name GPA

101 Sally Smith 3.8

102 Mark Jones 3.24

```
<table>
  <tr>
    <th>ID</th>
    <th>Name</th>
    <th>GPA</th>
  </tr>
  <tr>
    <td>101</td>
    <td>Sally Smith</td>
    <td>3.8</td>
  </tr>
  <tr>
    <td>102</td>
    <td>Mark Jones</td>
    <td>3.24</td>
  </tr>
</table>
```

32.1.6 Learning More HTML

For most web pages, you can right-click on the page and choose View Page Source from the pop-up menu. You will see the basic HTML tags discussed above. Other tags you will see are script tags for javascript, and meta tags for keywords and other information used by web crawlers. Another common tag is div, which is just used to group elements together.

This is enough HTML background to start web scraping. There are many excellent free resources on the web for learning more about HTML.

32.2 How Web Sites Work

A **website** is just a collection of web pages and other files that share a common domain name, like amazon.com or wikipedia.org. A **web server** is a computer that receives requests for a web page from a browser and returns the content of the page. This works through a client-server system. The web browser is the client that makes requests from the web server that then makes a response. Clients and servers communicate through a protocol, HTTP Hypertext Transfer Protocol. Any transmission of information on the internet is done through TCP/IP Transmission Control Protocol and Internet Protocol that packages content for transmission and makes sure it arrives. A server will send web page information in chunks called packets, using TCP/IP protocol. A web browser collects these packets and displays the web page.

The HTTP protocol handles different types of requests and responses between clients and servers, including:

- GET - to request data from a resource
- POST - to create or update a resource
- PUT - similar to POST but will not create multiple resources if multiple PUTs are issued
- DELETE - to delete a resource

When a web browser wants to display a web page, the domain name is translated to an IP address through the DNS domain name system. DNS servers provide lookup of IP addresses for web sites. Each device connected to the Internet has a unique IP address. A common standard for IP addresses is IPv4 which looks like four numbers separated by periods, where each number can be 0 to 255. The following could be an IPv4 address:

```
72.14.255.255
```

IPv4 addresses are 32 bits which allows for about 4 billion unique addresses. With the growth of the Internet, more addresses are needed and so a new protocol IPv6 is being used. IPv6 is 128 bits, written in hex with each group of digits separated by colons. This could be an IPv6 address:

```
2001:0db8:85a3:0000:0000:8a2e:0370:7334
```

32.2.1 Status Codes

An HTTP request will return content if possible but other conditions return a status code. Here are some common status codes that indicate something went wrong with the request:

- 400 Bad Request - the server did not understand your request
- 403 Forbidden - your client does not have access to the content
- 404 Not Found - the URL is not recognized
- 408 Request Timeout - the server shut down a request it felt had gone idle

This section described a very high-level view of communication between web browsers and web servers, with a focus on just information needed to understand web scraping.

32.3 Respecting Web Sites

A common misconception about content on the web is that if it's on the web, it's free content. Legally, the content provider owns the content and has the right to determine how that content can be used by others. When scraping data of a particular web site, make sure to read that site's Terms and Conditions. Many sites do not allow others to use the data for commercial purposes.

Limiting the rate at which you are scraping a web site can help prevent access issues with the web site for all users, as well as prevent you from getting blocked from the site. Scraping during off-peak hours can limit the impact of your crawler.

Before scraping a particular web site, check if they have provided an API. Many popular websites such as Twitter and Wikipedia provide APIs to enable people to scrape data safely.

32.3.1 The robots.txt file

Many web sites have a robots.txt file that specifies how a site should be crawled or indexed for search engines. The file can be found on the root directory, ex: `www.some_website.com/robots.txt` The information in a robots.txt file can be hard to interpret. Here is an example to get started:

```
# Google
User-agent: Googlebot
Disallow: /nogooglebot/

# Everyone else
User-agent: *
Allow: /
```

This simple robots.txt file has a rule for Google's web crawler, specified in the User-agent field. The disallow field says that Google should not crawl in the /nogooglebot/ directory or any subdirectories. The second rule has * for User-agent, meaning everyone else. The Allow field has just the root /, so that means all directories can be crawled. In contrast, `Disallow: /` would mean that you can't access anything.

32.4 Downloading text using the urllib library

The urllib library is part of Python, so there is nothing to install. The urllib library has several modules that work with URLS:

- urllib.request for opening and reading URLs
- urllib.error for handling exceptions raised by URL requests
- urllib.parse to parse URLs
- urllib.robotparser to parse robot.txt files

The urllib library is simple to use, and is useful for downloading text, as shown in the code below. A url is specified as a string. In this case, it is the address of a novel. The url is opened with the urlopen() method, then the page is read. Optionally, the content can be decoded. The page below was decoded with utf-8 encoding.

Code 32.4.1 — Using urllib. Reading text

```
from urllib import request

url = "http://www.gutenberg.org/files/2554/2554-0.txt"

with request.urlopen(url) as f:
    raw = f.read().decode('utf-8-sig')
print('len=', len(raw))
raw[:200]
```

The output from the code is shown below. More output is shown on the Jupyter notebook in the GitHub.

```
len= 1176966
'The Project Gutenberg EBook of Crime and Punishment, by Fyodor Dostoevsky'
```

The urllib modules can be used to download everything from web pages, but when a web page contains html, another library such as Beautiful Soup make the process easier.

32.5 Using the Beautiful Soup library

The Beautiful Soup library makes it easier to extract information from html tags. The `beautifulsoup4` library can be installed with pip/pip3: `pip3 install beautifulsoup4`

The example below extracts text from a web page news article coded in html. The code below first uses urllib to open the web page and read the content. Then a soup object is created. Finally, text is extracted from the soup object.

Code 32.5.1 — Using Beautiful Soup. Extracting text

```
import urllib
from urllib import request
from bs4 import BeautifulSoup

url = 'https://nyti.ms/2uAQS89'
html = request.urlopen(url).read().decode('utf8')
soup = BeautifulSoup(html)

# extract text
text = soup.get_text()
```

The text that was extracted is fairly clean but as shown in the Jupyter notebook online, there is still a bit of web page navigation and other material that is not really the text of the news article. The notebook also shows some text processing in Python to clean up the text.

32.5.1 Extracting paragraph tags

The next code chunk shows how to extract all p tags. These could then be processed further to extract text within the paragraphs. The code uses the same soup object created above, but selects just p tags.

Code 32.5.2 — Using Beautiful Soup. Extracting p tags

```
for p in soup.select('p'):
    print(p)
```

32.5.2 Extracting links

The next code chunk shows how to extract hyperlinks, 'a' tags. Just 10 are printed with this code.

Code 32.5.3
```
counter = 0
for link in soup.find_all('a'):
    counter += 1
    if counter > 10:
        break
    print(link.get('href'))
```

32.6 Alternatives

This chapter showed how to do web scraping with Beautiful Soup. Other popular frameworks for web scraping in Python are Scrapy and Selenium.