Week 6 – URLLIB and BeautifulSoup DEMO

urllib

We have used the urllib.request package before to retrieve a file from an ftp server over the Internet. But we can use it more generally to connect with web servers using the HTTP protocol. For documentation, we look at

https://docs.python.org/3/howto/urllib2.html

The main types of data used by urllib are the Request and Response, mirroring the request/response structure of connecting with web servers. The urllib requests can build GET and POST requests that may include additional information besides the actual URL, which will be encoded after the ? in the URL request.

One standard use of this library is to first use "urlopen" to pass the request to the server, and then use the read() function on the response, which returns the text of the response. We'll try this on a newspaper story.

```
>>> from urllib import request
>>> snapchaturl = "http://www.bbc.com/news/business-39135278"
## or altogether html =
request.urlopen(snapchaturl).read().decode('utf8')
>>> response = request.urlopen(blondurl)
```

```
>>> type(response)
<class 'http.client.HTTPResponse'>
```

If we just read the response, we get a sequence of bytes, so instead we decode it to convert it to a Python string.

```
>>> html = response.read().decode('utf8')
>>> type(html)
<class 'str'>
>>> print(html[:500])
<!DOCTYPE html>
<html lang="en" id="responsive-news">
<head  prefix="og: http://ogp.me/ns#">
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1">
    <title>Snapchat owner shares priced at $17 despite losses - BBC News</title>
    <meta name="description" content="The firm behind the Snapchat raises $3.4bn in one of the biggest tech share sales in years.">

    <link rel="dns-prefetch" href="https://ssl.bbc.co.uk/">
    <link rel="dns-prefetch" href="http
```

BeautifulSoup

The Python package BeautifulSoup will parse the HTML document to find its structure, fixing any non-well-formed tag structure as best it can.

http://www.crummy.com/software/BeautifulSoup/bs4/doc/

[An alternate package sometimes used is lxml.]

The typical use is to use the BeautifulSoup parser on an HTML document to get a soup object:

```
>>> from bs4 import BeautifulSoup
>>> htmlsoup = BeautifulSoup(html, 'html.parser')
>>> type(htmlsoup)
<class 'bs4.BeautifulSoup'>
```

A BeautifulSoup object corresponds to a tag. Every tag has a name, (possibly) some attributes, a value, and (possibly) some children. When you have a tag, you can get information about that as a node in the tree structure generated by the tags of the document. You can then navigate anything below that tag in the tree, either by looking for specific tags or by following the children.

So every tag and tag structure can be found from the object that we obtained from the parser. If you give the name of a tag, it will find the first instance of that tag.

```
>>> firsttitle = htmlsoup.title
>>> firsttitle
```

<title>Snapchat owner shares priced at $17 despite losses - BBC News</title>

>>> type(firsttitle)

<class 'bs4.element.Tag'>

>>> firsttitle.name

'title'

Use the get_text() function to get the actual tag content:

>>> firsttitle.get_text()

'Snapchat owner shares priced at $17 despite losses - BBC News'

We can also use the find_all() function to return a ResultSet of all the instances of that tag:

>>> anchors = htmlsoup.find_all('a')

>>> type(anchors)

<class 'bs4.element.ResultSet'>

>>> len(anchors)

73

>>> print(anchors[:8])

[, Skip to content, Accessibility Help, <a

href="https://www.bbc.com/account?ptrt=http%3A%2F%2Fwww.bbc.com%2Fnews%2Fbusiness-39135278" id="idcta-link"> BBC iD ,

Notifications

, News, News, Sport]

For each anchor, ‘a’ tag, we use the ‘href’ attribute to get the actual anchor part, where I converted the result to an actual string:

>>> links = [str(link.get('href')) for link in htmlsoup.find_all('a')]

>>> print(links[:4])

['/', '#page', '/accessibility/', 'https://www.bbc.com/account?ptrt=http%3A%2F%2Fwww.bbc.com%2Fnews%2Fbusiness-39135278']

Or we can look at the anchor strings to just get the ones starting with http:

>>> outlinks = [link for link in links if link.startswith('http')]

>>> len(outlinks)

44

>>> print(outlinks[:4])

['https://www.bbc.com/account?ptrt=http%3A%2F%2Fwww.bbc.com%2Fnews%2Fbusiness-39135278', 'http://www.bbc.co.uk/news/', 'http://www.bbc.com/news/', 'http://shop.bbc.com/']

These are examples of navigating the document using tags. In addition, you can navigate the document using children functions to move in to the tag structure.

Look at the example program to show how to get data from an HTML document. This example uses the ESPN web page for NCAA men's basketball rankings.

```
ncaa_url = "http://www.espn.com/mens-college-basketball/rankings"
```

To figure out how to get the data that you want from an HTML page, you need to inspect the HTML source for the page. For each browser, there are "developer tools" that you can turn on that allow you to see page sources, among other things.

Or you can use the prettify function in BeautifulSoup in order to see what tags to collect. Note that once we get and save a tag, then we can access all its children tags and attributes directly from it.