EECS 182: Web Module #2: Opening URLs from within Python

Atul Prakash

With Python, it is easy to open a URL instead of a file. Here is how to open
import urllib2
f = urllib2.urlopen(<urlname>)
This creates a file handle, similar to that from opening a file. You can now read things from that URL.
Here is an example program that reads and displays the contents from my profile page on EECS web server:

```
import urllib2
f = urllib2.urlopen("https://www.eecs.umich.edu/eecs/etc/fac/facsearchform.cgi?aprakash+")
contents = f.read()
print contents
```

A web browser renders the page as follows:

If you examine the HTML source of the above page via the Python program or via your browser, there is a lot of stuff in there to produce the top bar and the left side bar. A small part of the entire source that renders the information about me is as follows:

```
<h4 class="red">Faculty Profiles</h4>

<table border="0" cellspacing="1" width="100%" cellpadding="0">
<tr><td width="130" valign="top"><img src="/eecs/images/people/photos/aprakash.jpg"
width="130" border="0"></td>
<td width="420" valign="top">
<a href="http://www.eecs.umich.edu/~aprakash/"><b>Atul  Prakash</b></a><br>
<span class="profile_text">
<b>Division:</b> CSE  <br>
<b>Address:</b> 4741 CSE <br>
<b>Phone:</b> (734) 763-1585<br>
<b>Fax:</b> (734) 763-8094<br>
<b>Email:</b> <img border="0" src="/eecs/images/people/pngs/aprakash.png"
align="absbottom"><br>
<b>Degree:</b> Ph.D. UC-Berkeley<br>
<b>Title:</b> Professor<br>
<b>Research Interests:</b> security policy management, software infrastructure to support
collaborative work, privacy in pervasive computing,  intrusion detection, group security,
```

```
operating system  security, scientific collaboratories.  <br>
<b>Research Areas:</b> Software Systems.<br>
<b>Areas of Specialty:</b> Networking and Distributed Systems; Security.<br>
</span></td></tr>
<tr><td bgcolor="#FFFFFF" height="14" colspan="2"> </td></tr>
<tr><td bgcolor="#F9F6F6" height="2" colspan="2"></td></tr>
<tr><td bgcolor="#FFFFFF" height="14" colspan="2"> </td></tr>
</table>
<table border="0" cellspacing="0" width="100%" cellpadding="0">
<tr><td width="100%" height="35" valign="bottom">
<a href="/eecs/faculty/faculty.html">EECS Faculty Page</a></td></tr></table>

</td></tr>
</table>
```

You don't have to understand all the HTML at this point. But, see if you can figure out why words like "Email", "Phone", etc., are rendered in Bold font and why my Email ID and photo is rendered as an image. To check your understanding, answer the following questions:

- What HTML tags are used to make some text bold?
- What HTML tag is used to break to a new line?
- How do you include an image in a web page?
- How do you embed a clickable URL in the page? An example is my name "Atul Prakash".
- Modify the above Python program to count and print the number of characters and words in the retrieved HTML file. Treat the HTML file as a plain-text file for this purpose (i.e., don't worry about eliminating all the HTML tags).

Feel free to go back to the HTML tutorial at http://www.w3schools.org to confirm your intuition.

Setting User-Agent

Some sites will reject requests that come from your script. The reason is that each browser sends a value called “User-Agent” as part of the web request that specifies the name of the browser and the platform from which the request is made. If the server does not like the browser or the platform, it is free to reject the request. Web servers need the User-Agent value because they may wish to send different content to different platforms or different browsers (e.g., Safari on iPhone should receive different content than Safari on Mac OS because of the form factor of the device).

If you find that your request is being rejected, you can specify a different User-Agent. Below is an example of how to do that with urllib2module:

```
headers = { 'User-Agent' : 'Mozilla/4.0 (compatible; MSIE 5.5; Windows NT)'}
req = urllib2.Request('www.example.com', None, headers)
```

htmlcontents = urllib2.urlopen(req).read()

The above says that your Python script is compatible with Mozilla/4.0-compatible browser on Windows platform, as far as handling of the received content is concerned. Mozilla was the first popular web browser. So, many browsers today, including Chrome, use Mozilla in their User-Agent settings. You can check what the User-Agent value of your browser is by visiting the following URL:

http://whatsmyuseragent.com/

If nothing else works, you can try setting the value to what the above site reports for a working browser.

(Obviously, you should avoid sending too many requests to a web site from your script. Otherwise, you could end-up getting blacklisted and the web site could stop serving you content.)

Cookies

When you login or connect to a web server for the first time, a web server may send something called a “cookie” back to your web browser. The cookie is just a string that identifies you in case you make subsequent requests to the same web server; it is sent with all subsequent requests by your browser to the web server. Cookies permit you to avoid having to type in your passwords every time you visit a site.

Occasionally, you may run into a situation when fetching a page that a previously-issued cookie is required in order to read a page. Here is how you can have Python remember any cookies that are received:

```
import urllib2
import urllib
import cookielib
cj = cookielib.CookieJar()

# How to specify a cookie jar when opening URLs. Any received
# cookies go into the cookie jar.
opener = urllib2.build_opener(urllib2.HTTPCookieProcessor(cj))
urlhandle = opener.open('http://www.yahoo.com')
contents = urlhandle.read()

print cj


# Now send the cookie from the jar in a subsequent request
# Any cookie in the cookie jar is
# automatically sent with the request for the same server.
```

```
sendopener = urllib2.build_opener()
sendopener.addheaders_append(urllib2.HTTPCookieProcess(cj))
r = opener.open("http://www.yahoo.com")
```