12.6: Parsing HTML using regular expressions

One simple way to parse HTML is to use regular expressions to repeatedly search for and extract substrings that match a particular pattern.

Here is a simple web page:

```html
<h1>The First Page</h1>
<p>
If you like, you can switch to the
</p>
```

We can construct a well-formed regular expression to match and extract the link values from the above text as follows:

```
href="http://.+?"
```

Our regular expression looks for strings that start with "href="http:/", followed by one or more characters (".+?"), followed by another double quote. The question mark added to the ".+?" indicates that the match is to be done in a "non-greedy" fashion instead of a "greedy" fashion. A non-greedy match tries to find the *smallest* possible matching string and a greedy match tries to find the *largest* possible matching string.

We add parentheses to our regular expression to indicate which part of our matched string we would like to extract, and produce the following program:

Code \{|PageIndex\{1\}|} (Python):

https://eng.libretexts.org/Bookshelves/Computer_Science/Book%3A_Python_for_Everybody_(Severance)/12\%3A_Networked_P
Updated: Sun, 28 Jul 2019 00:04:12 GMT
Powered by
# Search for lines that start with From and have an at sign
import urllib.request, urllib.parse, urllib.error
import re
url = input('Enter - ')
html = urllib.request.urlopen(url).read()
links = re.findall(b'href="(http://.*?)"', html)
for link in links:
    print(link.decode())

# Code: http://www.py4e.com/code3/urlregex.py

The `findall` regular expression method will give us a list of all of the strings that match our regular expression, returning only the link text between the double quotes.

When we run the program, we get the following output:

```
python urlregex.py
Enter - http://www.dr-chuck.com/page1.htm
http://www.dr-chuck.com/page2.htm

python urlregex.py
Enter - http://allendowney.com/

http://gmpg.org/xfn/11
http://www.allendowney.com/wp/xmlrpc.php
http://www.allendowney.com/wp/feed/
http://www.allendowney.com/wp/comments/feed/
http://www.allendowney.com/wp/wp-includes/wlwmmanifest.xml
http://www.allendowney.com/wp/
http://www.allendowney.com/wp/
http://www.allendowney.com/wp/books/
http://www.allendowney.com/wp/classes/
http://www.allendowney.com/wp/projects/
http://www.allendowney.com/wp/
http://www.olin.edu
http://allendowney.com/downeyCV.pdf
http://allendowney.blogspot.com/
http://www.oreilly.com/
http://greenteapress.com/
http://allendowney.com/research
http://www.allendowney.com/wp/bayesian-statistics-for-undergrads/
```
Regular expressions work very nicely when your HTML is well formatted and predictable. But since there are a lot of "broken" HTML pages out there, a solution only using regular expressions might either miss some valid links or end up with bad data.

This can be solved by using a robust HTML parsing library.