Web Crawler Project for CS 111 Law Fall 2019

Release version 1.1 (some details might change a little)
Due: *11:59 pm*, Monday, November 18, 2019.

(Correcting error in earlier version that said "Monday" but had wrong date, a date that wasn't even a Monday)

Themes: Working with lists, connecting to the internet from Python, crawling and scraping the web

Web crawler

It's finally time to put all the pieces together and build our web crawler. And the web crawler should do something, so we're going to have you harvest email addresses.

BE SURE TO TURN IN SOMETHING THAT WORKS! If your program fails with an error, it becomes very hard for the TA doing the grader to give you any points at all! Turn in something that works, even if it doesn't do EVERYTHING, and you will get partial credit. If your program stops with an error without getting anything done, it's hard for us to give you much partial credit.

Assignment: Online Email Harvester

For this assignment, you are to write a Python program that will search the web for email addresses.

You should write a function called `crawl(start, limit)` that will return a list of all the *unique* email addresses found on all the webpages reachable from the webpage whose URL is `start`, except that you should stop if you have visited `limit` distinct web pages and return the list from that many pages.

Types: `start` is a string. `limit` is an integer. You will return a list of strings (one string for each email address you find).

You should not test your program on the open "real" full web. You can use https://www.cs.uic.edu/~sloan/CS111Law/crawlerstart.html as a starting point, and you can also use pages reachable from that page.

Below you will find some code that you can use in your programs that gets the email addresses found inside a given page of URL.

Remember to make your function work, and to use the email address finder that you need to add the following

two lines to the start of your file:

```
import urllib.request as ur
import re
```

Your program may not import any other modules that we have not *explicitly* discussed in a lecture. You should not actually have any reason to import any module other than urllib.request and re.

In order to run the code below, you need to be connected to the internet...

Also make sure you **start early!!**

Code to find email addresses in content

(Feel free to use the code below, or write your own...)

```
# This function takes a webpage's content (i.e., some text) as input.
# It then searches through the text, grabbing all the email addresses,
# and storing them in a list called strings that is returned to the caller.

import re

def get_addresses(content):
    '''Input can be either str type or bytes type from a urllib read'''

    #re package wants str type, not bytes
    strings = re.findall('[a-zA-Z0-9_.]*[@][a-zA-Z0-9-]+\.[a-zA-Z0-9-.]+',
                         str(content))

   #Some addresses have a trailing period. Need to get rid of it...
   for x in range(0, len(strings)):
     if strings[x].endswith("."):
       strings[x] = strings[x][0:len(strings[x])-1]

   return strings
```

Note: If you test this function on the content of Professor Sloan's homepage, https://www.cs.uic.edu/Sloan you will notice (as of this writing) two things:

1. The function returns a list that includes *two* copies of the same webmaster address, and
2. The function does not give back Prof. Sloan's own email address, which is listed as my-last-name at host: uic.edu because Prof. Sloan doesn't like getting the email usually sent to addresses found by automated

email harvesters.

(You have my permission to connect to the page. You do *not* have my permission to crawl onward from that page.)

Advice

Classic CS advice says: "Design top down; implement bottom up."

For this project, that would translate to using the hierarchical decomposition design for this crawler and harvester that we did earlier in the semester for the design.

For the implementation, a logical progression might be:

1. Get your old function to find the first link on a page to work with taking a URL to open as its input.
2. Use that function, or a minor modification of it, as a subroutine in a function that takes a URL as input and returns *a list of all* the URLs in the text of that page.
3. Next, write a program that just crawls, and perhaps returns the list of all the URLs of all the pages it visited.
4. Once you are sure the above works, then add in the harvesting of email addresses.

Submission

Coming soon.