Introduction to Web Science 101: Assignment #1

Dr. Nelson

Alexander Nwala

Thursday, September 11, 2014

Contents

Problem 1

Demonstrate that you know how to use "curl" well enough to correctly POST data to a form. Show that the HTML response that is returned is "correct". That is, the server should take the arguments you POSTed and build a response accordingly. Save the HTML response to a file and then view that file in a browser and take a screen shot.

Listing 1: Curl Demo

```
#curl demonstration
import commands

co = 'curl  -i --data "message=L\'enfant qui voulait etre un ours"  http://www.cs.odu.
    edu/~anwala/files/CS895Assignment1/problem1_CS895Assignment1form.php >
    problem1_curlDemonstrationOutput.html'
data = commands.getoutput(co)

print data
```

Listing 2: Web App

```
<!DOCTYPE html>
<html>
<body>

<?php
      echo("<br />\n");
      echo("<br />\n");
     echo("<b>Message Posted: </b>" . $_POST['message'] . "<br />\n");
?>

</body>
</html>
```

The curl command supports submitting information to the server through a POST method by means of the "–data" option followed by attribute values pairs separated by the ampersand symbol. In Listing 1, the message (L'enfant qui voulait etre un ours) submitted to the server is subsequently printed by the web application (Listing 2). The response of the server prepends the message sent to the server (Figure 1).

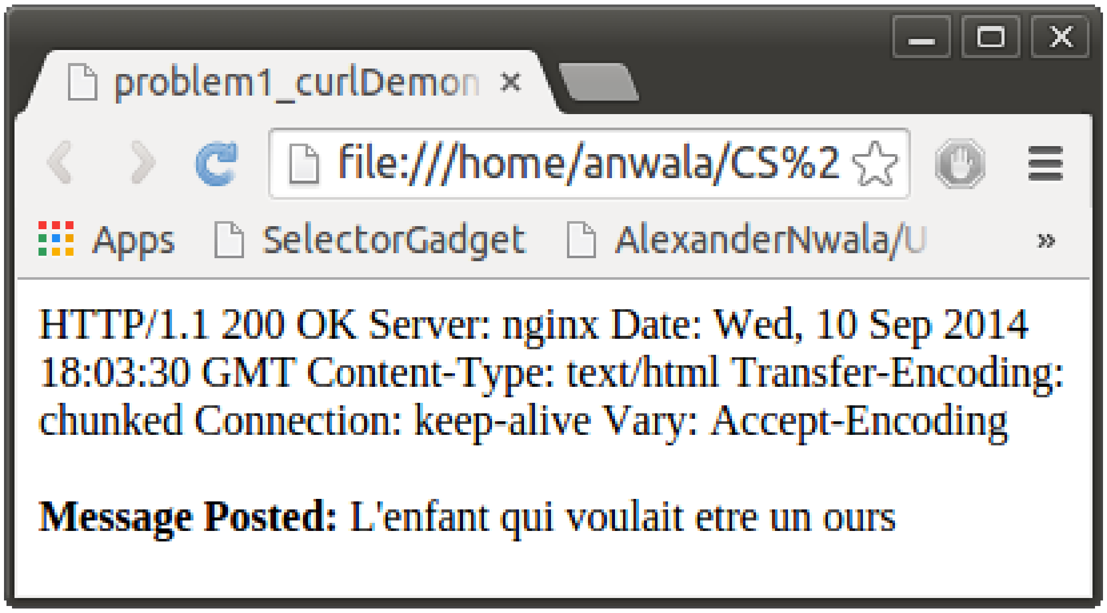
Problem 2

Write a Python program that:

1. takes one argument, like "Old Dominion" or "Virginia Tech"
2. takes another argument specified in seconds (e.g., "60" for one minute).
3. takes a URI as a third argument: `http://sports.yahoo.com/college-football/scoreboard/` or `http://sports.yahoo.com/college-football/scoreboard/?week=2&conf=all` or `http://sports.yahoo.com/college-football/scoreboard/?week=1&conf=72` etc.
4. dereferences the URI, finds the game corresponding to the team argument, prints out the current score (e.g., "Old Dominion 27, East Carolina 17"), sleeps for the specified seconds, and then repeats (until control-C is hit).

Problem 2 continued on next page. . .

Figure 1: curlDemoOutput



HTTP/1.1 200 OK Server: nginx Date: Wed, 10 Sep 2014 18:03:30 GMT Content-Type: text/html Transfer-Encoding: chunked Connection: keep-alive Vary: Accept-Encoding

Message Posted: L'enfant qui voulait etre un ours

Listing 3: Format of Html Containing Scores

```
<tr class="game  link" ...>

  <td class="summary">...</td>
  <td class="away">
    ...
    <em>CollegeName</em>
    ...
  </td>

  <td class="score">
  <span class...>awayTeamScore</span>
  " - "
  <span class...>homeTeamScore</span>
  </td>

  <td class="home">
    ...
    <em>CollegeName</em>
    ...
  </td>
  <td class="details">...</td>

</tr>
```

With the aid of urllib2, the url with the score information is dereferenced into an html text document representation. The main challenge is tied to parsing the html document (with structure summarized in Listing 3) to retrieve the scores. Fortunately, Beautiful Soup makes this very easy as outlined by Listing 4. Listing 4 parses the html document from parent table to children rows, down the hierarchy to extract the scores. Figure 2 is a screenshot for the following input: “Old Dominion 5 http://sports.yahoo.com/college-football/scoreboard/?week=2&conf=all”

Listing 4: Get Score from Html Document

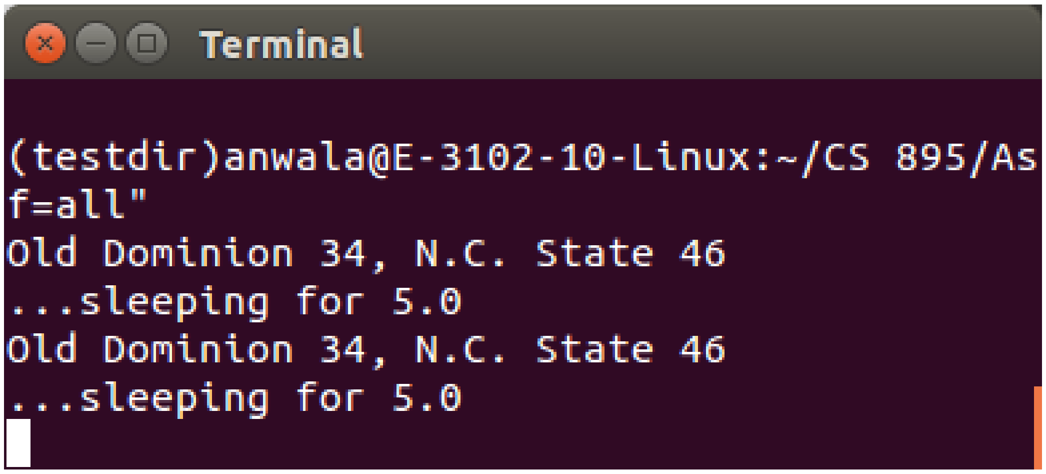
```
#get table containg scores, this table has class value "list"
parentScoresTables = soup('table', {"class": "list"})
     for table in parentScoresTables:
          #get array containing all games
          gameLinks = table.findAll('tr', {"class": "game  link"}) #for live games,
              use gameLinks = table.findAll('tr', {"class": "game live link"})
          for game in gameLinks:
               #get away team name
               awayTeam = game.findAll('td', {"class": "away"})[0].em.string
               awayTeam = str(awayTeam)
               awayTeam = awayTeam.strip()

               #get home team name
               homeTeam = game.findAll('td', {"class": "home"})[0].em.string

               #the span containing scores is indexible, with 0 containing the away
                   team score and 1 - home team
               awayTeamScoreIndexOHomeIndex1 = game.findAll('td', {"class": "score"})
                   [0].findAll('span')
               awayTeamScore = awayTeamScoreIndexOHomeIndex1[0].string
               homeTeamScore = awayTeamScoreIndexOHomeIndex1[1].string

               awayTeamScore = str(awayTeamScore)
               awayTeamScore = awayTeamScore.strip()

               homeTeamScore = str(homeTeamScore)
               homeTeamScore = homeTeamScore.strip()

               #fix for beautiful soup semicolon when ampersand is seen issue:
               #BeautifulSoup parser appends semicolons to naked ampersands, mangling
                   URLs?
               #http://stackoverflow.com/questions/7187744/beautifulsoup-parser-
                   appends-semicolons-to-naked-ampersands-mangling-urls
               if( homeTeam[-1] == ';' ):
                    homeTeam = homeTeam[:-1]

               if( awayTeam[-1] == ';' ):
                    awayTeam = awayTeam[:-1]

               #apply restriction to only print user input college name
               if(homeTeam.lower() == collegeName.lower() or awayTeam.lower() ==
                   collegeName.lower()):
                    print awayTeam, awayTeamScore + ', ' + homeTeam, homeTeamScore
```

Problem 3

Consider the “bow-tie” graph in the Broder et al. paper (fig 9): http://www9.org/w9cdrom/160/160.html. Now consider the following graph:

Problem 3 continued on next page. . .

Figure 2: Program output



```
(testdir)anwala@E-3102-10-Linux:~/CS 895/As
f=all"
Old Dominion 34, N.C. State 46
...sleeping for 5.0
Old Dominion 34, N.C. State 46
...sleeping for 5.0
```

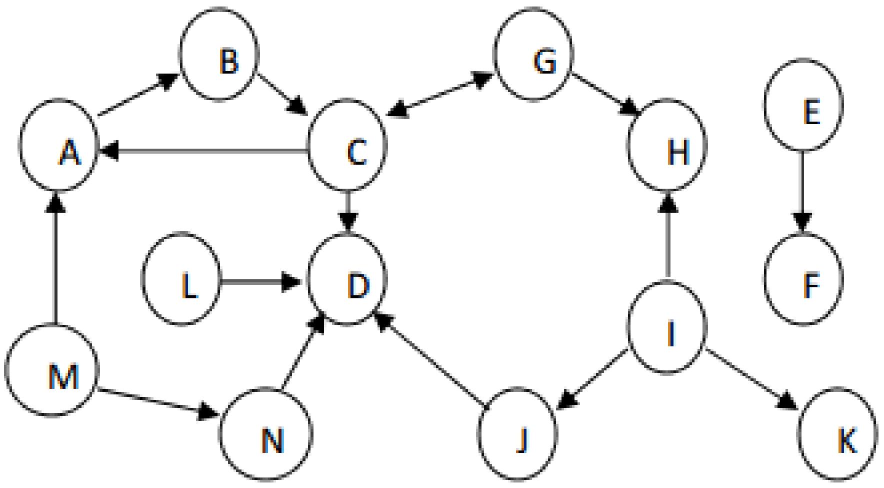
```
A --> B
B --> C
C --> D
C --> A
C --> G
E --> F
G --> C
G --> H
I --> H
I --> J
I --> K
J --> D
L --> D
M --> A
M --> N
N --> D
```

For the above graph, give the values for:

```
IN:            { M }
SCC:           { A, B, C, G }
OUT:           { D, H }
Tendrils:      { L, K, I, J }
Tubes:         { N }
Disconnected:  { E, F }
```

The links in Problem 3 are represented by Figure 3
IN: Pages with no in-links or in-links from IN pages and out-links to pages in IN, SCC, Tendrils, or Tubes. (http://www.harding.edu/fmccown/classes/comp475-s13/web-structure-homework.pdf)
{M}
SCC: All of whose pages can reach one another along directed links. (http://www9.org/w9cdrom/160/160.html)
{A, B, C, G}

Figure 3: Graph of Pages



```
A - B: A -> B
A - C: A -> B -> C
A - G: A -> B -> C -> G

B - A: B -> C -> A
B - C: B -> C
B - G: B -> C -> G

G - A: G -> C -> A
G - B: B -> G -> C -> A -> B
G - C: G -> C
```

OUT: Pages with no out-links or out-links to other pages in OUT, and all in-links come from OUT, SCC, Tendrils, or Tubes. (http://www.harding.edu/fmccown/classes/comp475-s13/web-structure-homework.pdf)
{D, H}

```
D: is linked from SCC through {C},
                  Tube through {N}
                  Tendrils through {L,J}

H: is linked from SCC through {G},
                  Tendrils through {I}
```

TENDRILS: Pages that can only be reached from IN or have only out-links to OUT (http://www.harding.edu/fmccown/classes/comp475-s13/web-structure-homework.pdf)
{L, K, I, J}
TUBES: Pages that have in-links from IN or other pages in Tubes and out-links to pages in Tubes or OUT. (http://www.harding.edu/fmccown/classes/comp475-s13/web-structure-homework.pdf)
{N}

```
N: is linked from IN through {M}, and links out through {D}
```

DISCONNECTED: Pages that have no in-links from any other components and no out-links to other components. These pages may be linked to each other. (http://www.harding.edu/fmccown/classes/comp475-s13/web-structure-homework.pdf)
{E, F}