Introduction to Web Science: Assignment #1

Due on Thursday, January 26, 2017

Dr. Nelson

Udochukwu Nweke

Contents

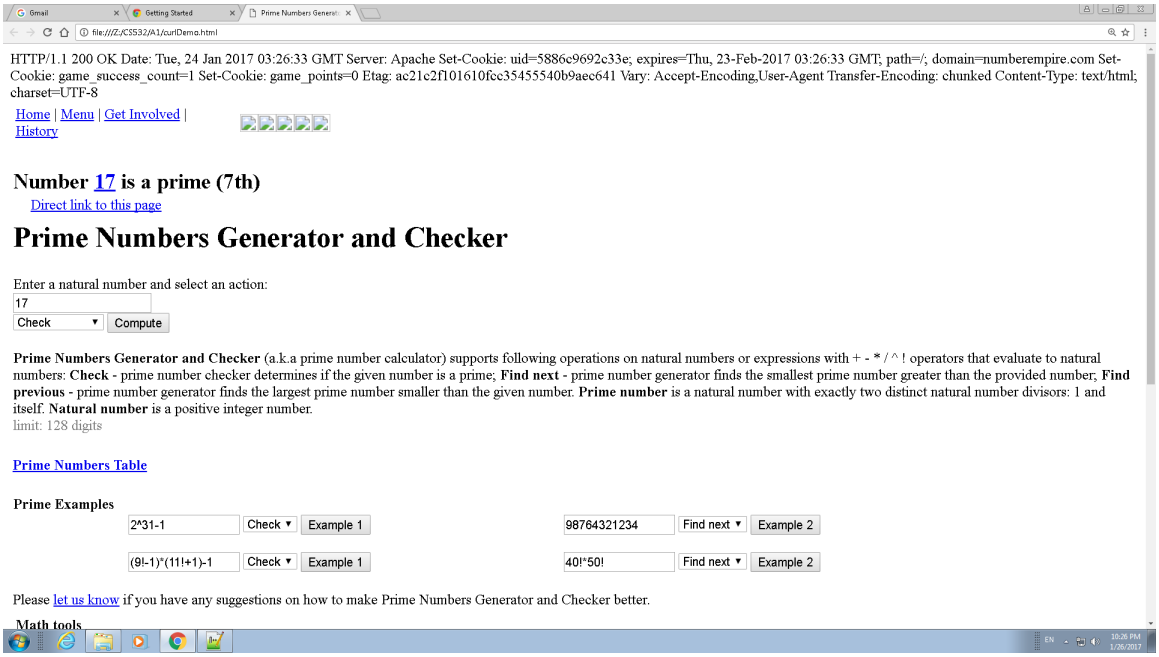

Figure 1: Showing the value 17 correctly received from the server

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: Shows curl Demonstration Script With Highlighting

```
#curl demonstration
import commands
def curlDemonstration():
      co = 'curl  -i --data "number=17" http://www.numberempire.com/primenumbers.php
      > output.html'
      data = commands.getoutput(co)
      print data
```

The Listing 1. shows how I used curl command to submit data to the server through a POST method by using –data option. In listing 1, the number 17 submitted to the server at address: "http://www.numberempire.com /primenumbers.php" is printed by the web application.

Problem 2

Write a Python code that:
1. takes as a command line argument a web page
2. extracts all the links from the page
3. lists all the links that result in PDF files, and prints out the bytes for each of the links. (note: be sure to follow all the redirects until the link terminates with a "200 ok".)

Problem 2 continued on next page. . .

4. Show that the program works on 3 different URIs, one of which needs to be:
http://www.cs.odu.edu/ mln/teaching/cs532-s17/test/pdf/html

Listing 2: Shows curl Demonstration Script With Highlighting

```
import commands
import sys
from bs4 import BeautifulSoup

def getHeadAttr(attr, head):

     head = head.lower()
     indexOfStr = head.find(attr)
     if( indexOfStr != -1 ):

          indexOfNewline = head.find('\n', indexOfStr)
          if( indexOfNewline != -1 ):

               headAttr = head[indexOfStr:indexOfNewline]
               headAttr = headAttr.replace(attr, '').strip()
               return headAttr.lower()

     return -1

def getPDFLinks(links):
     for i in range(0, len(links)):
          head = derefURL_head(links[i])
          if( getHeadAttr('content-type:', head) == 'application/pdf' ):
               print links[i]
               print '\t', i, 'bytesize:', getHeadAttr('content-length:', head)
               print '\ttype: PDF'
               print ''

def getLinks(htmlText):
     soup = BeautifulSoup(htmlText, 'html.parser')
     links = soup.findAll('a')

     hrefs = []
     for i in range(0, len(links)):
          hrefs.append(links[i]['href'])

     return hrefs

def derefURL_head(url):
     co = 'curl  -I -L --silent ' + url
     data = commands.getoutput(co)
     return data

def derefURL(url):
     co = 'curl -L --silent ' + url
     data = commands.getoutput(co)
     return data

webpage = ''
```

```
if len(sys.argv) != 2:
    print 'Invalid arg list'
else:
    #1.
    webpage = sys.argv[1]
    print 'webpage:', webpage
    htmlText = derefURL(url=webpage)
    allLinks = getLinks(htmlText)
    getPDFLinks(allLinks)
```

I extracted the command line arguments with the python sys.argv array. Next, a the function called *derefURL* extracts the HTML content using curl with the -L option to follow redirects. I used BeautifulSoup to parse the HTML content and extract links with function *getLinks*. I made a HEAD HTML request using curl -I option to process the extracted URLs in other to get the PDF files and bytesize (function *getPDFLinks and getHeadAttr*). The following below is a sample output from "http://www.cs.odu.edu/ mweigle/Main/PubsByYear". I have included all the ouput (pdflinks.txt, pdflink1.txt and pdflinks2.txt) in my submission.

```
webpage: http://www.cs.odu.edu/~mweigle/Main/PubsByYear
http://www.cs.odu.edu/~mweigle/files/CV.pdf
7 bytesize: 93364
type: PDF

http://www.cs.odu.edu/~mweigle/papers/pardue-vis16-2pg-poster.pdf
17 bytesize: 583877
type: PDF

http://www.cs.odu.edu/~mweigle/papers/pardue-vis16-poster.pdf
18 bytesize: 614446
type: PDF

http://www.cs.odu.edu/~mweigle/papers/aturban-tpdl15.pdf
33 bytesize: 622981
type: PDF

http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-stories.pdf
35 bytesize: 1274604
type: PDF

http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-off-topic.pdf
37 bytesize: 4308768
type: PDF
```

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:

```
A --> B
B --> C
```

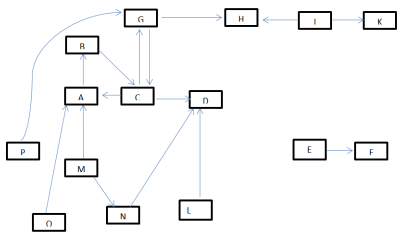

Figure 2: bow-tie graph

```
C --> D
C --> A
C --> G
E --> F
G --> C
G --> H
I --> H
I --> K
L --> D
M --> A
M --> N
N --> D
O --> A
P --> G
```

For the above graph, give the values for:
IN: M, P, O
SCC: A, B, C, G
OUT: D, H
Tendrils: L, I, K
Tubes: N
Disconnected: E, F
SCC: This is the Strongly Connected Component and it is refers to all the links that are connected to one another along a directed link. Hence, from our graph (http://www9.org/w9cdrom/160/160.html).
SCC: A, B, C, D
IN: These are points that can be reached by SCC but cannot be reached from SCC (http://www9.org/w9cdrom/160/160.html)
IN: M, P, O
OUT: These are the pages that can be accessed from the SCC but they do not have any link back to the SCC (http://www9.org/w9cdrom/160/160.html)
OUT: D,H
TENDRIL: These are pages that cannot get to the SCC and they cannot be cannot be reached from SCC as well. (http://www9.org/w9cdrom/160/160.html)
TENDRIL: L, I, K
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)
Tube: N
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.

Problem 3 continued on next page...

(http://www.harding.edu/fmccown/classes/ comp475-s13/web-structure-homework.pdf)
DISCONNECTED E, F