Assignment 1

David Bayard

January 31, 2019

QUESTION 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.

Solution:

The URI: http://www.cs.odu.edu/ anwala/files/temp/namesEcho.php
maintains the ability to mirror the output of a POST method.

1.) Using curl -L, we have the ability to follow all redirections.

2.)Next we add curl -L -X POST which specifies a custom request method of POST

3.)Adding the -d option specifies that data is being sent, and the final result is
curl -L -X POST -d "fname"="David" -d "lname=bayard" https://www.cs.odu.edu/ anwala/files/temp/-namesEcho.php.

This sends a POST method to the server with specialized data fields fname and lname to be submitted to the form.

```
curl -L -X POST -d "fname=David" -d "lname=bayard" https://www.cs.odu.edu/~anwala/files/
    temp/namesEcho.php
```

Listing 1: Curl Command
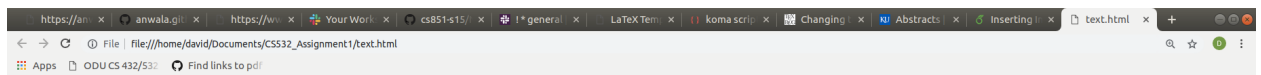
fname Posted: David
lname Posted: bayard

(a) Website Image

QUESTION 2

Write a Python program 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".)
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/pdfs.html

Solution:

1.) Was accomplished by including the sys library and registering user input from sys.argv[1].
2.) To extract all of the links from the page, the Requests library was used.

```
try:
    response = requests.get(uri, headers = headers, timeout=10)

  except Exception as e:
    print('Error: ', str(e))

  return response
```
Listing 2: Create Response Object

Listed above is an example of a response object created from the Requests library. This code is responsible for sending the server a request and storing the response inside a Response object. We can then dereference the HTMl, parse it, and scan for the href elements within the 'a' tags using the Beautiful Soup library.

```
def parseTags(response, uri):

  #break uri into seperate components
  scheme, netloc, path, params, query, fragment = urlparse(uri)
  mainPage = scheme + "://" + netloc

  #dereference the response object
  redditHtml = response.text

  listLinks = []

  #Parse the document and retrieve all <a> tags
  soup = BeautifulSoup(redditHtml, "html.parser")

  for links in soup.find_all('a'):

    #Retrieve the link from the a tag
    x = links.get('href')
```
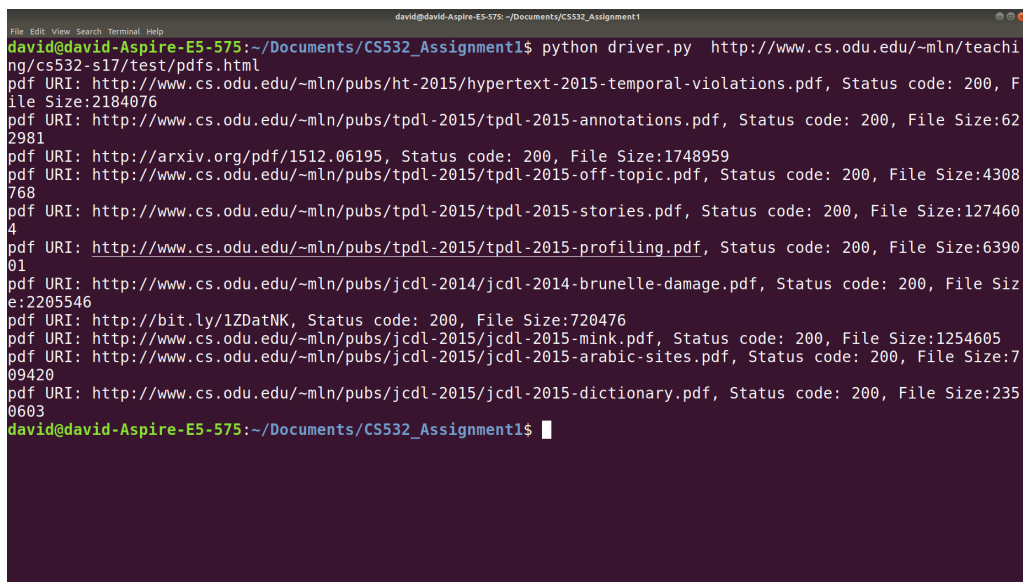Listing 3: Dereference and scan for tags

3.)In order to list all of the links that result in PDF files, it is necessary to filter the list of links. This is done creating a new list of links and adding only the links that have the correct "Content-Type" header value.

This value should match the "application/pdf" value, and can be retrieved from the returned Response object. Accomplishing this requires checking the Response instance header, by using the .headers.get('content-type') method on the instance of Response. This produces a list of links to pdf files.

The rest can be easily accomplished by making a get request with the Requests library for each link. This will result in a response that automatically follows redirections and provides the file size and status code.

```python
#loop through links and check if they are .pdf files
for link in links:
    try:
        responseObj = makeRequest(link)

        #check the response header for the correct content-type
        if("application/pdf" in (responseObj.headers.get('content-type'))):
            listPdfs.append(link);

    #general exception
    except Exception as e:
        print('error', str(e))

if(not listPdfs):
    print('No pdf files')

else:

    for pdf in listPdfs:
        res = makeRequest(pdf)
        print(("pdf URI: {0}, Status code: {1}, File Size:{2}").format(pdf, res.status_code,
            res.headers['content-length']))
```

Listing 4: Filtering pdf links

(a) Sending a request

Consider the "bow-tie" graph in the Broder et al. paper:
http://snap.stanford.edu/class/cs224w-readings/broder00bowtie.pdf give the
values for:

A –> B
B –> C
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

IN: P,O,M
SCC: A,B,C,G
OUT: H,K,D
Tendrils: I,L
Tubes: N
Disconnected:E,F