Assignment 1

Introduction to Web Science
Dr. Michael Nelson
Fall 2014

Sybil Melton

September 11, 2014

Contents

List of Figures

Listings

1 Question

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.

1.1 The Answer

I work with networks, so I found a subnetting web page that use that uses POST. The correct syntax was found by looking at the cURL Tutorial at http://curl.haxx.se/docs/httpscripting.html#POST. The command execution is in Listing 1.

```
curl --data "majornet=192.168.1.0/24&size_1=32&size_2=64&netnum=2&subm=Submit" http://vlsm-calc.net/ | tee vlsm.html
```

Listing 1: cURL command used

As Figure 1 shows, a correct response was built by the server and saved to my local file, vlsm.html.

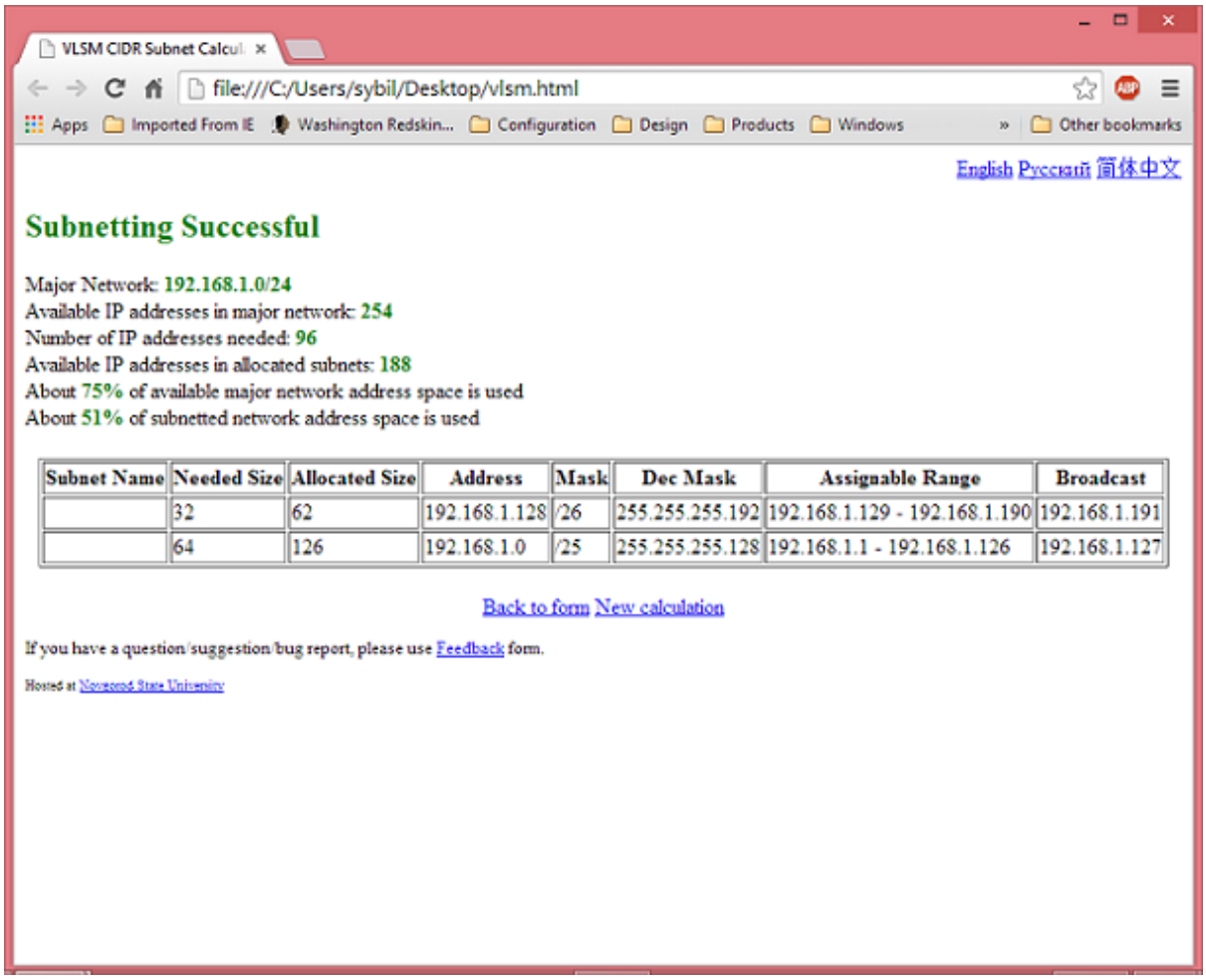

Figure 1: Screenshot of HTML response opened in Chrome

2 Question

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).

2.1 Answer

2.1.1 Argument Parsing

The first three requirements of the question involve the program handing input arguments given to it. I found" argparse" in the Python documentation online, https://docs.python.org/2/howto/argparse.html. I liked that description and usage help messages are available inheritently. It enabled me to use names that made sense for the variables. Listing 2 is the implementation of argparse used.

```
#https://docs.python.org/2/howto/argparse.html
parser = argparse.ArgumentParser(description='Get scores for a team')
parser.add_argument('team', help='team to search for')
parser.add_argument('sleepTime', type=int, help='integer, in seconds to sleep')
parser.add_argument('uri', help='URI to get scores from')
args = parser.parse_args()
```

Listing 2: Argument Parsing

2.1.2 URL Parsing

The next requirement involved dereferencing the URI, which I accomplished by find urlparse in the Python documentation, https://docs.python.org/2/library/urlparse.html. This functionality splits up the inputl into scheme, netloc, path, and query (if available.) The URL was put back together and if the query was input, it was encoded properly with urllib.urlencode, as stated in the power point slides from lecture. Listing 3 is the implementation.

```
  parsed = urlparse.urlsplit(urlp) # https://docs.python.org/2/library/urlparse.html
  query_arg = urlparse.parse_qsl(parsed.query)
  urli = parsed.scheme+"://"+parsed.netloc+parsed.path
  data = urllib.urlencode(query_arg)
  full_url = urli+"?"+data
```

Listing 3: URL Parsing

2.1.3 Web Scraping

In order to find the game for the input team argument, BeautifulSoup was used. My first attempt at an implementation was with HTMLParser, but there were difficulties with the team names that had a '&' character and teams with rankings. Then I started to read about BeautifulSoup, which has built-in functionality to handle this text. I found an example at the link provided in the class slides, http://www.pythonforbeginners.com/python-on-the-web/web-scraping-with-beautifulsoup/. Further research led me to two forums,http://stackoverflow.com/questions/2957013/beautifulsoup-just-get-inside-of-a-tag-no-matter-how-many-enclosing-tags-there and http://stackoverflow.com/questions/16835449/python-beautifulsoup-extract-text-between-element, which helped me formulate a function that could take the tag attribute and value, list name, and the BeautifulSoup object to make the data searchable. All whitespaces and the team rankings were removed to make the teams searchable. Listing 4 is the implementation.

```
#Function
#http://stackoverflow.com/questions/2957013/beautifulsoup-just-get-inside-of-a-tag-no-matter-how-many-enclosing-tags-there
def soupParser(attr, value, qlist, soup):
  for node in soup.findAll(attrs={attr : value}):
    #http://stackoverflow.com/questions/16835449/python-beautifulsoup-extract-text-between-element
    s = ''.join(node.findAll(text=True))
    s = s.strip(' \t\n\r') #remove whitespaces
    if s.startswith("("):
      s = s.strip('\(1234567890\) ') #remove ranking
    if s:
      qlist.append(s)
#####################################################
# How used:
  #http://www.pythonforbeginners.com/python-on-the-web/web-scraping-with-beautifulsoup/
  r = requests.get(full_url)
  #print "Got Status Code: ", r.status_code
  data = r.text
  soup = BeautifulSoup(data)

  #use soupParser Get all the teams and the scores, and store in lists
  soupParser('class', 'away', aways, soup)
  soupParser('class', 'home', homes, soup)
```

Listing 4: BeautifulSoup Implementation

2.1.4 Data Parsing and Output

The last step was to search through the data acquired for the input team. I found an example of finding the index of a list item in a forum at http://stackoverflow.com/questions/176918/finding-the-index-of-an-item-given-a-list-containing-it-in-python. An integer was used to keep track of the index found. If the team wasn't found in the "away" team list, the "home" list was searched. I added the output "Team wasn't found, try your search again" for an error message to help with debugging. Upon completion of a successful search, the index number was used to get the score and the opponent. The output was printed on the screen. Listing 5 is the implementation to find the team.

```
# Function
# find index of the team in the list
#http://stackoverflow.com/questions/176918/finding-the-index-of-an-item-given-a-list-containing-it-in-python
def find_team(value, tlist):
    i = -1
    while True:
        try:
```

```
            i = tlist.index(value, i+1)
        except ValueError:
            break
    return i
##################################################
# How used:
#find the team from input argument
  t = find_team(team,aways)
  away = True;
  if t == -1:
    away = False;
    t = find_team(team,homes)
  if t == -1:
    print "Team wasn't found, try your search again"
  else:
  #Print the score, with the searched for Team first
    if away:
      print aways[t] + " " + aways[t+1] + ", " + homes[t+1] + " " + homes[t]
    else:
      print homes[t] + " " + homes[t-1] + ", " + aways[t-1] + " " + aways[t]
```

Listing 5: Searching Data and Printing Output

The last requirement entailed the program to sleep for the input number of seconds and repeat. This was accomplished with a "while" loop with the condition "True" and time.sleep python functionality. The python documentation at https://docs.python.org/2/library/time.html was used to write this code.

I ran the program successfully with different team names, as shown in Figure 2.

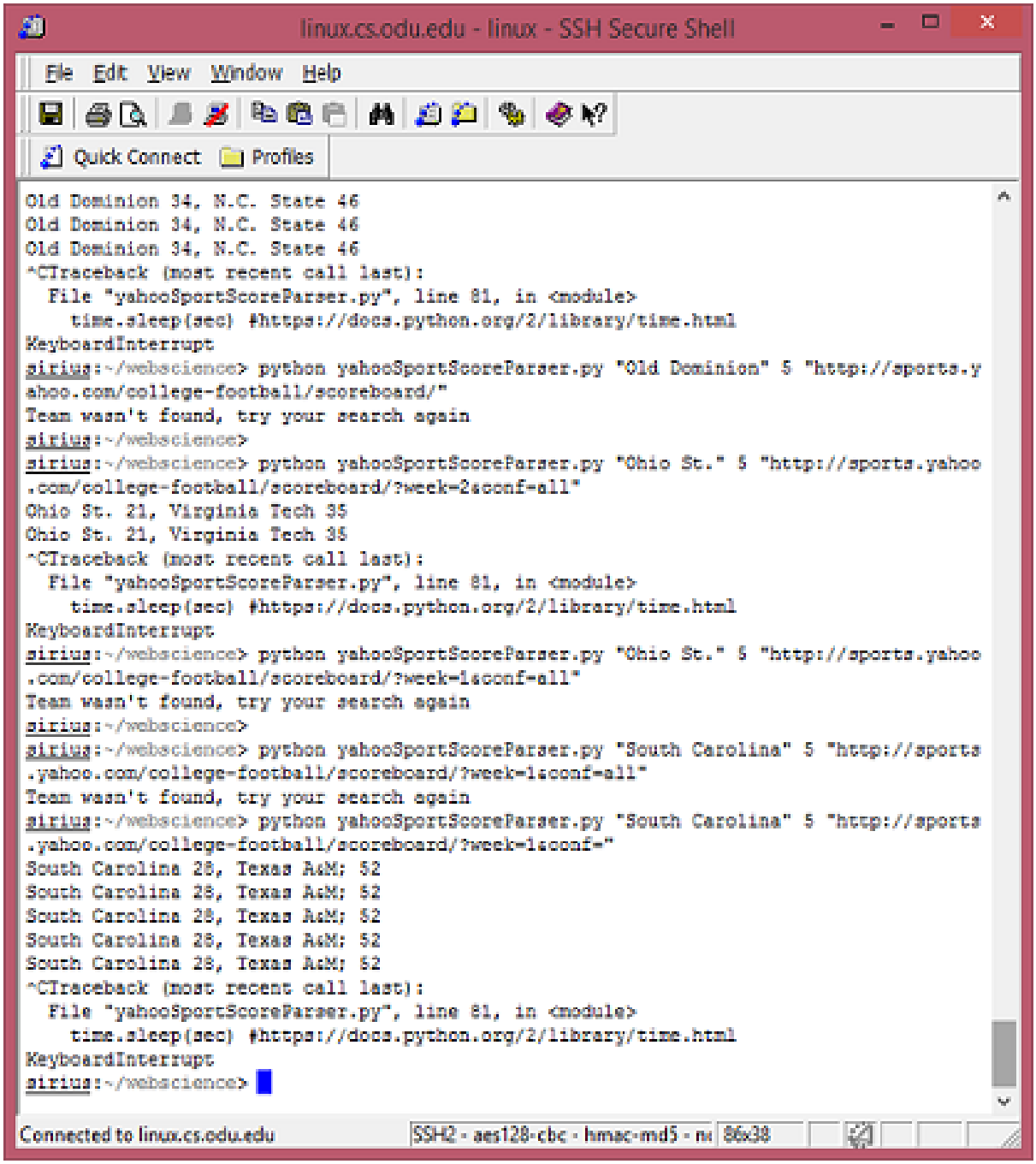

Figure 2: Screenshot of SSH client

2.1.5 Program

The entire program, as written.

```
import urllib2
import re
import requests
import argparse
import urllib
import urlparse
from bs4 import BeautifulSoup
import time

# find index of the team in the list
#http://stackoverflow.com/questions/176918/finding-the-index-of-an-item-given-a-list-containing-it-in-python
def find_team(value, tlist):
    i = -1
    while True:
        try:
            i = tlist.index(value, i+1)
        except ValueError:
            break
    return i

#http://stackoverflow.com/questions/2957013/beautifulsoup-just-get-inside-of-a-tag-no-matter-how-many-enclosing-tags-there
def soupParser(attr, value, qlist, soup):
  for node in soup.findAll(attrs={attr : value}):
    #http://stackoverflow.com/questions/16835449/python-beautifulsoup-extract-text-between-element
    s = ''.join(node.findAll(text=True))
    s = s.strip(' \t\n\r') #remove whitespaces
    if s.startswith("("):
      s = s.strip('\(1234567890\) ') #remove ranking
    if s:
      qlist.append(s)

#https://docs.python.org/2/howto/argparse.html
parser = argparse.ArgumentParser(description='Get scores for a team')
parser.add_argument('team', help='team to search for')
parser.add_argument('sleepTime', type=int, help='integer, in seconds to sleep')
parser.add_argument('uri', help='URI to get scores from')
args = parser.parse_args()
# initialize variables
urlp = args.uri
team = args.team
sec = args.sleepTime
aways = [] #list for away teams and scores
homes = [] #list for home teams and scores

while True:
  parsed = urlparse.urlsplit(urlp) # https://docs.python.org/2/library/urlparse.html
  query_arg = urlparse.parse_qsl(parsed.query)
  urli = parsed.scheme+"://"+parsed.netloc+parsed.path
  data = urllib.urlencode(query_arg)
  full_url = urli+"?"+data

  #http://www.pythonforbeginners.com/python-on-the-web/web-scraping-with-beautifulsoup/
  r = requests.get(full_url)
  #print "Got Status Code: ", r.status_code
  data = r.text
  soup = BeautifulSoup(data)

  #use soupParser Get all the teams and the scores, and store in lists
  soupParser('class', 'away', aways, soup)
  soupParser('class', 'home', homes, soup)

  #find the team from input argument
  t = -1
  t = find_team(team,aways)
  away = True;
  #print aways
```

```
  if t == -1:
    away = False;
    t = find_team(team,homes)
    #print homes
  if t == -1:
    print "Team wasn't found, try your search again"
    break
  else:
  #Print the score, with the searched for Team first
    if away:
      print aways[t] + " " + aways[t+1] + ", " + homes[t+1] + " " + homes[t]
    else:
      print homes[t] + " " + homes[t-1] + ", " + aways[t-1] + " " + aways[t]

  time.sleep(sec) #https://docs.python.org/2/library/time.html
```

Listing 6: Python program

3 Question

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

3.1 Answer

For the above graph, give the values for:

IN: 2.
Nodes M and I have edges leading away only.
I –> H
I –> J M –> A
I –> K M –> N

SCC: 6.
Nodes A, B, C, G, N, J have edges leading to and from them.

A –> B		I –> J
B –> C	C –> G	J –> D
C –> D	G –> C	M –> N
C –> A	G –> H	N –> D

OUT: 3
Nodes H, D, and K have edges only leading to them.

G –> H	J –> D
I –> H	L –> D
C –> D	N –> D

Tendrils: 1
Node L is a tendril to D, because D is apart of OUT.
L –> D

Tubes: 2
The edges I–>H and I–>K are tubes because I is apart of IN and H, K are in OUT.

Disconnected: 2.
Nodes E and F are only connected via E–>F.