

Assignment 1

Introduction to Web Science

Dr. Michael Nelson

Fall 2014

Sybil Melton

September 11, 2014

Contents

1	Question	4
1.1	The Answer	4
2	Question	5
2.1	Answer	5
2.1.1	Argument Parsing	5
2.1.2	URL Parsing	5
2.1.3	Web Scraping	6
2.1.4	Data Parsing and Output	6
2.1.5	Program	9
3	Question	11
3.1	Answer	11

List of Figures

1	Screenshot of HTML response opened in Chrome	4
2	Screenshot of SSH client	8

Listings

1	cURL command used	4
2	Argument Parsing	5
3	URL Parsing	5
4	BeautifulSoup Implementation	6
5	Searching Data and Printing Output	6
6	Python program	9

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.

```
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.

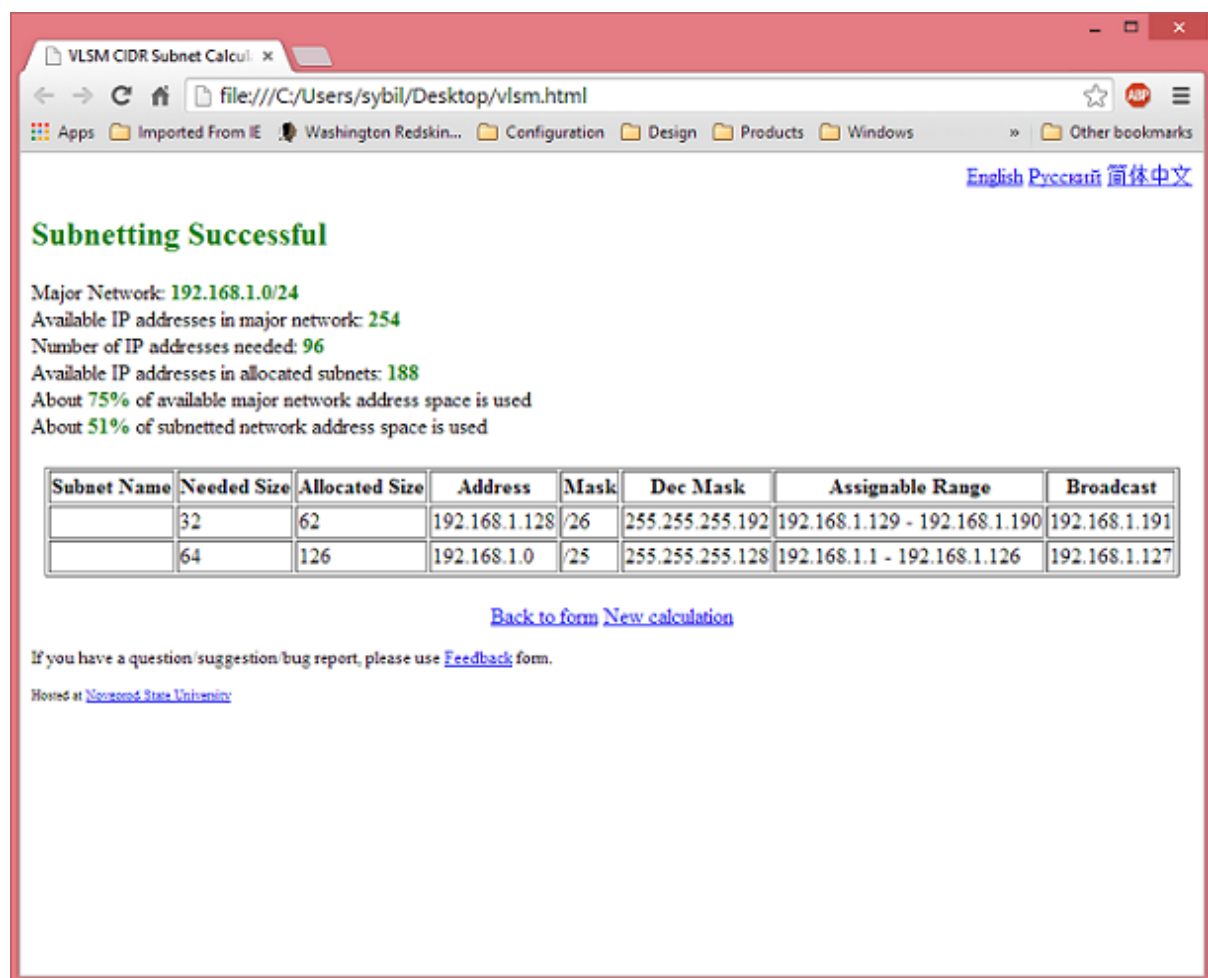


Figure 1: Screenshot of HTML response opened in Chrome

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 inherently. It enabled me to use names that made sense for the variables. Listing 2 is the implementation of argparse used.

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

```
1 parsed = urlparse.urlsplit(urlp) # https://docs.python.org/2/library/urlparse.html
2 query_arg = urlparse.parse_qs(parsed.query)
3 urli = parsed.scheme+"://"+parsed.netloc+parsed.path
4 data = urllib.urlencode(query_arg)
5 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.

```
1 #Function
2 #http://stackoverflow.com/questions/2957013/beautifulsoup-just-get-inside-of-a-tag-no-matter-
   how-many-enclosing-tags-there
3 def soupParser(attr, value, qlist, soup):
4     for node in soup.findAll(attrs={attr : value}):
5         #http://stackoverflow.com/questions/16835449/python-beautifulsoup-extract-text-between-
           element
6         s = ''.join(node.findAll(text=True))
7         s = s.strip(' \t\n\r') #remove whitespaces
8         if s.startswith("("):
9             s = s.strip('\(1234567890\ ') #remove ranking
10        if s:
11            qlist.append(s)
12 #####
13 # How used:
14 #http://www.pythonforbeginners.com/python-on-the-web/web-scraping-with-beautifulsoup/
15 r = requests.get(full_url)
16 #print "Got Status Code: ", r.status_code
17 data = r.text
18 soup = BeautifulSoup(data)
19
20 #use soupParser Get all the teams and the scores, and store in lists
21 soupParser('class', 'away', aways, soup)
22 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.

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

```

8         i = tlist.index(value, i+1)
9     except ValueError:
10         break
11     return i
12 #####
13 # How used:
14 #find the team from input argument
15 t = find_team(team,aways)
16 away = True;
17 if t == -1:
18     away = False;
19     t = find_team(team,homes)
20 if t == -1:
21     print "Team wasn't found, try your search again"
22 else:
23     #Print the score, with the searched for Team first
24     if away:
25         print aways[t] + " " + aways[t+1] + ", " + homes[t+1] + " " + homes[t]
26     else:
27         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.

```
linux.cs.odu.edu - linux - SSH Secure Shell
File Edit View Window Help
Quick Connect Profiles

Old Dominion 34, N.C. State 46
Old Dominion 34, N.C. State 46
Old Dominion 34, N.C. State 46
^CTraceback (most recent call last):
  File "yahooSportScoreParser.py", line 81, in <module>
    time.sleep(sec) #https://docs.python.org/2/library/time.html
KeyboardInterrupt
sirius:~/webscience> python yahooSportScoreParser.py "Old Dominion" 5 "http://sports.y
ahoo.com/college-football/scoreboard/"
Team wasn't found, try your search again
sirius:~/webscience>
sirius:~/webscience> python yahooSportScoreParser.py "Ohio St." 5 "http://sports.yahoo
.com/college-football/scoreboard/?week=2&conf=all"
Ohio St. 21, Virginia Tech 35
Ohio St. 21, Virginia Tech 35
^CTraceback (most recent call last):
  File "yahooSportScoreParser.py", line 81, in <module>
    time.sleep(sec) #https://docs.python.org/2/library/time.html
KeyboardInterrupt
sirius:~/webscience> python yahooSportScoreParser.py "Ohio St." 5 "http://sports.yahoo
.com/college-football/scoreboard/?week=1&conf=all"
Team wasn't found, try your search again
sirius:~/webscience>
sirius:~/webscience> python yahooSportScoreParser.py "South Carolina" 5 "http://sports
.yahoo.com/college-football/scoreboard/?week=1&conf=all"
Team wasn't found, try your search again
sirius:~/webscience> python yahooSportScoreParser.py "South Carolina" 5 "http://sports
.yahoo.com/college-football/scoreboard/?week=1&conf="
South Carolina 28, Texas A&M; 52
South Carolina 28, Texas A&M; 52
South Carolina 28, Texas A&M; 52
South Carolina 28, Texas A&M; 52
South Carolina 28, Texas A&M; 52
^CTraceback (most recent call last):
  File "yahooSportScoreParser.py", line 81, in <module>
    time.sleep(sec) #https://docs.python.org/2/library/time.html
KeyboardInterrupt
sirius:~/webscience>
```

Connected to linux.cs.odu.edu | SSH2 - aes128-cbc - hmac-md5 - n... 86x38

Figure 2: Screenshot of SSH client

2.1.5 Program

The entire program, as written.

```
1 import urllib2
2 import re
3 import requests
4 import argparse
5 import urllib
6 import urlparse
7 from bs4 import BeautifulSoup
8 import time
9
10 # find index of the team in the list
11 #http://stackoverflow.com/questions/176918/finding-the-index-of-an-item-given-a-list-
    containing-it-in-python
12 def find_team(value, tlist):
13     i = -1
14     while True:
15         try:
16             i = tlist.index(value, i+1)
17         except ValueError:
18             break
19     return i
20
21 #http://stackoverflow.com/questions/2957013/beautifulsoup-just-get-inside-of-a-tag-no-matter-
    how-many-enclosing-tags-there
22 def soupParser(attr, value, qlist, soup):
23     for node in soup.findAll(attrs={attr : value}):
24         #http://stackoverflow.com/questions/16835449/python-beautifulsoup-extract-text-between-
            element
25         s = ''.join(node.findAll(text=True))
26         s = s.strip(' \t\n\r') #remove whitespaces
27         if s.startswith("("):
28             s = s.strip('\(1234567890\ ') #remove ranking
29         if s:
30             qlist.append(s)
31
32 #https://docs.python.org/2/howto/argparse.html
33 parser = argparse.ArgumentParser(description='Get scores for a team')
34 parser.add_argument('team', help='team to search for')
35 parser.add_argument('sleepTime', type=int, help='integer, in seconds to sleep')
36 parser.add_argument('uri', help='URI to get scores from')
37 args = parser.parse_args()
38 # initialize variables
39 urlp = args.uri
40 team = args.team
41 sec = args.sleepTime
42 aways = [] #list for away teams and scores
43 homes = [] #list for home teams and scores
44
45 while True:
46     parsed = urlparse.urlsplit(urlp) # https://docs.python.org/2/library/urlparse.html
47     query_arg = urlparse.parse_qs(parsed.query)
48     urli = parsed.scheme+"://"+parsed.netloc+parsed.path
49     data = urllib.urlencode(query_arg)
50     full_url = urli+"?" +data
51
52     #http://www.pythonforbeginners.com/python-on-the-web/web-scraping-with-beautifulsoup/
53     r = requests.get(full_url)
54     #print "Got Status Code: ", r.status_code
55     data = r.text
56     soup = BeautifulSoup(data)
57
58     #use soupParser Get all the teams and the scores, and store in lists
59     soupParser('class', 'away', aways, soup)
60     soupParser('class', 'home', homes, soup)
61
62     #find the team from input argument
63     t = -1
64     t = find_team(team, aways)
65     away = True;
66     #print aways
```



```

67 | if t == -1:
68 |     away = False;
69 |     t = find_team(team,homes)
70 |     #print homes
71 | if t == -1:
72 |     print "Team wasn't found, try your search again"
73 |     break
74 | else:
75 | #Print the score, with the searched for Team first
76 |     if away:
77 |         print ways[t] + " " + ways[t+1] + ", " + homes[t+1] + " " + homes[t]
78 |     else:
79 |         print homes[t] + " " + homes[t-1] + ", " + ways[t-1] + " " + ways[t]
80 |
81 | 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 \rightarrow B	C \rightarrow G	I \rightarrow H	L \rightarrow D
B \rightarrow C	E \rightarrow F	I \rightarrow J	M \rightarrow A
C \rightarrow D	G \rightarrow C	I \rightarrow K	M \rightarrow N
C \rightarrow A	G \rightarrow H	J \rightarrow D	N \rightarrow D

3.1 Answer

For the above graph, give the values for:

IN: 2.

Nodes M and I have edges leading away only.

I \rightarrow H

I \rightarrow J M \rightarrow A

I \rightarrow K M \rightarrow N

SCC: 6.

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

A \rightarrow B I \rightarrow J

B \rightarrow C C \rightarrow G J \rightarrow D

C \rightarrow D G \rightarrow C M \rightarrow N

C \rightarrow A G \rightarrow H N \rightarrow D

OUT: 3

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

G \rightarrow H J \rightarrow D

I \rightarrow H L \rightarrow D

C \rightarrow D N \rightarrow D

Tendrils: 1

Node L is a tendril to D, because D is apart of OUT.

L \rightarrow D

Tubes: 2

The edges I \rightarrow H and I \rightarrow 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 \rightarrow F.