CS896 Introduction to Web Science
Fall 2013
Report for Assignment 1

Corren G. McCoy

September 12, 2013

Contents

1 Assignment 1 4
1.1 Question 1 4
Problem 4
Response 4
1.2 Question 2 4
Problem 4
Response 4
1.3 Question 3 5
Problem 5
Response 6

Appendices **9**

A R Code for Bow-tie Graph **10**

B Python Code for ScoreCenter **11**

List of Figures

1 Browser Query Results 5
2 cURL HTML Response 6
3 Score Center Session Output 7
4 Question 3 Graph 8

List of Tables

1 Bow-tie Graph Analysis 7

1 Assignment 1

1.1 Question 1

Problem 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" (e.g., save it to a file and then view that file in a browser and take a screen shot).

Response For this question, the NFL website `http://www.NFL.com/` was selected. The page contains a simple form which has a text box that allows the user to search for NFL-related articles using keywords. The following steps were followed to script the HTTP request to POST to the form to initiate the search.

1. Issued a cURL command to retrieve the HTML source code for the web page which was subsequently saved to a local file on the web server (fast.cs.odu.edu).

   curl www.nfl.com -O

2. Used the Unix grep command to search the local file for all <form>and <input>tags in order to identify the form field names and the action when the form is submitted.

   <input type="text" name="query" maxlength="50" value="Search NFL.com" data-placeholder="Search NFL.com"/>

3. The NFL.com page is a GET form that uses the GET method as shown below.

   <form class="ui3-searchbox yui3-skin-sam" action="`http://search.nfl.com/search`" method="get">

4. Issued a query for information concerning "tim tebow patriots." To post to the form, we used the *–data-urlencode* option to send name-value pairs to the website. The urlencode option properly handles our request by replacing spaces with %20.

   curl –data-urlencode "query=tim tebow patriots" –location –output nflresults.html "http://search.nfl.com/search"

5. The *-output* option saves the HTML response to a local file so we can review the result in a browser. Figures 1 and 2 show the query results when requested directly from www.NFL.com and the results from the scripted HTTP request. In both instances, we can see the same number of query results (i.e., 34068) in response to the query.

1.2 Question 2

Problem Write a Python program that accepts three arguments: college team, sleep time in seconds, and a URI. The program should download the URI, find the game corresponding to the team argument, print out the current score, sleeps for the specified seconds, and then repeats (until control-C is hit).

Response We designed a modular program which contains functions that present menus to select a college football team, weekly scoreboard, and prompt the user to enter the sleep time between updates. The user-selected menu options form the arguments for the function that searches the ESPN website (`http://scores.espn.go/ncf/scoreboard`) for specific games played during the year and week indicated by the user's selections. In order to efficiently search the website, we first studied

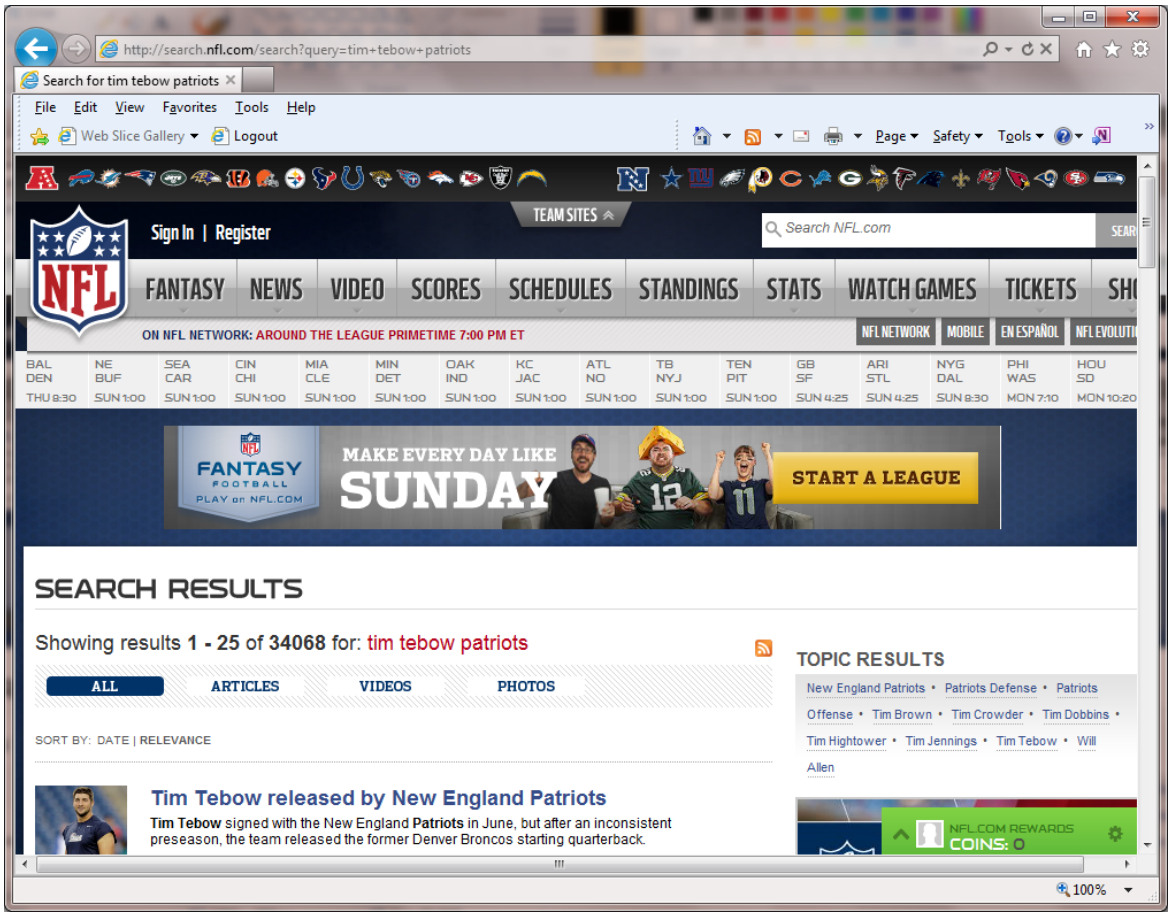

Figure 1: Browser Query Results

the HTML using the *view source* feature in the browser to identify patterns which indicated team pairings and the final score. We discovered the home and visiting team for each game is wrapped in a tag with a class attribute of either *'team visitor'* or *'team home'*. Using the anchor (<a>) tag immediately after the allowed us to extract the name of the football team in each position. We could then iterate over a series of list (<li>) items to locate the final score using a class attribute of *'final'*. Once a pattern was determined, we used the BeautifulSoup (`https://pypi.python.org/pypi/BeautifulSoup`) library to parse the HTML and extract the text that we needed (i.e., visiting team, home team). As we navigated through the HTML, we built two parallel lists, indexed by game number, to store each team and final score. Finally, we iterated over the parallel lists, searching for any game where the user's team (e.g., Old Dominion) was either the home or visiting team to display the final score. The source code for our ScoreCenter program is shown in Appendix B. A typical user session is shown in Figure 3. No change in score is noted between subsequent updates because no live games were being played.

1.3 Question 3

Problem Consider the "bow-tie" graph shown in Figure 9 of the Broder et al. [1] paper. Now consider the following directed graph whose node connections are described below. Perform a graph analysis as discussed in the referenced paper.

$$A \rightarrow B,\ B \rightarrow C,\ C \rightarrow D,\ C \rightarrow A,\ C \rightarrow G,\ E \rightarrow F,\ G \rightarrow C,\ G \rightarrow H,\ I \rightarrow H,\ I \rightarrow J,\ I \rightarrow K,$$

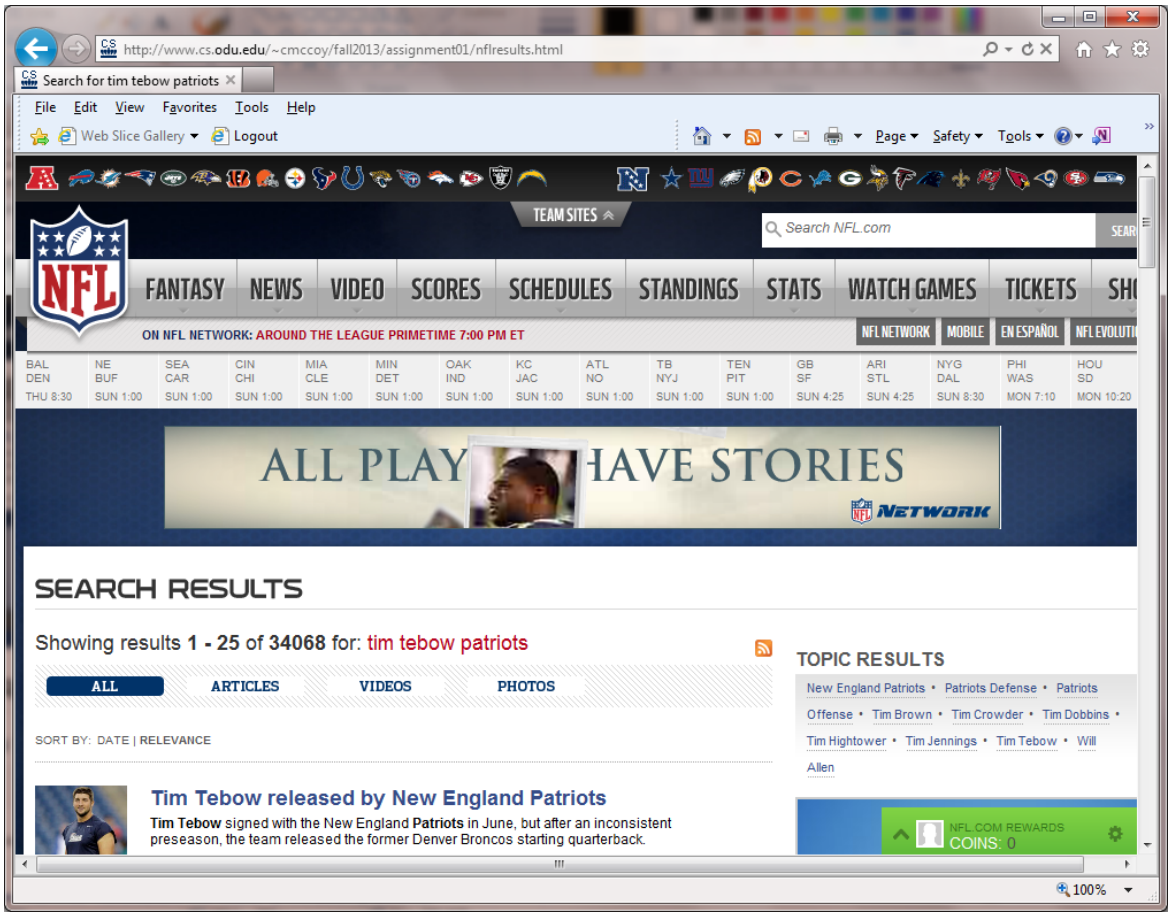

Figure 2: cURL HTML Response

J → D, L → D, M → A, M → N, N → D

Response We used the igraph library in R, which is a package for network analysis, to produce graphical representation of the node connections as shown in Figure 4. The source code which generated the graph is shown in Appendix A. We then applied the definitions defined in Levene [2] to assign each pair of nodes in the directed graph to a component of the bow-tie structure as shown in Table 1.

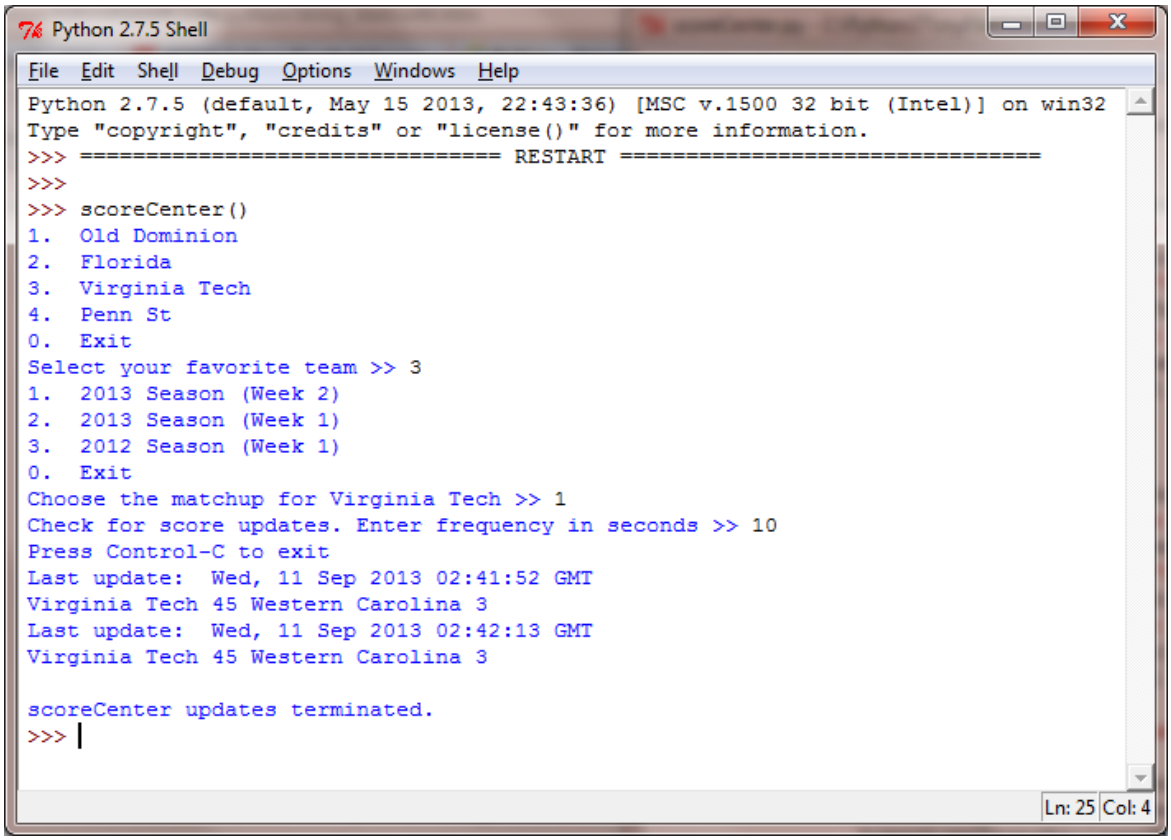

Figure 3: Score Center Session Output

Graph Structure	Characteristic (Levene [2])	Nodes
IN	IN consists of pages that can reach the strongly connected component (SCC) but cannot be reached from it.	(M, A)
SCC	A strongly connected component (SCC) consists of pages that reach one another along directed links.	(C, A) (A, B) (B, C) (C, G) (G, C)
OUT	OUT consists of pages that are accessible from the SCC but do not link back to it.	(C, D) (G, H)
Tendrils	Tendrils contain pages that cannot reach the SCC and cannot be reached from the SCC.	(L,D) (J, D) (I, J) (I, H) (I, K)
Tubes	A tube has a directed path from IN to OUT bypassing the SCC.	(M, N) (N, D)
Disconnected	The pages in the disconnected area are not even weakly connected to the SCC.	(E, F)

Table 1: Bow-tie Graph Analysis

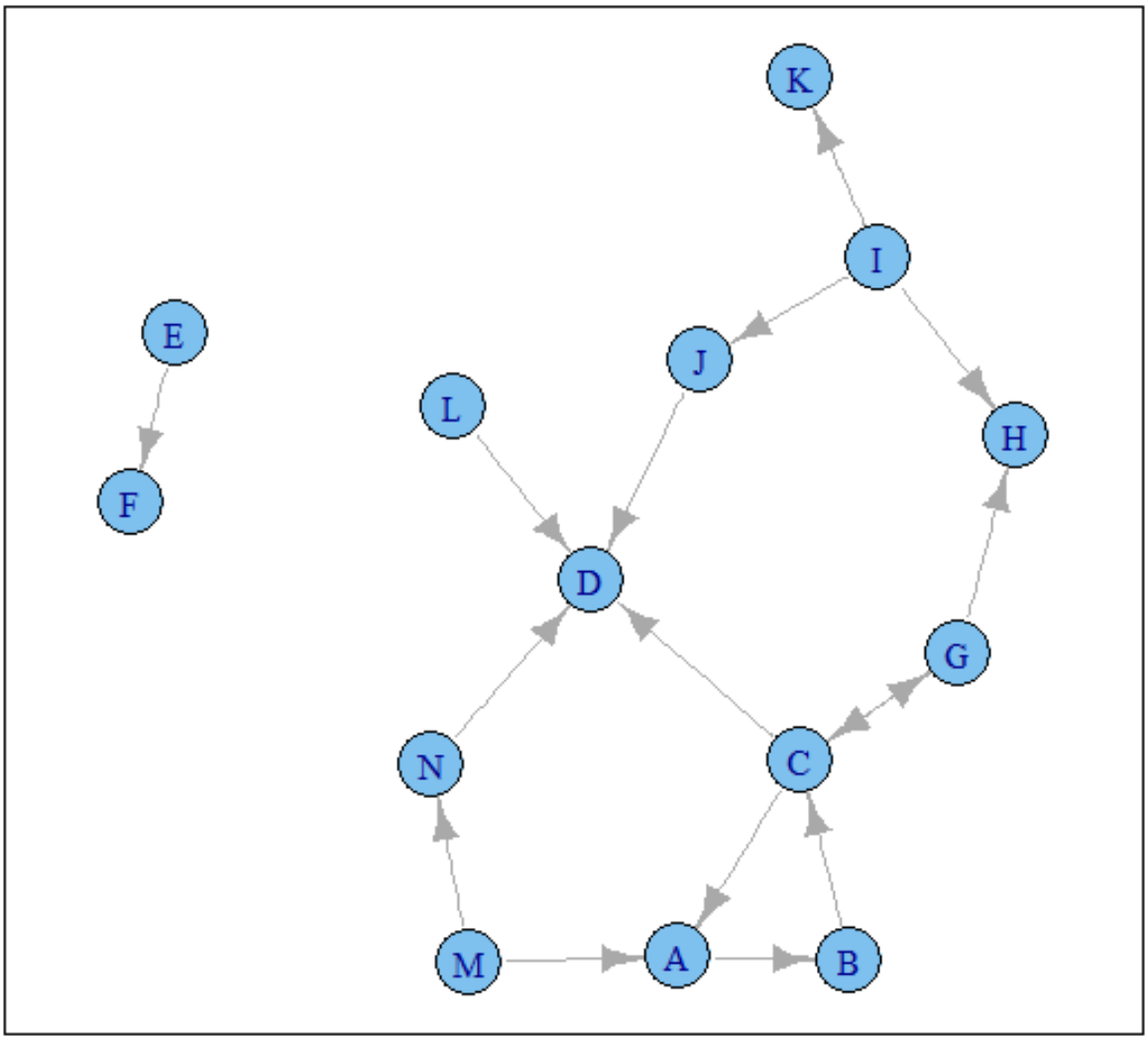

Figure 4: Question 3 Graph

Bibliography

[1] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. *Computer networks*, 33(1):309–320, 2000.

[2] M. Levene. *An introduction to search engines and web navigation*. Wiley. com, 2011.

Appendix A

R Code for Bow-tie Graph

```
library("igraph")
xlist<-read.table("graph.txt")
xlist<-graph.data.frame(xlist)
plot(xlist)
box()
# Trim off excess margin space (bottom, left, top, right)
par(mar=c(0.2, 0.2, 0.2, 0.2))
```

Appendix B

Python Code for ScoreCenter

```
#!/usr/bin/python
from BeautifulSoup import BeautifulSoup
import os
import sys
import time
import urllib2
import urllib
def teamSelectionMenu():
    #Display menu until user selects a valid option
    while True:
        #Use an associative array to store the team names and menu options.
        #The values (team names) must match how the names are displayed
        #on the http://scores.epsn.go.com
        teamName={0: "Exit", 1:"Old Dominion", 2:"Florida", 3:"Virginia Tech", 4:"Penn St"}
        print "1. ",teamName[1]
        print "2. ",teamName[2]
        print "3. ",teamName[3]
        print "4. ",teamName[4]
        print "0. ",teamName[0]
        #raw_input() function reads a line from input (i.e. the user) and returns
        #a string by stripping a trailing newline
        try:
            #convert the entered value to an integer to match the key of our associative array
            teamOption=int(raw_input("Select your favorite team" +" >> "))
            if teamOption >= 0 and teamOption <= 4:
                team=teamName[teamOption]
                if teamOption == 0:
                    sys.exit(0)
                else:
                    return team
            else:
                raise ValueError
        except (ValueError):
            print ""
            print "Option must be a number between 0 and 4. Please try again."
            print ""
```

```
def scoreboardMenu(p_teamName):
    #Display the menu until user selects a valid option
    while True:
        #Set the baseURI. Add the parameters after user selects an option
        baseURI="http://scores.espn.go.com/ncf/scoreboard?confId=80&"
        #Associative array stores the menu options
        scoreboardOption={0:"Exit", 1:"2013 Season (Week 2)", 2:"2013 Season (Week 1)",
        3:"2012 Season (Week 1)"}
        #These are the parameters which will be appended to the baseURI
        parameters={1:"seasonYear=2013&seasonType=2&weekNumber=2",
        2:"seasonYear=2013&seasonType=2&weekNumber=1",
        3:"seasonYear=2012&seasonType=2&weekNumber=1"}
        print "1. ",scoreboardOption[1]
        print "2. ",scoreboardOption[2]
        print "3. ",scoreboardOption[3]
        print "0. ",scoreboardOption[0]
        try:
            #convert the entered value to an integer to match the keys of parameters list
            scoreboardOption=int(raw_input("Choose the matchup for " + p_teamName+" >> "))
            if scoreboardOption >= 0 and scoreboardOption <= 3:
                fullURI=baseURI+parameters[scoreboardOption]
                if scoreboardOption == 0:
                    sys.exit(0)
                else:
                    return fullURI
            else:
                raise ValueError
        except (ValueError):
             print ""
             print "Option must be a number between 0 and 3. Please try again."
             print ""
def sleepMenu():
    #Display the menu until user selects a valid option
    while True:
        try:
            #convert the entered value to an integer
            sleepOption=int(raw_input("Check for score updates.
            Enter frequency in seconds" + " >> "))
            if sleepOption > 0:
                return sleepOption
            else:
                raise ValueError
        except (ValueError):
             print ""
             print "Update frequency must be a number greater than 0. Please try again."
             print ""
##############################################################################
#  This is main procedure in this package
##############################################################################
```

```
def scoreCenter():
    #Print the team selection menu
    team=teamSelectionMenu()
    #Select the weekly matchup
    uri=scoreboardMenu(team)
    #Set the update frequency
    sleepSeconds=sleepMenu()

    print "Press Control-C to exit"
    #display updated score until Control-C is entered from keyboard
    while True:
        try:
            #package the request
            request=urllib2.Request(uri)
            request.add_header('User-agent','Mozilla 5.10')
            response=urllib2.urlopen(request)
            html=response.read()
            print "Last update: ", response.info()['date']
            soup=BeautifulSoup(html)
            soup=BeautifulSoup(soup.prettify())
            visitingTeamsRec={}
            gameNumber=1
            for visitingTeam in soup.html.body.findAll('div',{'class' : 'team visitor'}):
                #first anchor after the <div> is the name of the visiting team
                visitingTeamName=str(visitingTeam.find('a').get('title')).strip()
                #<li class="final" id="332412579-aTotal">20</li>
                visitingScore=visitingTeam.find('li', {'class' : 'final'}).string.strip()
                #create a parallel list for the visiting teams
                visitingTeamsRec[gameNumber] = visitingTeamName + " " + visitingScore
                gameNumber=gameNumber+1

            homeTeamsRec={}
            gameNumber=1
            for homeTeam in soup.html.body.findAll('div',{'class' : 'team home'}):
                #first anchor after the <div> is the name of the home team
                homeTeamName=str(homeTeam.find('a').get('title')).strip()
                #<li class="final" id="332412579-aTotal">20</li>
                homeScore=homeTeam.find('li', {'class' : 'final'}).string.strip()
                #create a parallel list for the home teams
                homeTeamsRec[gameNumber] = homeTeamName + " " + homeScore
                gameNumber=gameNumber+1

            #iterate through all the game scores until the selected team is found
            gameNumber=1
            while gameNumber <= len(visitingTeamsRec):
                visitor=visitingTeamsRec[gameNumber].strip()
                home=homeTeamsRec[gameNumber].strip()
                if home.startswith(team) or visitor.startswith(team):
                    print home, visitor
```

```
            gameNumber=gameNumber+1
        time.sleep(sleepSeconds)
    #Raised when the user hits the interrupt key (normally Control-C or Delete).
    except KeyboardInterrupt:
        print ""
        print "scoreCenter updates terminated."
        sys.exit(0)
response.close()
```