Web Science: Assignment #1

Alexander Nwala

Daniel W. Moore

Sunday, Janury 28, 2018

Problem 1:

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

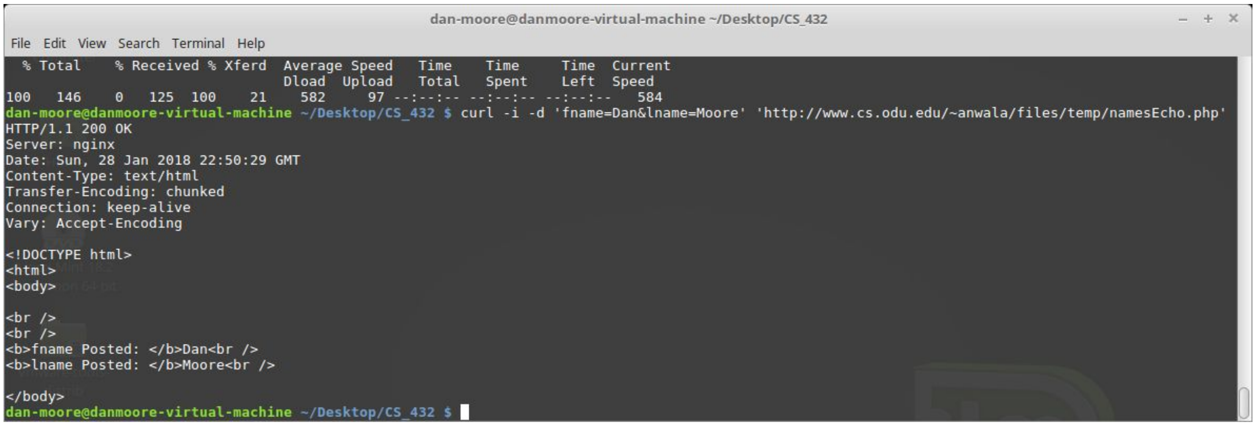
take a screenshot.

Solution:

The solution for this problem is outlined below:

1. Execute curl command from Linux command line in order to post form data to the following url:
   http://www.cs.odu.edu/~anwala/files/temp/namesEcho.php
   See the figure below for command details and output.

Figure 1

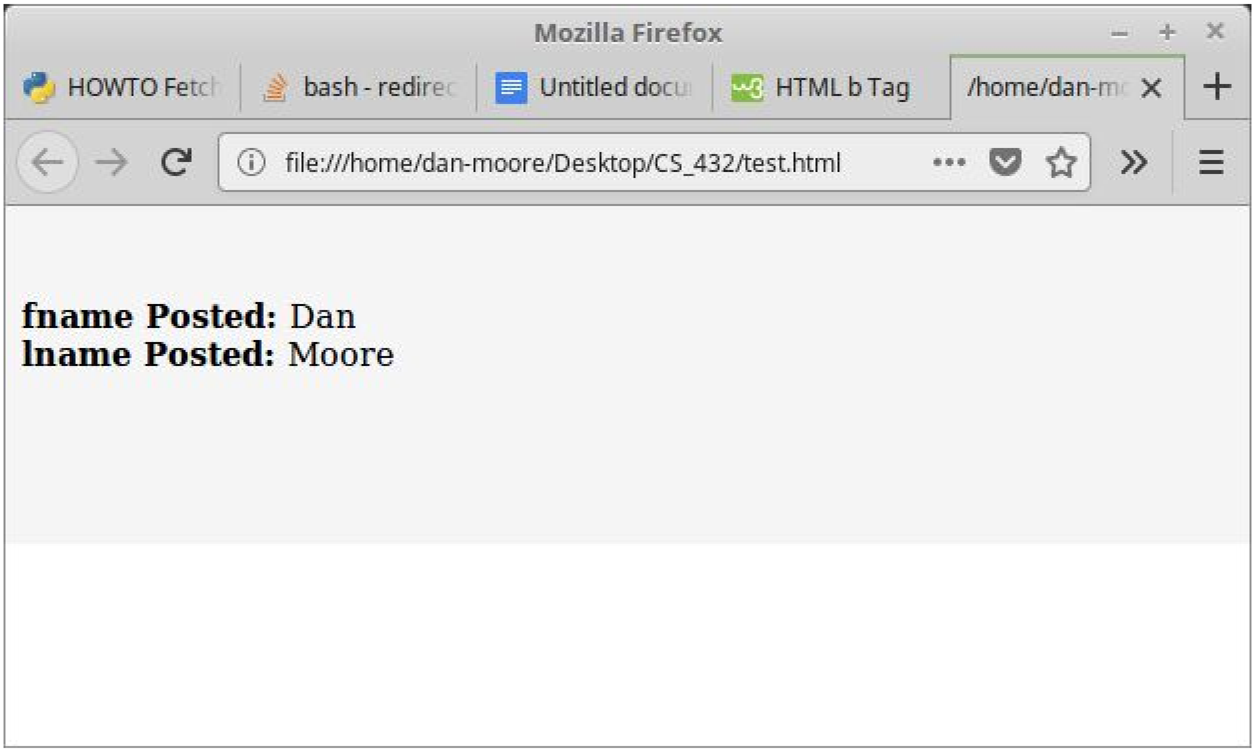


dan-moore@danmoore-virtual-machine ~/Desktop/CS_432
File Edit View Search Terminal Help

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   146    0   125  100    21    582     97 --:--:-- --:--:-- --:--:--   584
dan-moore@danmoore-virtual-machine ~/Desktop/CS_432 $ curl -i -d 'fname=Dan&lname=Moore' 'http://www.cs.odu.edu/~anwala/files/temp/namesEcho.php'
HTTP/1.1 200 OK
Server: nginx
Date: Sun, 28 Jan 2018 22:50:29 GMT
Content-Type: text/html
Transfer-Encoding: chunked
Connection: keep-alive
Vary: Accept-Encoding

<!DOCTYPE html>
<html>
<body>

<br />
<br />
<b>fname Posted: </b>Dan<br />
<b>lname Posted: </b>Moore<br />

</body>
dan-moore@danmoore-virtual-machine ~/Desktop/CS_432 $
```

2. Redirect output of the above curl command into a html file which may be viewed in a web browser. The figure below contains a screenshot of the above html-encoded information when viewed in a web browser.

Figure 2: Web Browser Representation



fname Posted: Dan
lname Posted: Moore

Problem 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:

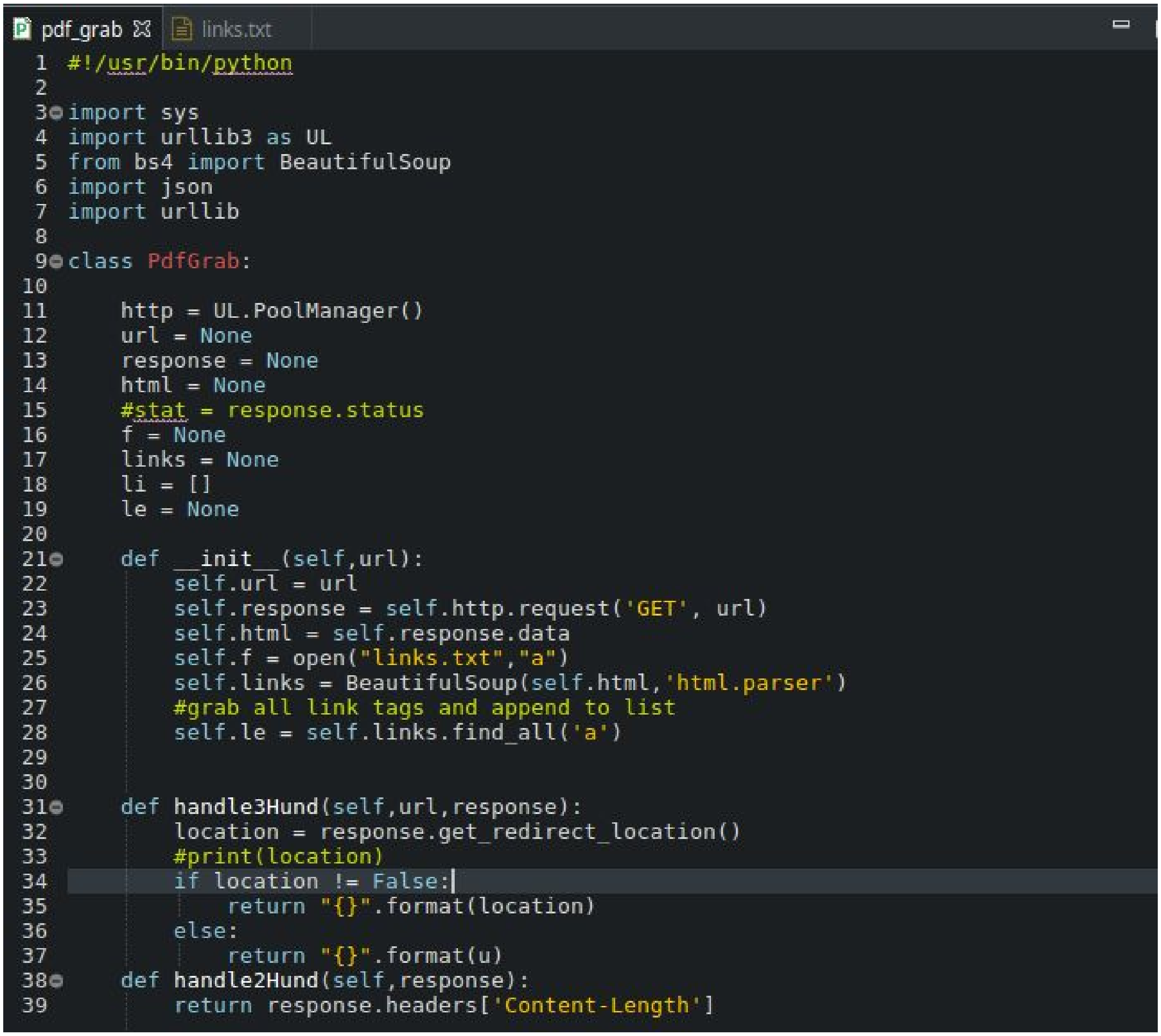
   http://www.cs.odu.edu/~mln/teaching/cs532-s17/test/pdfs.html

Solution:

The solution for this problem is outlined below:

1. Wrote and executed the following Python program which accepts a url as a command line parameter.

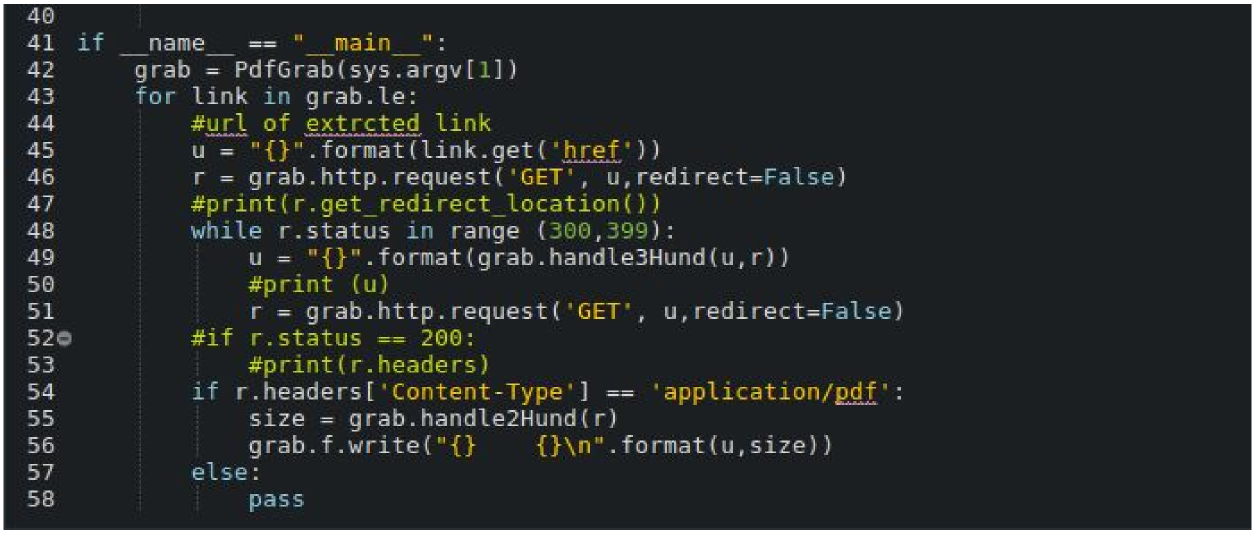
Figure 3:pdf_grab.py



```
#!/usr/bin/python

import sys
import urllib3 as UL
from bs4 import BeautifulSoup
import json
import urllib

class PdfGrab:

    http = UL.PoolManager()
    url = None
    response = None
    html = None
    #stat = response.status
    f = None
    links = None
    li = []
    le = None

    def __init__(self,url):
        self.url = url
        self.response = self.http.request('GET', url)
        self.html = self.response.data
        self.f = open("links.txt","a")
        self.links = BeautifulSoup(self.html,'html.parser')
        #grab all link tags and append to list
        self.le = self.links.find_all('a')


    def handle3Hund(self,url,response):
        location = response.get_redirect_location()
        #print(location)
        if location != False:
            return "{}".format(location)
        else:
            return "{}".format(u)
    def handle2Hund(self,response):
        return response.headers['Content-Length']
```

(This space left intentionally blank)

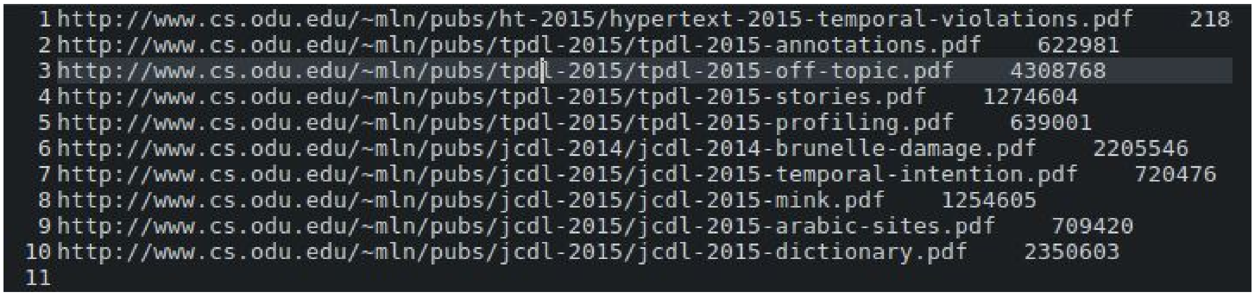
Figure 3 Continued



```
if __name__ == "__main__":
    grab = PdfGrab(sys.argv[1])
    for link in grab.le:
        #url of extrcted link
        u = "{}".format(link.get('href'))
        r = grab.http.request('GET', u,redirect=False)
        #print(r.get_redirect_location())
        while r.status in range (300,399):
            u = "{}".format(grab.handle3Hund(u,r))
            #print (u)
            r = grab.http.request('GET', u,redirect=False)
        #if r.status == 200:
            #print(r.headers)
        if r.headers['Content-Type'] == 'application/pdf':
            size = grab.handle2Hund(r)
            grab.f.write("{}    {}\n".format(u,size))
        else:
            pass
```

2. The above program extracts all links from the url supplied at command line and prints the url and file size of any links which result in a pdf file to a file. The program will output the redirect location of any links which respond with status code in the 3xx range. The output of this program using the following url as a command line parameter is shown in the figure below:
http://www.cs.odu.edu/~mln/teaching/cs532-s17/test/pdfs.html

Figure 4 links.txt



```
http://www.cs.odu.edu/~mln/pubs/ht-2015/hypertext-2015-temporal-violations.pdf    218
http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-annotations.pdf    622981
http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-off-topic.pdf    4308768
http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-stories.pdf    1274604
http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-profiling.pdf    639001
http://www.cs.odu.edu/~mln/pubs/jcdl-2014/jcdl-2014-brunelle-damage.pdf    2205546
http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-temporal-intention.pdf    720476
http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-mink.pdf    1254605
http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-arabic-sites.pdf    709420
http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-dictionary.pdf    2350603
```

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

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:

SCC:

OUT:

Tendrils:

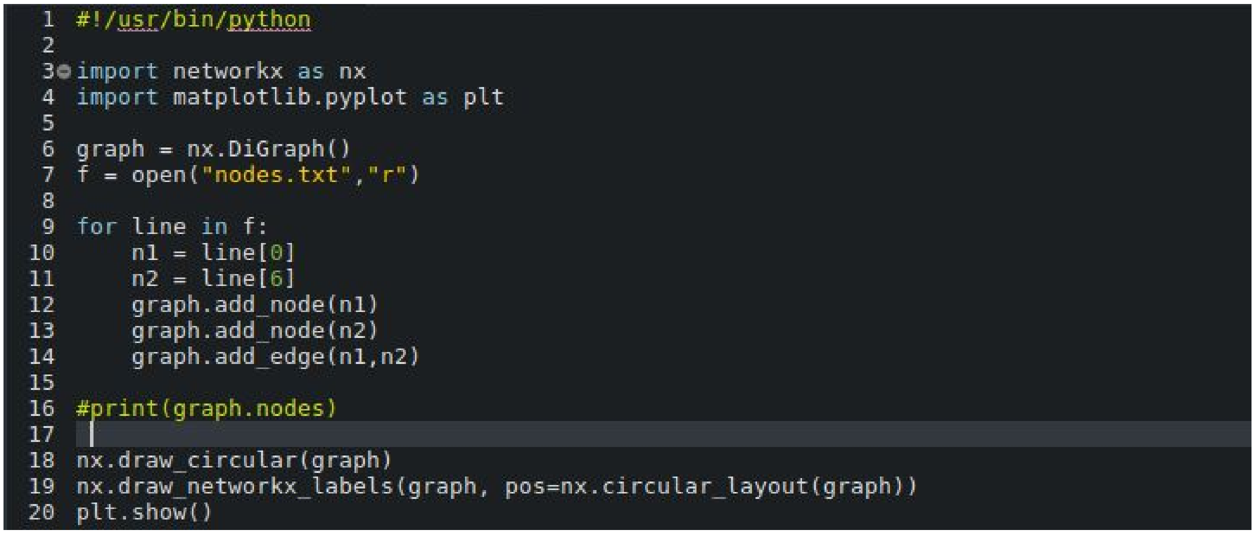
Tubes:

Disconnected:

Solution:

1. Placed provided graph data into a text file to be read by Python program.
2. Wrote and executed the following program which utilizes the networkx and matplotlib libraries to produce a visual representation of the provided graph data.

Figure 5 web_graph.py

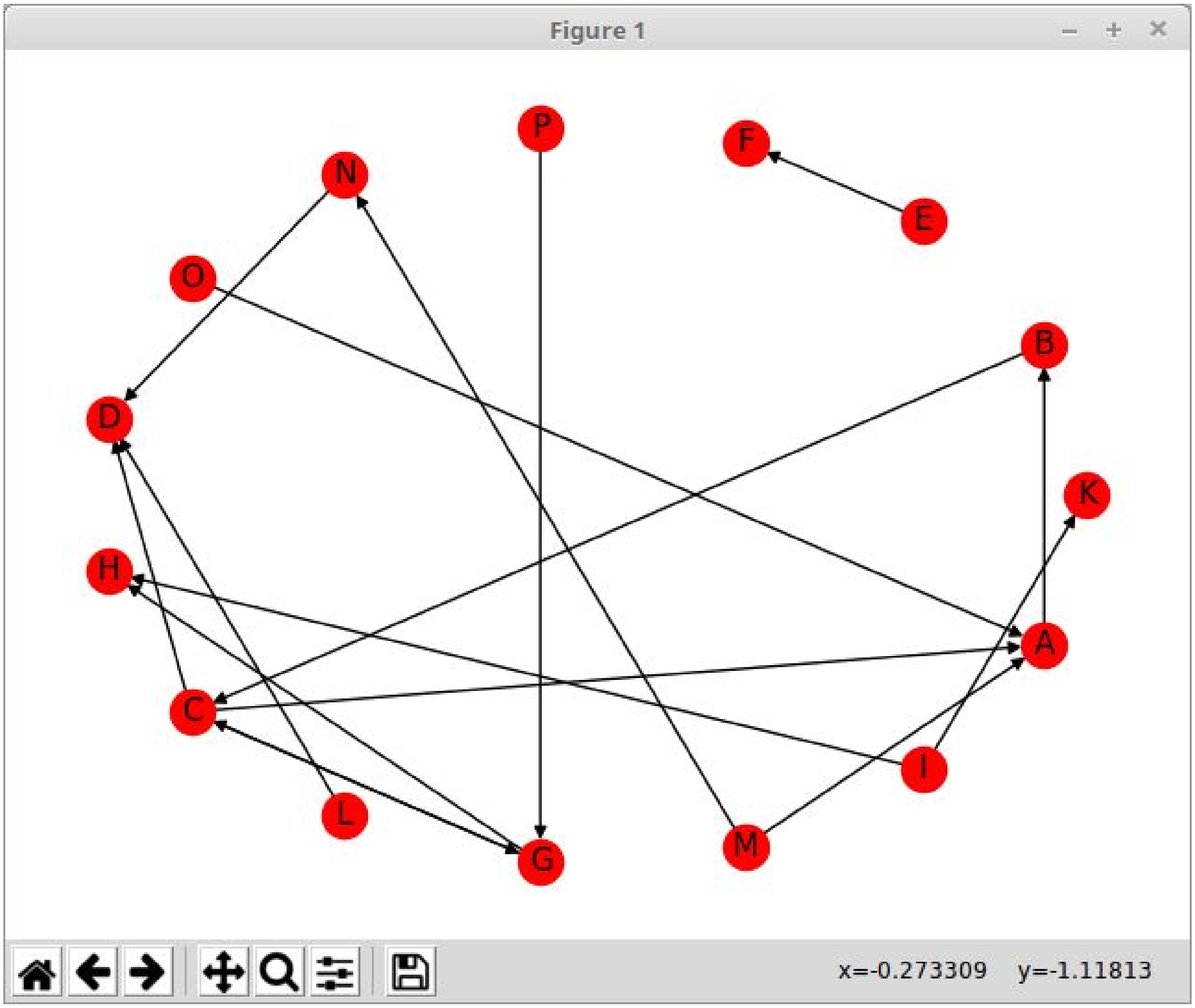


```
#!/usr/bin/python

import networkx as nx
import matplotlib.pyplot as plt

graph = nx.DiGraph()
f = open("nodes.txt","r")

for line in f:
    n1 = line[0]
    n2 = line[6]
    graph.add_node(n1)
    graph.add_node(n2)
    graph.add_edge(n1,n2)

#print(graph.nodes)

nx.draw_circular(graph)
nx.draw_networkx_labels(graph, pos=nx.circular_layout(graph))
plt.show()
```

3. Visually examined the graph below to determine to which group each node belongs

(This space left intentionally blank)

Figure 6 Directional Node Graph



SCC: A, B, C, G

IN: M, O, P

OUT: D, H, K

TENDRILS: L, I

TUBES: N

DISCONNECTED: E, F