Justin Schaffner (JASchaff)
CS532 — Spring 2017 **Assignment 1** January 23, 2017

Problem 1: Posting with Curl

I chose Hillary Clintons page because I knew it had an email subscription form. https://my.democrats.org/page/s/say-you-re-in-for-what-s-next-sticker-h/ I used cURL to pull the page data:

```
curl
   https://my.democrats.org/page/s/say-you-re-in-for-what-s-next-sticker-h/
   > ~/Desktop/Hillary.txt
```

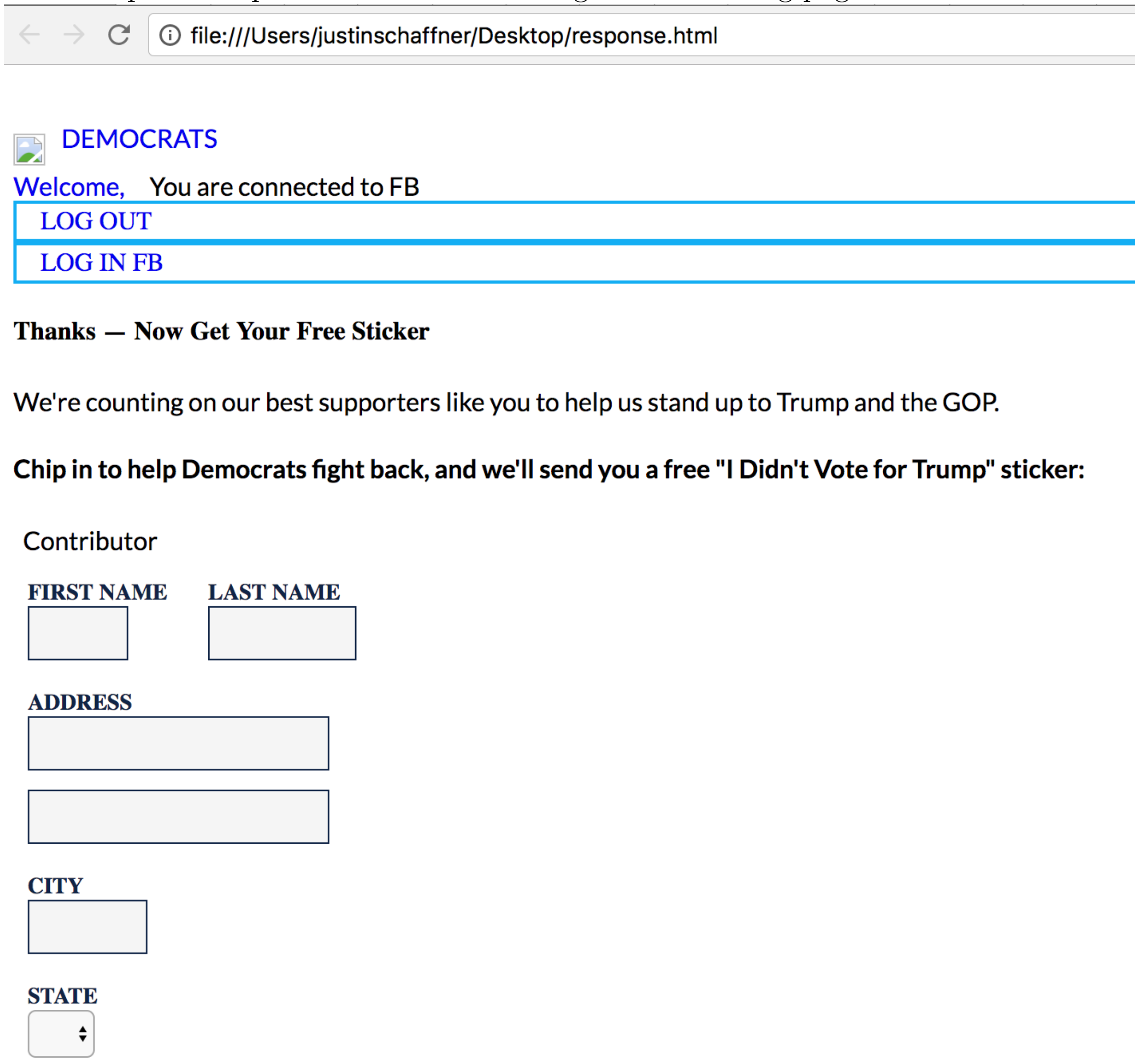
Scanning through the text with comm-f with ”input” as a search parameter I found a list of the names of the fields. A blog post had led me to a script called formfinder.pl , but when I tried to run it I got an error. Rather than try to fix it I just searched manually. The field names the server was expecting were firstname, lastname, email and zip. I used cURL to post the data:

```
curl -d firstname="charlie" -d lastname="chaplin" -d
   email="charlie.chaplin@mail.com" -d zip="23510"
   https://my.democrat.org/page/s/say-you-re-in-for-what-s-next-sticker-h/
   > ~/Desktop/response.html
```

The first attempt gave me a redirect address, so I added -L to follow the redirect and get the page

```
curl -L -d firstname="charlie" -d lastname="chaplin" -d
   email="charlie.chaplin@mail.com" -d zip="23510"
   https://my.democrat.org/page/s/say-you-re-in-for-what-s-next-sticker-h/
   > ~/Desktop/response.html
```

When I opened response.html in Chrome I got the following page:


file:///Users/justinschaffner/Desktop/response.html

DEMOCRATS

Welcome, You are connected to FB

LOG OUT

LOG IN FB

Thanks — Now Get Your Free Sticker

We're counting on our best supporters like you to help us stand up to Trump and the GOP.

Chip in to help Democrats fight back, and we'll send you a free "I Didn't Vote for Trump" sticker:

Contributor

FIRST NAME LAST NAME

ADDRESS

CITY

STATE

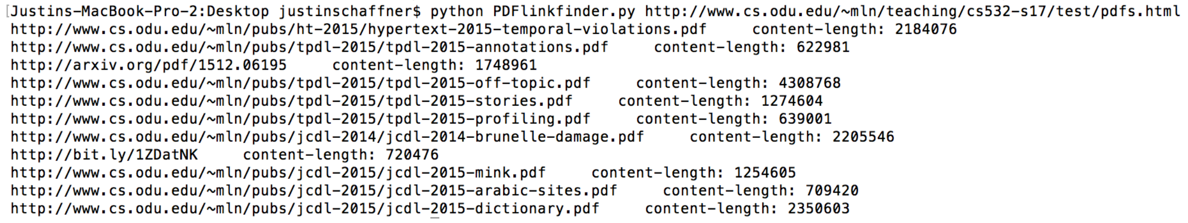
Problem 2: Python Link Finder

I built the program on my Mac using Python 3.6. For HTP handling I downloaded the requests library and I installed BeautifulSoup4 to read the response body. I used lxml as the parser for beautifulsoup. Several stackexchange posts had recommended it over the built-in parser for being faster. The code ended up being pretty simple to write.

```
import requests
from bs4 import BeautifulSoup
import lxml
import sys
```

```
if not(len(sys.argv)==2):
    raise ValueError(”Missing or multiple arguments”)
address=sys.argv[1]
if not (address.startswith(”http://”) or address.startswith(”https
   ://”)):
    raise ValueError(”Argument not an URI”)
page=requests.get(address, allow_redirects=True)
soup=BeautifulSoup(page.content, ”lxml”)
linklist=[]
pdflinklist=[]
numlinks=0
numpdfs=0
for link in soup.find_all('a'):
    linklist.append(link.get('href'))
    numlinks+=1
print(”Number of links found: ”, numlinks)
for i in linklist:
    tpage=requests.get(i, allow_redirects=True, stream=True)
    temp=tpage.headers.get('content-type')
    if ”pdf” in temp:
        pdflinklist.append((i,tpage.headers.get('content-length')))
        numpdfs+=1
    tpage.close
print(”Number of links to pdfs found: ”, numpdfs)
for i in pdflinklist:
    print (”\t\tcontent-length: ”.join(i))
```

The ”raise ValueError” solution to jumping out of the program if the args are wrong also came from a stack exchange post. I thought it would be less problematic than calling quit or exit from sys, though I could be wrong. I know that was always a big NO from C++. I ran the program with the required URI for the test page:

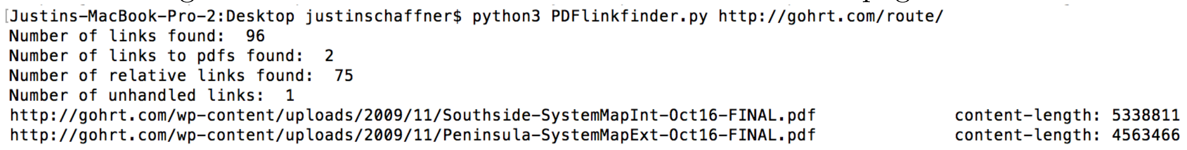


```
Justins-MacBook-Pro-2:Desktop justinschaffner$ python PDFlinkfinder.py http://www.cs.odu.edu/~mln/teaching/cs532-s17/test/pdfs.html
http://www.cs.odu.edu/~mln/pubs/ht-2015/hypertext-2015-temporal-violations.pdf     content-length: 2184076
http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-annotations.pdf     content-length: 622981
http://arxiv.org/pdf/1512.06195     content-length: 1748961
http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-off-topic.pdf     content-length: 4308768
http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-stories.pdf     content-length: 1274604
http://www.cs.odu.edu/~mln/pubs/tpdl-2015/tpdl-2015-profiling.pdf     content-length: 639001
http://www.cs.odu.edu/~mln/pubs/jcdl-2014/jcdl-2014-brunelle-damage.pdf     content-length: 2205546
http://bit.ly/1ZDatNK     content-length: 720476
http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-mink.pdf     content-length: 1254605
http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-arabic-sites.pdf     content-length: 709420
http://www.cs.odu.edu/~mln/pubs/jcdl-2015/jcdl-2015-dictionary.pdf     content-length: 2350603
```

For the second URI I went with the HRT bus schedule page. Initially I ran into a number of errors dealing with missing schema. A lot of the links seem to be relative urls don't have the full http:// address. For this issue I imported urllib.parse.urljoin, which gave me the following change to the ”for i in linklist:” function:

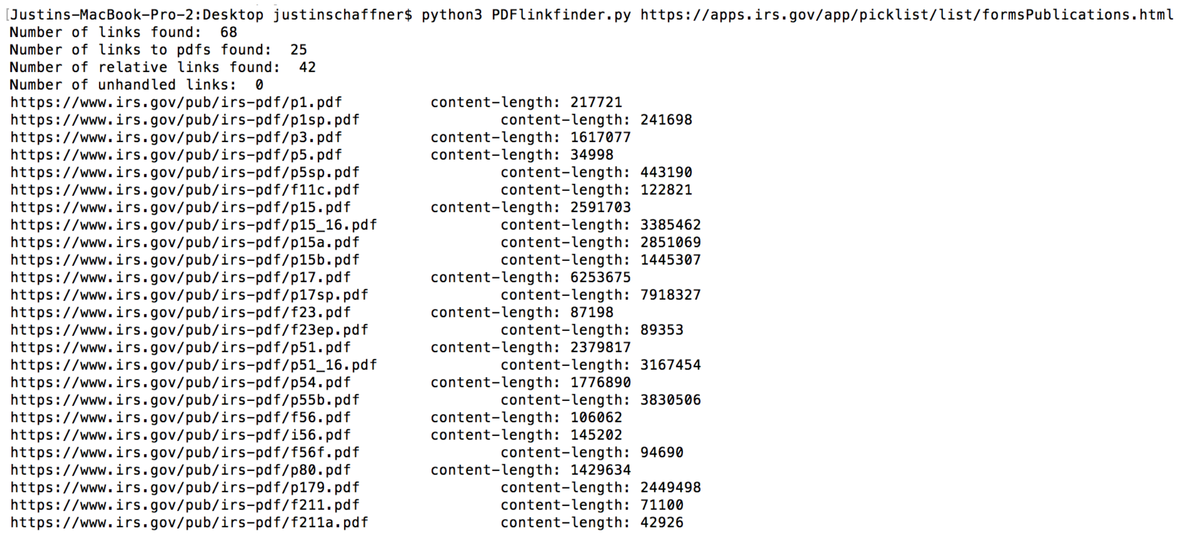
```
for i in linklist:
    if (i.startswith(”http://”) or i.startswith(”https://”)):
        tpage=requests.get(i, allow_redirects=True, stream=True)
        temp=tpage.headers.get('content-type')
        if "pdf" in temp:
            pdflinklist.append((i,tpage.headers.get('content-length
               ')))
            numpdfs+=1
        tpage.close
    else:
        try:
            tpage=requests.get((urljoin(address, i)),
               allow_redirects=True, stream=True)
            temp=page.headers.get('content-type')
            if "pdf" in temp:
                pdflinklist.append((i, tpage.headers.get('content-
                   length')))
                numpdfs+=1
        except Exception:
            unhandledlinks+=1
            pass
        rellinks+=1
print("Number of links to pdfs found: ", numpdfs)
print("Number of relative links found: ", rellinks)
print("Number of unhandled links: ", unhandledlinks)
```

Some of the links without full paths were relative, and could be handled with urljoin, but there was an email link on the page that was still throwing an exception, so I ignored it with the 'try' since it was unlikely to yeild a pdf. I also added a few print lines detailing how stuff was handled. the results for the HRT page:



```
Justins-MacBook-Pro-2:Desktop justinschaffner$ python3 PDFlinkfinder.py http://gohrt.com/route/
Number of links found:  96
Number of links to pdfs found:  2
Number of relative links found:  75
Number of unhandled links:  1
http://gohrt.com/wp-content/uploads/2009/11/Southside-SystemMapInt-Oct16-FINAL.pdf            content-length: 5338811
http://gohrt.com/wp-content/uploads/2009/11/Peninsula-SystemMapExt-Oct16-FINAL.pdf            content-length: 4563466
```

For the third run I used the IRS PUBS page. My initial run gave me an AttributeError for one of the elements in the link list. My interpretation was that one of the hrefs that Beautifulsoup found was empty. I fixed that with an if statement, but then got an error for SSL certification failure. For that, I encased the first if statement in a try as well and added one to the unhandled counter if it throws and exception. Once I got that all working, the results said there were no PDF's on that particular IRS page, so I followed one of the links to a list of FORMS and got the following results:



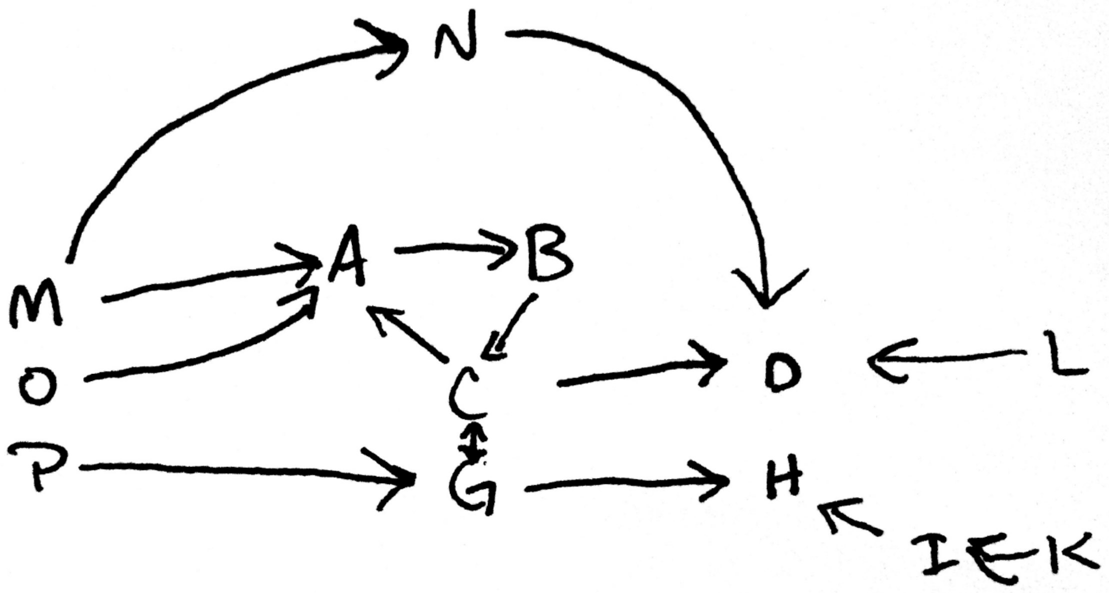
```
Justins-MacBook-Pro-2:Desktop justinschaffner$ python3 PDFlinkfinder.py https://apps.irs.gov/app/picklist/list/formsPublications.html
Number of links found:  68
Number of links to pdfs found:  25
Number of relative links found:  42
Number of unhandled links:  0
https://www.irs.gov/pub/irs-pdf/p1.pdf          content-length: 217721
https://www.irs.gov/pub/irs-pdf/p1sp.pdf                content-length: 241698
https://www.irs.gov/pub/irs-pdf/p3.pdf          content-length: 1617077
https://www.irs.gov/pub/irs-pdf/p5.pdf          content-length: 34998
https://www.irs.gov/pub/irs-pdf/p5sp.pdf                content-length: 443190
https://www.irs.gov/pub/irs-pdf/f11c.pdf                content-length: 122821
https://www.irs.gov/pub/irs-pdf/p15.pdf         content-length: 2591703
https://www.irs.gov/pub/irs-pdf/p15_16.pdf              content-length: 3385462
https://www.irs.gov/pub/irs-pdf/p15a.pdf                content-length: 2851069
https://www.irs.gov/pub/irs-pdf/p15b.pdf                content-length: 1445307
https://www.irs.gov/pub/irs-pdf/p17.pdf         content-length: 6253675
https://www.irs.gov/pub/irs-pdf/p17sp.pdf               content-length: 7918327
https://www.irs.gov/pub/irs-pdf/f23.pdf         content-length: 87198
https://www.irs.gov/pub/irs-pdf/f23ep.pdf               content-length: 89353
https://www.irs.gov/pub/irs-pdf/p51.pdf         content-length: 2379817
https://www.irs.gov/pub/irs-pdf/p51_16.pdf              content-length: 3167454
https://www.irs.gov/pub/irs-pdf/p54.pdf         content-length: 1776890
https://www.irs.gov/pub/irs-pdf/p55b.pdf                content-length: 3830506
https://www.irs.gov/pub/irs-pdf/f56.pdf         content-length: 106062
https://www.irs.gov/pub/irs-pdf/i56.pdf         content-length: 145202
https://www.irs.gov/pub/irs-pdf/f56f.pdf                content-length: 94690
https://www.irs.gov/pub/irs-pdf/p80.pdf         content-length: 1429634
https://www.irs.gov/pub/irs-pdf/p179.pdf                content-length: 2449498
https://www.irs.gov/pub/irs-pdf/f211.pdf                content-length: 71100
https://www.irs.gov/pub/irs-pdf/f211a.pdf               content-length: 42926
```

The final code for PDFlinkfinder.py

```python
import requests
from bs4 import BeautifulSoup
import lxml
import sys
from urllib.parse import urljoin


if not(len(sys.argv)==2):
    raise ValueError("Missing or multiple arguments")
address=sys.argv[1]
if not (address.startswith("http://") or address.startswith("https
    ://")):
    raise ValueError("Argument not an URI")
page=requests.get(address, allow_redirects=True)
soup=BeautifulSoup(page.content, "lxml")
linklist=[]
pdflinklist=[]
numlinks=0
numpdfs=0
rellinks=0
unhandledlinks=0;
for link in soup.find_all('a'):
    linklist.append(link.get('href'))
    numlinks+=1
print("Number of links found: ", numlinks)
for i in linklist:
    if not i:
        unhandledlinks+=1
        continue
    if (i.startswith("http://") or i.startswith("https://")):
        try:
            tpage=requests.get(i, allow_redirects=True, stream=True
                )
            temp=tpage.headers.get('content-type')
            if "pdf" in temp:
                pdflinklist.append((i,tpage.headers.get('content-
                    length')))
                numpdfs+=1
            tpage.close
        except Exception:
            unhandledlinks+=1
            pass
    else:
        try:
            tpage=requests.get((urljoin(address, i)),
                allow_redirects=True, stream=True)
            temp=page.headers.get('content-type')
            if "pdf" in temp:
                pdflinklist.append((i, tpage.headers.get('content-
                    length')))
                numpdfs+=1
        except Exception:
            unhandledlinks+=1
            pass
        rellinks+=1
print("Number of links to pdfs found: ", numpdfs)
print("Number of relative links found: ", rellinks)
print("Number of unhandled links: ", unhandledlinks)
for i in pdflinklist:
    print ("\\t\\tcontent-length: ".join(i))
```

Problem 3: Bowtie



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

'O M P' are all IN since they point into the SCC but nothing comes back out to them. 'A B C G' are SCC because they all connected both ways, even if it has to go around the loop to get there. 'D H' are both out, since data comes out from the SCC but does not go back in. 'I' and 'L' are both tendrils since they both point to nodes that are OUT. They're connected but their data never makes it into the SCC and SCC data never makes it to them. 'K' would be considered disconnected if 'I' wasn't a tendril, so I guess it just gets included as part of the 'I' tendril. 'E' and 'F' are disconnected. They have no relationships connecting them to other nodes.