Assignment #2

Wednesday, February 14TH, 2018

RANDY DO

Contents

Problem 1 2-3

Problem 2 4-6

Problem 3 7-8

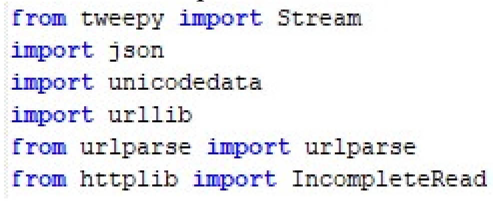
[Date]

Problem 1

```
1.  Write a Python program that extracts 1000 unique links from
Twitter. *Avoid Twitter sites.

This is run in Python 2.7
```

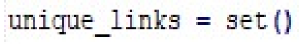
Answer:
I created a python code name: Getlink.py.
The first thing to do is:
Get
#consumer key, consumer secret, access token, access secret.
ckey = xxxx
csecret = xxx
atoken = xxxx
asecret = xxxx

I used these imports:



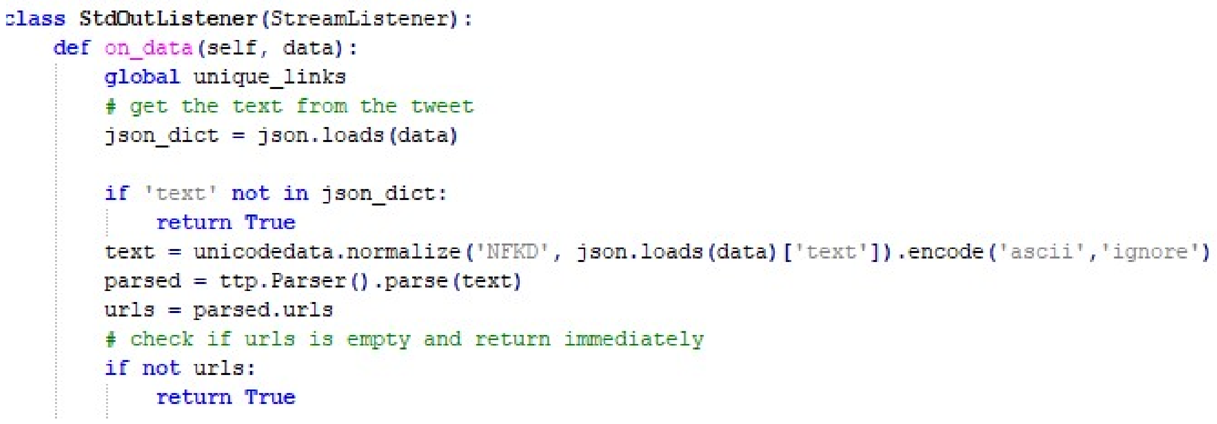
```python
from tweepy import Stream
import json
import unicodedata
import urllib
from urlparse import urlparse
from httplib import IncompleteRead
```

To prevent any duplicates URLs:



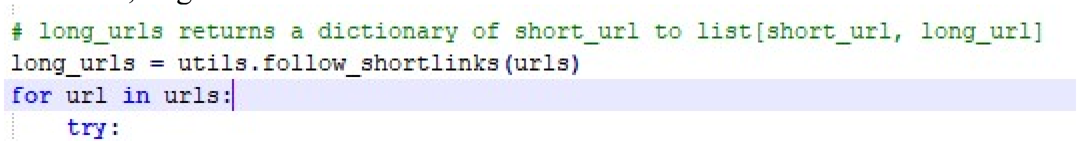
```python
unique_links = set()
```

To utilities tweepy: (I used stream and parsed the url)



```python
class StdOutListener(StreamListener):
    def on_data(self, data):
        global unique_links
        # get the text from the tweet
        json_dict = json.loads(data)

        if 'text' not in json_dict:
            return True
        text = unicodedata.normalize('NFKD', json.loads(data)['text']).encode('ascii','ignore')
        parsed = ttp.Parser().parse(text)
        urls = parsed.urls
        # check if urls is empty and return immediately
        if not urls:
            return True
```

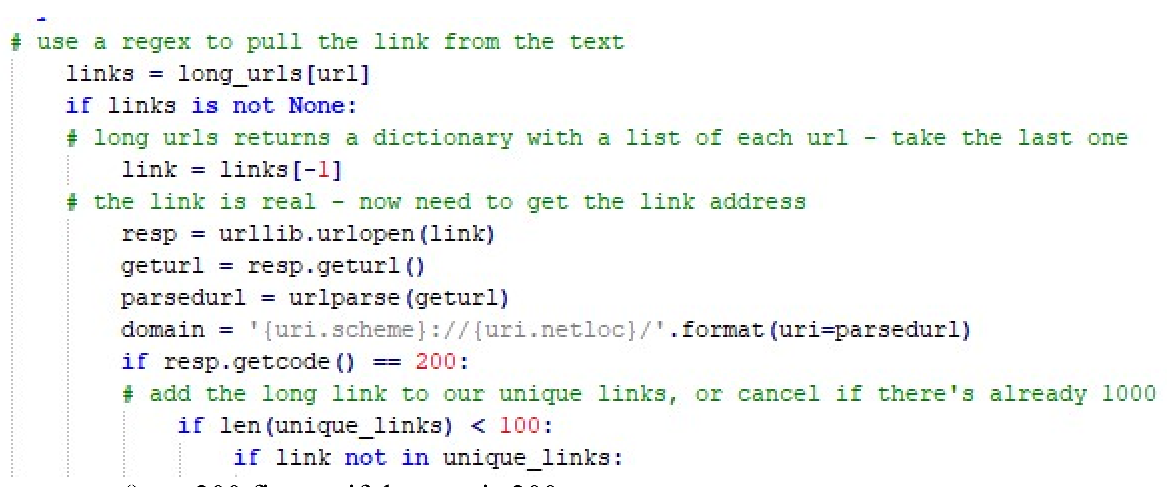
From this, to get rid of the shorter links



```python
# long_urls returns a dictionary of short_url to list[short_url, long_url]
long_urls = utils.follow_shortlinks(urls)
for url in urls:
    try:
```

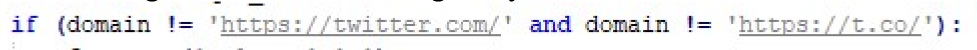
[Date]

Now for the obtaining links:



```
# use a regex to pull the link from the text
    links = long_urls[url]
    if links is not None:
    # long urls returns a dictionary with a list of each url - take the last one
        link = links[-1]
    # the link is real - now need to get the link address
        resp = urllib.urlopen(link)
        geturl = resp.geturl()
        parsedurl = urlparse(geturl)
        domain = '{uri.scheme}://{uri.netloc}/'.format(uri=parsedurl)
        if resp.getcode() == 200:
        # add the long link to our unique links, or cancel if there's already 1000
            if len(unique_links) < 100:
                if link not in unique_links:
```

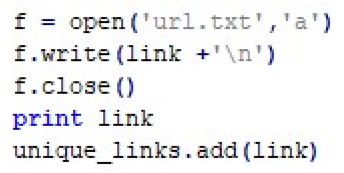
The resp.get() == 200 figures if the port is 200.
For this assignment, we are to not get any twitter URLs



```
if (domain != 'https://twitter.com/' and domain != 'https://t.co/'):
```

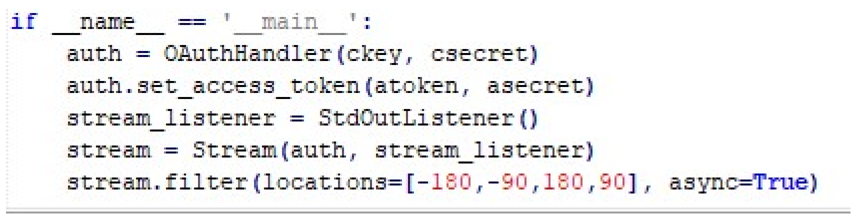
:

In this stream, once we obtain the unique link, we now write the files into a text file



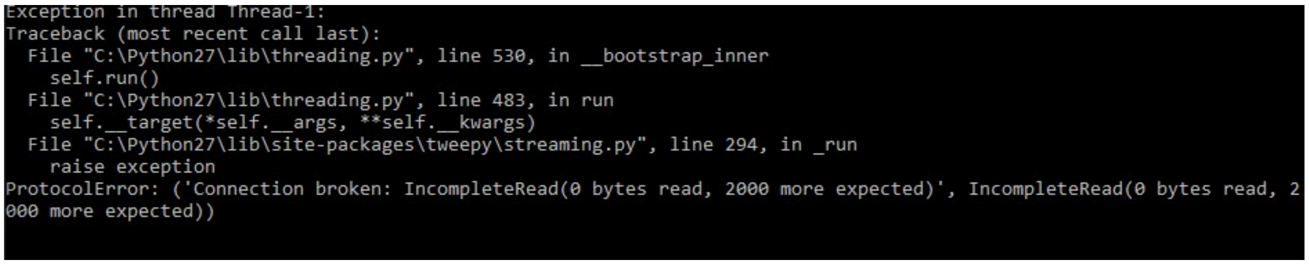
```
f = open('url.txt','a')
f.write(link +'\n')
f.close()
print link
unique_links.add(link)
```

Because this is class we need to call it from the main



```
if __name__ == '__main__':
    auth = OAuthHandler(ckey, csecret)
    auth.set_access_token(atoken, asecret)
    stream_listener = StdOutListener()
    stream = Stream(auth, stream_listener)
    stream.filter(locations=[-180,-90,180,90], async=True)
```

Issue:
I been getting this error whenever I reach between 20-50 links.
My assumption is that using the tweepy's streaming can't keep up with the amount of links it is getting from this program. Instead, what I did is ran it until I reached 1,000 links. Then used a unique.py to find any duplicate links in url.txt and remove it

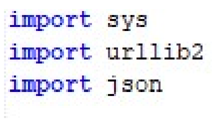


```
Exception in thread Thread-1:
Traceback (most recent call last):
  File "C:\Python27\lib\threading.py", line 530, in __bootstrap_inner
    self.run()
  File "C:\Python27\lib\threading.py", line 483, in run
    self.__target(*self.__args, **self.__kwargs)
  File "C:\Python27\lib\site-packages\tweepy\streaming.py", line 294, in _run
    raise exception
ProtocolError: ('Connection broken: IncompleteRead(0 bytes read, 2000 more expected)', IncompleteRead(0 bytes read, 2
000 more expected))
```

Problem 2:

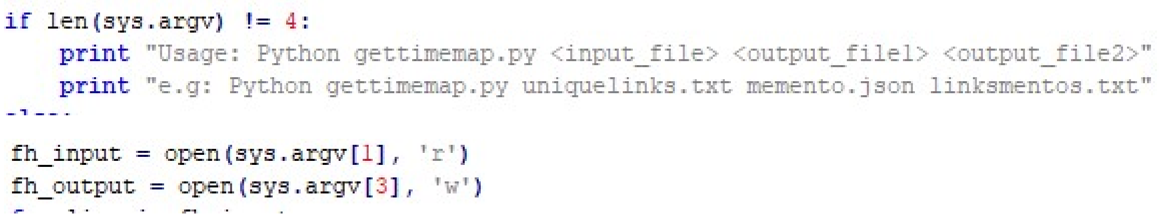
*I couldn't really understood this problem, so I asked a couple tutors.

For this, we used the import:



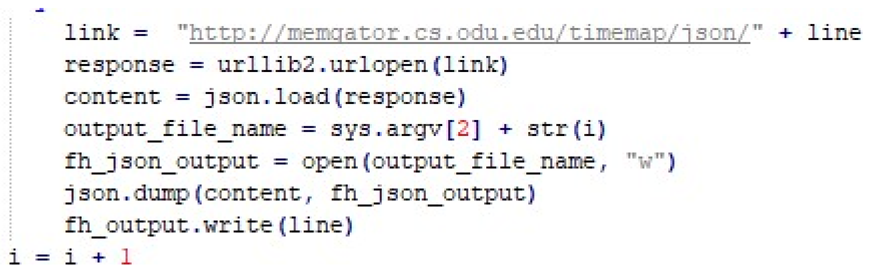
```
import sys
import urllib2
import json
```

In this, the output should be



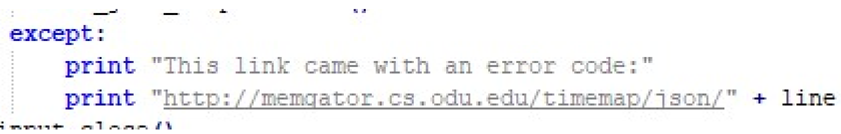
```
if len(sys.argv) != 4:
    print "Usage: Python gettimemap.py <input_file> <output_file1> <output_file2>"
    print "e.g: Python gettimemap.py uniquelinks.txt memento.json linksmentos.txt"

fh_input = open(sys.argv[1], 'r')
fh_output = open(sys.argv[3], 'w')
```

The uniquelinks.txt is what we are reading and linksmentos.txt is what we are writing in



```
        link =  "http://memgator.cs.odu.edu/timemap/json/" + line
        response = urllib2.urlopen(link)
        content = json.load(response)
        output_file_name = sys.argv[2] + str(i)
        fh_json_output = open(output_file_name, "w")
        json.dump(content, fh_json_output)
        fh_output.write(line)
    i = i + 1
```

I used memgator.cs.odu.edu/timemapp/json
if it has any text other than Error code: it would load and write it in the json file from sys.argv[2]

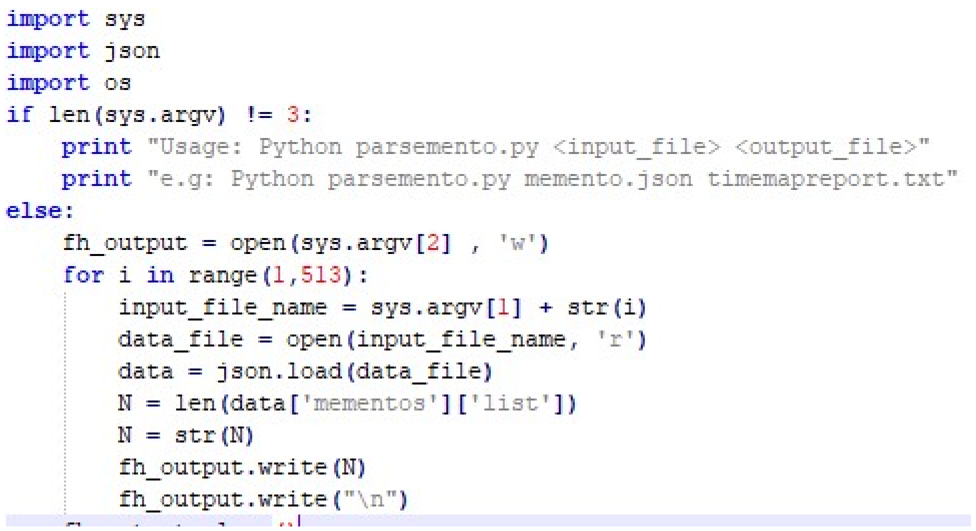


```
    except:
        print "This link came with an error code:"
        print "http://memgator.cs.odu.edu/timemap/json/" + line
```

Here I now have the json files that contains 'momentos'

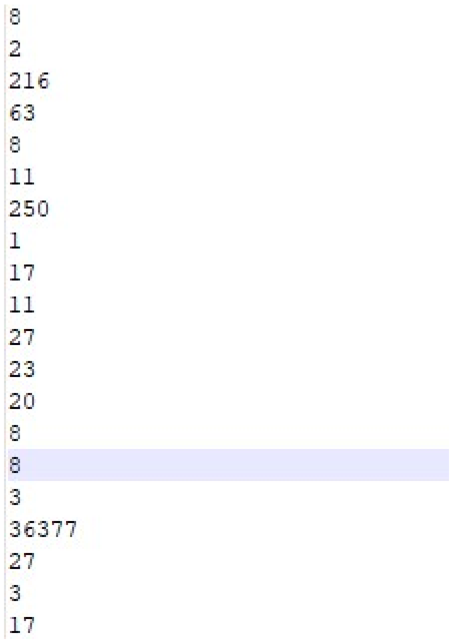
Now, I have to figure out how many mementos in each link.

[Date]



```
import sys
import json
import os
if len(sys.argv) != 3:
    print "Usage: Python parsemento.py <input_file> <output_file>"
    print "e.g: Python parsemento.py memento.json timemapreport.txt"
else:
    fh_output = open(sys.argv[2] , 'w')
    for i in range(1,513):
        input_file_name = sys.argv[1] + str(i)
        data_file = open(input_file_name, 'r')
        data = json.load(data_file)
        N = len(data['mementos']['list'])
        N = str(N)
        fh_output.write(N)
        fh_output.write("\n")
```

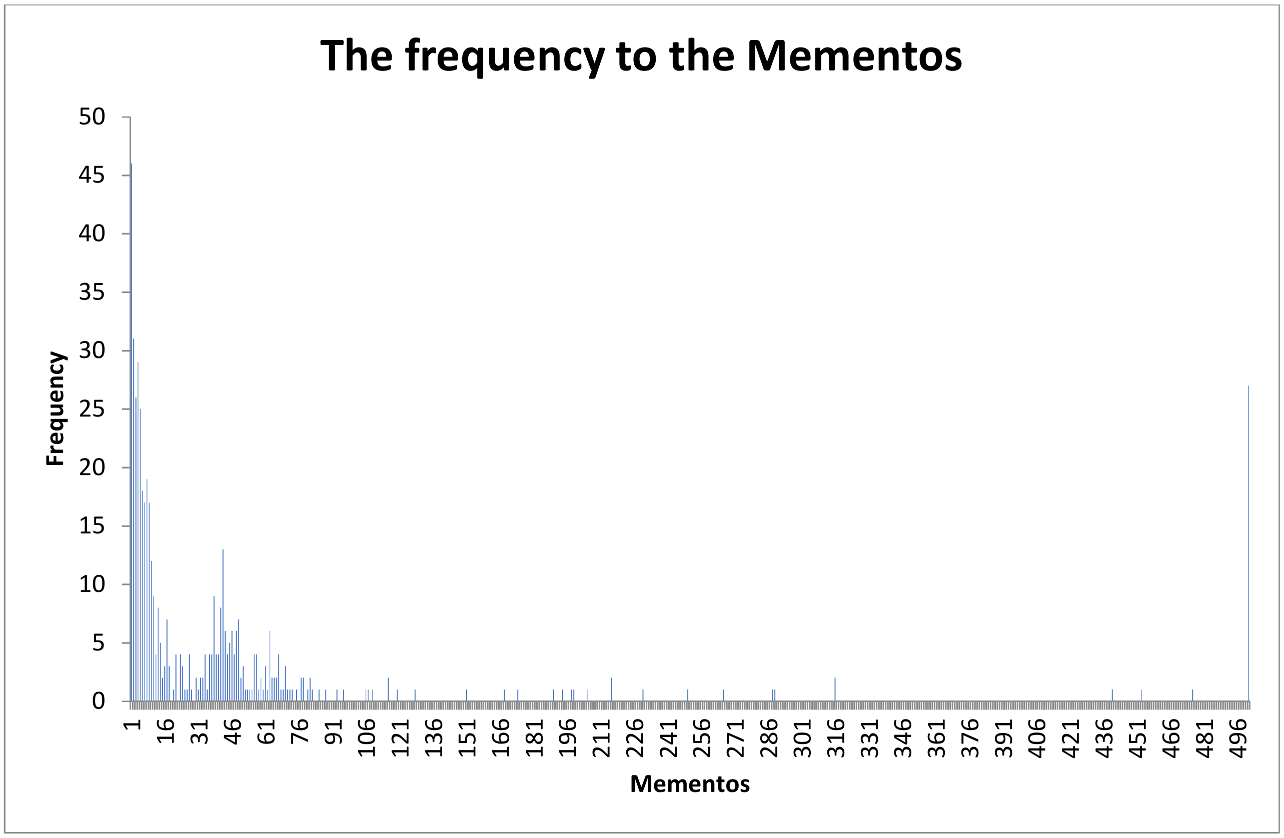
This figures out how many mementos does each link has



```
8
2
216
63
8
11
250
1
17
11
27
23
20
8
8
3
36377
27
3
17
```

Now, that I have the number of mementos of each link. I can create the histogram now.

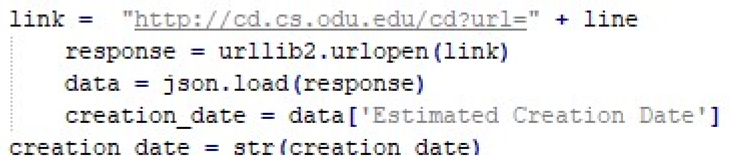
Because there are some that goes over 10,000, I am going to set a range of mementos to **500**



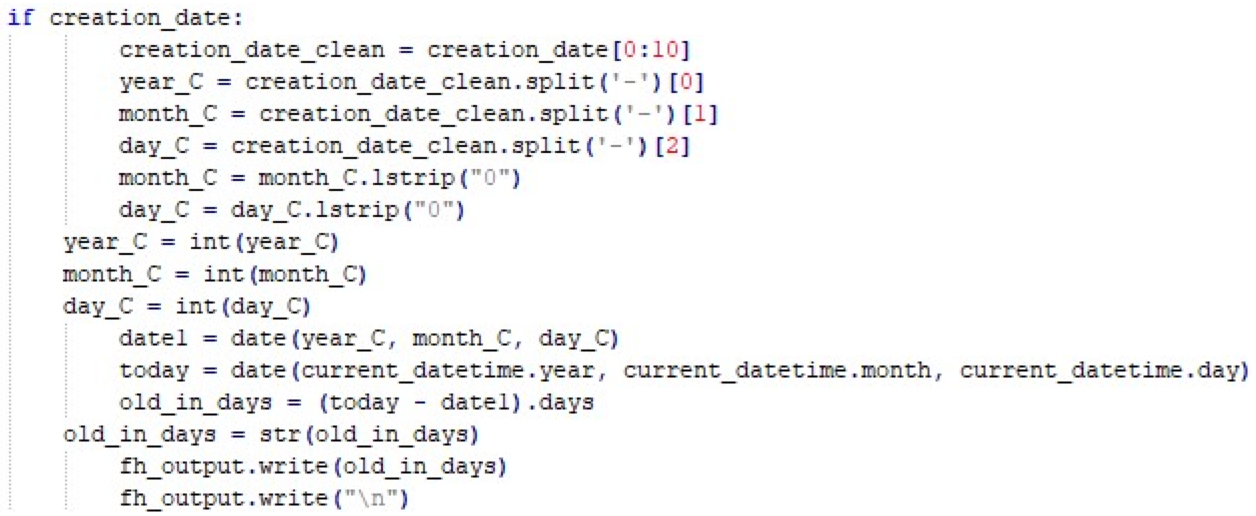
Pass 500, there are more frequency.

Problem 3

Unfortunately, I didn't have much time to work on this Part. I can see it as the same approach as the Problem 2.
This uses cd.cs.odu.edu/cd?url=
From the text file from I got from Problem one. It can get the creation date.



```
link =  "http://cd.cs.odu.edu/cd?url=" + line
    response = urllib2.urlopen(link)
    data = json.load(response)
    creation_date = data['Estimated Creation Date']
creation date = str(creation date)
```



```
if creation_date:
        creation_date_clean = creation_date[0:10]
        year_C = creation_date_clean.split('-')[0]
        month_C = creation_date_clean.split('-')[1]
        day_C = creation_date_clean.split('-')[2]
        month_C = month_C.lstrip("0")
        day_C = day_C.lstrip("0")
    year_C = int(year_C)
    month_C = int(month_C)
    day_C = int(day_C)
        date1 = date(year_C, month_C, day_C)
        today = date(current_datetime.year, current_datetime.month, current_datetime.day)
        old_in_days = (today - date1).days
    old_in_days = str(old_in_days)
        fh_output.write(old_in_days)
        fh_output.write("\n")
```

I couldn’t really figure what to do after to do this. Will work on this even if the assignment is past due.