RACHEL MCCREARY
CS 432 ASSIGNMENT 2 REPORT

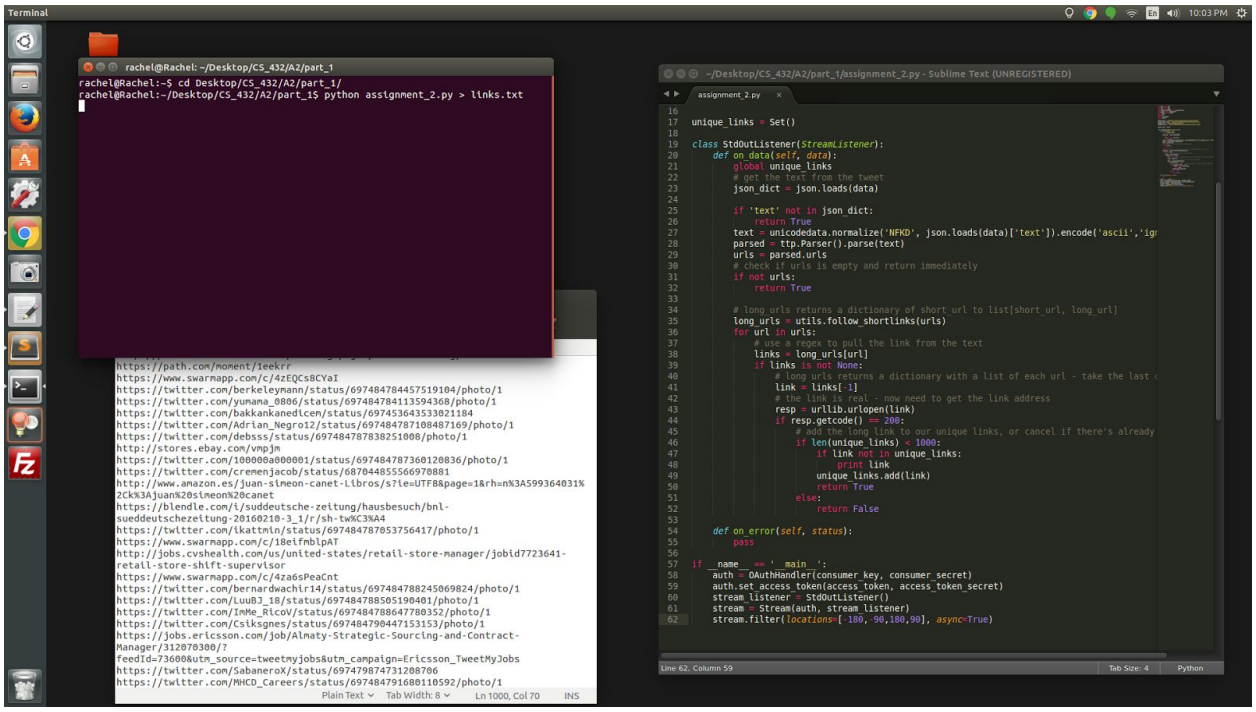
PART 1

This program extracts 1,000 unique links from Twitter. Tokens have been changed.

```
from sets import Set
from ttp import ttp
from ttp import utils
from tweepy.streaming import StreamListener
from tweepy import OAuthHandler
from tweepy import Stream
import json
import re
import unicodedata
import urllib

access_token = "XXXXXXX"
access_token_secret = "XXXXXXX"
consumer_key = "XXXXXXX"
consumer_secret = "XXXXXXX"
unique_links = Set()

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

                # long_urls returns a dictionary of short_url to list[short_url, long_url]
                long_urls = utils.follow_shortlinks(urls)
                for url in urls:
                        # 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)
                                if resp.getcode() == 200:
                                        # add the long link to our unique links, or cancel if there's already 1000
                                        if len(unique_links) < 1000:
                                                if link not in unique_links:
                                                        print link
                                                unique_links.add(link)
                                                return True
                                        else:
                                                return False
```

```
        def on_error(self, status):
                pass

if __name__ == '__main__':
        auth = OAuthHandler(consumer_key, consumer_secret)
        auth.set_access_token(access_token, access_token_secret)
        stream_listener = StdOutListener()
        stream = Stream(auth, stream_listener)
        stream.filter(locations=[-180,-90,180,90], async=True)
```

Demo screenshot:



When I first ran the program, it would spit out maybe 100 links into the file and quit, like this:

```
rachel@Rachel:~$ cd Desktop/python/
rachel@Rachel:~/Desktop/python$ python assignment_2.py > twitter_unique_links.txt
Exception in thread Thread-1:
Traceback (most recent call last):
  File "/usr/lib/python2.7/threading.py", line 810, in __bootstrap_inner
    self.run()
  File "/usr/lib/python2.7/threading.py", line 763, in run
    self.__target(*self.__args, **self.__kwargs)
  File "/home/rachel/.local/lib/python2.7/site-packages/tweepy/streaming.py", line 294, in _run
    raise exception
ProtocolError: ('Connection broken: IncompleteRead(0 bytes read, 512 more expected)', Incomplete
Read(0 bytes read, 512 more expected))

rachel@Rachel:~/Desktop/python$
```

To be honest, I only needed 1,000 links and was going to be working at my computer anyways, so I just ran it several more times in the background until I had my 1,000 links. If I needed more than that, I would have immediately tried to figure out how to run it automatically, but my thoughts were “get links now, learn how to automate later.”

I connected to Twitter Streaming API following steps 1 & 2 of Adil Moujahid’s Streaming API Tutorial here.

Libraries used: Set (used to ensure unique links)
ttp (for parsing text and following short links)
tweepy (for streaming)
json (tweet text returned as json objects)
unicodedata (unicode was annoying)
urllib (for opening links)

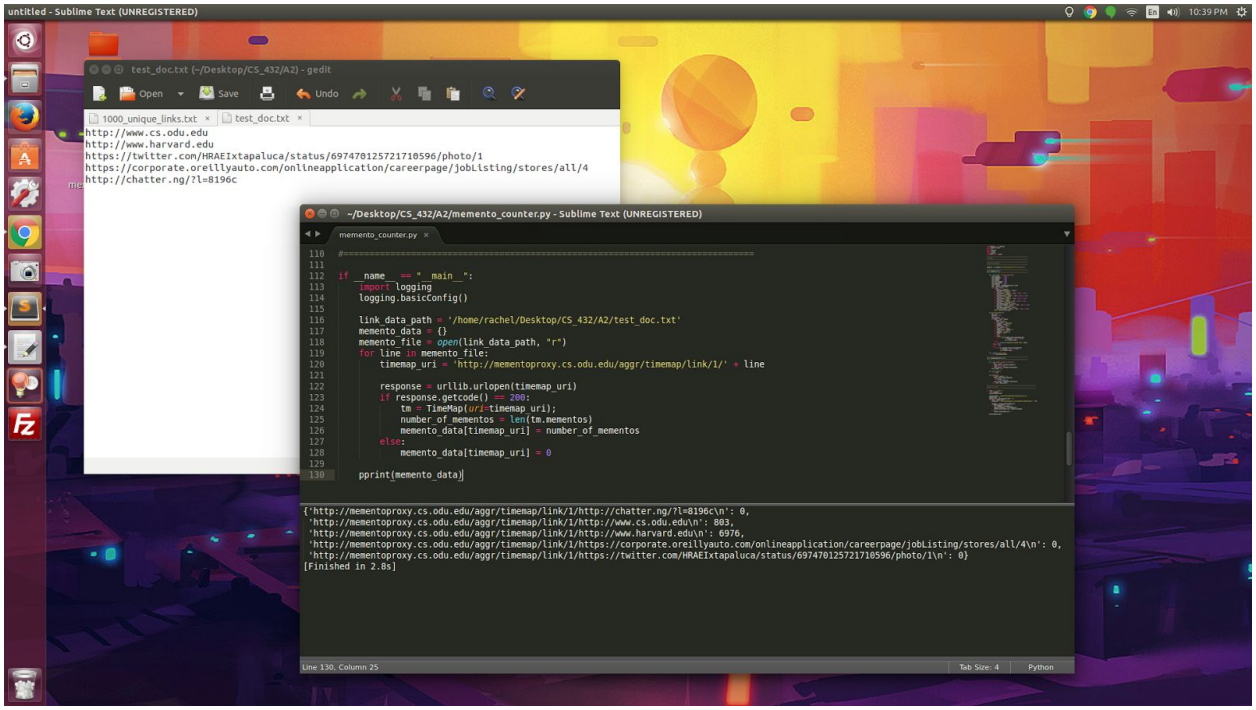
PART 2

I used James Chamberlain’s updated timemap parser.

The only changes I made to it were in the main function. This part takes a path to a text file and goes through it line by line, making key-value pairs containing the URI and the number of mementos for that URI.

```
if __name__ == "__main__":
        import logging
        logging.basicConfig()

        link_data_path = '/home/rachel/Desktop/CS_432/A2/test_doc.txt'
        memento_data = {}
        memento_file = open(link_data_path, "r")
        for line in memento_file:
                timemap_uri = 'http://mementoproxy.cs.odu.edu/aggr/timemap/link/1/' + line

                response = urllib.urlopen(timemap_uri)
                if response.getcode() == 200:
                        tm = TimeMap(uri=timemap_uri);
                        number_of_mementos = len(tm.mementos)
                        memento_data[timemap_uri] = number_of_mementos
                else:
                        memento_data[timemap_uri] = 0

        pprint(memento_data)
```

I gave it a path to a test file containing a much shorter chunk of links, since I was running out of time and just wanted to show how it worked:



The output is a set of URIs with their corresponding number of mementos. One URI had 803 mementos, another had 6976 mementos, and three of them had no mementos.

This is as far as I made it for this assignment before the due date. I'll keep chipping away at it anyways and make a histogram with my 1000 links just to see what happens, along with the carbon dating tool.