

## PART 1

This program goes through the facebook friend graph emailed to the class and prints the friend counts. I used [this](#) and [this](#) for guidance (stackoverflow questions).

Libraries used: ElementTree (for iterating over graphml file)  
re (for extracting only the ints from the friend count data)

Demo screenshot, including the full program:

The screenshot shows a terminal window on the left and a code editor on the right. The terminal displays the command to run the program and the resulting list of friend counts. The code editor shows the Python script that parses the graphml file and extracts the friend counts.

```
graph_parser.py
1 import xml.etree.ElementTree as ET
2 import re
3
4 if __name__ == "__main__":
5     tree = ET.parse('mln.graphml')
6     root = tree.getroot()
7
8     graphml = {
9         "graph": "http://graphml.graphdrawing.org/xmlns/graph",
10        "node": "http://graphml.graphdrawing.org/xmlns/node",
11        "data": "http://graphml.graphdrawing.org/xmlns/data",
12        "friend_count": "http://graphml.graphdrawing.org/xmlns/data[key='friend_count']"
13    }
14
15    graph = tree.find(graphml.get("graph"))
16    nodes = graph.findall(graphml.get("node"))
17
18    friend_counts = []
19
20    for node in nodes:
21        for data in node.findall(graphml.get("data")):
22            attribs[data.get("key") = "friend_count"] = data.text
23
24            friend_counts.append(map(int, re.findall('\d+', attribs[data.get("key") = "friend_count"])))
25
26    print friend_counts
```

```
friend_counts.txt
[244], [575], [421], [539], [784], [317], [448], [236], [561], [833], [752],
[763], [155], [195], [2], [555], [404], [242], [425], [366], [321], [1194], [259],
[427], [297], [400], [592], [424], [555], [97], [387], [0], [622], [337], [496],
[705], [819], [229], [1521], [324], [208], [619], [227], [3187], [351], [181],
[104], [295], [233], [348], [87], [1512], [8], [11], [190], [363], [62], [315],
[449], [186], [19], [380], [297], [359], [274], [245], [425], [562], [275], [240],
[510], [409], [245], [1626], [58], [89], [278], [30], [580], [197], [321], [276],
[68], [168], [182], [124], [233], [552], [1], [131], [615], [1346], [436], [770],
[322], [93], [0], [94], [106], [568], [170], [143], [128], [220], [312], [844],
[255], [420], [624], [204], [576], [524], [168], [873], [231], [68], [443], [65],
[3], [241], [86], [144], [2], [54], [172], [60], [250], [43], [183], [909], [94],
[38], [528], [17], [3], [40], [80], [108], [231], [458], [41], [42], [235], [327],
[15], [187], [207], [165], [7], [77], [308], [415], [111], [328], [123], [104],
[538], [147], [353], [59], [41], [96], [85], [25], [39]]
```

This next program computes the mean, standard deviation, and median of the number of friends that your friends have, using the [statistics](#) library. Before doing this, I used regex to get rid of the brackets:

The screenshot shows a terminal window on the left and a code editor on the right. The terminal displays the command to run the program and the resulting statistics. The code editor shows the Python script that reads the friend counts, removes brackets using regex, and calculates the mean, standard deviation, and median.

```
part_1_math.py
1 import statistics
2 import re
3
4 if __name__ == "__main__":
5     with open('friend_counts.txt') as f:
6         data = f.read()
7
8     data = map(int, re.findall('\d+', data))
9     print 'mean: ', statistics.mean(data)
10    print 'standard deviation: ', statistics.stdev(data)
11    print 'median: ', statistics.median(data)
```

```
clean_friend_counts.txt
244, 575, 421, 539, 784, 317, 448, 236, 561, 833, 752, 763, 155, 195, 2, 555, 404, 242, 425, 366, 321, 1194, 259, 427,
297, 400, 592, 424, 555, 97, 387, 0, 622, 337, 496, 705, 819, 229, 1521, 324, 208, 619, 227, 3187, 351, 181, 104, 295,
233, 348, 87, 1512, 8, 11, 190, 363, 62, 315, 449, 186, 19, 380, 297, 359, 274, 245, 425, 562, 275, 240, 510, 409, 245,
1626, 58, 89, 278, 30, 580, 197, 321, 276, 68, 168, 182, 124, 233, 552, 1, 131, 615, 1346, 436, 770, 322, 93, 0, 94, 106,
568, 170, 143, 128, 220, 312, 844, 255, 420, 624, 204, 576, 524, 168, 873, 231, 68, 443, 65, 3, 241, 86, 144, 2, 54, 172,
60, 250, 43, 183, 909, 94, 38, 528, 17, 3, 40, 80, 108, 231, 458, 41, 42, 235, 327, 15, 187, 207, 165, 7, 77, 308, 415,
111, 328, 123, 104, 538, 147, 353, 59, 41, 96, 85, 25, 39]
```

Update: There were a total of 165 friends in the original list. After reading a group email about some of these friends only showing a mutual friend count, I went back and deleted those from the list, saving them in a "deleted from list" file (there were 11 people), then ran the programs on the updated friend list:

The screenshot shows a terminal window on the left and a Sublime Text editor on the right. The terminal shows the command `python graph_parser.py > updated_friend_counts.txt` being executed. The Sublime Text editor shows the code for `graph_parser.py`, which uses the `xml.etree.ElementTree` module to parse an XML file and extract friend counts. The `updated_friend_counts.txt` file is open in a separate window, displaying a large list of numbers representing friend counts.

```

1 import xml.etree.ElementTree as ET
2 import re
3
4 if __name__ == "__main__":
5     tree = ET.parse('updated_mln.graphml')
6     root = tree.getroot()
7
8     graphml = {
9         "graph": "http://graphml.graphdrawing.org/xmlns/graph",
10        "node": "http://graphml.graphdrawing.org/xmlns/node",
11        "data": "http://graphml.graphdrawing.org/xmlns/data",
12        "friend_count": "http://graphml.graphdrawing.org/xmlns/data[key='friend_count']"
13    }
14
15    graph = tree.find(graphml.get("graph"))
16    nodes = graph.findall(graphml.get("node"))
17
18    friend_counts = []
19
20    for node in nodes:
21        attribs = {}
22        for data in node.findall(graphml.get("data")):
23            attribs[data.get("key") = "friend_count"] = data.text
24
25        friend_counts.append(map(int, re.findall('\d+', attribs[data.get("key") = "friend_count"])))
26
27    print friend_counts

```

The screenshot shows a terminal window on the left and a Sublime Text editor on the right. The terminal shows the command `python part_1_math.py > updated_clean_friends_count.txt` being executed. The Sublime Text editor shows the code for `part_1_math.py`, which uses the `statistics` module to calculate the mean, standard deviation, and median of the data. The `updated_clean_friends_count.txt` file is open in a separate window, displaying a large list of numbers representing the cleaned friend counts.

```

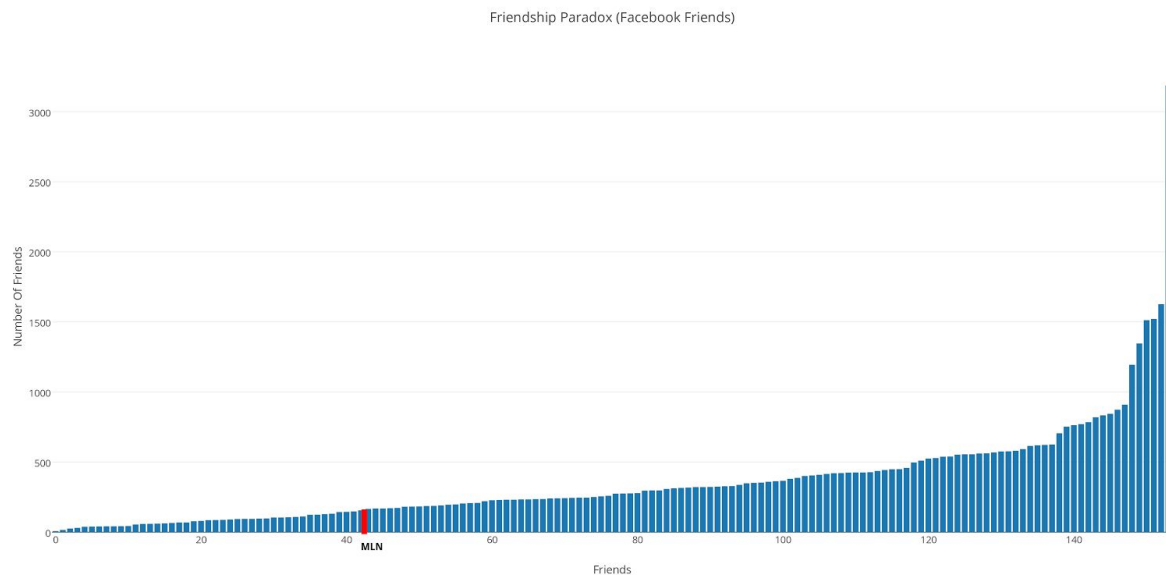
1 import statistics
2 import re
3
4 if __name__ == "__main__":
5     with open('updated_friend_counts.txt') as f:
6         data = f.read()
7         data = map(int, re.findall('\d+', data))
8
9     #print data
10    print 'mean: ', statistics.mean(data)
11    print 'standard deviation: ', statistics.stdev(data)
12    print 'median: ', statistics.median(data)

```

After using the updated friend list:

mean: 358.987012987  
 standard deviation: 371.585298171  
 median: 266.5

Using [plot.ly](https://plot.ly) to graph, MLN (165 friends) is in red:



## PART 2

Getting the followers of followers for [phonedude\\_mln](#), using same tweepy library from assignment 2 with Twitter credentials (I just noticed it says “import time” in the screenshot, but I ended up not using it.):

```
rachel@Rachel: ~/Desktop/CS_432/A4/part 2
rachel@Rachel:~/Desktop/CS_432/A4/part 2$ python part_2.py
Followers: 491
Shaaban_Migo 4
KevinClemmons 52
knnmyn 102
mgome0072 6
majetisiri 7
futuresma 684
dhomb3 10
SIGIR17 188
lmaccork 546
bltfield 73177
LarryWilson1942 38
JayNuetron 1079
wospworkshop 28
NetLab_dk 45
SalimChemlal 34
JonathanBrendle 88
EmeraldIKM 597
ShathaJY 1221
CrowNaito 0
EmeraldLibrary 689
clem_kev 25
diglarchiveteam 131
9ulovesu 53
manoj_chandra_k 8
```

```
~/Desktop/CS_432/A4/part 2/part_2.py - Sublime Text (UNREGISTERED)
part_1_math.py x part_2.py x
1 import tweepy
2 from tweepy import OAuthHandler
3 import time
4
5 # Authentication details. To obtain these visit dev.twitter.com
6 access_token = "4"
7 access_token_secret = " "
8 consumer_key = " "
9 consumer_secret = " "
10
11 if __name__ == '__main__':
12     # Create authentication token
13     auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
14     auth.set_access_token(access_token, access_token_secret)
15     api = tweepy.API(auth)
16
17     data = api.get_user('phonedude_mln')
18     print 'Followers: ' + str(data.followers_count)
19
20     for user in tweepy.Cursor(api.followers, screen_name="phonedude_mln", count = 200).items():
21         print user.screen_name, user.followers_count
22
```



[illegible]

```
mean: 169270.835951
standard deviation: 3109535.51029
median: 231
```

Using [plot.ly](https://plot.ly) to graph, MLN (491 followers) is in red:

