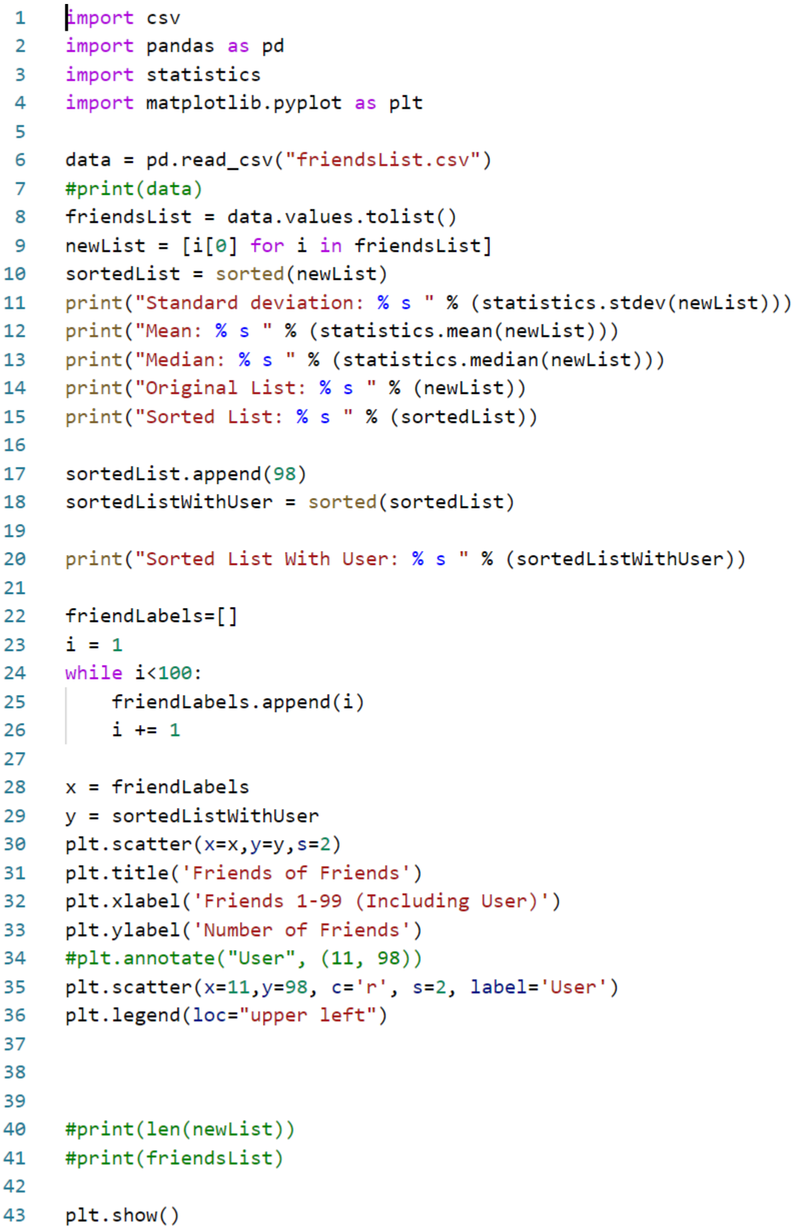
HW4 CS432

Ondra Torkilson

October 25, 2020

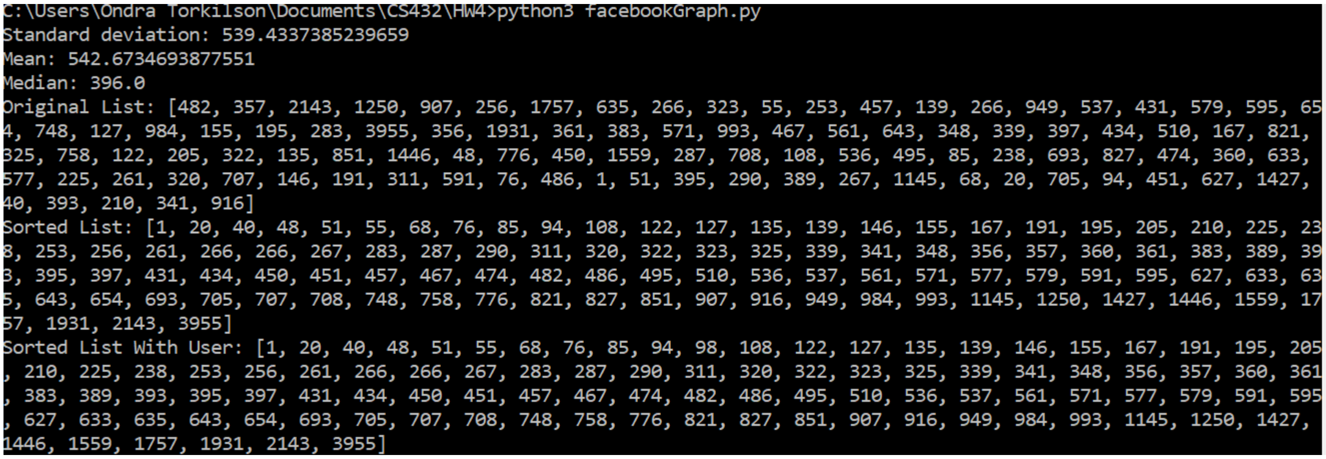
Question 1

For question one, I wrote a python script to import the data from the CSV. I put it into a list, sorted it, and then calculated the statistics and displayed them to the console. You can see these results in Figure 2. The standard deviation was 539.43, the mean was 542.67, and the median was 396.0. After computing the statistics, I inserted the user into the list and sorted it. In order to create the scatter plot, I needed the x values which I created using a for loop seen in lines 22-26. Then I scripted the scatter plot to show the data which you can see in Figure 3. The user's item is highlighted red, and the legend in the upper left conveys that. The friendship paradox did hold in this case, as the user has only 98 friends, but the mean number of friends was 542.



```python
import csv
import pandas as pd
import statistics
import matplotlib.pyplot as plt

data = pd.read_csv("friendsList.csv")
#print(data)
friendsList = data.values.tolist()
newList = [i[0] for i in friendsList]
sortedList = sorted(newList)
print("Standard deviation: % s " % (statistics.stdev(newList)))
print("Mean: % s " % (statistics.mean(newList)))
print("Median: % s " % (statistics.median(newList)))
print("Original List: % s " % (newList))
print("Sorted List: % s " % (sortedList))

sortedList.append(98)
sortedListWithUser = sorted(sortedList)

print("Sorted List With User: % s " % (sortedListWithUser))

friendLabels=[]
i = 1
while i<100:
    friendLabels.append(i)
    i += 1

x = friendLabels
y = sortedListWithUser
plt.scatter(x=x,y=y,s=2)
plt.title('Friends of Friends')
plt.xlabel('Friends 1-99 (Including User)')
plt.ylabel('Number of Friends')
#plt.annotate("User", (11, 98))
plt.scatter(x=11,y=98, c='r', s=2, label='User')
plt.legend(loc="upper left")



#print(len(newList))
#print(friendsList)

plt.show()
```

Figure 1: facebookGraph.py



```
C:\Users\Ondra Torkilson\Documents\CS432\HW4>python3 facebookGraph.py
Standard deviation: 539.4337385239659
Mean: 542.6734693877551
Median: 396.0
Original List: [482, 357, 2143, 1250, 907, 256, 1757, 635, 266, 323, 55, 253, 457, 139, 266, 949, 537, 431, 579, 595, 65
4, 748, 127, 984, 155, 195, 283, 3955, 356, 1931, 361, 383, 571, 993, 467, 561, 643, 348, 339, 397, 434, 510, 167, 821,
325, 758, 122, 205, 322, 135, 851, 1446, 48, 776, 450, 1559, 287, 708, 108, 536, 495, 85, 238, 693, 827, 474, 360, 633,
577, 225, 261, 320, 707, 146, 191, 311, 591, 76, 486, 1, 51, 395, 290, 389, 267, 1145, 68, 20, 705, 94, 451, 627, 1427,
40, 393, 210, 341, 916]
Sorted List: [1, 20, 40, 48, 51, 55, 68, 76, 85, 94, 108, 122, 127, 135, 139, 146, 155, 167, 191, 195, 205, 210, 225, 23
8, 253, 256, 261, 266, 266, 267, 283, 287, 290, 311, 320, 322, 323, 325, 339, 341, 348, 356, 357, 360, 361, 383, 389, 39
3, 395, 397, 431, 434, 450, 451, 457, 467, 474, 482, 486, 495, 510, 536, 537, 561, 571, 577, 579, 591, 595, 627, 633, 63
5, 643, 654, 693, 705, 707, 708, 748, 758, 776, 821, 827, 851, 907, 916, 949, 984, 993, 1145, 1250, 1427, 1446, 1559, 17
57, 1931, 2143, 3955]
Sorted List With User: [1, 20, 40, 48, 51, 55, 68, 76, 85, 94, 98, 108, 122, 127, 135, 139, 146, 155, 167, 191, 195, 205
, 210, 225, 238, 253, 256, 261, 266, 266, 267, 283, 287, 290, 311, 320, 322, 323, 325, 339, 341, 348, 356, 357, 360, 361
, 383, 389, 393, 395, 397, 431, 434, 450, 451, 457, 467, 474, 482, 486, 495, 510, 536, 537, 561, 571, 577, 579, 591, 595
, 627, 633, 635, 643, 654, 693, 705, 707, 708, 748, 758, 776, 821, 827, 851, 907, 916, 949, 984, 993, 1145, 1250, 1427,
1446, 1559, 1757, 1931, 2143, 3955]
```

Figure 2: Q1 Facebook Statistics

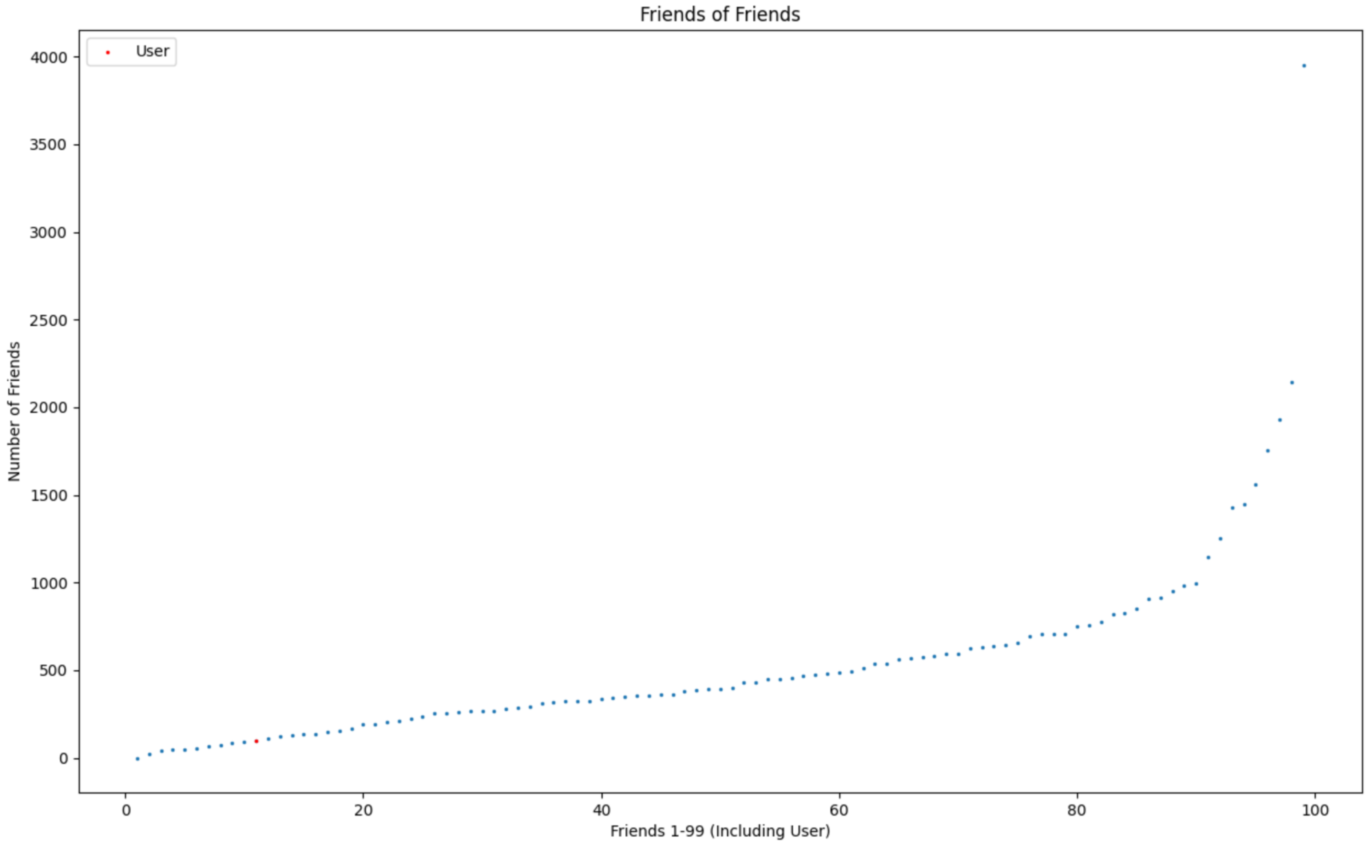
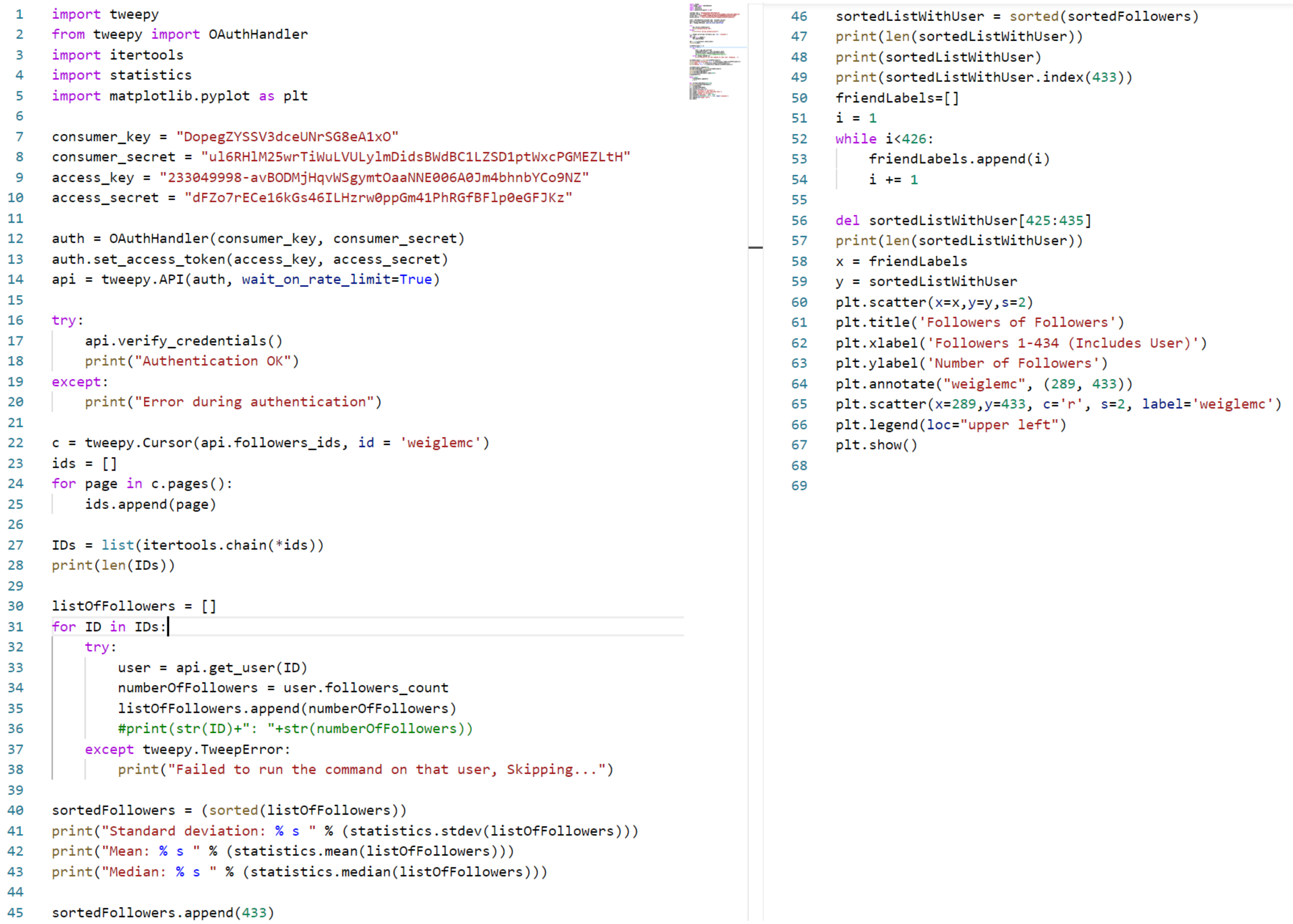


Figure 3: Facebook Friends

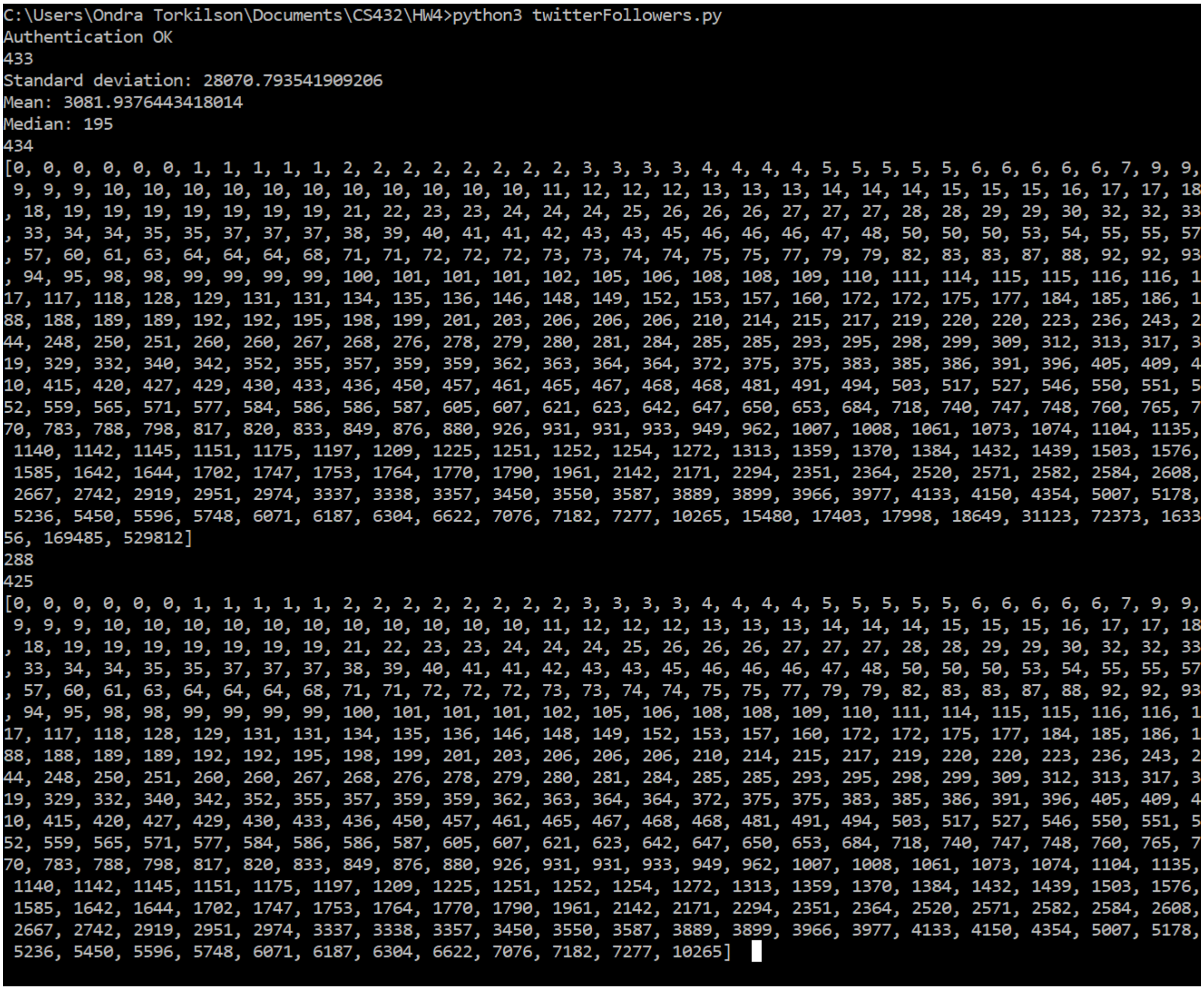
Questions 2

I wrote another python script to obtain the followers of weiglemc on Twitter as seen in Figure 4. Using the Twitter API made it simply a matter of accessing the cursor on the user object for weiglemc and returning the followers' IDs as a list (lines 22-25). Then, I formatted the list of follower IDs to be parsed more easily by python using the itertools module chain function (line 27). Next, I accessed each follower's user object and obtained their followers_count, and appended that to a new list (lines 33-38). That was everything needed to perform the statistics seen in Figure 5, at the bottom of the command line output. The standard deviation was 28070.794, the mean was 3081.938, and the median was 195. After obtaining the statistics, I did the same as before with Q1, and I inserted the user's number of followers (433 on October 22, 2020), and then sorted the list. Also, I created the x values seen in lines 52-54. After plotting the data the first time, as seen in Figure 7, I decided to remove the outliers from the data which was easy, as the list was sorted. I removed indexes 426-434, which cut off at about 10,300 followers. This made the data in the graph easier to view, seen in Figure 6. The friendship paradox for weiglemc's twitter followers did not hold as strongly as in question one. It did still hold in regards to some of the average statistics, as the mean number of followers was over 3000, and weiglemc's followers was 433. Although, looking at the median number of followers, we see that weiglemc passes that average statistic by a large amount. It was only 195. Looking at the original graph as well, it shows that the user falls in the middle of the data, and has more followers than at least half of their own followers. In the ranking, weiglemc had more followers than 287 of their own followers, which contradicts the friendship paradox. To conclude, I believe that in the case of weiglemc's twitter followers, the friendship paradox did not occur, because the user had more followers than over half of their own followers.



```python
import tweepy
from tweepy import OAuthHandler
import itertools
import statistics
import matplotlib.pyplot as plt

consumer_key = "DopegZYSSV3dceUNrSG8eA1xO"
consumer_secret = "ul6RHlM25wrTiWuLVULylmDidsBWdBC1LZSD1ptWxcPGMEZLtH"
access_key = "233049998-avBODMjHqvWSgymtOaaNNE006A0Jm4bhnbYCo9NZ"
access_secret = "dFZo7rECe16kGs46ILHzrw0ppGm41PhRGfBFlp0eGFJKz"

auth = OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_key, access_secret)
api = tweepy.API(auth, wait_on_rate_limit=True)

try:
    api.verify_credentials()
    print("Authentication OK")
except:
    print("Error during authentication")

c = tweepy.Cursor(api.followers_ids, id = 'weiglemc')
ids = []
for page in c.pages():
    ids.append(page)

IDs = list(itertools.chain(*ids))
print(len(IDs))

listOfFollowers = []
for ID in IDs:
    try:
        user = api.get_user(ID)
        numberOfFollowers = user.followers_count
        listOfFollowers.append(numberOfFollowers)
        #print(str(ID)+": "+str(numberOfFollowers))
    except tweepy.TweepError:
        print("Failed to run the command on that user, Skipping...")

sortedFollowers = (sorted(listOfFollowers))
print("Standard deviation: % s " % (statistics.stdev(listOfFollowers)))
print("Mean: % s " % (statistics.mean(listOfFollowers)))
print("Median: % s " % (statistics.median(listOfFollowers)))

sortedFollowers.append(433)
sortedListWithUser = sorted(sortedFollowers)
print(len(sortedListWithUser))
print(sortedListWithUser)
print(sortedListWithUser.index(433))
friendLabels=[]
i = 1
while i<426:
    friendLabels.append(i)
    i += 1

del sortedListWithUser[425:435]
print(len(sortedListWithUser))
x = friendLabels
y = sortedListWithUser
plt.scatter(x=x,y=y,s=2)
plt.title('Followers of Followers')
plt.xlabel('Followers 1-434 (Includes User)')
plt.ylabel('Number of Followers')
plt.annotate("weiglemc", (289, 433))
plt.scatter(x=289,y=433, c='r', s=2, label='weiglemc')
plt.legend(loc="upper left")
plt.show()
```

Figure 4: twitter.py



```
C:\Users\Ondra Torkilson\Documents\CS432\HW4>python3 twitterFollowers.py
Authentication OK
433
Standard deviation: 28070.793541909206
Mean: 3081.9376443418014
Median: 195
434
[0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 7, 9, 9,
 9, 9, 9, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 11, 12, 12, 12, 13, 13, 13, 14, 14, 14, 15, 15, 15, 16, 17, 17, 18
, 18, 19, 19, 19, 19, 19, 19, 19, 21, 22, 23, 23, 24, 24, 24, 25, 26, 26, 26, 27, 27, 27, 28, 28, 29, 29, 30, 32, 32, 33
, 33, 34, 34, 35, 35, 37, 37, 37, 38, 39, 40, 41, 41, 42, 43, 43, 45, 46, 46, 46, 47, 48, 50, 50, 50, 53, 54, 55, 55, 57
, 57, 60, 61, 63, 64, 64, 64, 68, 71, 71, 72, 72, 72, 73, 73, 74, 74, 75, 75, 77, 79, 79, 82, 83, 83, 87, 88, 92, 92, 93
, 94, 95, 98, 98, 99, 99, 99, 99, 100, 101, 101, 101, 102, 105, 106, 108, 108, 109, 110, 111, 114, 115, 115, 116, 116, 1
17, 117, 118, 128, 129, 131, 131, 134, 135, 136, 146, 148, 149, 152, 153, 157, 160, 172, 172, 175, 177, 184, 185, 186, 1
88, 188, 189, 189, 192, 192, 195, 198, 199, 201, 203, 206, 206, 206, 210, 214, 215, 217, 219, 220, 220, 223, 236, 243, 2
44, 248, 250, 251, 260, 260, 267, 268, 276, 278, 279, 280, 281, 284, 285, 285, 293, 295, 298, 299, 309, 312, 313, 317, 3
19, 329, 332, 340, 342, 352, 355, 357, 359, 359, 362, 363, 364, 364, 372, 375, 375, 383, 385, 386, 391, 396, 405, 409, 4
10, 415, 420, 427, 429, 430, 433, 436, 450, 457, 461, 465, 467, 468, 468, 481, 491, 494, 503, 517, 527, 546, 550, 551, 5
52, 559, 565, 571, 577, 584, 586, 586, 587, 605, 607, 621, 623, 642, 647, 650, 653, 684, 718, 740, 747, 748, 760, 765, 7
70, 783, 788, 798, 817, 820, 833, 849, 876, 880, 926, 931, 931, 933, 949, 962, 1007, 1008, 1061, 1073, 1074, 1104, 1135,
 1140, 1142, 1145, 1151, 1175, 1197, 1209, 1225, 1251, 1252, 1254, 1272, 1313, 1359, 1370, 1384, 1432, 1439, 1503, 1576,
 1585, 1642, 1644, 1702, 1747, 1753, 1764, 1770, 1790, 1961, 2142, 2171, 2294, 2351, 2364, 2520, 2571, 2582, 2584, 2608,
 2667, 2742, 2919, 2951, 2974, 3337, 3338, 3357, 3450, 3550, 3587, 3889, 3899, 3966, 3977, 4133, 4150, 4354, 5007, 5178,
 5236, 5450, 5596, 5748, 6071, 6187, 6304, 6622, 7076, 7182, 7277, 10265, 15480, 17403, 17998, 18649, 31123, 72373, 1633
56, 169485, 529812]
288
425
[0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 3, 3, 3, 3, 4, 4, 4, 4, 5, 5, 5, 5, 5, 6, 6, 6, 6, 6, 7, 9, 9,
 9, 9, 9, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 10, 11, 12, 12, 12, 13, 13, 13, 14, 14, 14, 15, 15, 15, 16, 17, 17, 18
, 18, 19, 19, 19, 19, 19, 19, 19, 21, 22, 23, 23, 24, 24, 24, 25, 26, 26, 26, 27, 27, 27, 28, 28, 29, 29, 30, 32, 32, 33
, 33, 34, 34, 35, 35, 37, 37, 37, 38, 39, 40, 41, 41, 42, 43, 43, 45, 46, 46, 46, 47, 48, 50, 50, 50, 53, 54, 55, 55, 57
, 57, 60, 61, 63, 64, 64, 64, 68, 71, 71, 72, 72, 72, 73, 73, 74, 74, 75, 75, 77, 79, 79, 82, 83, 83, 87, 88, 92, 92, 93
, 94, 95, 98, 98, 99, 99, 99, 99, 100, 101, 101, 101, 102, 105, 106, 108, 108, 109, 110, 111, 114, 115, 115, 116, 116, 1
17, 117, 118, 128, 129, 131, 131, 134, 135, 136, 146, 148, 149, 152, 153, 157, 160, 172, 172, 175, 177, 184, 185, 186, 1
88, 188, 189, 189, 192, 192, 195, 198, 199, 201, 203, 206, 206, 206, 210, 214, 215, 217, 219, 220, 220, 223, 236, 243, 2
44, 248, 250, 251, 260, 260, 267, 268, 276, 278, 279, 280, 281, 284, 285, 285, 293, 295, 298, 299, 309, 312, 313, 317, 3
19, 329, 332, 340, 342, 352, 355, 357, 359, 359, 362, 363, 364, 364, 372, 375, 375, 383, 385, 386, 391, 396, 405, 409, 4
10, 415, 420, 427, 429, 430, 433, 436, 450, 457, 461, 465, 467, 468, 468, 481, 491, 494, 503, 517, 527, 546, 550, 551, 5
52, 559, 565, 571, 577, 584, 586, 586, 587, 605, 607, 621, 623, 642, 647, 650, 653, 684, 718, 740, 747, 748, 760, 765, 7
70, 783, 788, 798, 817, 820, 833, 849, 876, 880, 926, 931, 931, 933, 949, 962, 1007, 1008, 1061, 1073, 1074, 1104, 1135,
 1140, 1142, 1145, 1151, 1175, 1197, 1209, 1225, 1251, 1252, 1254, 1272, 1313, 1359, 1370, 1384, 1432, 1439, 1503, 1576,
 1585, 1642, 1644, 1702, 1747, 1753, 1764, 1770, 1790, 1961, 2142, 2171, 2294, 2351, 2364, 2520, 2571, 2582, 2584, 2608,
 2667, 2742, 2919, 2951, 2974, 3337, 3338, 3357, 3450, 3550, 3587, 3889, 3899, 3966, 3977, 4133, 4150, 4354, 5007, 5178,
 5236, 5450, 5596, 5748, 6071, 6187, 6304, 6622, 7076, 7182, 7277, 10265]
```

Figure 5: Q2 Twitter Statistics

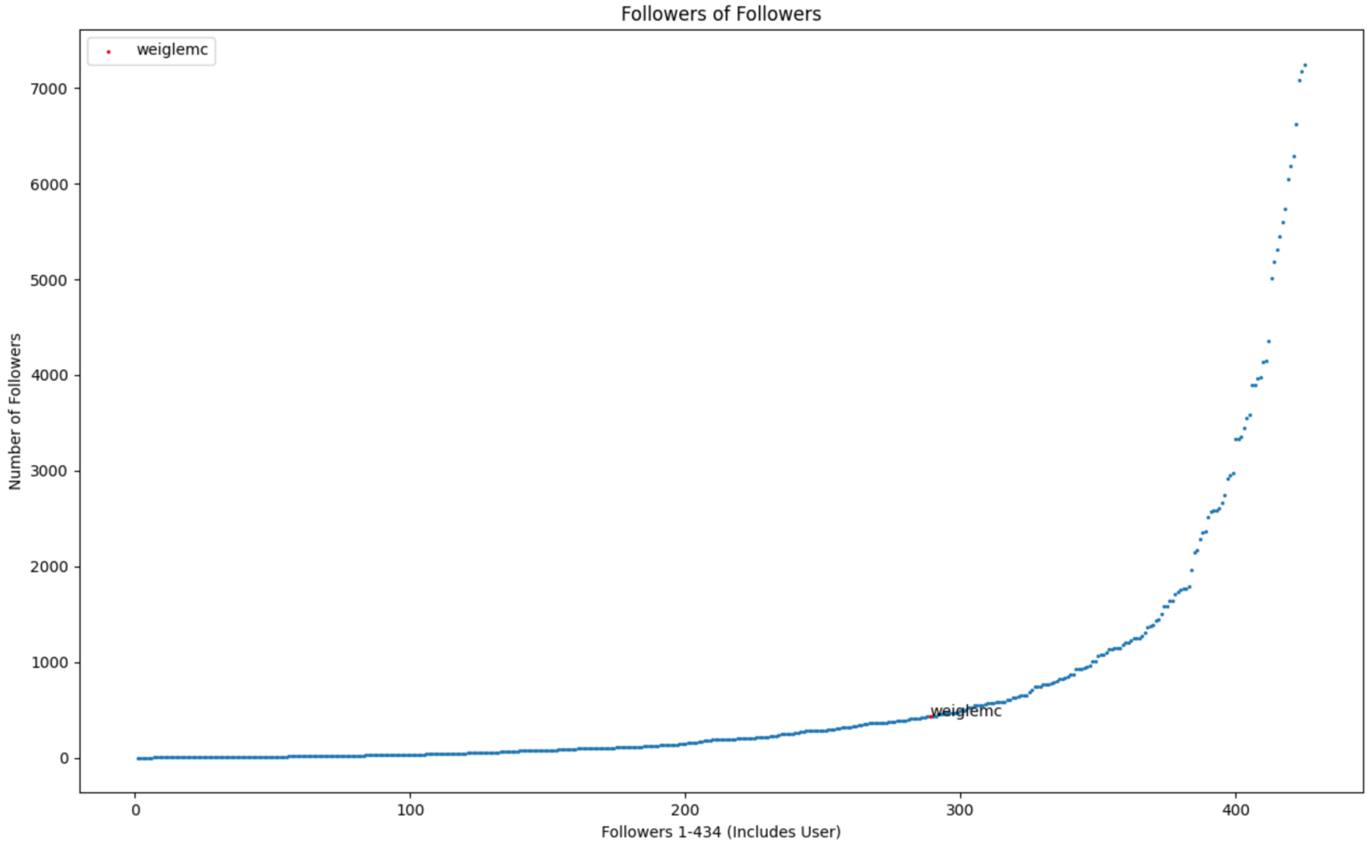


Figure 6: Twitter Followers (excludes outliers)

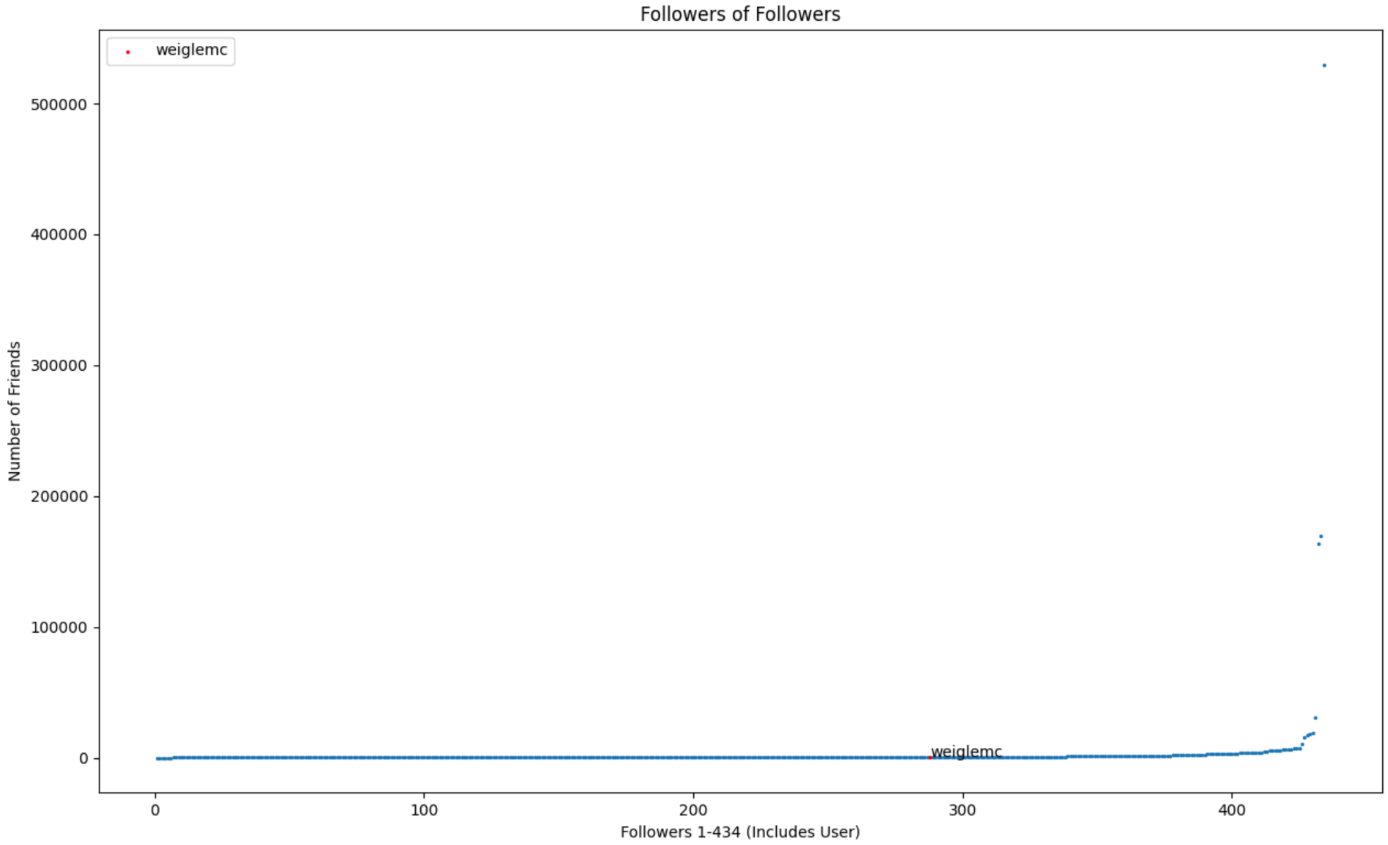


Figure 7: Twitter Followers