Assignment 4 (60 pts)

The goal of this project is to get familiar with networking/threading and starting web scraping.

Task 1: A little bit of networking (10pts)

Create a TCP server using the library socketserver using the following code:

```
import socketserver
import numpy as np
from time import sleep

class Handler_TCPServer(socketserver.BaseRequestHandler):
     def handle(self):
        try:
            mu, sigma = 1, 0.1
            s = np.random.normal(mu, sigma, 10)
            for i in s:
               self.request.sendall((str(i) + ' ').encode())
               sleep(1)
        except:
            pass

if __name__ == "__main__":
    HOST, PORT = "localhost", 9999
    tcp_server = socketserver.TCPServer((HOST, PORT), Handler_TCPServer)
    tcp_server.serve_forever()
```

This server will send you 10 random values of a normal distribution for each connection.

Your mission, should you choose to accept it is to find the standard deviation and the mean of this distribution with a TCP client using the smallest amount of time.

For this task, give the 'telnet' command you can use to get connected to the server. Give the output of this command (for those using windows, you can use telnet from this website: https://technet.microsoft.com/en-us/library/cc771275(v=ws.10).aspx).

Task 2: Create a TCP client (10 pts)

In this part, you will create a TCP client based on the code below to collect the data from the TCP server.

```
import socket
import statistics
import threading

host_ip, server_port = "127.0.0.1", 9999


    def work_with_server():
        global res_mean
        global res_stdev
        tcp_client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        try:
            tcp_client.connect((host_ip, server_port))
            while True:
                received = tcp_client.recv(1024)
                if not received: break
       # store received into a list
       finally:
            tcp_client.close()
```

hints: received needs to be casted into a float. Floating data received will be used to create a list with received numbers. When the data have been sent (after 10 data points), calculate the average and standard deviation.

Task 3: Create threaded TCP clients (20pts)

You can get concurrently more connections with threads.
You can create a global list to store the means/std deviation calculated by the different threads. You can use the following template:

```
import socket
import statistics
import threading

host_ip, server_port = "127.0.0.1", 9999

res_mean=[]
res_stdev=[]
thread_list=[]


class tcp_client(threading.Thread):
    def __init__(self,offset):
        threading.Thread.__init__(self)
```

```
        self.offset=offset
    def work_with_server(self):
         # Code this part
    def run(self):
        self.work_with_server()


thread_number=#pick the number of threads you would like to use
res_mean can be list to store all the means
res_stdev can be list to store all the standard deviation

for i in range(0,thread_number):
    thread_list.append(tcp_client(i))

for i in range(0,thread_number):
    thread_list[i].start()

for i in range(0,thread_number):
    thread_list[i].join()

print('RESULT')
#print the mean of all the means
#print the mean of all the std deviations
```

Averaging the 2 lists, you will get a more accurate result for the mean and the standard deviation of the distribution from the server.

Task 4: Introduction to web scraping (20pts)

Parse the number of hyperlink you can find from the site stackoverflow.com. You can use the function count.

```
import socket
request = b"GET / HTTP/1.1\nHost: stackoverflow.com\n\n"
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(("stackoverflow.com", 80))
s.send(request)
while True:
    result = s.recv(512)
    if ( len(result) < 1 ) :
        break
    #count the number of hyperlink
```

You will also use the library urlib and count the number of words.

```
import urllib.request
with urllib.request.urlopen("http://www.stackoverflow.com") as url:
    s = url.read()
```