```
# Task 1: A little big of networking
# Create a TCP server

import socketserver # diff in p2.7
import numpy as np
from time import sleep
import socket
import statistics
import threading

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()

# In command prompt enter    telnet 127.0.0.1 9999
# Output from Client(command prompt)
#    1.00094400388 0.895617696392 1.09301557998 1.10805728882 0.955695804086
#    0.986706074736 1.06383527808 0928869628808 1.02471673176
#    Connection to host lost.
```

```
# Task 2: Create a TCP client

import statistics as st
import socket
# set host and port, create data as an empty list
host_ip, server_port = "127.0.0.1", 9999
data =[]

# function to create a client
def work_with_server():
    global data
    global res_mean
    global res_stdev
    # creates tcp_client as a socket object
    tcp_client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    try:
        # connects using host and port
```

```
        tcp_client.connect((host_ip, server_port))
        while True:
            # stores recieved data in variable received
            received = tcp_client.recv(1024)
            # will break loop if data stops
            if not received:
                break
            # appends received as a float into the list data
            data.append(float(received))

    # closes the connection and prints the list
    finally:
        tcp_client.close()
        print(data)

# calls work_with_server function, print the mean and standard deviation of the list data
work_with_server()
print(st.mean(data))
print(st.stdev(data))


#Output:

# C:\Users\JohntheGreat\Anaconda3\python.exe
C:/Users/JohntheGreat/Documents/MSCA/Python3forStreamingAnalytics/Week4/Assignment4_Task2.p
y
# [1.01641677, 0.915754791165, 0.759171955394, 0.949588610492, 0.939346036072, 1.12396555513,
1.08119365413, 0.900009853287, 0.965018654557, 1.05002770126]
# 0.9700493581487
# 0.10450894585239735
#
# Process finished with exit code 0
```

```
# Task 3: Create threaded TCP clients

import threading
import statistics as st
import socket
# set the tuple for host and port
host_ip, server_port = "127.0.0.1", 9999
# create empty lists for res_mean, res_stdev and thread_list
res_mean=[]
res_stdev=[]
thread_list=[]
# create subclass TcpClient of parent class threading.Thread
class TcpClient(threading.Thread):
    def __init__(self, offset):
        threading.Thread.__init__(self)
        self.offset=offset  # int between 0 and 5
```

```
    # This is my function from task #2
    def work_with_server(self):
        #create empty list,  set tcp_client as socket object
        data = []
        global res_mean
        global res_stdev
        tcp_client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

        # This will store the recieved data in variable received
        try:
            tcp_client.connect((host_ip, server_port))
            while True:
                received = tcp_client.recv(1024)
                #it will break when the information stops
                if not received:
                    break
                # this appends received as a float into data list
                data.append(float(received))

        finally:
            # takes the mean of data and appends into res_mean
            res_mean.append(st.mean(data))
            # takes the standard deviation of data and appends into res_stdev
            res_stdev.append(st.stdev(data))
            # ends the connection
            tcp_client.close()
            print(data)

        return data


    def run(self):
        self.work_with_server()

# Set number of threads to 5
thread_number = 5

# calls the function for each thread, appends into thread_list
for i in range(0,thread_number):
    thread_list.append(TcpClient(i))

# starts the threads
for i in range(0, thread_number):
    thread_list[i].start()

# makes the threads wait
for i in range(0, thread_number):
    thread_list[i].join()

# shows that res_mean has 5 means from each thread
print(res_mean)
```

```
# print the mean of res_mean and the mean of res_stdev
print(st.mean(res_mean))
print(st.mean(res_stdev))

# Output:
# C:\Users\JohntheGreat\Anaconda3\python.exe
C:/Users/JohntheGreat/Documents/MSCA/Python3forStreamingAnalytics/Week4/Assignment4_Task3.p
y
#
# [0.9962661781696, 0.9860156937055999, 0.9893152328904, 0.9647682937555, 1.0038831954592]
# 0.98804971879606
# 0.10806280301072893
#
# Process finished with exit code 0
```

```
# Task 4: Introdcution to web scraping

import socket
# will be used to get the byte data from the site stackoverflow.com
request = b"GET / HTTP/1.1\nHost: stackoverflow.com\n\n"
# create a socket object: s
s =  socket.socket(socket.AF_INET, socket.SOCK_STREAM)
#connects the socket to the site
s.connect(("stackoverflow.com", 80))
# sends the request for the data
s.send(request)
# set tally count to 0
tally = 0

# while loop, stores the data received into result
while True:
    result = s.recv(512)
    # use count function on hyperlinks and aggregates each time into tally
    tally += result.count(b'http')
    # will break the loop
    if len(result) < 1:
        break
    # returns the tally count each time, final number is the final number of hyperlinks on the page
    print(tally)

# Outcome:
# final line is 89
```

```
# Task 4b: Count the words

# Alot of this code is from
# http://stackoverflow.com/questions/1936466/beautifulsoup-grab-visible-webpage-text

import urllib.request # gets the information from a web page
```

```
from bs4 import BeautifulSoup # takes html and turns into bs object you can manipulate
import re # regular expressions is used to identify patterns

# reads the stackoverflow website and stores its html data into variable html
html = urllib.request.urlopen('http://www.stackoverflow.com').read()
# creating a beautiful soup object that still looks like html (has tags, etc)
soup = BeautifulSoup(html, 'html.parser')
# uses findAll texts to create a list of the texts of the page
texts = soup.findAll(text=True)

# Create a function that gets rid of style, script etc tags also uses regular expressions to clean the texts
def visible(element):
    if element.parent.name in ['style', 'script', '[document]', 'head', 'title']:
        return False
    elif re.match('<!--.*-->', str(element)):
        return False
    return True

#use the filter function to run visible on texts
visible_texts = filter(visible, texts)
# make visible_texts a list
l_visible_texts = list(visible_texts)
# use ''.join to join all the elements in the list
j_visible_texts = ''.join(l_visible_texts)
# use .split() to split the items and put back into a list
final_visible_texts = j_visible_texts.split()
# get the length of the list
print(len(final_visible_texts))

# Output:
#C:\Users\JohntheGreat\Anaconda3\python.exe
C:/Users/JohntheGreat/Documents/MSCA/Python3forStreamingAnalytics/Week4/Assignment4_Task4B.
py
#2876

#Process finished with exit code 0
```