Wade Moulton

10/12/2020

CS 4720 W01

Assignment 4 Sockets Report

Weblink: <http://studentweb.kennesaw.edu/~jmoulto2/>

**Assignment Description**

The purpose of the sockets assignment is to give an introduction to exchanging data between a server and a client. This assignment also uses a premade polynomials file that contains functions for evaluating a polynomial and a bisection. The source code for this assignment can be found at my weblink as well as the bottom of this assignment report.

**Assignment Objectives**

This assignment is broken into two main parts: the server and the client. The client holds the logic for sending requests and receiving responses from the server. The server holds the logic for receiving requests and sending responses for the client. The server is also always running so that it is able to receive incoming connections. In this model this is done through the use of an infinite loop.

The client creates a socket and connects to the server. Using the supplied testing strings list, the client iterates through each item, encodes the message, makes the request, receives the response, decodes the response, outputs the response, and then closes the connection.

The server also creates a socket and then opens up a listener. Once a connection and request are made, the server gathers the request, decodes the data, decides what to do with that data ( in the case of this assignment this is either evaluate, bisect, or nothing ), builds a response, encodes that response, sends that response back to the client, and then closes that connection.

**Source Code**

# server.py

# Name: Wade Moulton  
# Date: 10/11/2020  
# Class: CS 4720  
# Section: W01  
# Instructor: Dr. Sarah North  
# Assignment\_4-Sockets  
  
import socket  
from polynomials import evaluate, bisection  
  
# creating constant variables for the server port and address using localhost address  
SERVER\_PORT = 12345  
SERVER\_ADDR = '127.0.0.1'  
  
# create globals for various errors that might be encountered  
ARG\_EXCEPTION = "XToo few arguments"  
VALUE\_EXCEPTION = "XCould not convert string to float"  
REQUEST\_EXCEPTION = "XIncorrect command type"  
TOLERANCE\_EXCEPTION = "XInvalid tolerance"  
  
# create socket and start listening  
listener = socket.socket()  
listener.bind((SERVER\_ADDR, SERVER\_PORT))  
listener.listen(0)  
print("Server started. Listening on " + SERVER\_ADDR + ":" + str(SERVER\_PORT))  
  
  
# infinite loop so that server can always accept connections  
while True:  
 # assign and print client connection info  
 (sock, address) = listener.accept()  
 print(f"Received connection from: {address}")  
 # assign encoded data to variable and create a new variable to store decode data  
 encoded\_data = sock.recv(2048)  
 client\_data = ""  
  
 # iterate and decode through data and assign to decoded variable  
 while len(encoded\_data) > 0:  
 client\_data += encoded\_data.decode()  
 encoded\_data = sock.recv(2048)  
  
 try:  
 # print received data and split data into list  
 print(f"RECV: {client\_data}")  
 client\_data = client\_data.split(" ")  
 if len(client\_data) < 1:  
 raise Exception(ARG\_EXCEPTION)  
  
 # splitting first value to get the command and storing in request\_start  
 request = client\_data[0]  
 request\_start = request[0]  
  
 # if request is for evaluation function  
 if request\_start == "E":  
 if len(client\_data) < 2:  
 raise Exception(ARG\_EXCEPTION)  
  
 # evaluation function expects an x value and a polynomial list  
 # first convert x\_value to float  
 x\_value = float(request[1:])  
 # convert remaining list values to floats for the polynomial values  
 poly\_values = [float(i) for i in client\_data[1:]]  
 # call function evaluate and assign return to response  
 response = f"E{evaluate(x\_value, poly\_values)}"  
  
 # if request is for bisection function  
 elif request\_start == "S":  
 if len(client\_data) < 3:  
 raise Exception(ARG\_EXCEPTION)  
  
 # bisection function expects a value, b value, polynomial list, and tolerance value  
 # first convert a and b values to floats  
 a\_value = float(request[1:])  
 b\_value = float(client\_data[1])  
 # convert remaining list values to floats for the polynomial values except the last value  
 poly\_values = [float(i) for i in client\_data[2:len(client\_data) - 1]]  
 # convert tolerance value to float and check to make sure it is non-negative and not equal to 0  
 tolerance = float(client\_data[-1])  
 if tolerance <= 0:  
 raise Exception(TOLERANCE\_EXCEPTION)  
 # call function bisection and assign return response  
 response = f"S{bisection(a\_value, b\_value, poly\_values, tolerance)}"  
  
 # this exception will be raised when first value is not E or S and no other exceptions have been raised  
 else:  
 raise Exception(REQUEST\_EXCEPTION)  
  
 # using the built in ValueError exception for issues converting strings to floats  
 except ValueError:  
 response = VALUE\_EXCEPTION  
 # catch any other errors in case they need to be handled  
 except Exception as err:  
 response = str(err)  
  
 # send response to client and close socket  
 print(f"SEND: {response}")  
 sock.sendall(response.encode())  
 sock.close()

# client.py

# Name: Wade Moulton  
# Date: 10/11/2020  
# Class: CS 4720  
# Section: W01  
# Instructor: Dr. Sarah North  
# Assignment\_4-Sockets  
  
import socket  
  
# creating constant variables for the server port and address using localhost address  
SERVER\_PORT = 12345  
SERVER\_ADDR = "127.0.0.1"  
  
# create sender socket and connect to server  
sender = socket.socket()  
sender.connect((SERVER\_ADDR, SERVER\_PORT))  
  
# supplied testing strings list for this assignment  
testing\_strings = ["E1.0 -945 1689 -950 230 -25 1",  
 "S0 2 -945 1689 -950 230 -25 1 1e-15",  
 "G4.1 0 0",  
 "4 1 0",  
 "E1.0",  
 "S1.0",  
 "S0 2 -945 1689 -950 230 -25 1 -1e-15",  
 "Not a number",  
 "S0 2 -945 1689 -950 230 -25 1 0",  
 "S0 2 -945 1689 -950 230 G 1 1e-15"]  
  
# iterate through each item in testing strings list  
for msg in testing\_strings:  
 # first print message being sent and then assign encoded message to variable and send to server  
 print(f"SEND: {msg}")  
 encoded\_msg = msg.encode()  
 sender.sendall(encoded\_msg)  
  
 # close connection and receive encoded data from server  
 # assign encoded data to variable  
 sender.shutdown(1)  
 encoded\_data = sender.recv(2048)  
 response = ""  
  
 # iterate through encoded data, decode, and assign to string variable  
 while len(encoded\_data) > 0:  
 response += encoded\_data.decode()  
 encoded\_data = sender.recv(2048)  
  
 # output response string from server  
 print(f"RECV: {response}")  
  
 # close sender socket  
 sender.close()  
 # on the last iteration a new connection is opened and server responds with string index out of range  
 # create if statement to ensure new connection is not made on last iteration of testing strings list  
 if msg != testing\_strings[len(testing\_strings) - 1]:  
 sender = socket.socket()  
 sender.connect((SERVER\_ADDR, SERVER\_PORT))  
  
# make sure sender socket is closed  
sender.close()