CS4720 Assignment 4

Client/Server Using Sockets

Develop two Python scripts: one a server; the other a client that will use the server. You will use the polynomial function module "polynomials.py". Do not include those functions directly into your server code, but import them into the server (see Additional Notes below).

Server

The server will listen on a specific port number (ex. 12345). It will carry out polynomial computations using the functions in the provided module. Requests are in one of two formats:

- Evaluate Request
    - Request starts with 'E'
    - Followed by an argument value
    - Followed by a single space
    - Followed by the coefficients of a polynomial, separated by single spaces
- Bisection Request
    - Requests starts with 'S'
    - Followed by 'a', 'b', polynomial, tolerance separated by single spaces

For example, here is a sample evaluate request:

```
E1.0 -945 1689 -950 230 -25 1
```

This is a request to evaluate the polynomial
$-945 + 1689x - 950x^2 + 230x^3 - 25x^4 + x^5$
at the value $x = 1.0$

Here is a sample bisection request:

```
S0 2 -945 1689 -950 230 -25 1 1e-15
```

This requests the use of the bisection function with a = 0, b = 2, tolerance = 1e-15 and using the same polynomial as in the evaluate example.

The server will create a response to the client for each request. The first character of the response will indicate the type of response:

- 'X' indicates an error response. The remainder of the response is an error message
- 'E' indicates a successful response to an evaluate request. This is immediately followed by the value returned by the evaluate function
- 'S' indicates a successful response to a bisection request. This is immediately followed by the value returned by the bisection function

The response to the example evaluate request would be[2]

```
    E2.2737367544323206e-13
```

The response to the example bisection request would be

```
    S1.0000000000000004
```

The server should operate in a continuing manner: the server should repeatedly accept a connection, get a request, send a response and close the connection to the client. In particular, only one request is handled for each connection.

Server Error Checking

The server should respond properly in the case of an erroneous request. Not only should the server not 'crash' but the server should send back an appropriate error response to the client.

For example, the request string

```
E1.0 -945xx 1689 -950 230 -25 1
```

should result in a response something like this:

```
    Xinvalid format numeric data
```

Note the 'X' that flags this as an error message.

Include error checking in the server. **Also** document the error checking by including a string at the end of the server code that describes the errors caught. That string might look something like this (to start with):

```
'''
Invalid numeric format

Wrong number of fields in request

'''
```

Client

You are **strongly urged** to write small clients that will test aspects of your server as you work on that. Once you are satisfied that you can send an evaluation request and get a good result and also send a bisection request and get a good result then work on the client described here.

The client should define four variables at the beginning: the first, named `poly` with a list of numbers representing the coefficients of a polynomial;[3] another, named 'a', representing a value a to use in bisection; another, named b representing a value b to use in bisection; another, named tol representing a tolerance value to use in bisection.

Note: in testing your client, the values assigned to these variables will be changed so make sure they are evident.

The client should first make a request to the server for a bisection. Provide the data defined in the variables just described. Display the value returned.

The result from the first request should be used as the x value in another request to the server to evaluate the polynomial (defined in the variable described above).
The result of the evaluation should be displayed. This value should be very close to 0.

Java Version

The “assign02-java.zip” contains a Java version of the assignment. This version works with the Python version (Java client to Python server and Python server to Java client). You may find this helpful in testing your code.

Java version of client

```
package cs4320.assign02;

import java.io.*;
import java.net.Socket;

public class Client2 {

    public static final int PORT = 12321;

    public static void main(String[] args) throws IOException {
        double  a = 8.0;
        double b = 10.0;
        double tolerance = 1e-13;
        double[] poly = {-945, 1689, -950, 230, -25, 1};

        StringBuilder req1Sb = new StringBuilder("S");
        req1Sb.append(a).append(" ").append(b).append(" ");
        for(double coeff: poly)
            req1Sb.append(coeff).append(" ");
        req1Sb.append(tolerance);
        String resp1 = sendAndReceive(req1Sb.toString());
        if(resp1.charAt(0) == 'S') {
            double val1 = Double.parseDouble(resp1.substring(1));
            System.out.println("Value returned from bisection: " + val1 );
            StringBuilder req2Sb = new StringBuilder("E");
            req2Sb.append(val1).append(" ");
            for(double coeff: poly)
                req2Sb.append(coeff).append(" ");
            String resp2 = sendAndReceive(req2Sb.toString());
            if(resp2.charAt(0) == 'E') {
                double val2 = Double.parseDouble(resp2.substring(1));
                System.out.println("Value returned from eval: " + val2 );
            } else {
                System.out.println("Error in evaluation call: " +
resp2.substring(1));

            }
        } else {
            System.out.println("Error in bisection call: " +
resp1.substring(1));
        }
    }


    /**
     * Sends a request to the server, prints it and then prints the response.
     *
     */
    public static String sendAndReceive(String request) throws IOException {
        System.out.println("request: " + request);
        Socket conn = new Socket("localhost", PORT);
        OutputStream os = conn.getOutputStream();
        Writer wr = new OutputStreamWriter(os, "UTF-8");
        wr.write(request);
        wr.flush();
        conn.shutdownOutput();

        InputStream is = conn.getInputStream();
        Reader rdr = new InputStreamReader(is,"UTF-8");
        StringBuilder sb = new StringBuilder();
        int c = rdr.read();
        while(c >= 0) {
            sb.append((char)c);
            c = rdr.read();
        }
        // System.out.println("response: " + sb);
        conn.shutdownInput();

        conn.close();
        return sb.toString();

    }


}
```

Submitting the Assignment

The assignment should have two scripts: one for the server, one for the client. The assignment should include the polynomials module.

Submission Instruction: Please compress all of your files in a single zip file. Then, rename its title to #Spring2017-CS4720#Assignment04#Your Student ID#Your Name and submit through D2L assignment box (previously dropbox).

Due on 11:30 pm, March 21, 2017

Additional notes:

1. The client will do not numeric computations at all, it is just sending and receiving data. The most it does is get substrings and concatenate strings.
2. The result is in scientific notation. The value of the result is very near 0.
3. This value *should not be string*, it should be a list of numeric values.