EECE 7374: Fundamentals of Networks
Programming Assignment 1: Socket Programming with Python

Introduction

You are tasked with implementing the client side of a network application designed according to the client-server application architecture. (The server is already implemented and running on a remote machine.) The description of the application and of its application layer protocol is provided below.

In this assignment, you must use **Python 3** to write your client.

Application and Application Layer Protocol

Overview

The application concerns remote *evaluation of simple arithmetical expressions* using only one of the operators `+`, `-`, `*` and `/` (no parenthesis or other operators). Examples: `3 + 5`, `5 * 8`, etc. After an initial greeting message used by your client to start the connection to the server, the server asks your client to evaluate a (server-chosen, unspecified) number of expressions one after the other. Each of these expressions will be sent as a separate message. For each expression message, the server expects a response message containing the result of the evaluation of the expression. If your program is able to correctly evaluate all the expressions, then the server will return a *secret flag* to the client. This flag is unique to each student, and should be saved for submission (see Section **Submission**). The message containing the flag will signify the end of the communication session. At this point, the client can close the connection to the server, and the application terminates.

Protocol Actions and Messages

Upon starting, the client must setup a **TCP connection** with the server. Then, the client must send the *introductory* message to the server. The server will respond with an *expression* message. The client must evaluate the expression and send back a *result* message. If the result is incorrect, the server will send back a *failure* message and close the connection to the client (this means that your client is not correctly designed and you have to modify it and try again). Otherwise, the server will either send another *expression* message or, if enough expressions have been evaluated, a *success* message. The *success* carries the secret flag, which you should save and keep for submission.

The application uses the following types of messages:

- The *introductory* message is sent from the client to the server to signal the start of communications. It is also used authenticate the sender. This message uses the following format:

  `EECE7374 INTR` ***`nuid`***

  Where ***`nuid`*** is your Northeastern University identification number. Make sure to enter your NU ID number as a string that includes all leading zeroes.

- The *expression* message is sent from the server to the client, carrying the expression to be evaluated. This message uses the following format:

  `EECE7374 EXPR` ***`expression`***

  Where ***`expression`*** is the expression to be evaluated.

- The *result* message is sent from the client to the server, carrying the result of the last expression to be evaluated. This message uses the following format:

  `EECE7374 RSLT` *`result`*

  Where *`result`* is the value of the evaluated expression.

- The *failure* message is sent from the server to the client to indicate that the last expression was evaluated incorrectly. After the server sends this message it closes the connection to the client. This message has the following format:

  `EECE7374 FAIL`

- The *success* message is sent from the server to the client, indicating that the application has completed successfully. It carries the secret flag for submission. This message uses the following format:

  `EECE7374 SUCC` *`flag`*

  Where *`flag`* is the secret flag.

All of these messages must be case-sensitive strings that have been encoded using UTF-8.

Servers

The server is running on a remote machine with the host name `kopi.ece.neu.edu`. The server uses the port numbers in the range $[5203, 5212]$. If you experience difficulty connecting to a port, try another. The server will be listening for TCP connection requests. Therefore, ensure that the client uses TCP sockets.

Testing

A sample server `sample_server.py` is available to test your client locally, on your machine, as you design the client. This server behaves almost exactly like the actual server. However, the *secret flag* it returns **is not** the one required for the assignment. You can test your client by setting your host-name to `localhost` and the port number to the same as the one in the sample server. This way, you can test your client as many times as you like, without needing to worry about connectivity or server load.

Submission

To receive full credit for the assignment, you need to provide this secret flag in your assignment submission, along with the source code for your client. For your submission, please name your source file as follows:

`FirstName_LastName``_Proj.py`

Where *FirstName* and *LastName* are your first and last names, respectively. It is very important that your submission complies with this naming format. Any other file name will result in a loss of points.

In your source file, please include a comment header that contains:

- A comprehensive description of your high-level approach to the client design. Here, you need to explain what your program is doing and how it is doing that. This needs to be written as if you are trying to explain your program to someone who has not taken this course.
- Your NU ID number, clearly labelled.
- The secret flag that you obtained by running this program, also clearly labelled.

This header can be succinct, but **please make sure that all of the information in this header is accurate.**

Please ensure that your file can be run! If your submission is not a Python 3 source file and cannot be run from the command line (or IDE of your choice), then you will lose points.

You may resubmit your source file as many times as you want. However, only the last submission will be graded.

Grading

Points will be given to your solution based on the following:

- *Filename:* 5 points. You will receive full points if your file is named correctly.
- *Code works:* 35 points. You will receive full points if your code runs without errors.
- *High-level approach:* 25 points. You will receive full points if you provide sufficient details on how you designed the client.
- *NU ID:* 5 points. You will receive full points if the listed NU ID is correct.
- *Secret flag:* 30 points. You will receive full points if the listed secret flag is the correct one.