PIC 16, Spring 2018 – Assignment 10M

Assigned 6/4/2018. Code (a single `.py` file) due by the end of class 6/8/2018 on CCLE. Hand in a printout of this document with the self-assessment portion completed by the end of class on 6/8/2018.

In this assignment, you will create a simple chat program with a partner.

Task

Write a GUI for sending messages to and receiving messages from a partner on another computer. The goal is for the behavior to be similar to the example demonstrated in this video. (Note that this chat program is not very good, but we'll improve it next time after we've learned about setting up multiple threads.) I suggest that you and your partner develop a *single* program together on one computer. Only when it is working well between two Python consoles on the same computer should you begin to test it between your two computers. Again, you should only write *one* program rather than a separate client and server; a technique for having your program automatically act as a client or server as needed is explained in step 2.

1. You've already seen the code for establishing a connection between two computers, so let's start with the GUI. Create a window with a text box for sending messages and a label for displaying incoming messages.
2. Create a `socket` and attempt to `connect` to a socket on your partner's computer. If that fails (because their server is not running), `bind` the socket to your IP address and a valid port, `listen` for connection requests, and `accept` one if it is received. If all goes well, the first instance of the program to run will become the server, and the second will be the client. When the connection is established, show "`<connected>`" in the label.
3. Connect the appropriate signal of the text box to a slot that `send`s the message contained in the text box when the return key is pressed.
4. Set up a `QTimer` such that incoming messages are `recv`eived every two seconds. Beforehand, use your socket's `settimeout` method (see documentation) such that the socket will not wait for messages indefinitely but instead timeout after 0.25s. (Otherwise, the `recv` method would block program execution permanently, the GUI would be totally unresponsive, and no messages could be sent or received.) Display incoming messages in the label.
5. When one of the connected pair of GUIs is closed, the other should show "`<connection closed>`" with the label. You should be able to close both GUIs, run them again, and have the connection established any number of times without restarting the Python consoles.
6. If you can figure out a way to improve the responsiveness of the program (beyond tweaking the `QTimer` and `socket` timeouts and without using multithreading), I'd love to see it.

Self-Assessment

Demonstrate (to me) the operation of your program between two computers during class. Check off the tasks you completed above and indicate your total score below. (20 pts each.)