CPSC 235: Assign 1: Knock-Knock Jokes   Feb 3

Knock-Knock
        Who's there?
NetCentric Student
        NetCentric Student who?
No Joke – ***you*** are writing a Knock-Knock Program

Objectives:

- Experiment with the Client-Server program pattern
- Understand the Python socket module
- Generally improve familiarity with Python

The Standard Knock-Knock Protocol:

Knock-Knock jokes can come in various structures – in fact sometimes varying the structure is the cause of the humor. However, we have to start with the standard delivery of a Knock-Knock joke.

1. When a connection is established, the server initiates the joke by sending the string: "Knock-Knock".
2. The client responds with the phrase "\tWho's there?"
   Recall that the "\t" represents a tab character.
3. The server responds with a name.
4. The client replies with a tab ("\t") followed by the server's response followed by " who?"
5. When it receives the last question, the server delivers the punch line.

We need to be careful to follow the protocol exactly; pay close attention to capitalization, spacing and punctuation. Any deviations from the protocol should result in the server terminating the connection.

Specification:

1. Write a program that implements the standard Knock-Knock protocol between the client and the server.
   a. The client and the server will both communicate over port 2000.
   b. For now, the joke should be stored in the server as a literal value.
   c. Your server should
      i. Find its own IP address and display it (i.e. "Listening on 192.168.1.36")
      ii. Set up a server socket.
      iii. Be able to accept multiple requests in series
         1. It doesn't shut down after telling a single joke.
         2. It doesn't need to simultaneously accept requests from multiple clients.
      iv. When the server is not busy – it should display a "Waiting" message.
      v. When the server is connected, it should display the last data sent or received.
      vi. Close the socket after the punch line is delivered and wait for the next request.

   d. Your client should
      i. Accept an IP address as an input parameter.
      ii. Use the input parameter as the IP address of the server to which you want to connect.
      iii. Display any messages received from the server
      iv. Display any messages sent to the server.
      v. Close the socket after it receives the punch line.

2. Stop and verify that your server works with a partner.
3. Alter the Server to allow it to tell multiple jokes. You should:
   a. Store a collection of names and punch lines in a file (see jokes.txt)
   b. Read the file into a structure that helps manage the jokes (one time)
   c. When a connection is established, randomly select a joke to tell.
4. Create a second program[1] that implements the extended Knock-Knock Protocol. This protocol is the same as the standard protocol, with one exception; In addition to the other jokes, it also implements the (in)famous banana exception. According to this protocol:
   a. On stage 3 of the protocol, the server responds with the name "Banana".
   b. When the client responds "\tBanana who?, the server resends the name "Banana".
   c. The acceptable responses from the client are "\tBanana who?" or "\tBANANA WHO!?"
   d. As long as the client responds "\tBanana who?" the server will continue responding "Banana". When the client responds "\tBANANA WHO!?", the server responds "Orange".
   e. The punchline associated with "\tOrange who?" is "Orange you glad I didn't say Banana?"
5. Create a third program[2] that implements the generalized Knock-knock Protocol. This version allows either the client or the server to tell the joke.
   a. After establishing the connection, the Client sends a message
      i. "Hear" indicates that the client wants to hear a joke – your program proceeds with the standard knock-knock protocol as before.
      ii. "Tell" indicates that the client wants to tell a joke.
         1. You should invert the messaging of the standard Knock-Knock protocol. (i.e. the client starts with Knock-Knock; the server responds "\tWho's there?", etc.)
         2. Your program should "learn" any new jokes that it hears from a client. This means you should update the jokes in memory as well as updating its joke file.

Submit:

Three versions of your KnockKnock Client-server pair (6 .py files). Each file should be named in a manner makes it clearly identifiable.

A copy of your updated Jokes.txt --- I ***expect*** to be entertained.

[1] This can (should) start as a blatant copy-paste; I want to see clearly that you were able to implement the first sections independent of the changes proposed here.

[2] Again, copy-paste. However, this one doesn't have to include the "extended Knock-Knock"; you can copy from the original program if you want.