****

Task Description

****

The goal is to implement an user interface for our Chat Server.
You must use Python 3.5+ (3.6 preferred). To create the GUI, you must use PyQt 5.
The server is a simple chat server that does not require authentication.

Key UI elements:
- Field for Username and button for connection and disconnection.
- Status Label showing if we are connected or not to the server.
- Combo box to select which user the message will be sent (Add option for ALL - Broadcast)

The design is free form. You have one week to conclude this work and send back to us the following items:
- Code Package (tar.gz, github repo, etc)
- Documentation on how to operate the UI including installation instructions.

****

Requirements

****

In order to properly run the server you must use Python 3.5+ (3.6 preferred).
To create the GUI, you must use PyQt 5.

****

Running the Server

****

```
python server.py --host 127.0.0.1 --port 33002
```

The default port is 33002 and the host is 127.0.0.1

****

Server Protocol

****

Here is the communication process between the client and the server:

The client establishes the connection to the server by registering the username to the server:

```
    {REGISTER}<USERNAME>
        Where <USERNAME> is the username provided by the chat user from a
GUI text box:
        e.g.:
            {REGISTER}Hugo
```

Note that the chat window will not display this message. The server will process this message internally.

The server sends back a welcome message:

```
    {<TARGET>}<message>
        Where <TARGET> will be the username just registered:
        e.g.:
            {Hugo}Welcome, Hugo!
```

Note that the message will be displayed to the chat window, without the command portion, i.e. “Welcome, Hugo!” will be displayed, without the {Hugo} command header.

The server sends a list of clients currently participating in the chat. {CLIENTS} is the command header of the message and <CLIENT NAMES> is a string of names separated by the | character.

```
    {CLIENTS}<CLIENT NAMES>
        When a client connects or disconnect the server will emit a
broadcast messsage
        with the current list of clients where {CLIENTS} is the identifier
of the
        message and <CLIENT NAMES> is a string of names separated by the |
character.
        e.g.:
            {CLIENTS}Hugo|User2|User3|User4|User5
```

The chat window will not display this message, which is to be processed internally. The GUI can use this client list to populate the chat user combo box.

The client can send a broadcast message or send a message to a specific participant (selected from the chat user combo box):

```
    {<TARGET>}<message>
        Where <TARGET> will be a username or ALL for a broadcast message.
        e.g.:
            {ALL}This is a test of the broadcast system...
            {Hugo}Hi... Only Hugo should see this message
```

The chat window will display just the message, without the command header.

The client can send the QUIT message to disconnect from the server. In this case, it doesn’t matter which chat participant the user is talking to, or whether the user is broadcasting:

```
    {QUIT}
        This will close the client connection and send a broadcast message
        that the client left the chat. Also, an empty message will result
        on the same disconnection.
        e.g.:
            {QUIT}
```

The chat window could display a “goodbye” message before terminating the connection.