Python 3.7 is used

1. Program design

All input follow specification format and after any command is processed, both client and server terminal print meaningful messages like did the action success or failed.

The client-side program is straightforward. At first it processes the arguments that user enters, check the arguments number, the port number is decimal and the IP is valid. Then it tries connecting the server, after that, it starts authentication process. Once user is successful login, it starts a thread for receive P2P UDP information and prompting user to enter one of the available commands. Every command result in make interactions with server. If user enter UPD command, it starts another thread for upload the file.

The server-side program empties the log file every time it starts. Then it reads username and password into a dictionary for credentials.txt, the program assume username is unique and not equal to empty string. The server then waits for the user to connect and start a thread for that user which process any command the user give. Note that, the server doesn't limit the number of users connected. The server uses two list of dictionaries to store the messages and active users and writes log file base on these two lists.

Json is used for transfer data from client to server. The client-side program forms a temporary dictionary and dumps into json. Every temporary dictionary has a key "Type" to indicate its type of command. For data transfer from server to client, it's just simple status code or the information needed.

2. Design trade off

The P2P file transfer between users is using UDP which save the time to build connection in between and improve the transfer rate but also make the transmission unreliable.

The user data and message data are stored inside a list structure, this make the program been produced faster but potential lower the performance when number of user increase.

3. Possible improvement

The reliability of P2P file transfer is unacceptable at this stage. To transfer a 100MB file, the possibility of file corrupted is approximately 80 to 90 percent. This could be improved by develop some reliable transmission method on UDP.

The readability of the code is also a concern. The server and client program right now contain a big if else statement which make it hard to read and is a violation of design principles. This could be improved by put the code inside a class or refactor the code.

4. Borrowed code

Between client.py line 260 to line 264

https://stackoverflow.com/questions/13993514/sending-receiving-file-udp-in-python