How to run the program: python program's name

In my TCP server, I first create a socket, make it listen to address at 127.0.0.1 with port number 12345, and set the maximum connect to 5. When a client connects to the server, I create a new thread the handle the server connect by using the thread API in python. The advantage of doing is to keep the server running even an error happened in a connection. When handling a client connect, the server first wait for the client to send a message in JSON format, then use the JSON API in python to convert the JSON object into dictionary and do operation on this data structure. If the OC or the value is not a valid format, the server will send an error message {'status':300,'value': -1} in JSON format back to the client and the socket is close after that. If all the format is valid, the server will do the simple operation on these number base on the given OC and send message {'status': 200, 'value': result} in JSON format back to the client and socket close after message is sent.

In my UDP server, most of operation is like the TCP server exception it has chance disregard a client. Before handle a client connects there is a random function creates number between 0 and 1, which both has 50% chance. If it is 0 the server handle the client connection, otherwise disregarded.

In my TCP client, I first create a socket, make it connect to address at 127.0.0.1 with port number 12345. Accept input for OC, number 1 and number 2 according from the user. Store these values in JSON format {'OC':OC,'num1':num1,'num2':num2} and sent to the server. Then wait for the server to reply. Once the client the get the message from the server, parse the JSON object according to the status code. If the status code is not 200, which indicate failure, the client will print out an error message, otherwise it prints out the result.

In my UDP client, everything is like the TCP client, except it has timeout. First set the timeout limit the 0.1, double the limit if doesn't get message from the server, reset the timeout to 0.1 if receive message from the server. If the timeout get over 2, the operation is disregarded and the user allows to enter new input. The timeout limit is within 2, the client will try to send the same operation message to the server until it success or disregarded.