**通信与网络实验一：Socket编程实验报告**

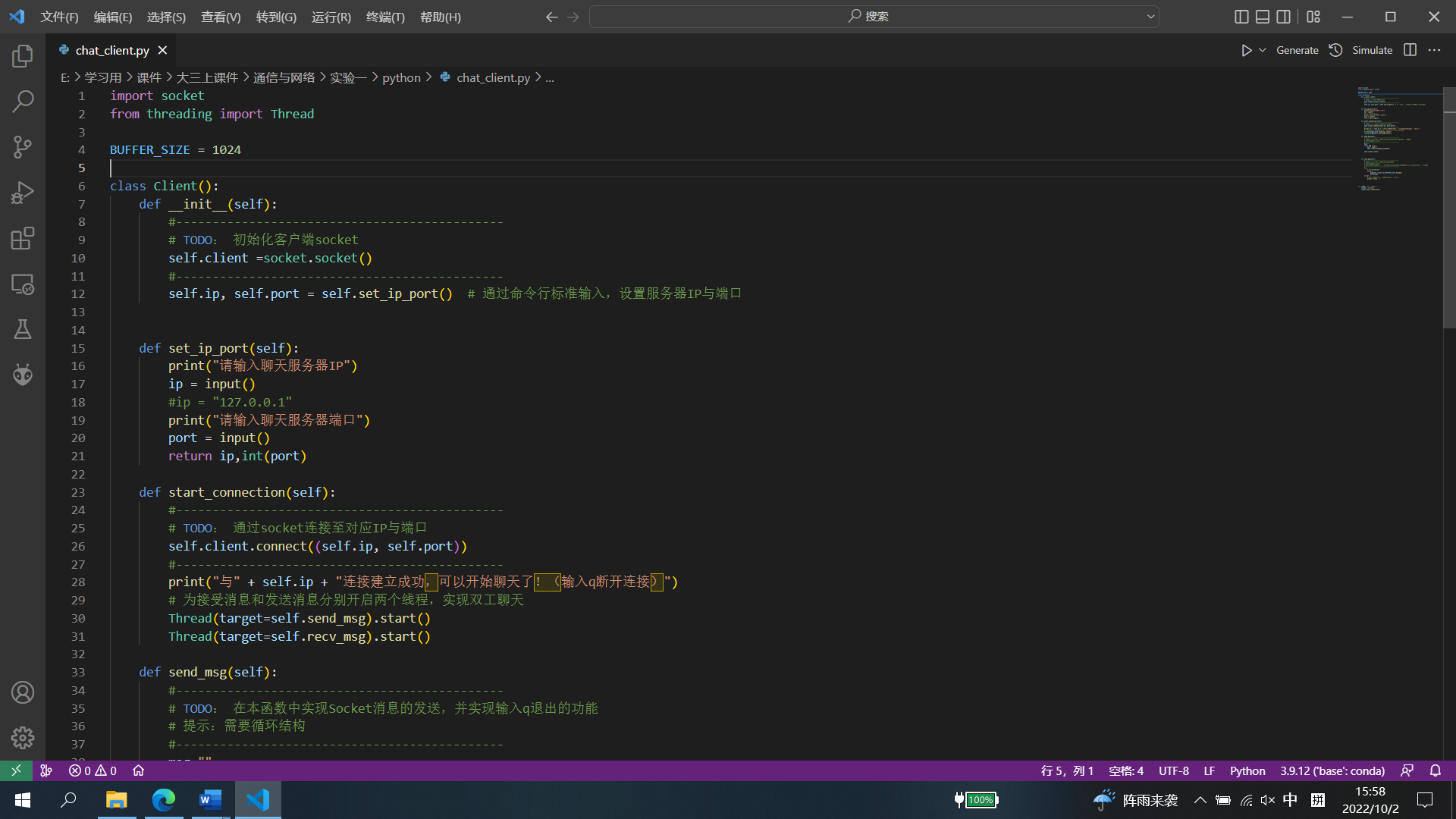
**无06**

**闫珺博**

**2020010796**

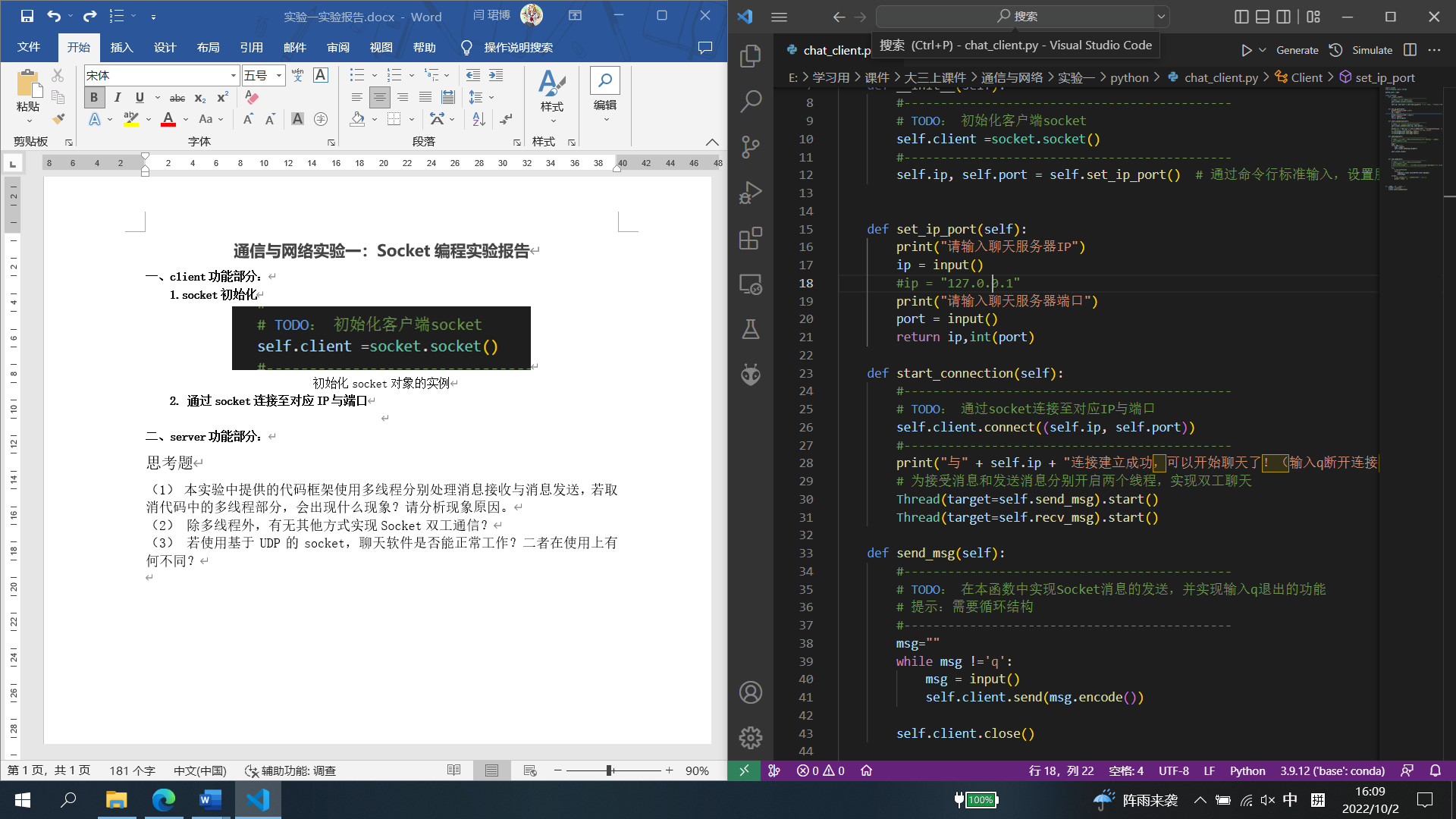
**一、client功能部分：**

**1.socket初始化**



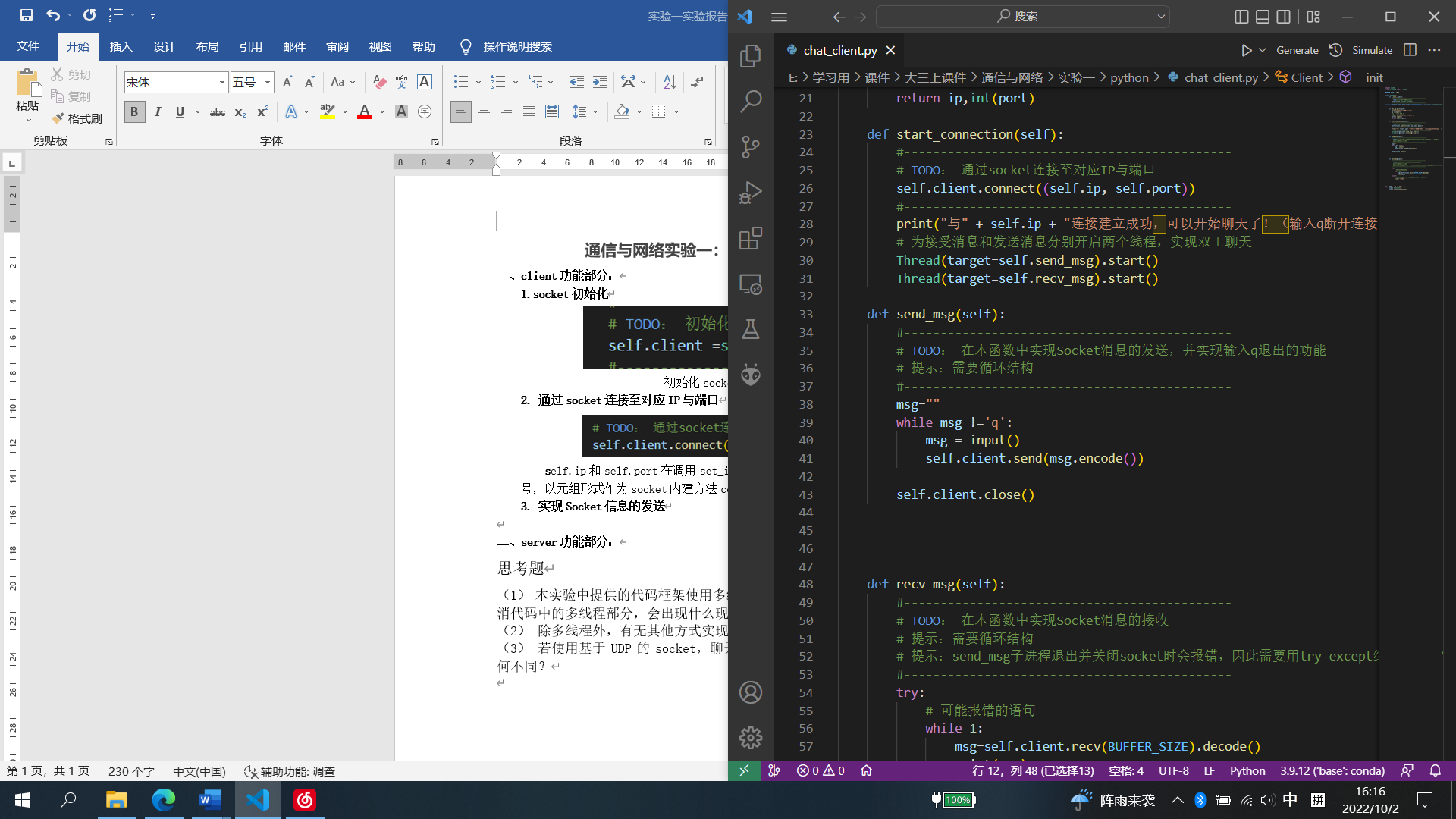
初始化socket对象的实例

**2.** **通过socket连接至对应IP与端口**



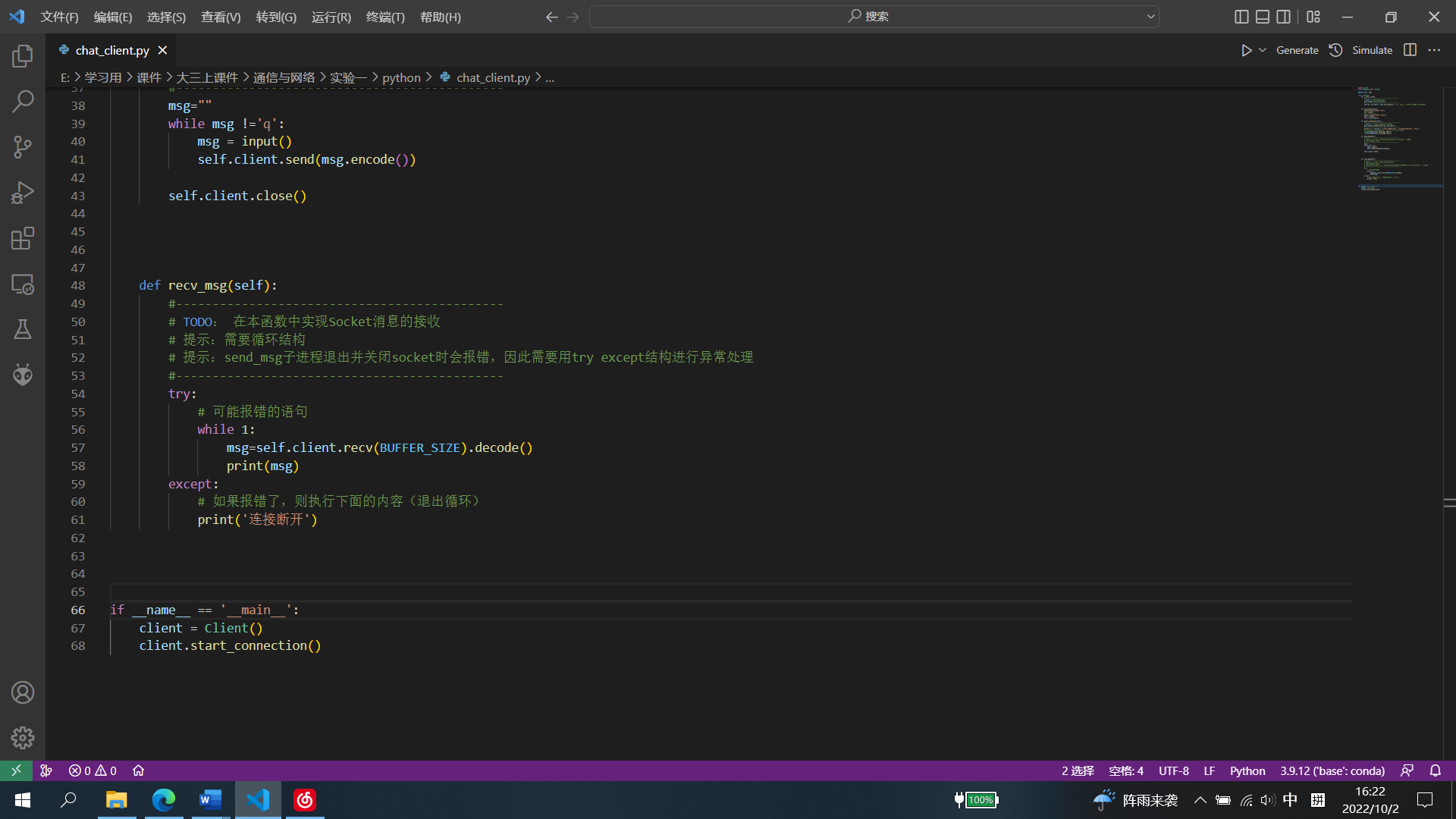
**s**elf.ip和self.port在调用set\_ip\_port()后被设置为服务器的ip地址和端口 号，以元组形式作为socket内建方法connect()的参数

**3.** **实现Socket信息的发送**



msg接收标准输入流中的信息，当输入信息不是q时，通过socket内建方法 send()向server发送UTF-8编码的msg

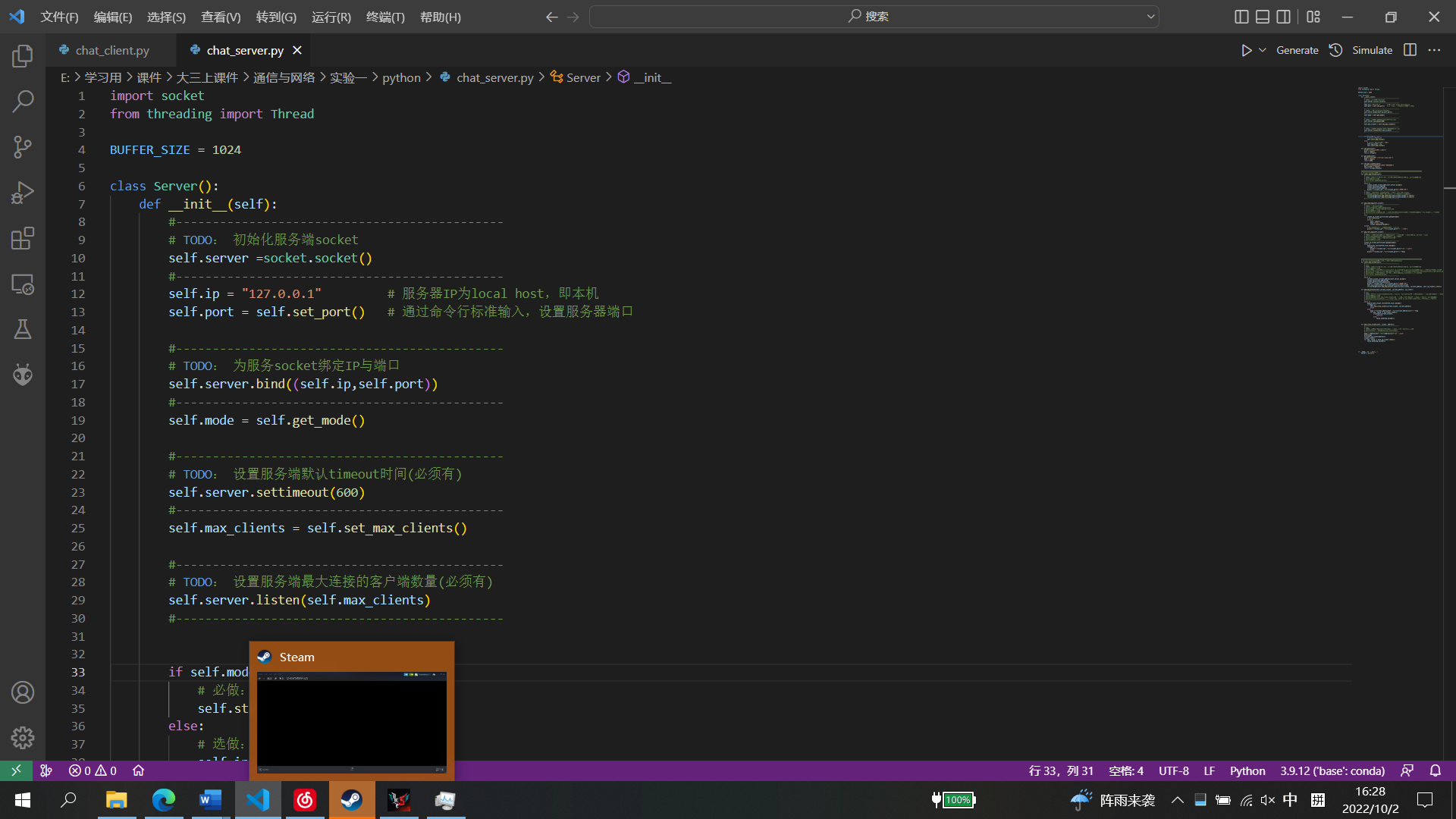
**4.实现Socket信息的接收**



由socket内建方法recv()实现信息接收，解码后赋给msg再打印出来

**二、server功能部分：**

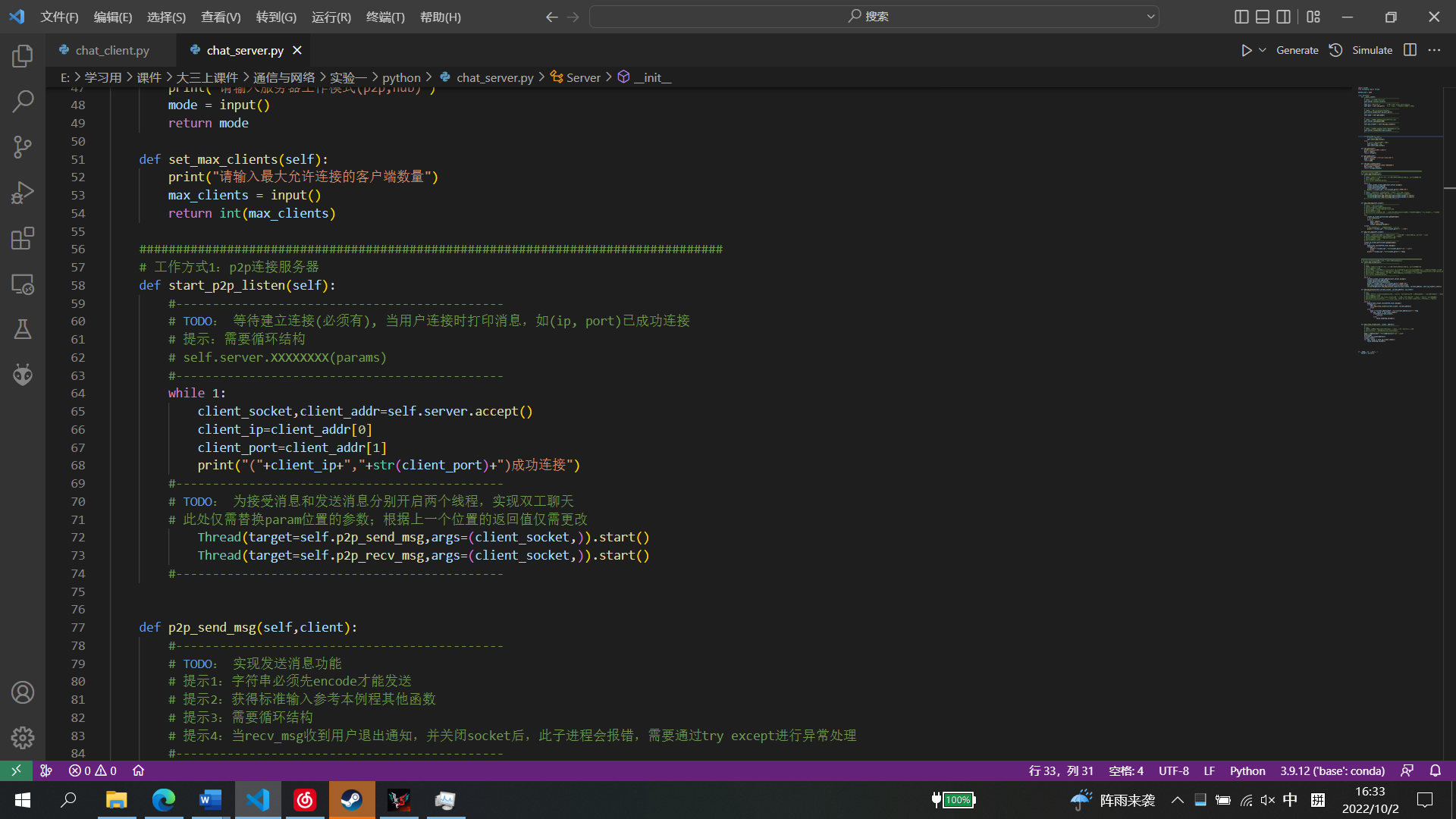
**1.server初始化**



初始化server端socket，绑定ip和port，设置timeout时间以及最大连接客户端数量

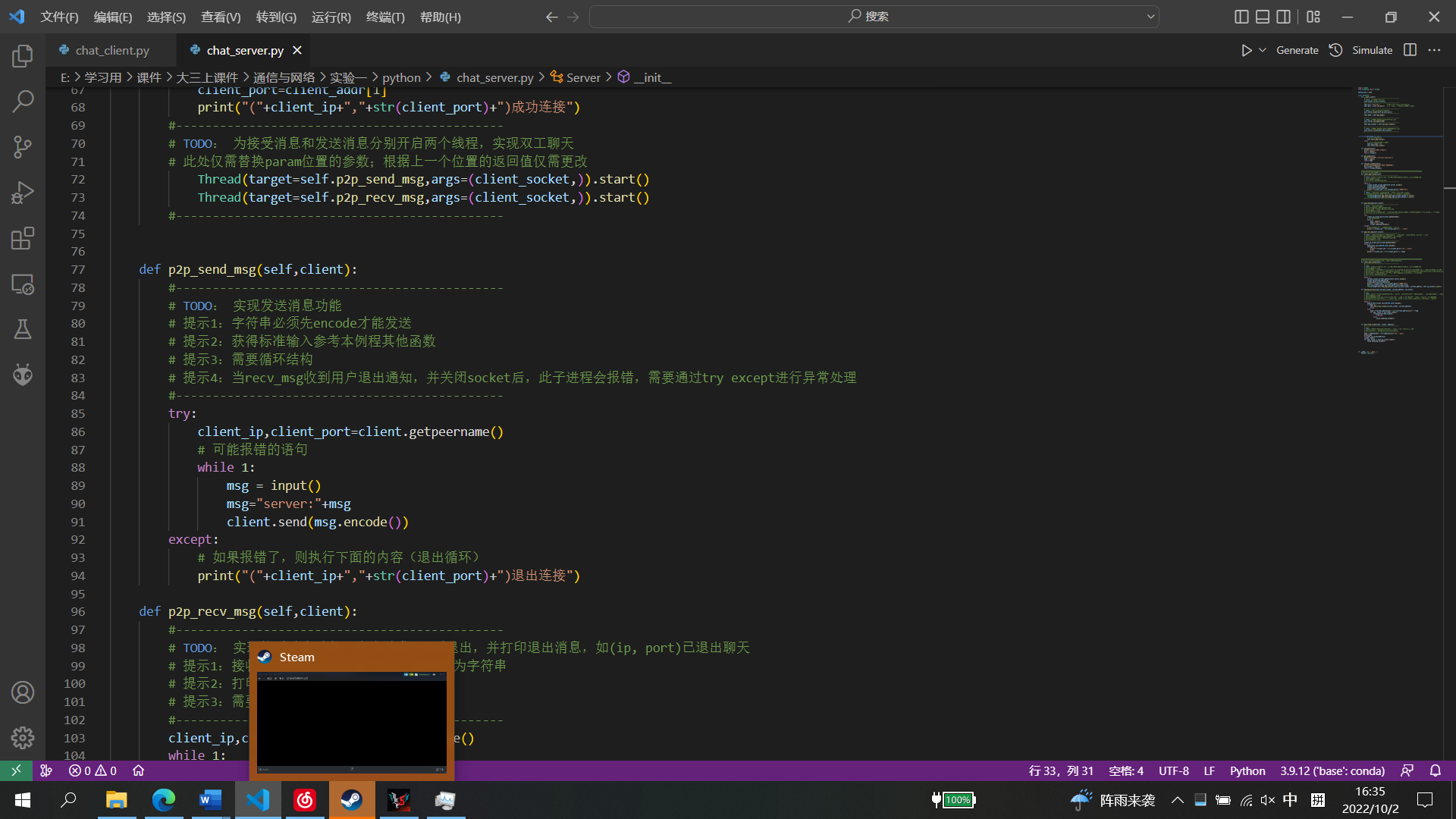
**2.p2p连接服务器**

**(1).等待建立连接**



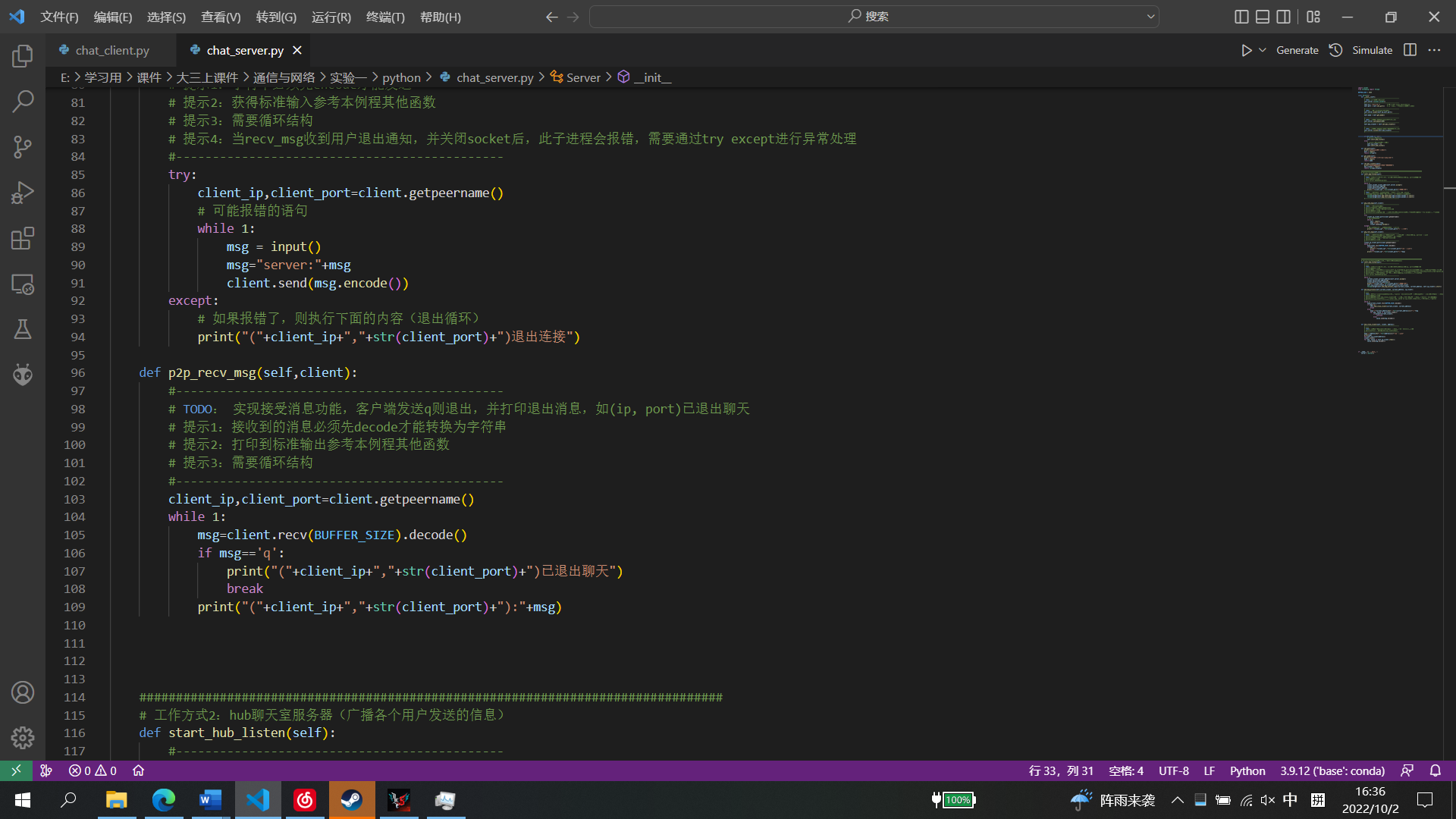
由socket内建方法accept()实现堵塞式等待连接并返回socket对象和地址(ip,port)

**(2).发送信息**



与客户端发送消息类似，利用send()方法

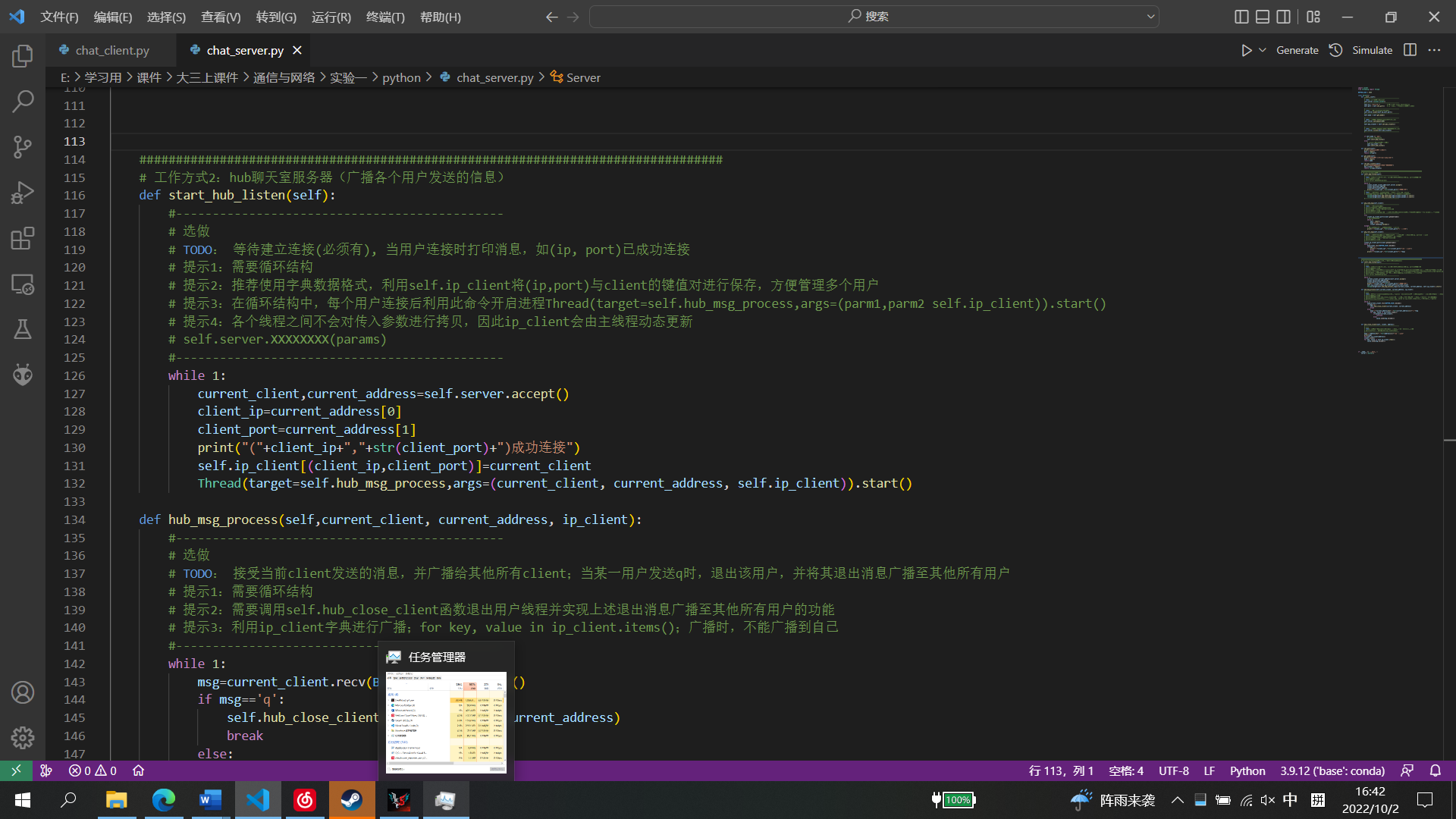
**(3).接收信息**



循环实现打印接收到的消息，如果接收到q则打印客户端退出消息

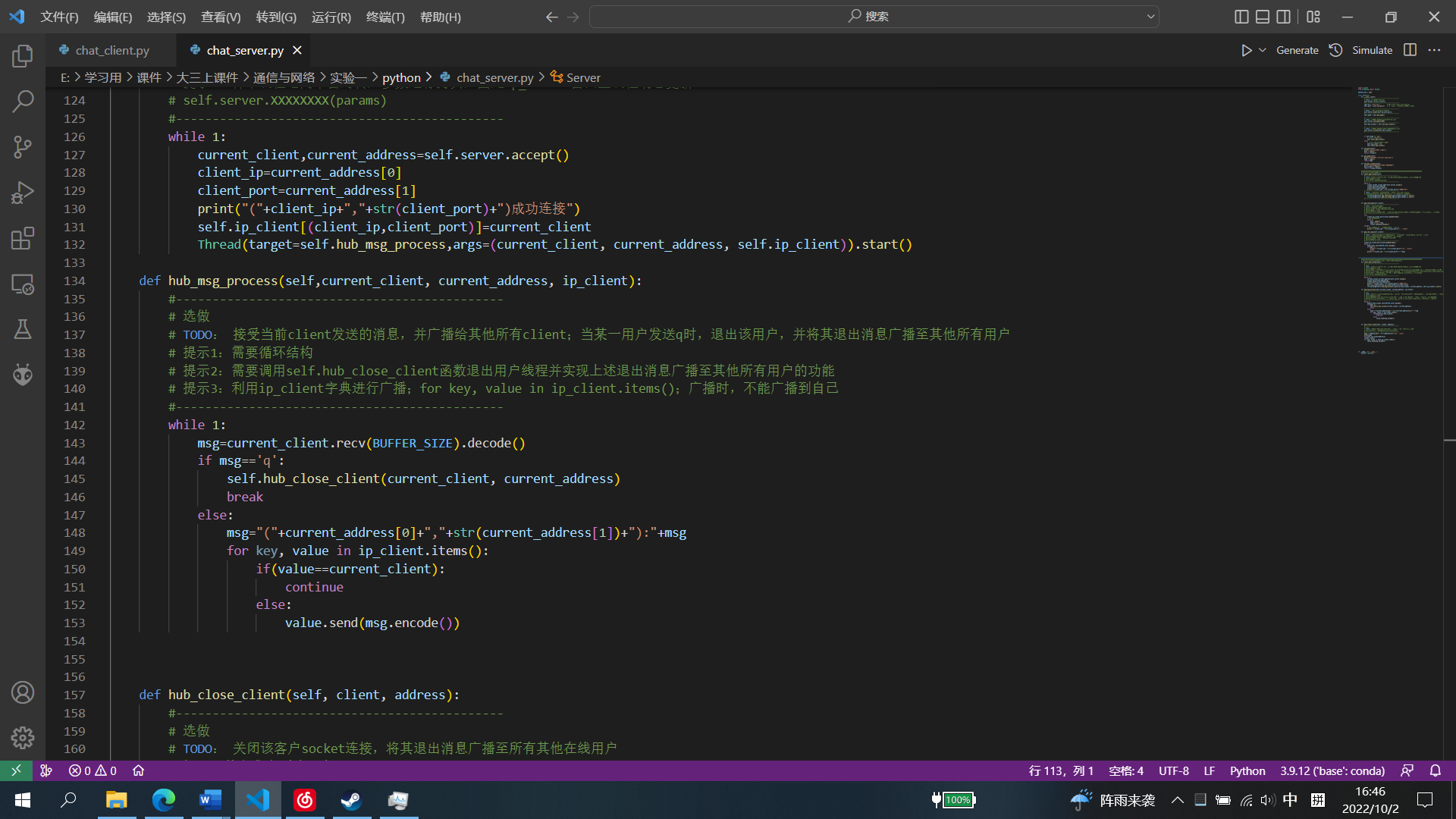
**3.hub聊天室服务器**

**(1).等待建立连接**



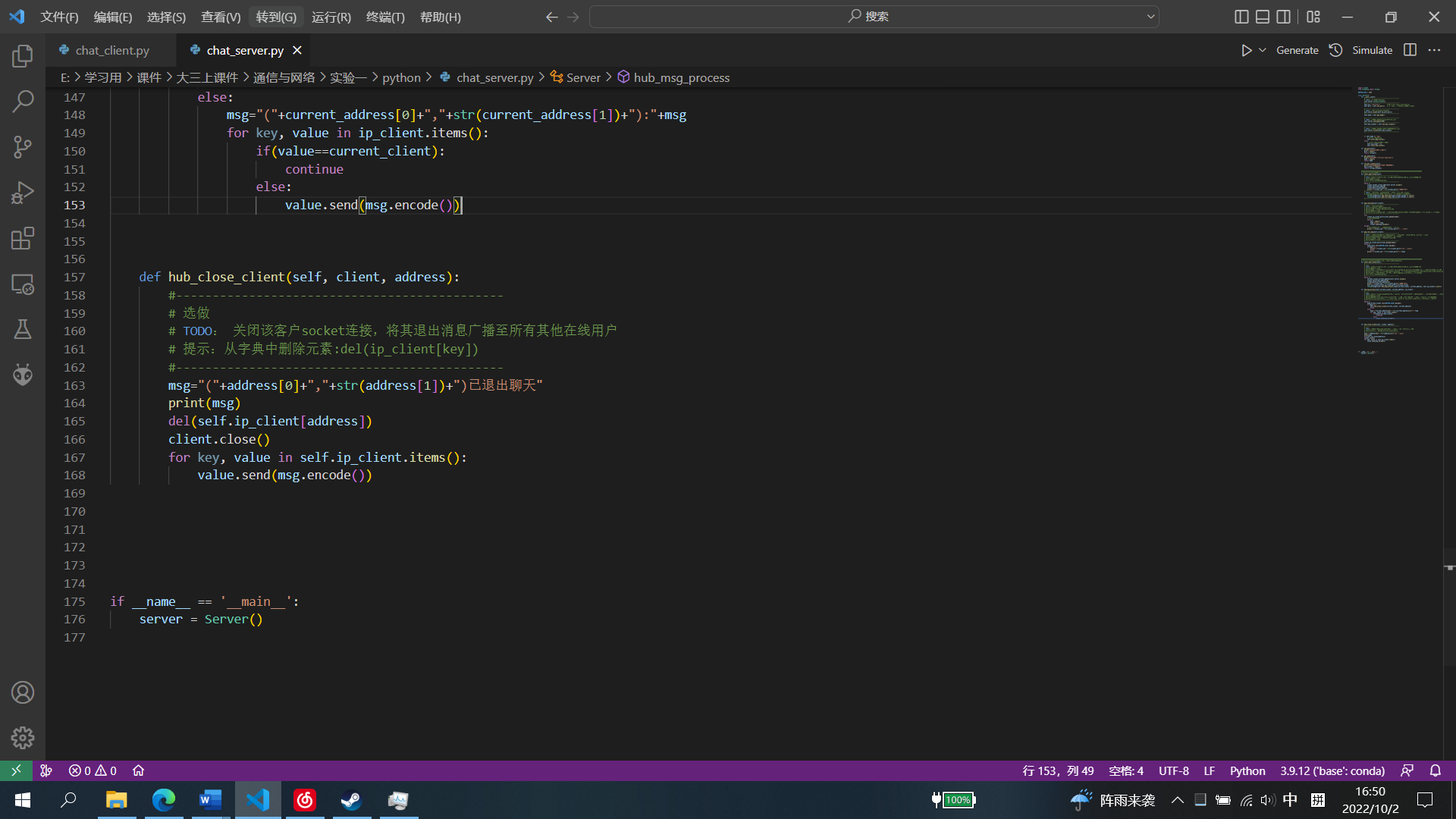
循环结构，每多一个client连接就令开启一个进程，以(ip，port)元组作为 key，对应用户的socket对象作为value构建字典

**(2).接收消息并广播**



利用ip\_client字典对除了当前ip,port的socket对象发送从当前客户端接收到 的信息，如果接收到q则调用关闭客户端socket的方法hub\_close\_client()

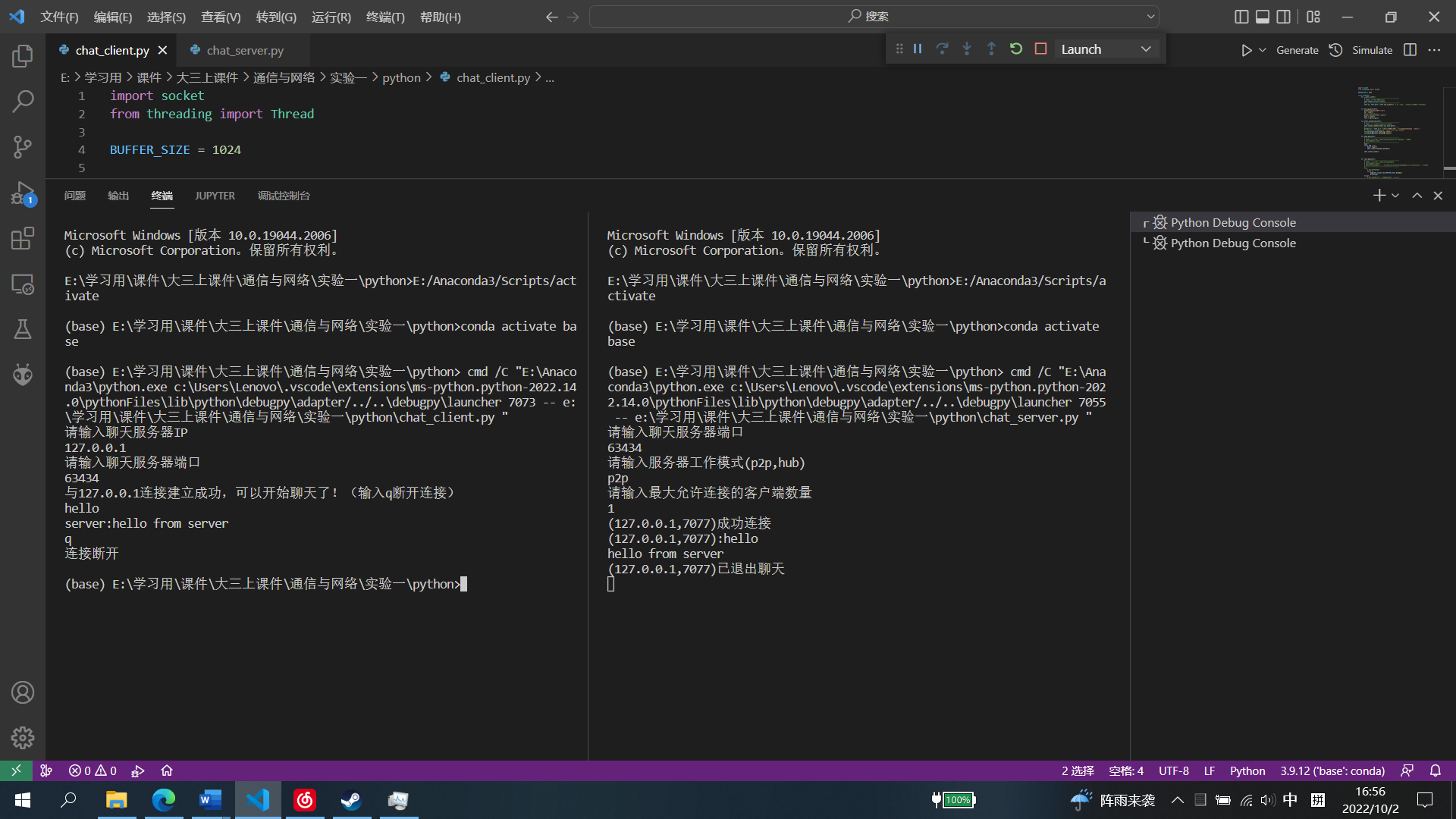
**(3).关闭当前客户端socket连接并广播退出消息**



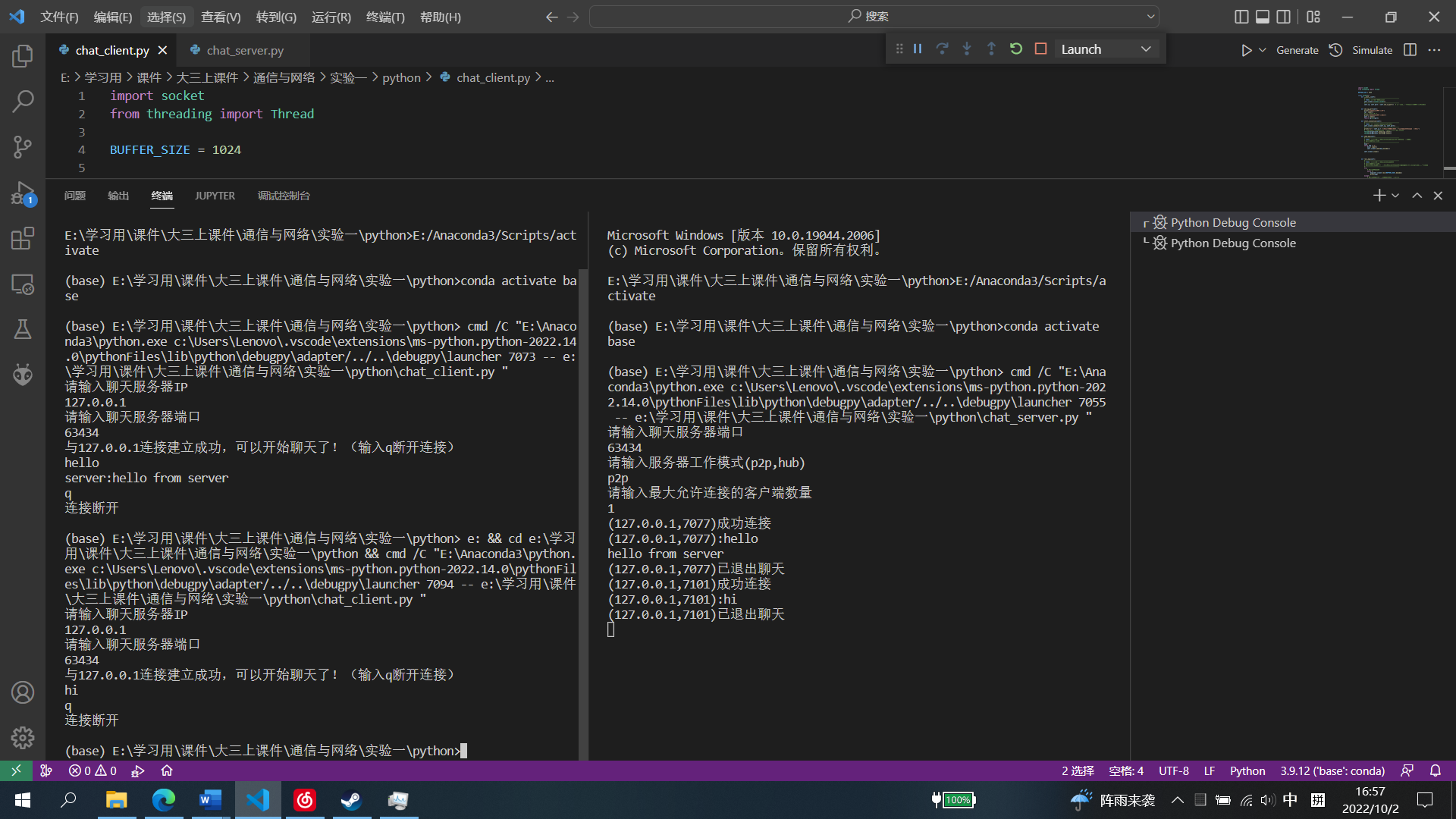
从字典中删去退出的客户端并广播退出消息

**三、client、server正确性验证：**

**1.client和server实现p2p通信**

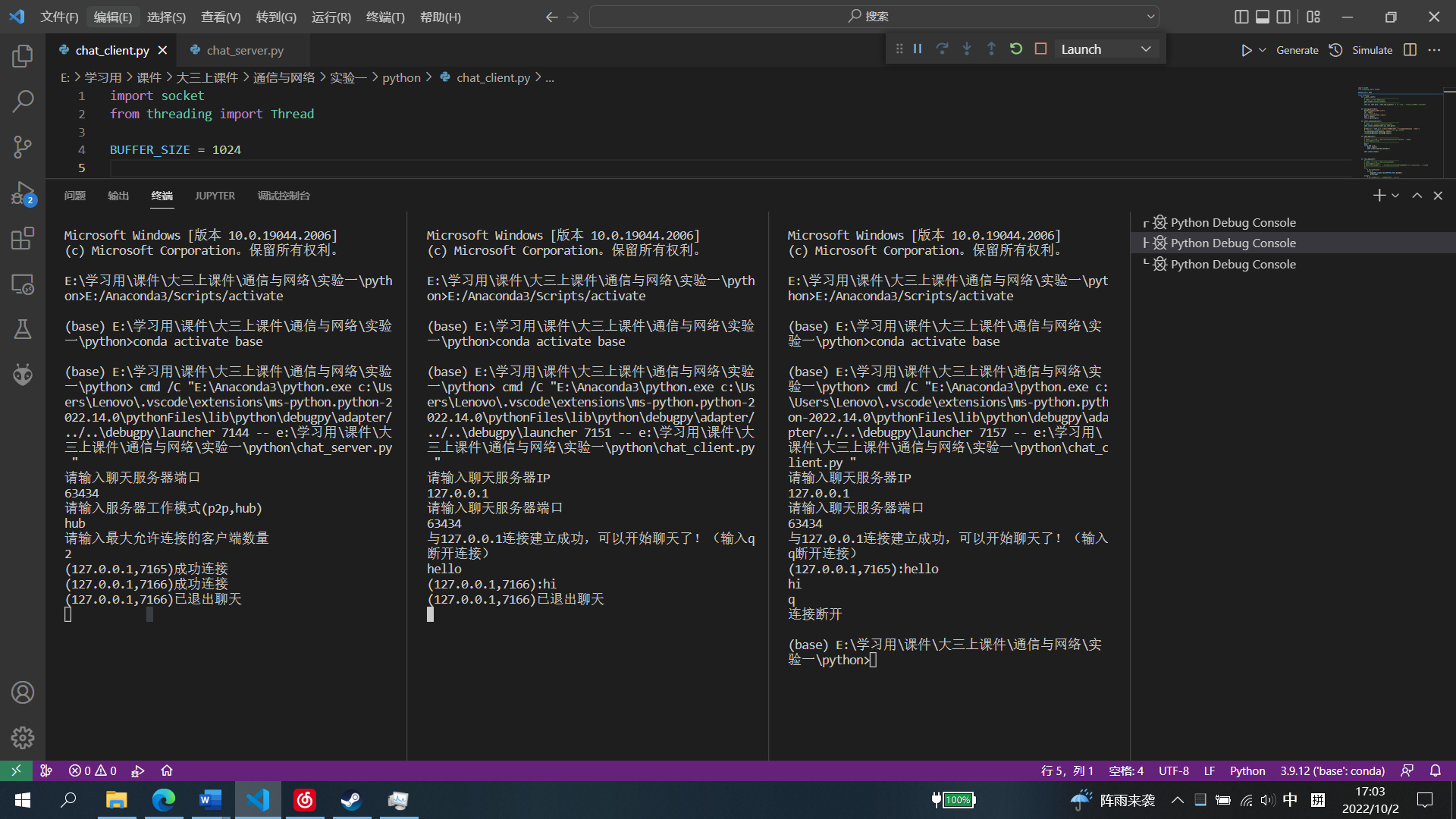


client和server可以正常地互相收发信息



当前client断开连接后新的client也能正常连接到server，验证了client和 server的p2p部分的正确性

**2.两个client连接到聊天室server**



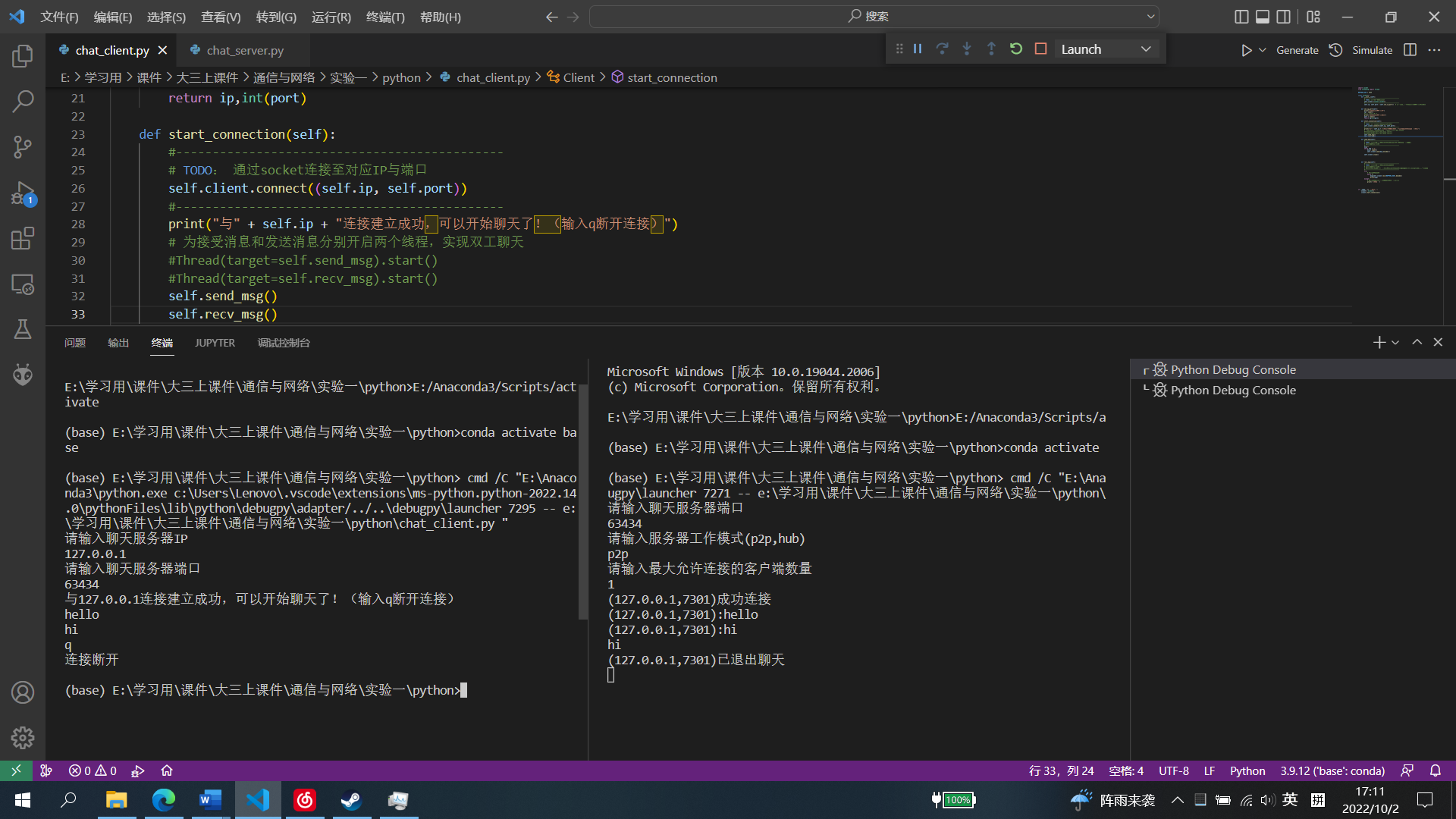
两个client连接到server，可以看到广播的消息，其中一个退出也可以正常收到广播的退出消息，验证了server的hub部分的正确性

**四、思考题**

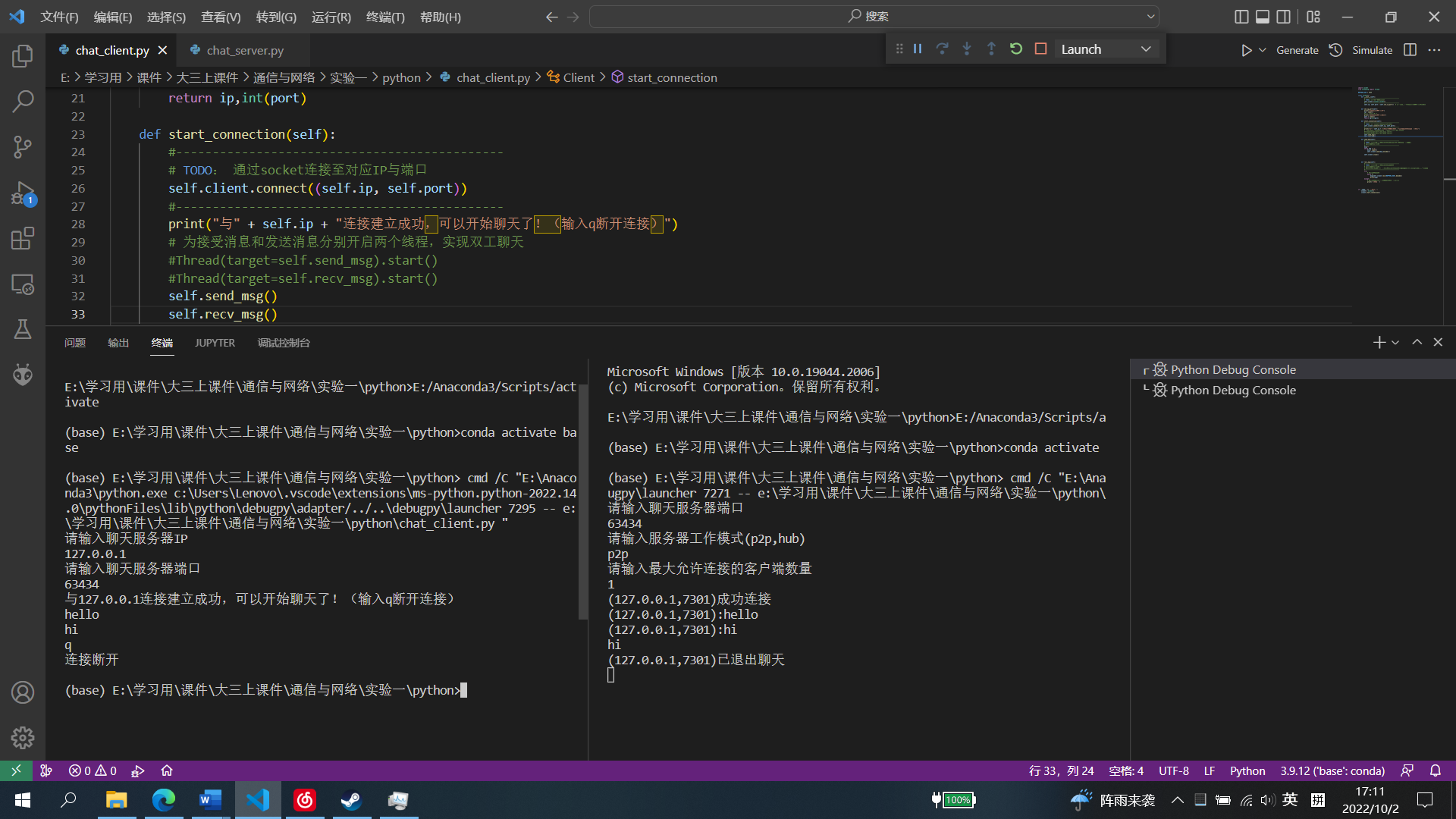
**（1）** 本实验中提供的代码框架使用多线程分别处理消息接收与消息发送，若取消代码中的多线程部分，会出现什么现象？请分析现象原因。

**答：**如果去掉多线程部分，则client只能收/发消息，p2p服务器只能收/发消息，聊天室服务器不能连接大于一个client，并且也只能收/发消息。

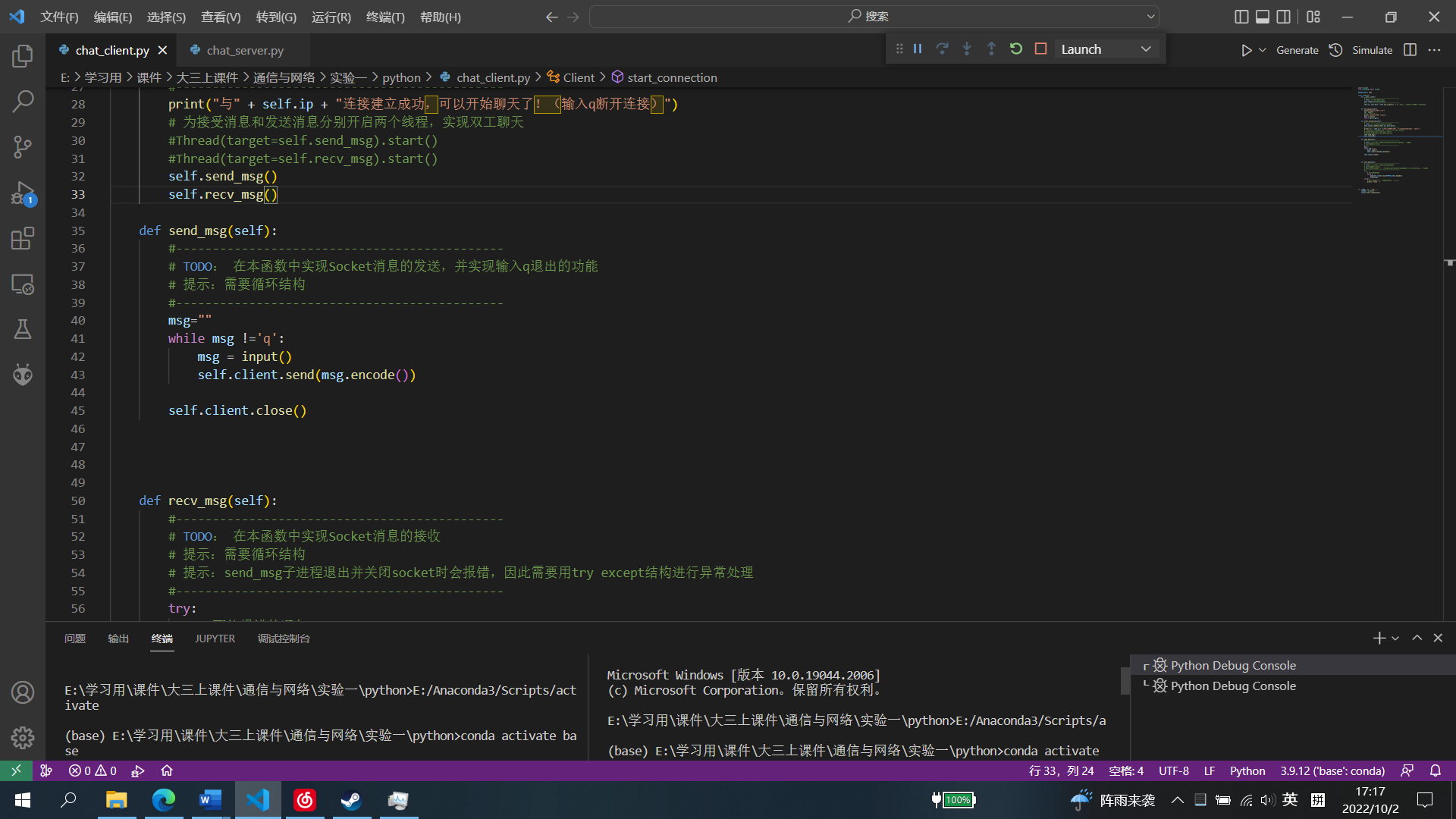
以client去掉多线程部分为例：



注释掉多线程部分，先调用send\_msg(),再调用recv\_msg()



如图，client发送消息server可以正常收到并打印出来，但client 不能接受到server发送的消息



观察send\_msg()实现，在通信过程中，while循环一直在进行。因此如 果不使用多进程，self.send\_msg()一直没有执行完成，程序运行不到 self.recv\_msg()，也就不能正常进行信息的接收。其他地方的多线程也类 似，都是为了同时处理多个在通信建立后一直执行的循环语句。如果去掉 收发方法的循环，也可以实现双向通信，但要等待对方发来信息才能发下 一条信息。

**（2）** 除多线程外，有无其他方式实现Socket双工通信？

**答：**根据网络资料，select模块可以在单线程网络服务器程序中管理多个套接字连接，进而实现双工通信。

**（3）** 若使用基于UDP的socket，聊天软件是否能正常工作？二者在使用上有何不同？

**答：**在更改部分代码后可以正常工作。

|  |  |  |
| --- | --- | --- |
|  | 客户端 | 服务器端 |
| 基于TCP的socket | 创建socket对象，设置服务器的ip地址和断开号，使用connect()连接到服务器，用send()和recv()向服务器发送信息和从服务器接收信息 | 创建socket对象，用bind()绑定ip地址和端口号到socket上，用listen()开始监听。用accept()阻塞接受客户端的连接请求，用send()和recv()发送信息和接收信息 |
| 基于UDP的socket | 创建socket对象，用bind()绑定ip地址和端口号到socket上，向服务器发送请求，并等待接收服务器回复的响应。 | 创建socket对象，并绑定ip和端口号。服务器用recvfrom()循环等待用户请求，并在接收到请求后发送对应响应。 |

基于UDP的socket客户端不需要事先connect到服务器端，但需要绑定ip地址和端口号；服务器端不需要阻塞接受客户端的连接请求