**实验报告：Lab7 Socket Programming(基于python)**

|  |  |  |
| --- | --- | --- |
| **课程名称**：计算机网络实践 | **年级：**大二 | **上机实践成绩：** |
| **指导教师**：章玥 | **姓名**：邱吉尔 |  |
|  | **学号**：10235101533 | **上机实践日期：2024/12/30** |

1. **需求分析**

目的：实现一个简单的客户端和服务端的通信系统

功能需求：客户端应能够向服务器发送消息，服务器应能够接收并显示消息，客户端和服务器应该支持多个连接，服务端也能向客服端发送信息

输入与输出： 系统的输入是客户端输入以及从文件读取的数据

输出是服务器和客户端都能打印接收到的消息并且每次收到信息都会自动换行以区分

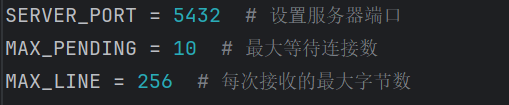
**初步设计思路：**

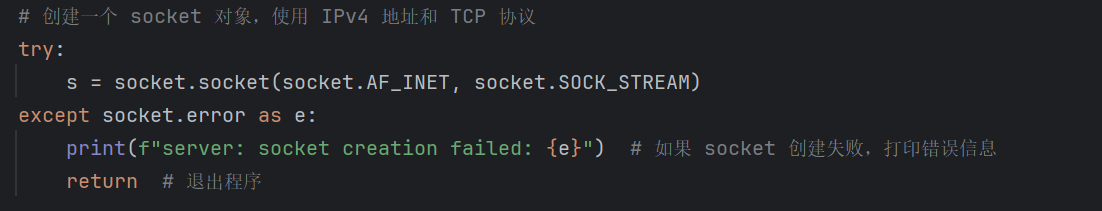
客户端需要连接到指定的服务器IP和端口，发送输入的消息

服务器需要监听指定的端口，接收并打印客户端发送的消息

1. **设计**
2. **Server服务器端**
3. **实现能在标准输出打印客户端发送的消息**  
   

首先导入socke模块，提供网络通信的基本功能，可以创建套接字、连接远程主机、接收和发送数据等



设置服务器监听的端口号以及每次从客户端接收的最大字节数  
这里为了完成后续实验的要求顺便也设置了下最大连接数  
  
接着使用socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)：创建一个 TCP 套接字（IPv4 地址族和 TCP 协议）

socket.AF\_INET 表示使用 IPv4 协议

socket.SOCK\_STREAM 表示使用 TCP 协议

如果创建套接字失败，捕获异常并打印错误信息，程序终止



使用bind将服务器的套接字绑定到本地的 IP 地址和端口号

'0.0.0.0' 表示绑定到所有可用的网络接口，服务器可以通过任何可用的网络接口接收请求

SERVER\_PORT 指定服务器监听的端口号,即之前设置的5432



调用listen() 方法将套接字设置为监听状态，准备接收客户端的连接请求

MAX\_PENDING 表示最大等待连接数。即最多允许 5 个客户端等待连接



输出一下正在监听哪个端口，便于debug



while True：使服务器不断接受新的客户端连接

s.accept() 阻塞直到有客户端连接到来，返回两个值：

new\_s：一个新的套接字，用于与客户端通信

addr：客户端的地址信息（如 IP 地址）

如果连接出现错误，会捕获异常并打印错误信息，然后继续等待新的连接



new\_s.recv(MAX\_LINE)：接收客户端发送的数据，最多接收 MAX\_LINE 字节的数据，即256字节

如果客户端关闭连接，recv() 返回空字节串，表示连接已关闭

data.decode('utf-8')：将接收到的字节数据解码为字符串，使用 UTF-8 编码

1. **实现支持5个以上客户端同时发送消息并逐一打印**

要使服务器能够支持多个客户端并发连接，并且能够同时接收消息并逐一打印，需要在服务器端实现多线程处理，每当一个客户端连接时，服务器会为该客户端创建一个独立的线程来处理通信，这样就能同时处理多个客户端



首先导入threading来实现线程相关操作



添加handle\_client函数用于处理接收到的信息，添加输出的信息便于后续的测试

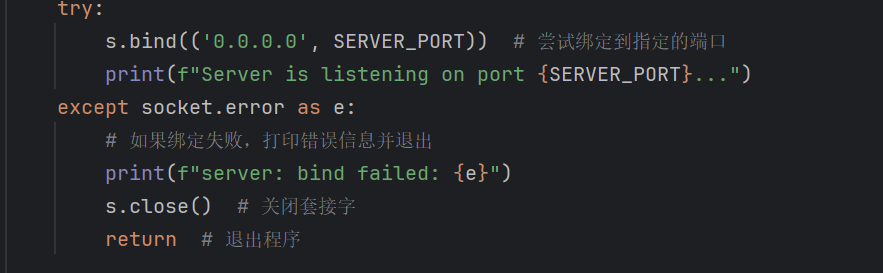
接着我们来看看main函数的改动：



在每次接受到新的客户端连接后，都会使用threading.Thread为该连接创建一个独立的线程来处理，这样就可以让服务器同时处理多个客户端的请求，同时每个客户端都通过独立的线程接收和打印消息

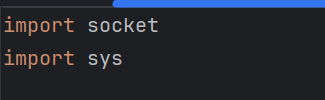
1. **绑定至错误的端口号时能提示出错信息**

为了实现这一功能就需要捕获 socket.error 异常。而socket.bind会抛出 OSError 或 socket.error 异常，如端口号已被占用，或没有足够权限绑定该端口，都会导致绑定失败  
修改代码如下：



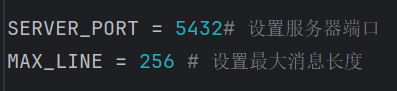
如果端口绑定失败，就会打印错误信息 server: bind failed: {e}

1. **Client用户端**
2. **实现能从标准输入或文件接收消息**



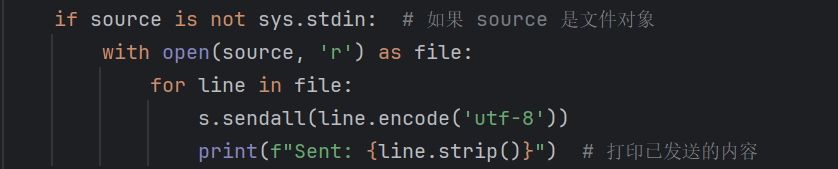
socket：用于进行网络通信（创建套接字，连接，发送/接收数据等）。

sys：用于处理命令行参数和系统退出。



指定客户端连接的服务器端口号和指定最大消息长度

接着定义了send\_message 函数，作用是将数据发送到服务器，两个参数分别是是已经建立连接的socket以及消息的来源，可以是标准输入或文件路径，来看看函数的具体实现：



如果消息来源不是标准输入，即提供了一个文件路径，程序会打开该文件并逐行读取内容

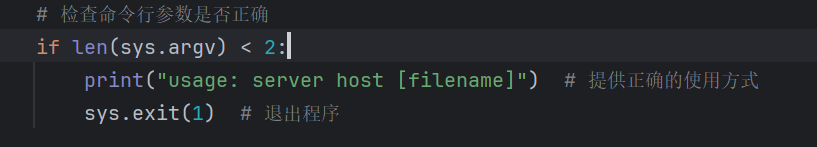


如果消息来源是标准输入，则从键盘逐行读取消息

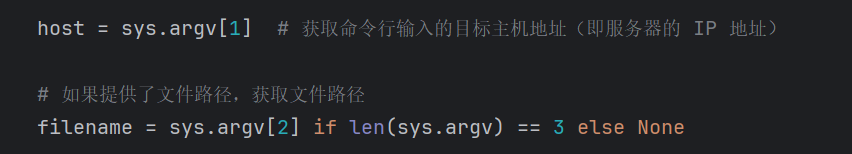
如果按下 Ctrl+C，则退出输入循环。

如果输入为空即连按两次回车，退出循环。

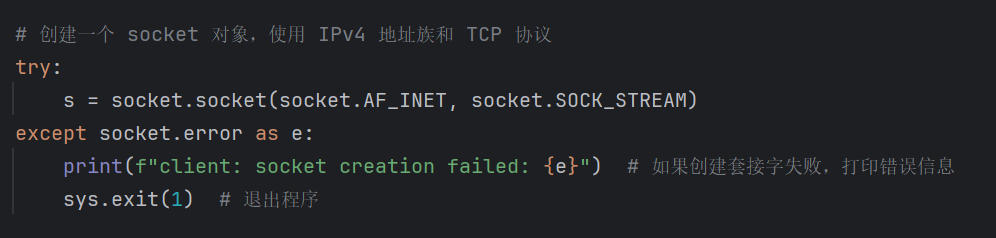
接着我们来看main函数的具体实现：



首先检查命令行参数，确保至少提供了一个目标服务器地址



获取命令行参数中的目标主机地址，且如果提供了文件路径作为第二个命令行参数，则 filename 为该路径；如果没有，则为 None



接着创建一个套接字，指定地址族为 AF\_INET（IPv4），协议为 SOCK\_STREAM（TCP）



然后使用gethostbyname函数将目标主机的域名解析为 IP 地址



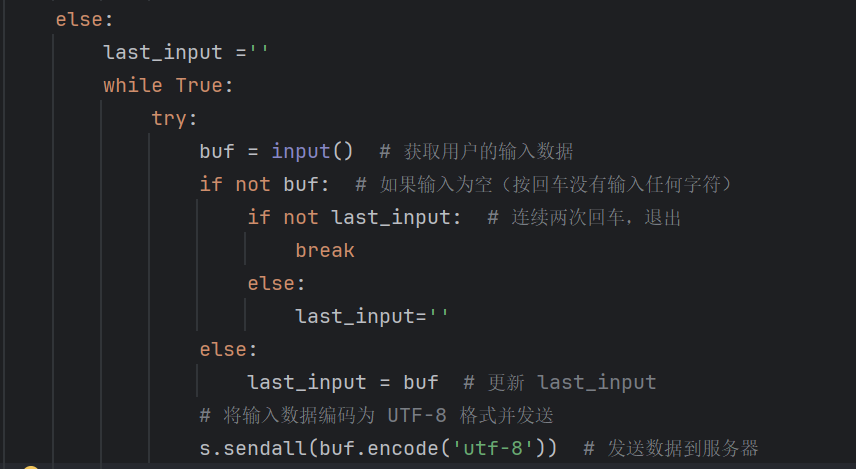
使用 connect 方法连接到目标服务器，连接的地址是 IP 地址和指定的端口，即5432



最后调用send\_message发送消息，就完成了客户端最基础的功能

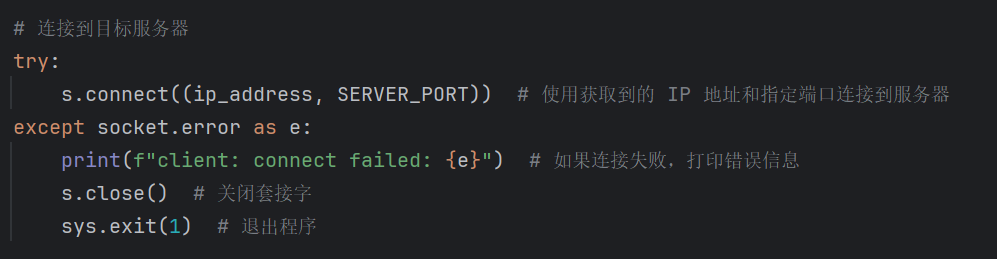
1. **实现标准输入消息以两次回车作为结束标志**

为了实现这个功能，则我们需要记录上一次的输入，只有当上一次输入以及当前输入都为空，也就是连按了两次回车后，才结束输入，所以要修改的是send\_message函数：



引入了last\_input用于记录上一次的输入，两次输入都为空时才break

1. **连接至错误的IP地址/端口号时能提示出错信息**

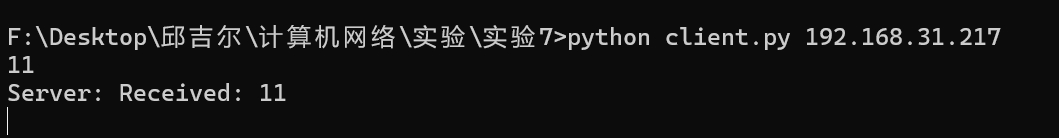


在 s.connect() 之前，添加了try-except 块。这样，当无法连接到指定的ip地址或端口时，能够捕获 socket.error 异常并打印详细的错误信息

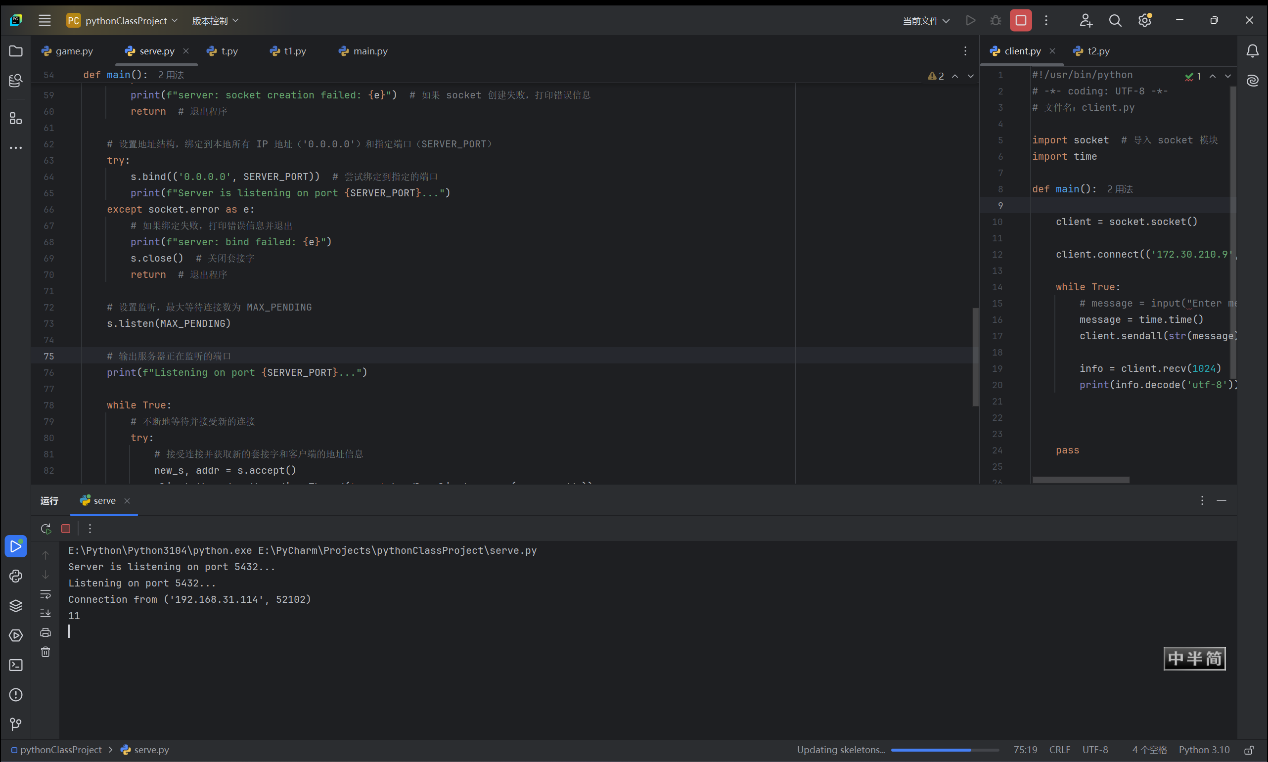
1. **整体**
2. **支持在localhost及在两台不同机器上运行**

之前的用例都是在localhost上运行，肯定支持，在两台机器上运行时只要在运行client.py时输入正确的服务器地址即可，如：python client.py 192.168.31.114

以下是在我的电脑上运行客户端：



在室友电脑上运行服务器端：



可以看到可以成功链接

1. **支持长文本消息 （不少于 20KB），有缓冲区管理**

要支持长文本消息，首先要将client和server端的缓冲区大小设置的大些，这里就以4096字节为例，先修改server.py,在其中增加数据拼接逻辑：

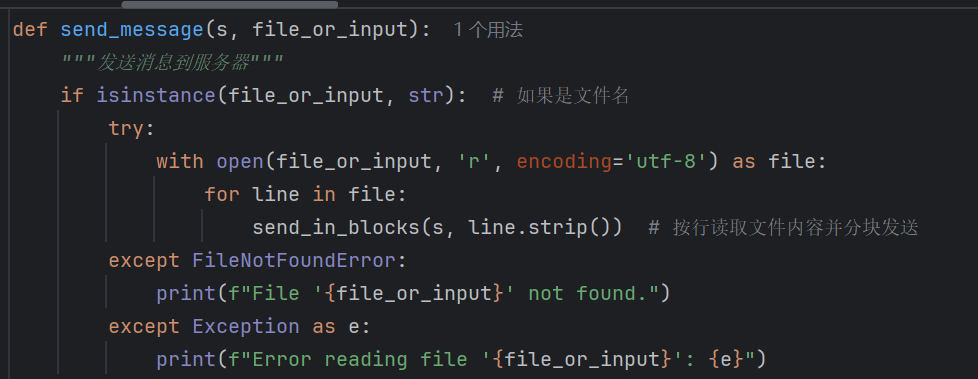
在 handle\_client 函数中使用循环不断调用 recv()

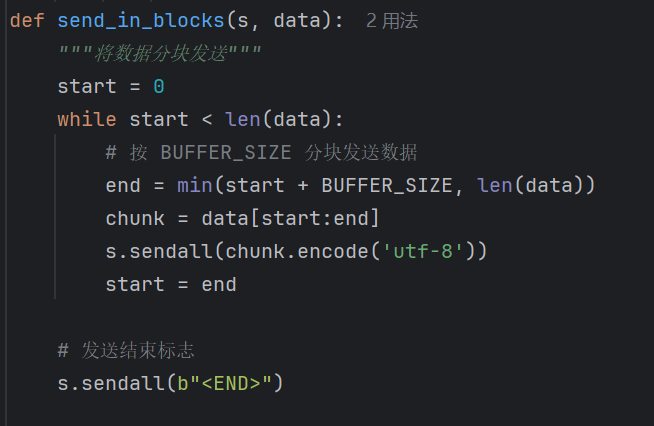
拼接收到的数据，直到客户端关闭连接或发送结束标志

以下为具体修改部分：

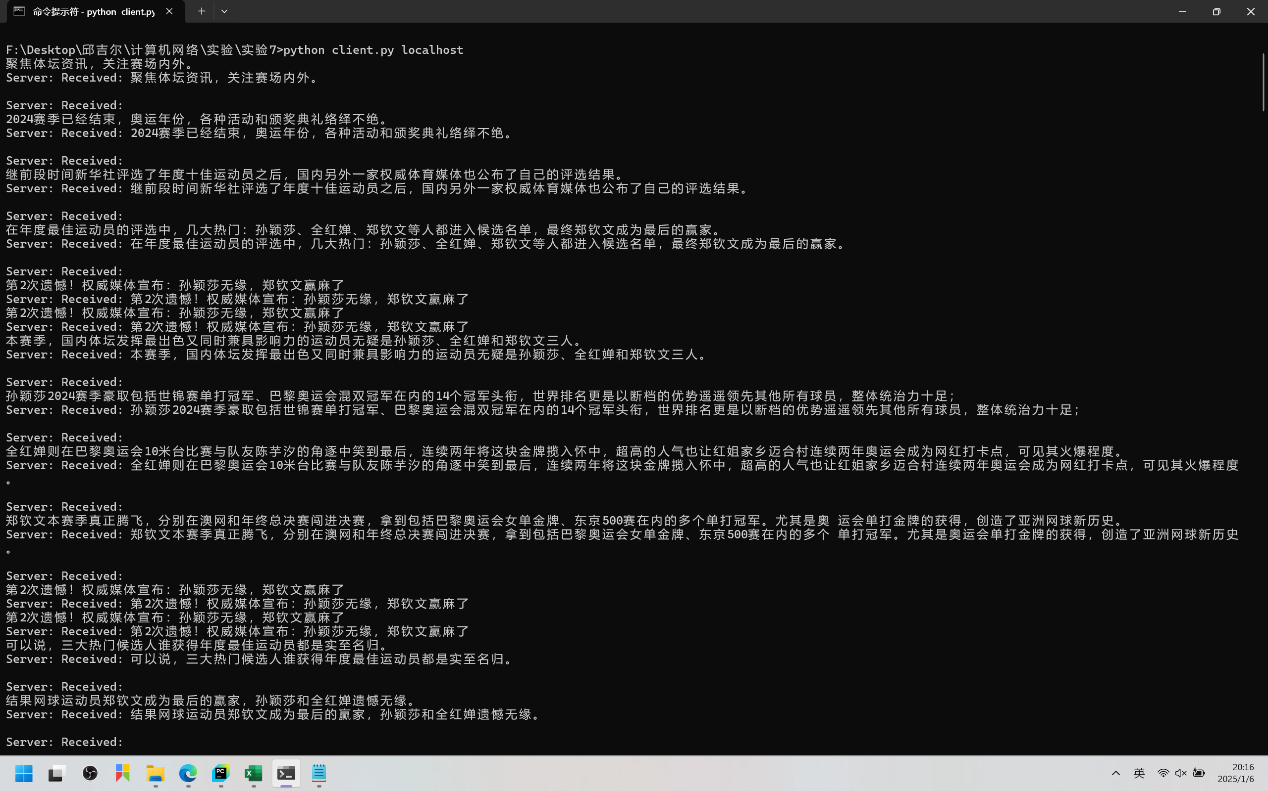


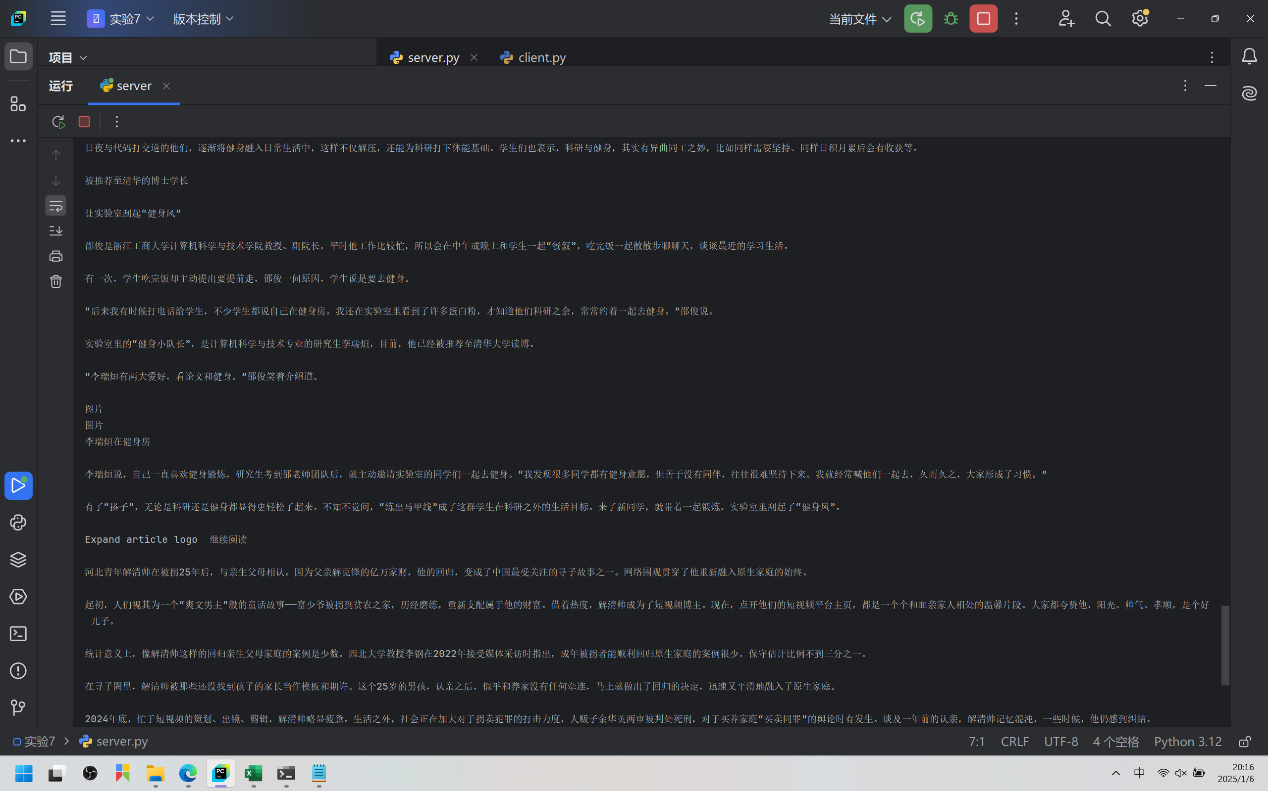
然后我们再修改客户端，需要自定义分块发送函数，将长于缓冲区的消息换分成最大为4096字节的块然后发送，另外，鉴于文本多是采用UTF-8编码，故显式指定发送消息的编码类型，以下为具体代码改动：





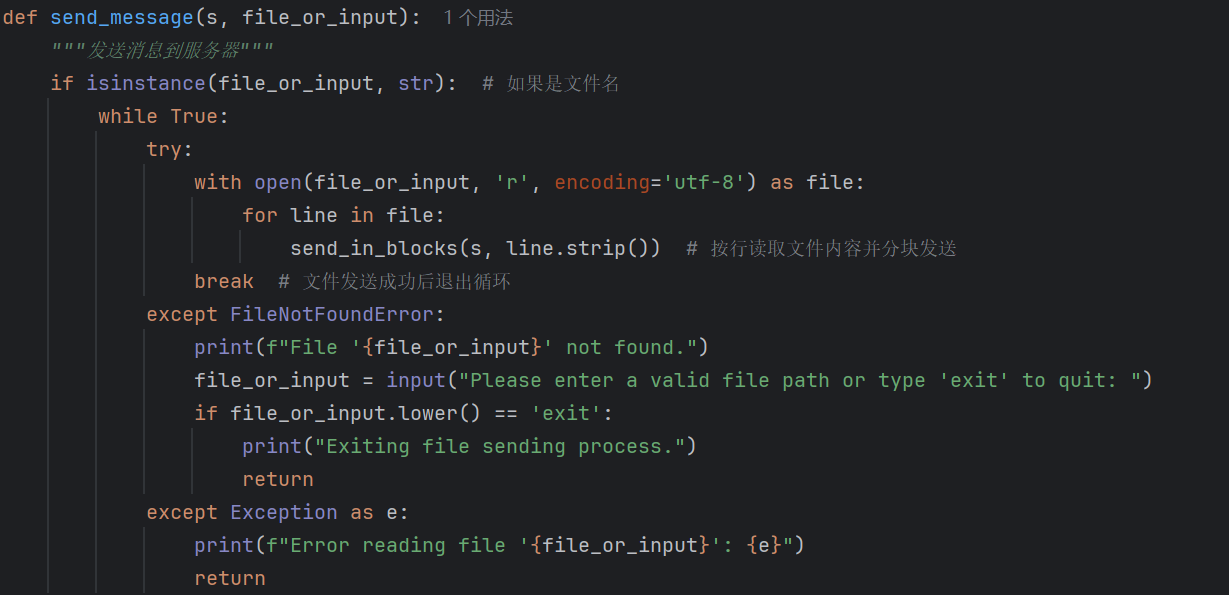
长文本发送演示：





可以看到能成功接受，（刚刚课上演示不成功是因为我长文本直接从网页上扒下来的，然后排版上有两个连续的回车所以识别成退出符了）

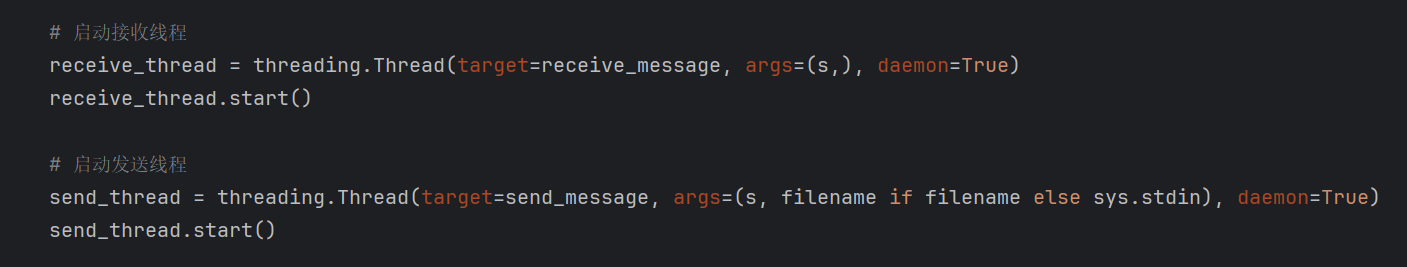
1. **容错性好，无闪退**

容错性方面有意在最初编写时就利用try-except来捕获异常，以此可以尽量避免闪退问题，以下是后面所新添加的文件不存在时的异常捕获：

1. **附加**
2. **实现双工通信**
3. 先来看客户端的改动：首先为了让客户端具有接收消息的功能，需要为客户端创建一个独立的线程用于接收来自服务器的消息，由此避免客户端在等待用户输入时阻塞接收数据，增加了**r**eceive\_message函数用于在一个单独的线程中接收数据并打印，以下参考server.py完成编写：



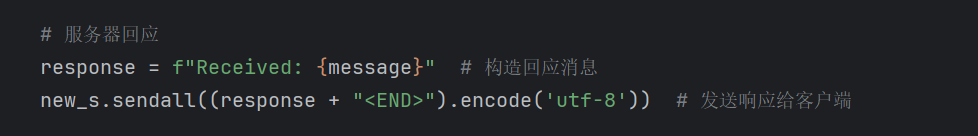
然后在 main 函数中，启动接收线程并等待它的执行。原来发送消息的逻辑保持不变，但现在在发送消息之前启动了接收线程，确保客户端可以同时接收和发送数据，为了确保不会错过服务端的消息，因此先启动接收线程：



1. 接着来看服务器端的改动：要让服务器也能发送消息给客户端，需要改造 handle\_client 函数，使其可以对接收到的消息构造响应并建立一个独立线程，允许服务器主动向客户端发送消息。

同时让服务器继续监听新的客户端连接，每个连接都会创建一个新的线程。这样，每个客户端的通信都能独立于其他客户端进行。



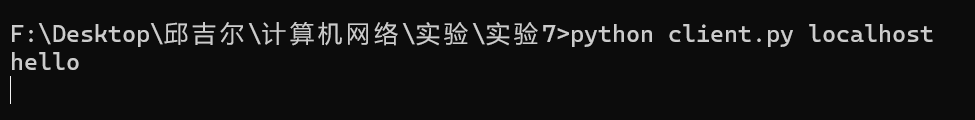


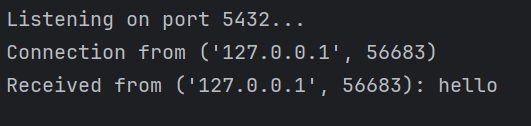
1. **实现**

源码见附录

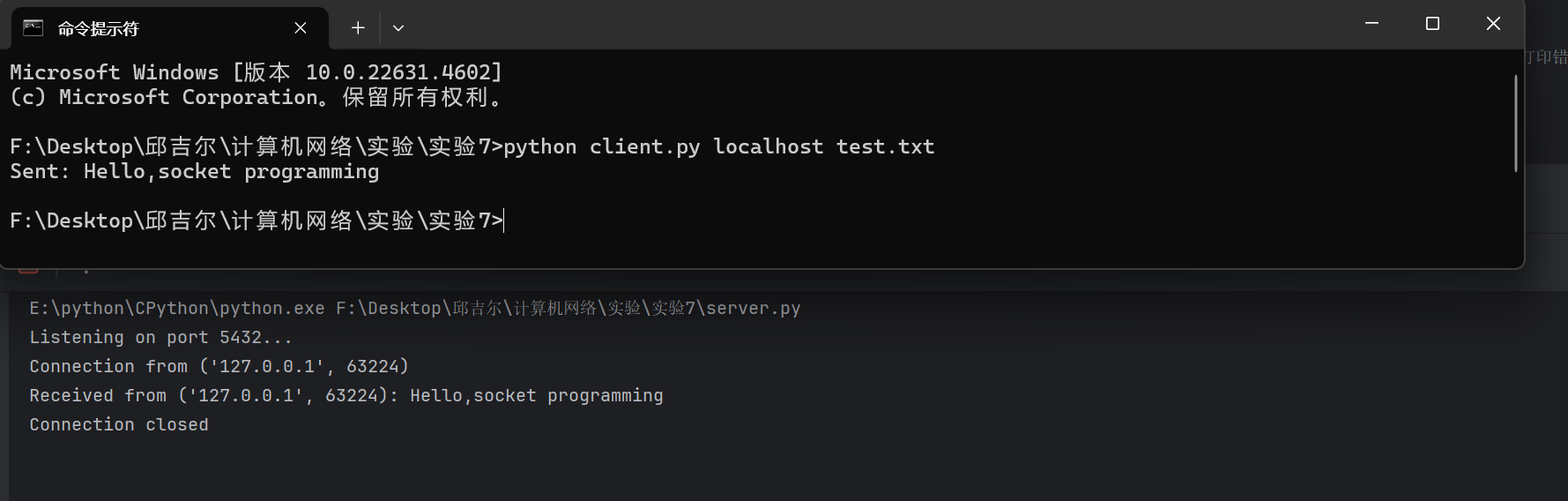
1. **测试**

* **首先是基本的输入输出：**



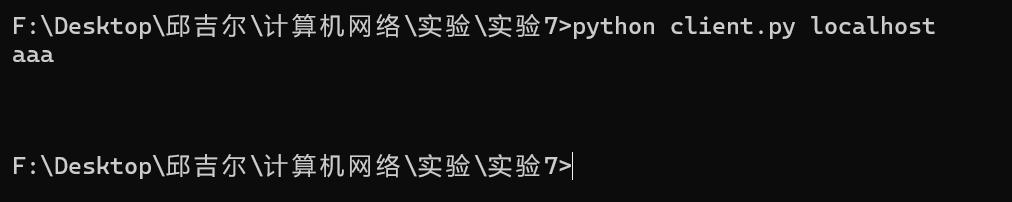


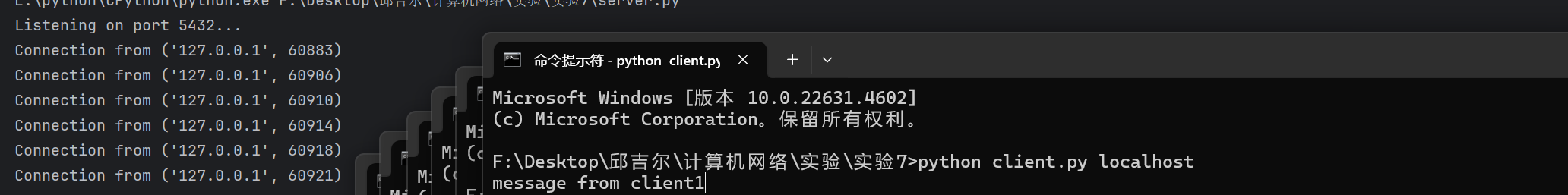
* **然后测试从文件接收信息输出：**



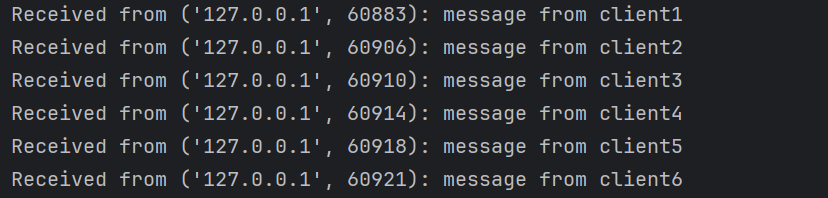
可以看到client支持从文件读入

* **接着是两次回车作为结束输入的标志：**



* **接着测试是否支持5个以上客户端同时发送消息并逐一打印：**  
  

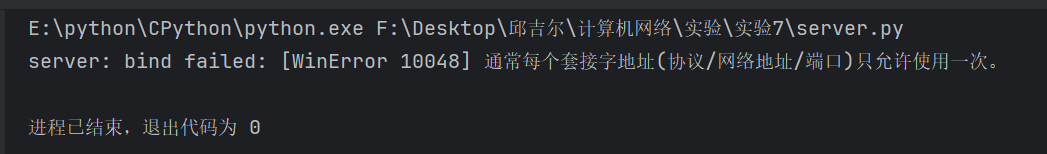
由服务端打印信息（左侧）可知，已经完成同时与6个client的连接



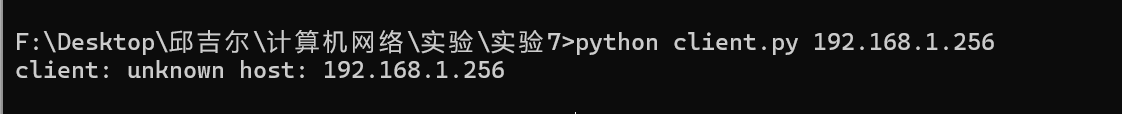
发送信息后可以发现已经逐一打印

* **然后测试服务器绑定至错误的端口号时是否能提示出错信息：**

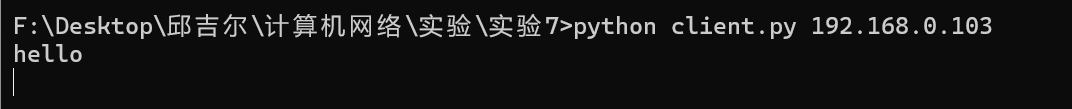
为了进行该测试我们首先需要编写一个同样占用5432端口的程序，这里为了偷懒（bushi）就直接把server.py的内容复制到了test.py里,然后我们先启动test.py再启动server.py，就会发现server.py已经报错：

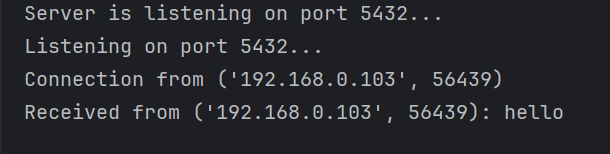


* **那么再测试客户端连接至错误的IP地址/端口号时能否提示出错信息：**



* **接着测试可否在不同主机上运行：**



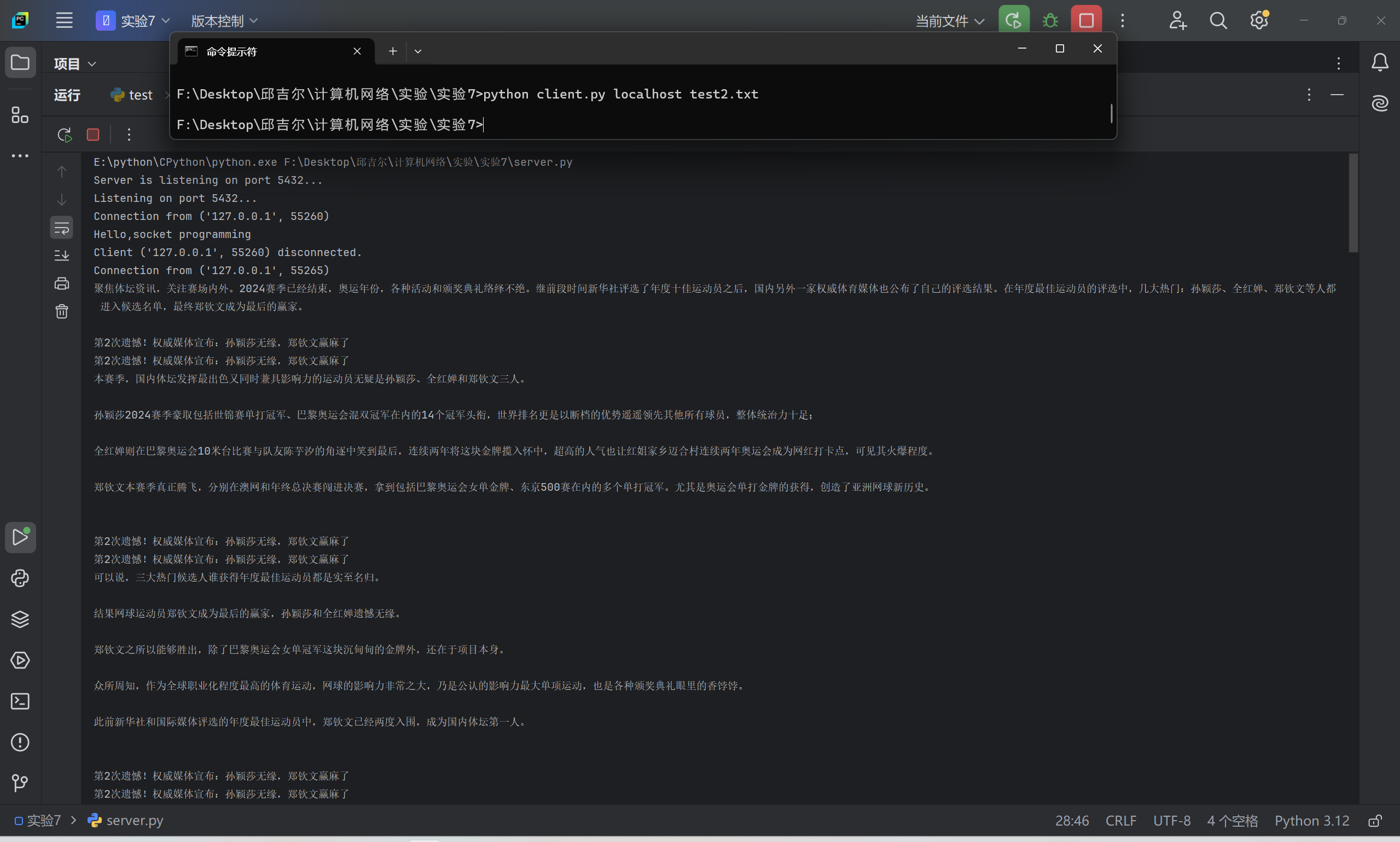


可见可以完成

* **测试是否支持长文本消息传输：**

这里以这个21KB的文件做测试





可以看到已经完整打印了出来（未完全截图）

* **测试是否支持双工通信：**



左侧绿色字体为服务器端发送的信息，可见已经在客户端显示出来，至此已完成所有功能的测试