

**《计算机网络》实验报告**

(2022~2023 **学年第一学期**)

## 实验名称：Socket编程

## 学 院：软件学院

## 姓 名：郁万祥

## 学 号：2013852 指导老师：张圣林

### 2022 年 12 月 6日

实验名称 (实验 4:Socket编程)

# 实验目的

熟悉基于Python进行UDP套接字编程的基础知识，掌握使用UDP套接字发送和接收数据包，以及设置正确的套接字超时，了解Ping应用程序的基本概念，并理解其在简单判断网络状态，例如计算数据包丢失率等统计数据方面的意义。

熟悉基于Python进行TCP套接字编程的基础知识，理解HTTP报文格式，能基于Python编写个可以一次响应一个HTTP请求，并返回静态文件的简单Wb服务器。

进一步理解和掌握基于Python进行TCP套接字编程的知识，理解SMTP报文格式，能基于Python编写一个简单的SMTP客户端程序。

# 实验条件

装有python环境的电脑两台

局域网环境

已经正常运行的邮件服务器

# 实验报告内容及原理

**3.1 套接字基础与UDP通信**

（源代码见附件）

实验原理：

基于UDP的无连接客户/服务器在Python实现中的工作流程如下：

1.首先在服务器端通过调用socket()创建套接字来启动一个服务器；

2.服务器调用bind()指定服务器的套接字地址，然后调用recvfrom()等待接收数据。

3.在客户端调用socket()创建套接字，然后调用sendto()向服务器发送数据。

4.服务器接收到客户端发来的数据后，调用sendto()向客户发送应答数据，

5.客户调用recvfrom()接收服务器发来的应答数据。

6.一旦数据传输结束，服务器和客户通过调用c1ose()来关闭套接字。

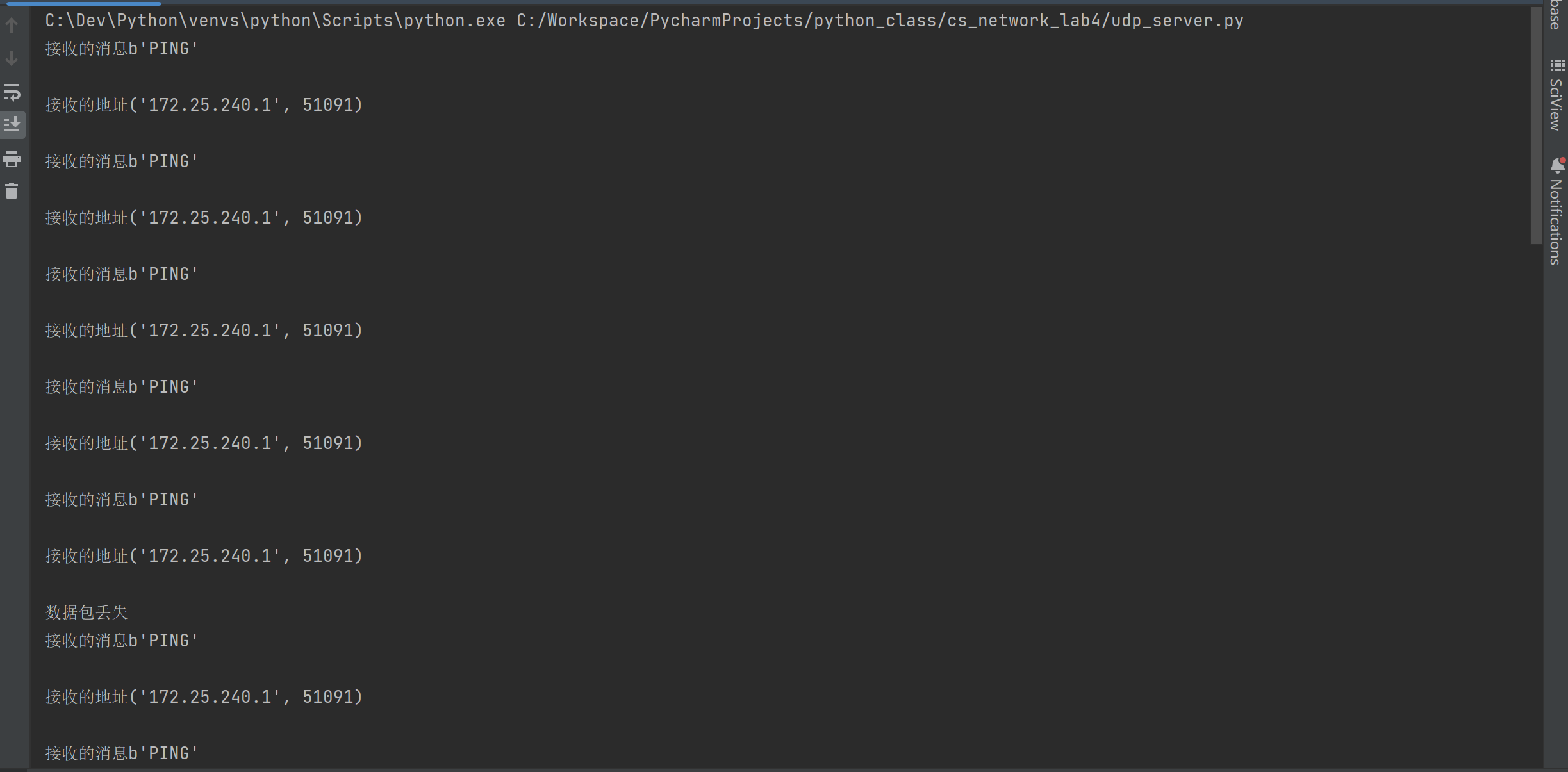
实验步骤：

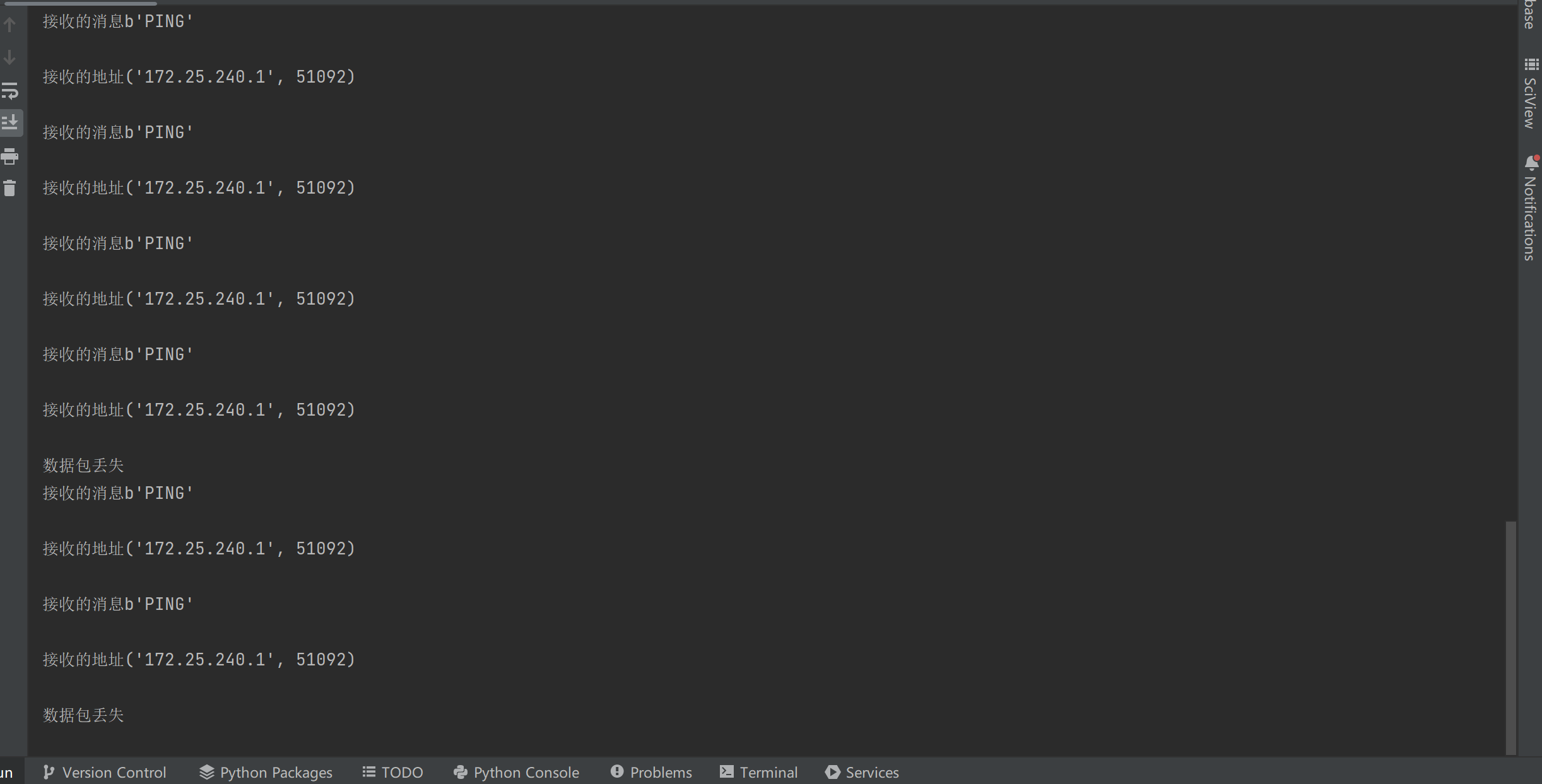
1、运行服务器端代码。

2、使用UDP发送ping消息（注意：因为UDP是无连接协议，不需要建立连接。）：

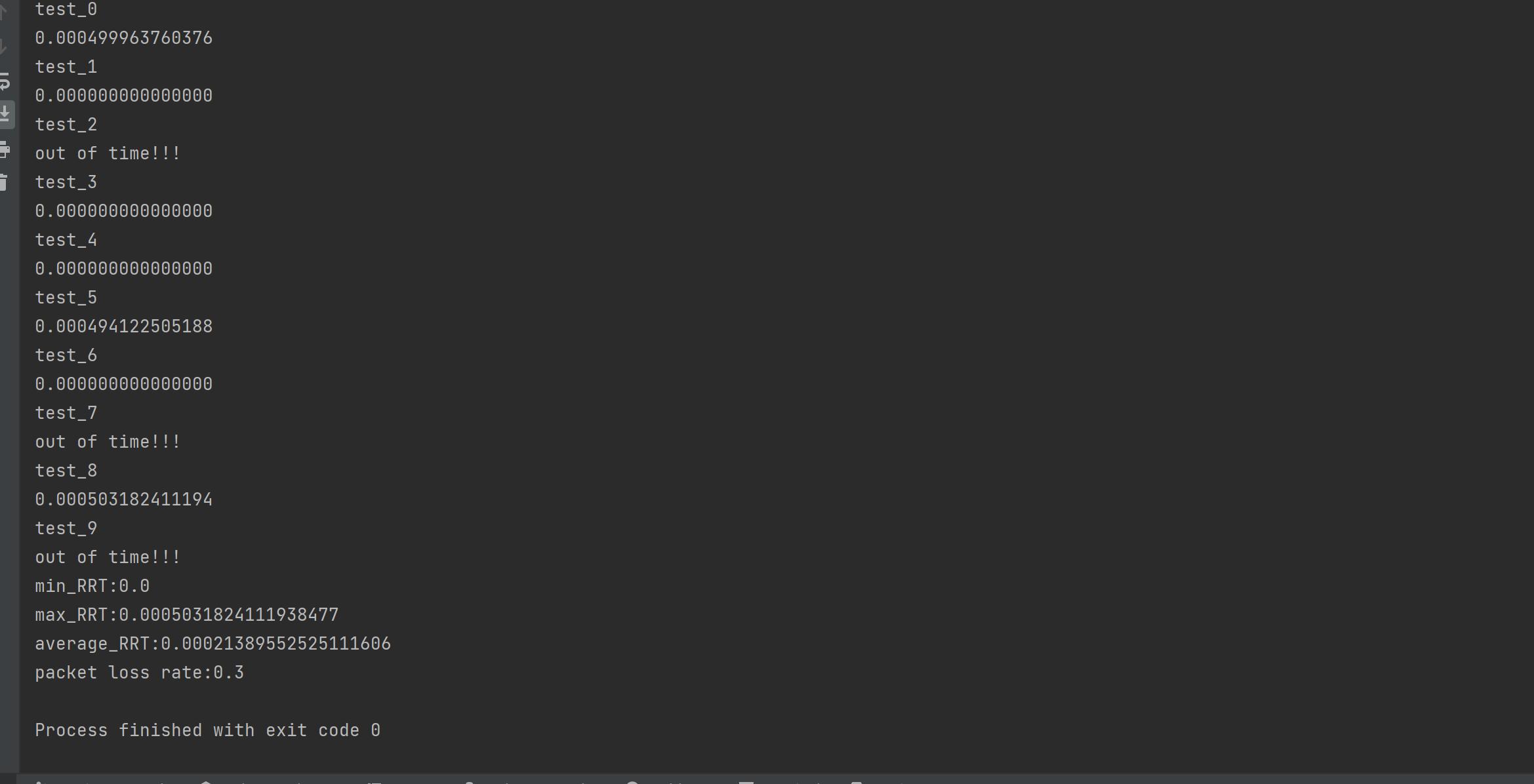
3、如果服务器在1秒内响应，则打印该响应消息；计算并打印每个数据包的往返时间RTT(以秒为单位)：

4、否则，打印“请求超时”（中英文皆可)。





我们发现，两次运行客户端命令，显示的随机生成的端口会不同，51091和51092



需要提到的是：此时使用的服务器端是使用本机的ip地址，所以，此时的实验，就相当于本机ping本机的命令，这与之前的实验相似，所以就算使用python套接字编程，本质上也是实现了本机ping本机的命令。

但是原理是一样的，如果将服务器端的代码运行在另一台装有python环境的服务器上，那么ip地址就会变化，这样的话，就可以进行正常的客户机ping服务器的过程了。

实验中问题以和解决方法

实验中遇到的问题：

1.判断该请求是否超时

2.怎么精确计算RTT时间

解决方法：

1.使用socket的内置函数settimeout来判断请求是否超时，并捕获异常打出超时。

2.利用python内置模块time获取发送和接受数据的时间戳，相减得到RTT时间。

**3.2 TCP通信与Web服务器**

（源代码见附件）

实验原理：

基于TCP的面向客户端/服务器在Python实现中的的工作流程是：

1.首先在服务器端通过调用socket()创建套接字来启动一个服务器；

2.服务器调用bind()绑定指定服务器的套接字地址(P地址+端口号)；

3.服务器调用1 isten()做好侦听准备，同时规定好请求队列的长度；

4.服务器进入阻塞状态，等待客户的连接请求；

5.服务器通过accept()来接收连接请求，并获得客户的socket地址。

6.在客户端通过调用socket()创建套接字；

7.客户端调用connect()和服务器建立连接。

8.连接建立成功后，客户端和服务器之间通过调用read()和write()来接收和发送数据。

9.数据传输结束后，服务器和客户各自通过调用c1ose()关闭套接字。

实验步骤：

1.服务器收到请求时能创建一个TCP套接字：

2.可以通过这个TCP套接字接收HTTP请求；

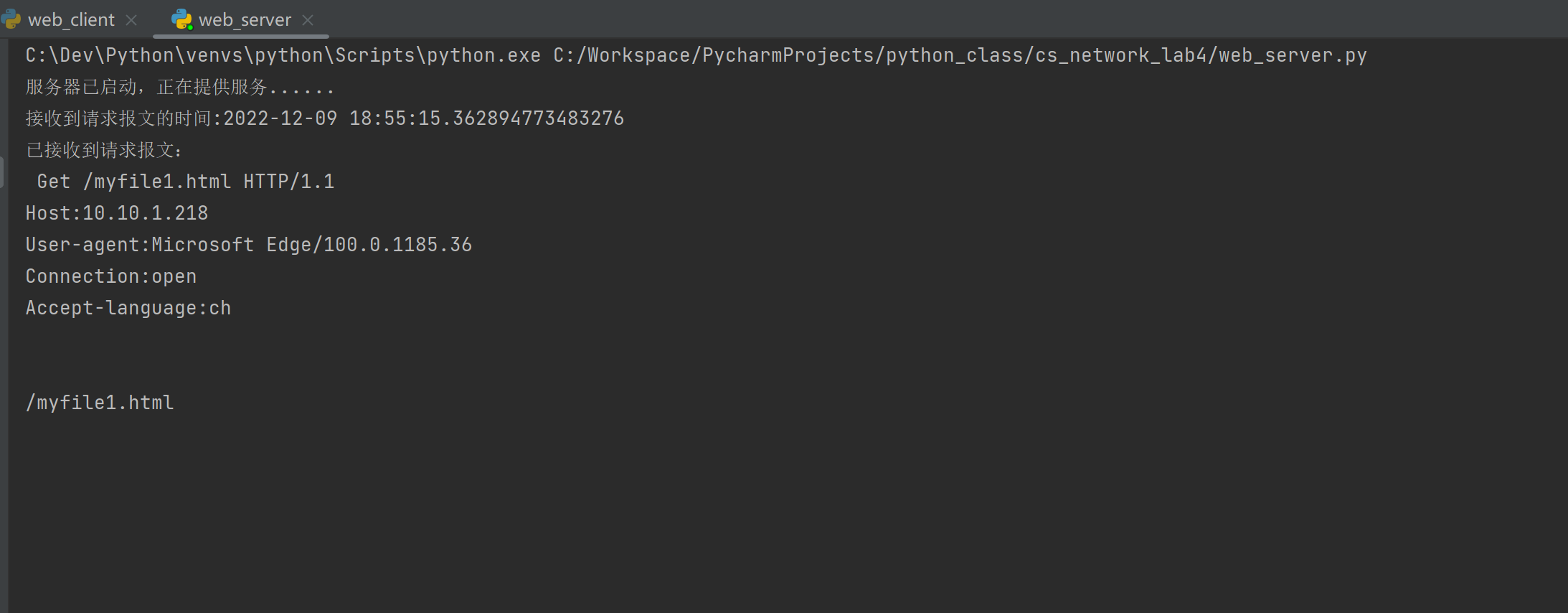
3.解析HTTP请求并在操作系统中确定客户端所请求的特定文件；

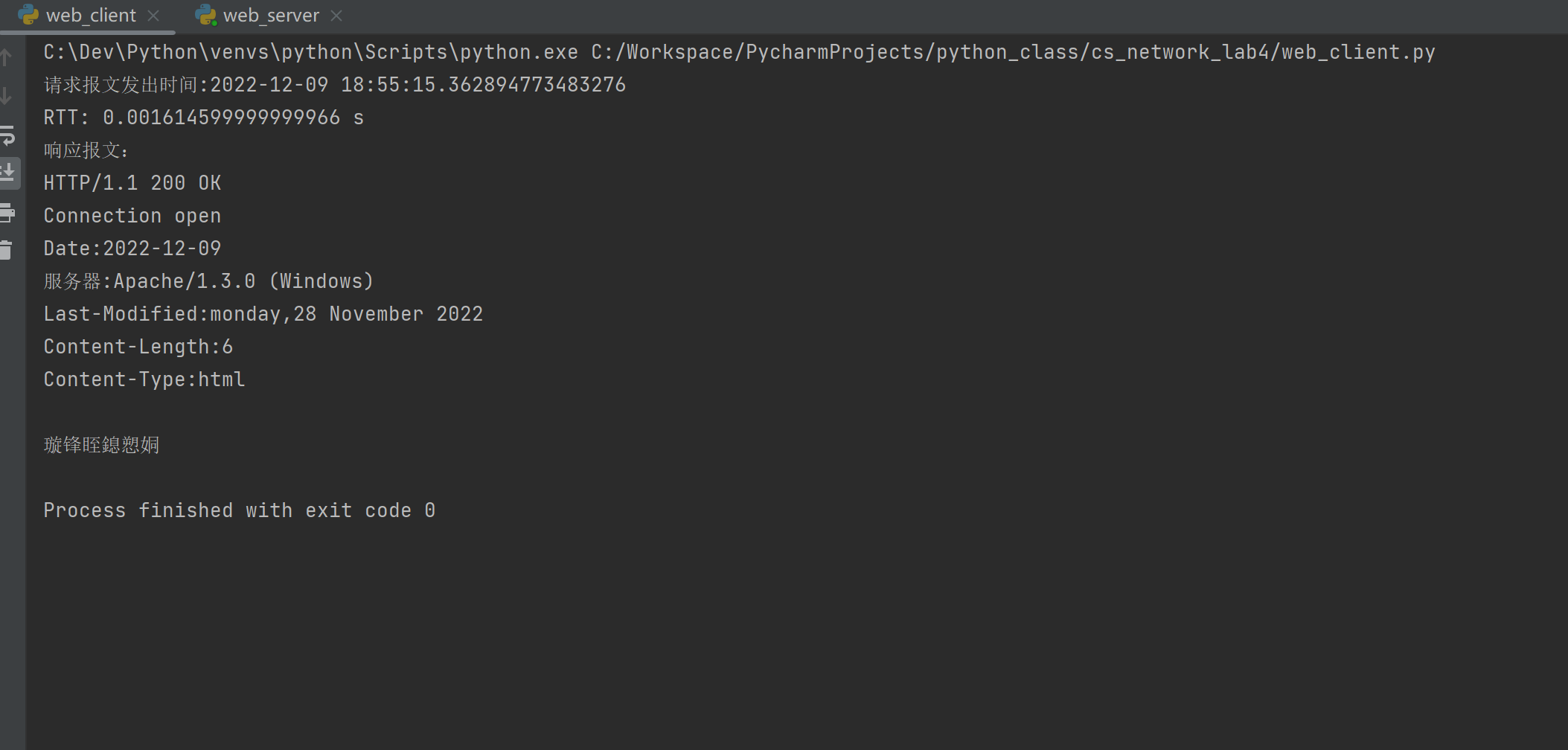
4.从服务器的文件系统读取客户端请求的文件；

5.当被请求文件存在时，创建一个由被请求的文件组成的“请求成功”HTTP响应报文；

6.当被请求文件不存在时，创建“请求目标不存在”HTTP响应报文；

7.通过TCP连接将响应报文发回客户端；

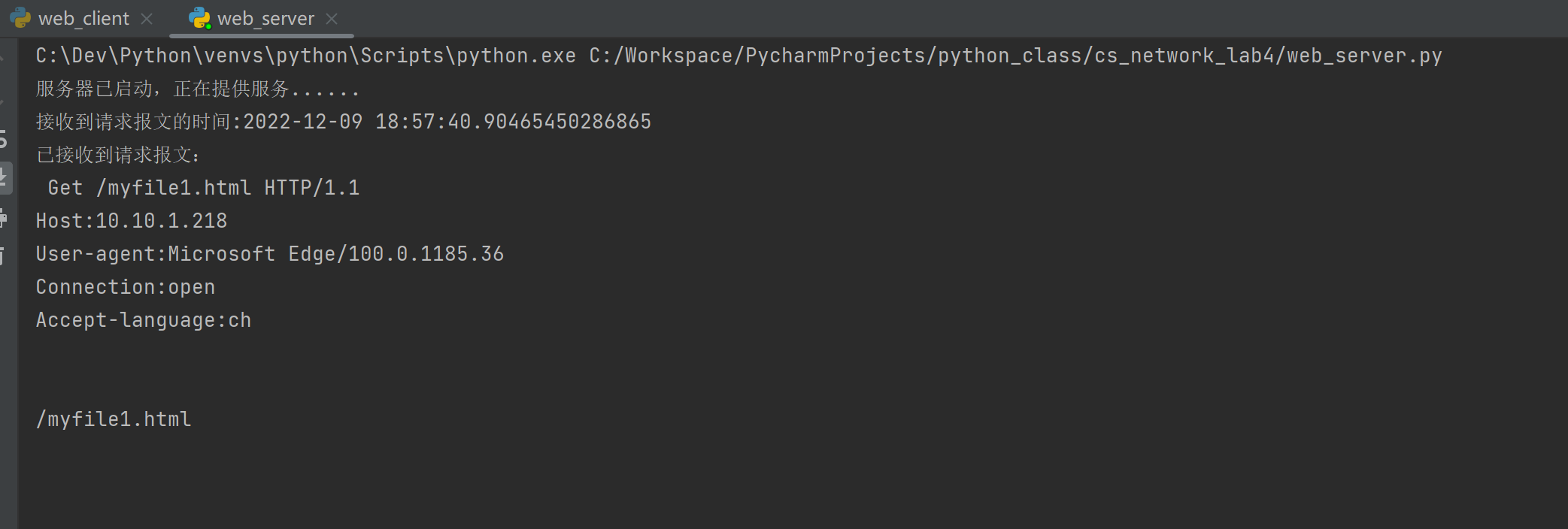


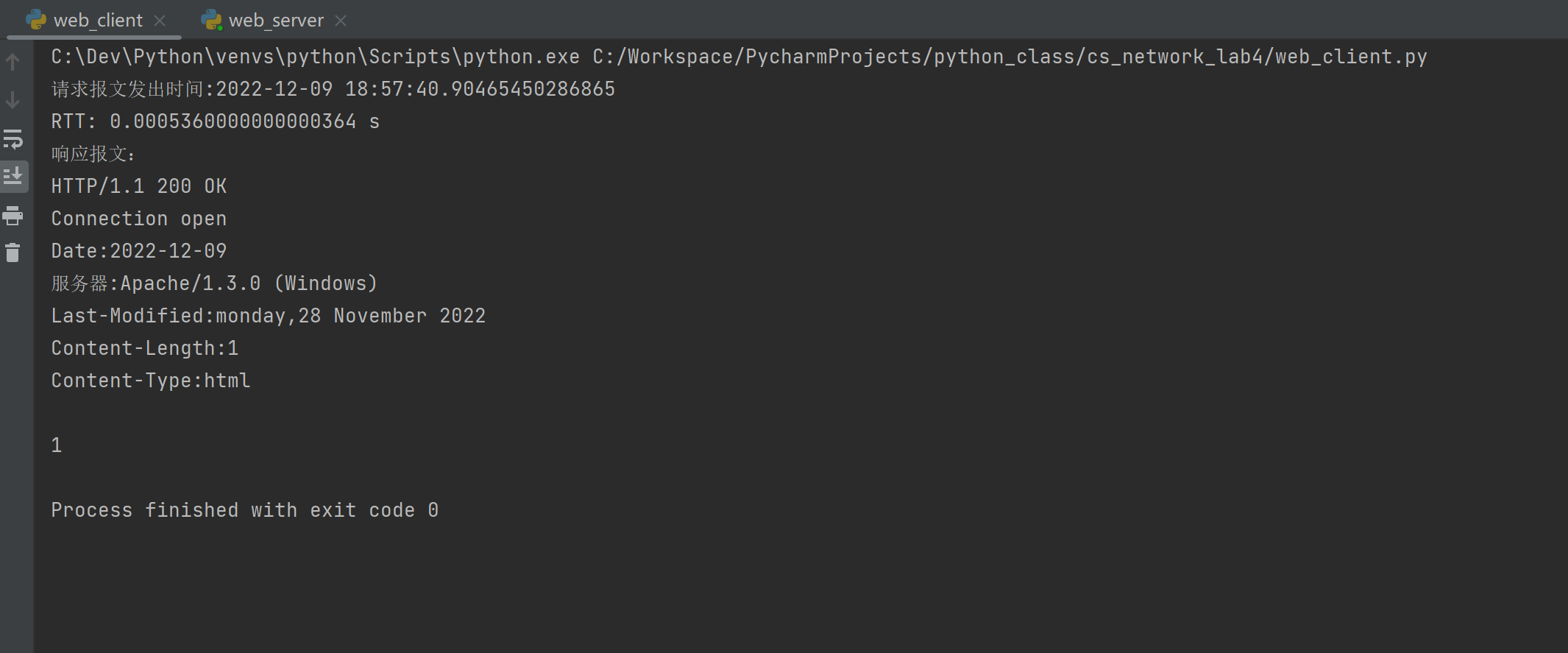


由于pycharm解码问题，正常输出的：

请求成功

结果输出了乱码，不过这并不影响，为了确保实验的正确性，我们改变成数字，进行测试





结果显示为：1（设定的请求成功页面），因此可以验证工作的正确性。

**3.3 SMTP客户端实现**

（源代码见附件）

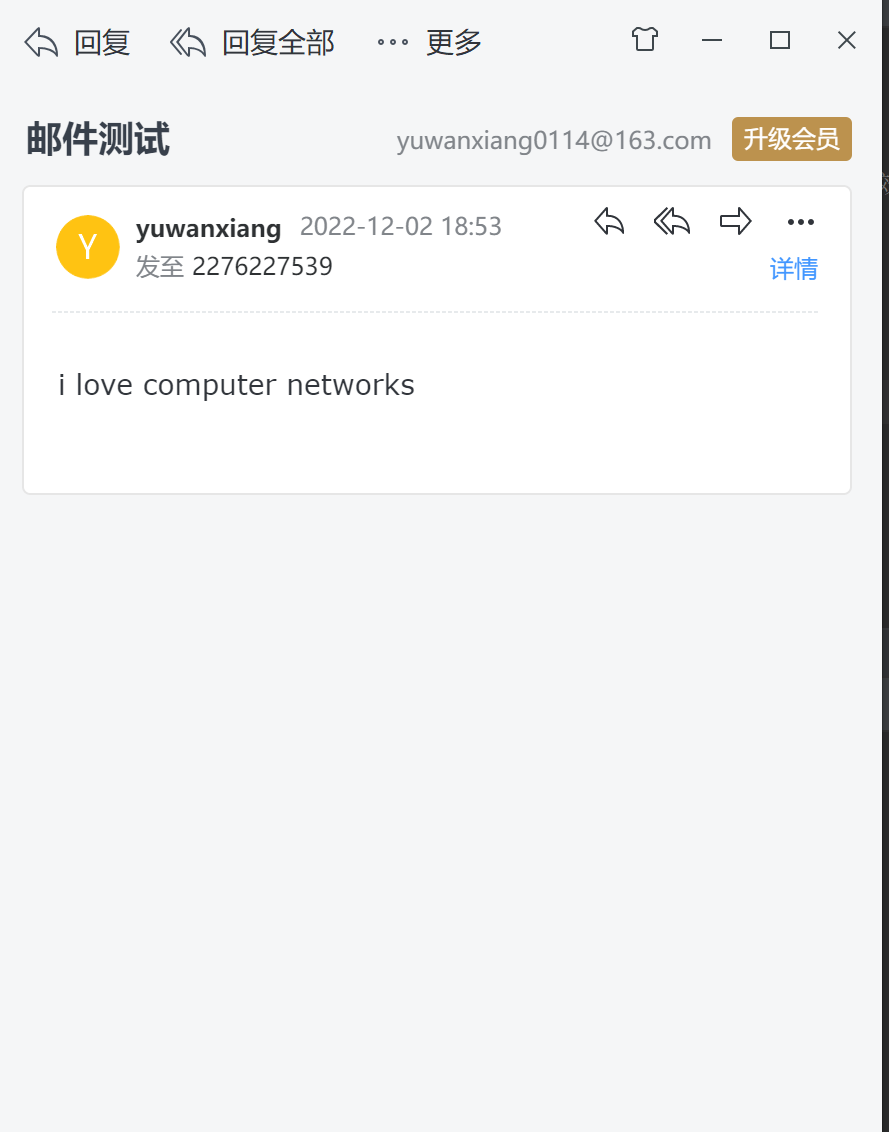
实验原理：

简单邮件传输协议(Simple Mail Transfer Protocol,SMTP)是实现电子邮件收发的主要应用层协议，它基于TCP提供的可靠数据传输连接，从发送方的邮件服务器向接收方的邮件服务器发送邮件。注意，虽然一般情况下邮件总是从发送方的邮件服务器中发出，但是工作在发送方邮件服务器上的发送程序是一个SMTP客户端，因此一个完整的SMTP程序总有两个部分参与工作：运行在发送方邮件服务器的SMTP客户端和运行在接收方邮件服务器的SMTP服务器。

实验步骤：

运行程序，完成SMTP客户端实现

完成邮箱设置后运行实验代码，检验结果。

****

# 实验结论及心得体会

实验中出现的问题总结一下：

1. 此时使用的服务器端是使用本机的ip地址，所以，此时的实验，就相当于本机ping本机的命令，这与之前的实验相似，所以就算使用python套接字编程，本质上也是实现了本机ping本机的命令。但是原理是一样的，如果将服务器端的代码运行在另一台装有python环境的服务器上，那么ip地址就会变化，这样的话，就可以进行正常的客户机ping服务器的过程了。
2. 使用socket的内置函数settimeout来判断请求是否超时，并捕获异常打出超时。
3. 利用python内置模块time获取发送和接受数据的时间戳，相减得到RTT时间。
4. 解决TCP与Web实验中html乱码不能验证实验正确性问题。

socket是一种特殊文件，说白了socket是应用层与TCPP协议族通信的中间软件抽象层，它是一组接口。在设计模式中，socket其实就是一个外观模式，它把复杂的TCP协议族隐藏在socket接口后面。对用户来说，一组简单的接口就是全部，让socket去组织数据，以符合指定的协议。其实socket也没有层的概念，它只是一个外观设计模式的应用，我们大量用的都是通过socket实现的。

我们平常说的TCP、UDP是指的行业规范好的通信协议，它们体现为具体的编程模型就是socket编程。Socket是编程接口，TCP和UDP是通信规范，二者相互对应，但不是一个层次的。所以sockwt和TCP的区别，其实就是对协议和接口的理解。