实验三　网络编程

## 一、实验目标

* 练习使用编程语言提供的网络接口；
* 自行设计方案解决 TCP 的“粘包”问题，复习TCP面向字节流的特点；
* 编写简单的客户-服务端程序、web服务器；
* 编写基于UDP的简单ping程序。

## 二、提前阅读

**2.1 实验报告书写要求（重要！）**

**1、需要回答问题的部分会使用红字标出，例如“请使用文字与截图（可选）回答”，没有红色标注的部分不需要回答；**

**2、请将你的回答中的文字设置为“深蓝色”，方便批改；**

**3、回答问题时不抄袭、不直接复制大模型的回答、不堆砌文字、保持简洁、体现自己的思考。**

**2.2 关于编程语言**

**我们的demo示例使用了Python语言。除了Python语言，感兴趣的同学可以使用C、C++、Java、go等编程语言，我们不做编程语言的限制。对于使用其他语言的同学，可以改掉提交实例的后缀，比如我们要求提交test.py文件，而你使用Java语言，则提交test.java文件，其他语言依此类推。**

**2.3 其他注意事项**

1、可以不参考demo代码的注释和示例，可以自由发挥，但是必须符合题目要求。

2、硬编码（按题目要求直接把结果一个个print出来，然后截图）的，本周的作业全部计零分。

## 实验要求：

### 1、简单UDP编程

利用UDP协议，完成下列的程序要求，客户端代码放在code/UDPClient\_Q1.py，服务端代码放在code/UDPServer\_Q1.py

注：我们提供code/UDPClient\_Q1.py作为示例，该代码已经写好，无需修改。

**程序要求：**

1. 服务端开启UDP监听；
2. 客户从其键盘读取一行字符（**小写字母，小于1024字节**）并将该数据向服务端发送；
3. 服务端接受该数据**（接受缓冲为1024字节）**并将这些字符转换为**大写；**
4. 服务端将修改的数据发送给客户；
5. 客户端接受修改的数据并在其控制台上将该行显示出来。

**注：养成良好编程的习惯，记得关闭IO流！**

### 2、简单TCP编程

利用TCP协议，完成下列的程序要求，客户端代码放在code/TCPClient\_Q2.py，服务端代码放在code/TCPServer\_Q2.py

**程序要求：**

1）服务端开启TCP监听；

2）客户从其键盘读取一行字符（**大写字母，小于1024字节**）并将该数据向服务端发送；

3）服务端接受该数据**（接受缓存为1024字节）**并将这些字符转换为**小写；**

4）服务端将修改的数据发送给客户，并关闭连接；

5）客户端接受修改的数据并在其控制台上将该行显示出来，并关闭连接；

6）服务端继续等待下次连接；

**注：养成良好编程的习惯，记得关闭IO流！**

### 3、TCP不中断续传

在2）中，服务端处理完一次响应后就关闭了本次连接，等待下次连接。在本小节中，我们要求服务端可以不断地处理客户端数据，直到客户端要求断开连接。

请利用TCP协议，实现下面的程序要求，客户端代码放在code/TCPClient\_Q3.py，服务端代码放在code/TCPServer\_Q3.py

**程序要求：**

1）服务端开启TCP监听；

2）客户从其键盘读取一行字符（**大写字母，小于1024字节**）并将该数据向服务端发送；

3）服务端接受该数据**（接受缓存为1024字节）**并将这些字符转换为**小写；**

4）服务端将修改的数据发送给客户；

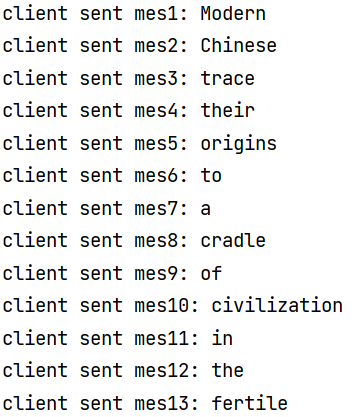
5）客户端持续步骤2），不断地发送数据给服务端，输入“bye”结束发送，并关闭连接；

6）服务端不断处理客户端发来的数据，当收到“bye”的时候关闭与客户端的连接，继续等待下一个TCP连接；

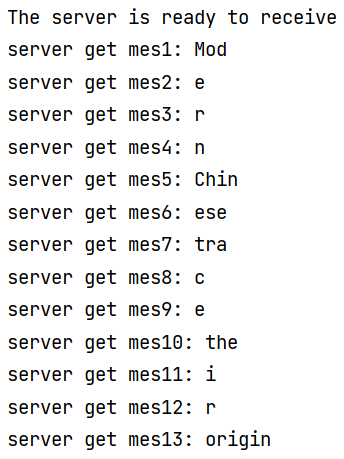
**注：养成良好编程的习惯，记得关闭IO流！**

### 4、TCP粘包问题

运行服务端code/TCPServer\_Q4.py和客户端code/TCPClient\_Q4.py，得到如下的结果。



**图1：客户端运行结果**



**图2：服务端运行结果（以上为示例，请以实际运行结果为准）**

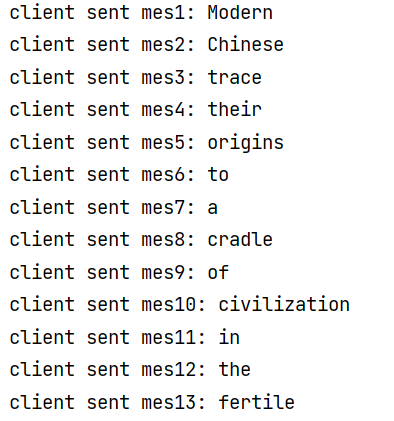
请问为什么客户端发送给服务端时，服务端显示的mes与客户端不同？请使用文字与截图（可选）回答。

请你在原来的代码（code/TCPServer\_Q4.py和code/TCPClient\_Q4.py）的基础上，解决这种问题，**方法不限，硬编码除外**，并保存为code/TCPServer\_Q4\_modi.py和code/TCPClient\_Q4\_modi.py。

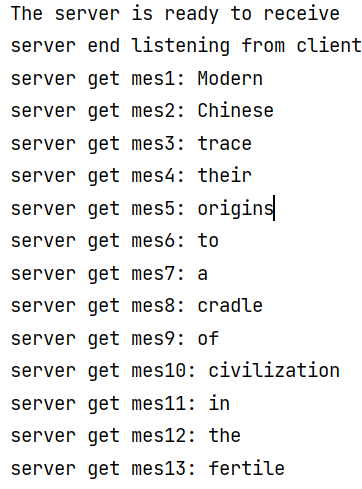
**程序要求：**

1）服务端的接受缓存区为5字节，即recv(5)；

2）客户端每次发送以回车作为结束，回车前的字符串作为一条信息，消息发送顺序必须与服务端打印到控制台的顺序一致，如下图所示：



**图3：客户端发送示例**



**图4：服务端接受示例，应当与客户端保持一致**

### 5、简单 Web 服务器

**要求：**

在上次实验中，你已经学会TCP编程。在本实验中，你将基于TCP开发一个处理HTTP请求的Web服务器。

你的Web服务器应该接受并解析HTTP请求，然后从服务器的文件系统中获取所请求的html文件，创建一个由响应文件组成的HTTP响应消息，然后将响应直接发送给客户端。如果请求的文件不存在于服务器中，则服务器应该向客户端发送“404”状态码的差错报文。

**代码：**

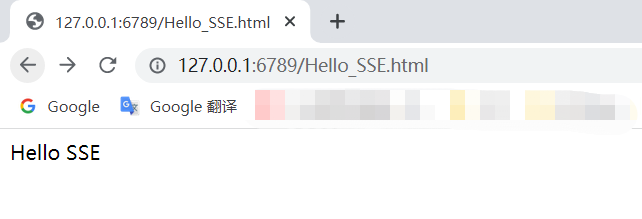
“web\_server\_sample.py”是一个Web服务器的代码框架。你需要在“........”处补充代码。另外，每个地方都可能需要不止一行代码。如果你不喜欢这个模板或者你使用其他语言，可以自行重新编写，只要能满足题目的要求即可。

**运行示例：**

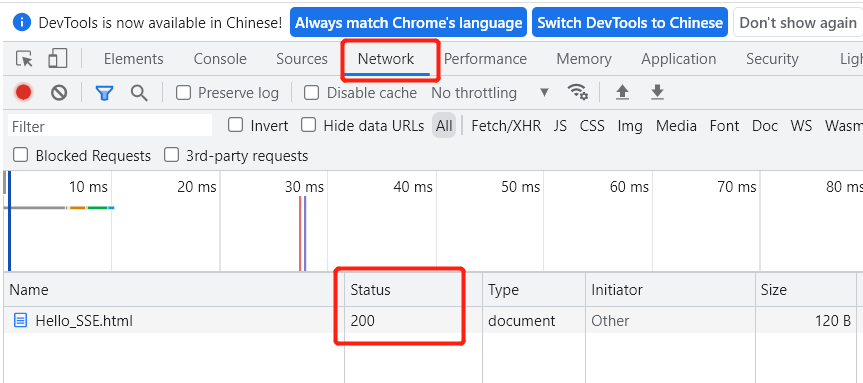
我们提供了一个简单的HTML文件（“web\_server\_code/Hello\_SSE.html”），和404状态码文件（web\_server\_code/404.HTML），请你放在服务器所在的目录中。编写并运行服务器程序。确认自己服务器主机监听的IP地址和端口（例如IP地址为127.0.0.1，端口为6789）。从另一个主机（或本机），打开浏览器并输入相应的URL。这里以Chrome浏览器为例：

1、输入“http://127.0.0.1:6789/Hello\_SSE.html”（不包括双引号）。

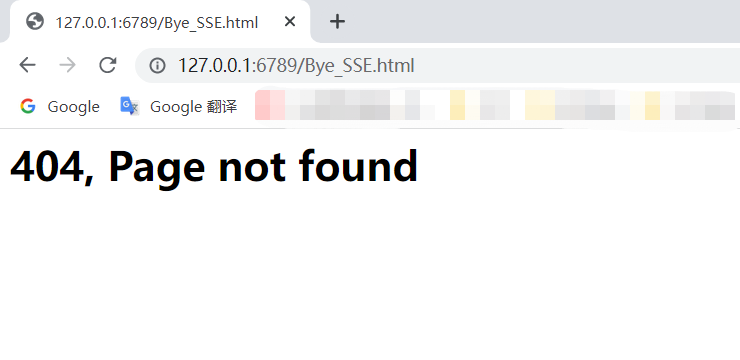
效果应如下图所示：

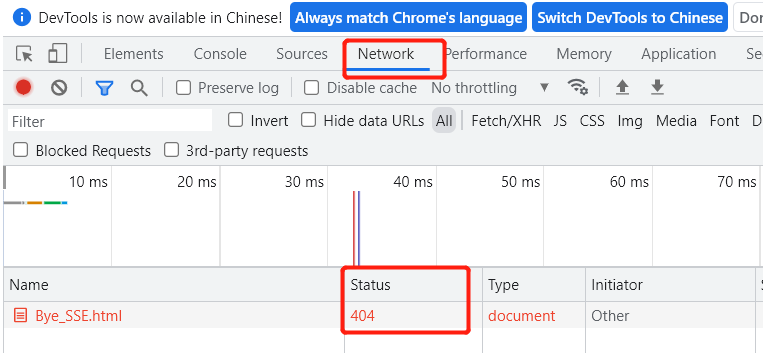


状态码应当是200。查看状态码，可以使用浏览器的开发者工具，以Chome浏览器为例：键盘按下F12->点击Network（网络），就能看到，这里和抓包一样，需要提前打开、开始抓包，才能抓到相应的http请求。



2、输入服务器不存在的资源：比如“http://127.0.0.1:6789/Bye\_SSE.html”,页面应该显示“404.html”文件的内容，状态码为404。





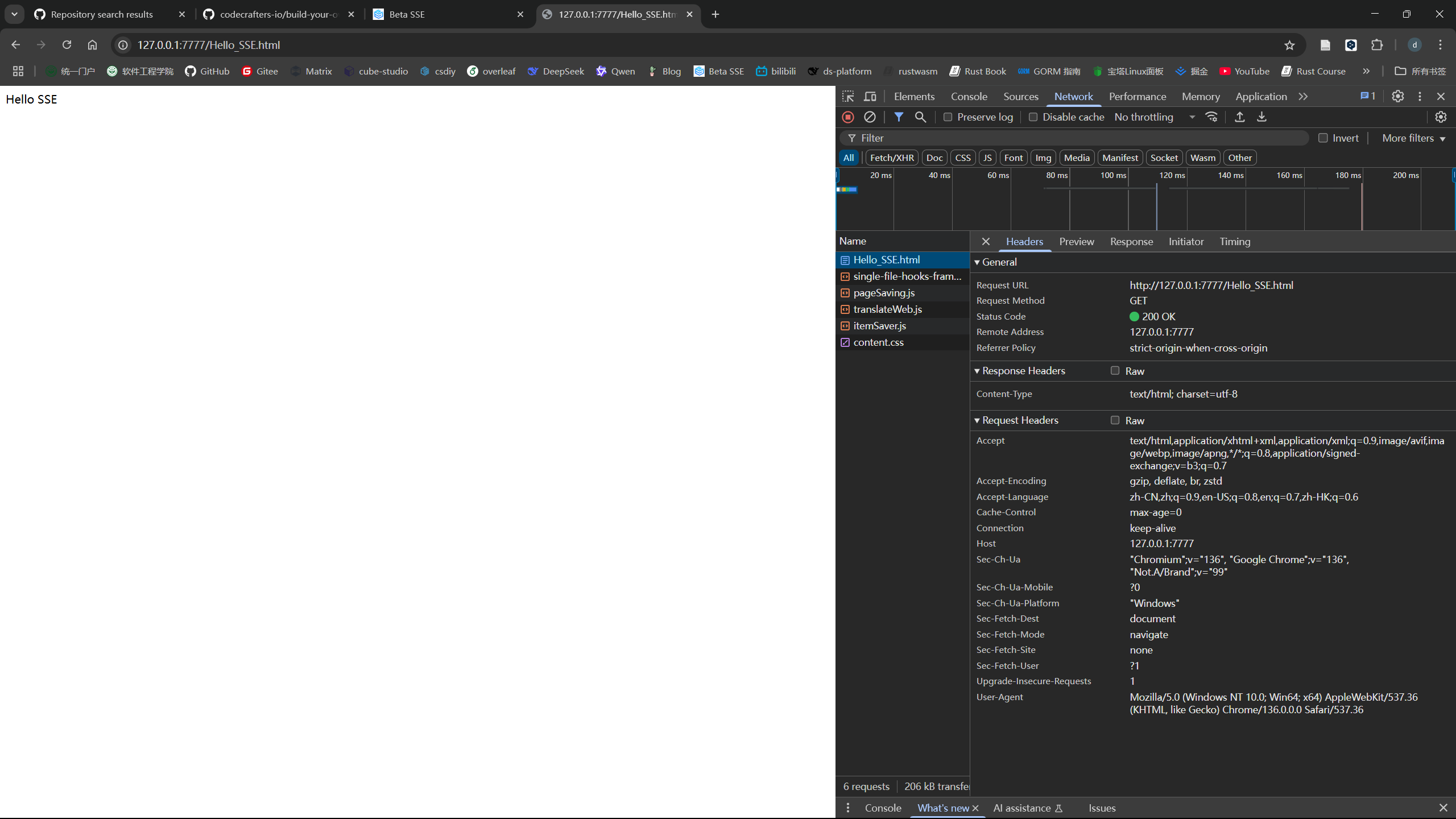
**提交内容：**

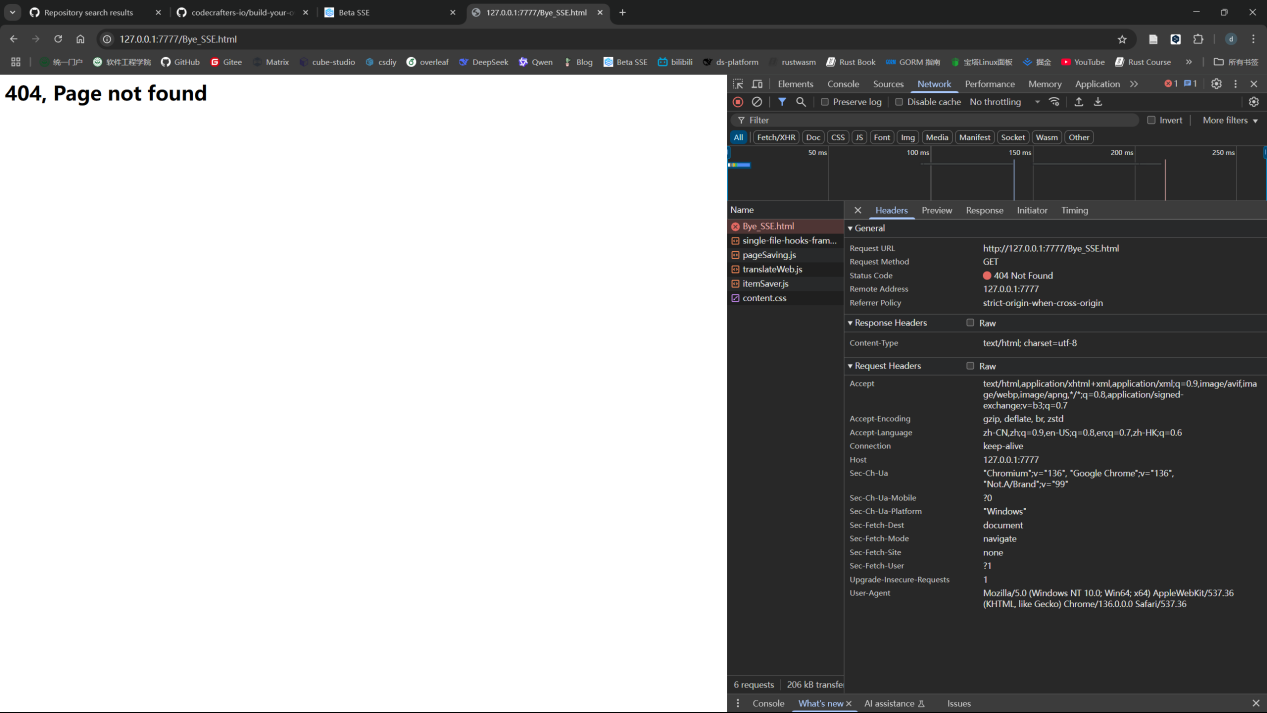
1、服务器代码，保存为“web\_server\_code/simple\_web\_server.py”

2、客户端浏览器输入“http://127.0.0.1:6789/Hello\_SSE.html”后的网页和状态码（200）截图，应与上述例子相似，将截图放在此处。

3、客户端浏览器输入服务器不存在的资源，比如“http://127.0.0.1:6789/Bye\_SSE.html”后的网页和状态码（404）截图，应与上述例子相似，将截图放在此处。

注：“Hello\_SSE.html”是你放在服务器目录中的文件。在使用浏览器访问你的服务时，你需要使用服务器代码中绑定的端口号。比如，在上面的例子中，我们服务器绑定端口号6789，浏览器应该显示Hello\_SSE.html的内容。如果省略“:6789”，浏览器将使用默认端口80，除非你的服务器就是监听80端口，才会从服务器获取网页，但我们不建议使用常用的端口。





### 6、UDP ping 程序

**概述：**

我们知道ping程序位于网络层，基于ICMP协议，其允许客户端机器发送一个数据包到远程机器，并使远程机器响应回客户（称为回显）。另外，ping程序还允许主机计算它到其他机器的往返时间。在上次实验中，你已经学会如何使用UDP编程。考虑到ICMP的复杂性，本次实验，你将基于UDP简单复现一个ping客户端应用程序。

**ping服务器程序代码说明：**

“udp\_pinger\_code/UDP\_pinger\_server.py”是ping服务器程序的代码，无需你进行修改。在这个服务器代码中，30％的客户端数据包会模拟实际路由场景而被人为地丢弃。

UDP为应用程序提供了不可靠的传输服务，消息可能因为路由器队列溢出，硬件错误等其他原因，而在网络中丢失。但由于在内网环境中一般不会丢包，因为经过的路由器足够少，甚至数据包没有离开你本机（目的地址是本地的环回地址，127.0.0.1），所以本次实验服务器程序认为添加了网络丢包模拟：服务器创建一个随机整数，由它确定传入的数据包是否丢失。另外，服务端程序还设置了回复时间，以模拟数据包在网络传输中的延迟。

**ping客户端代码要求：**

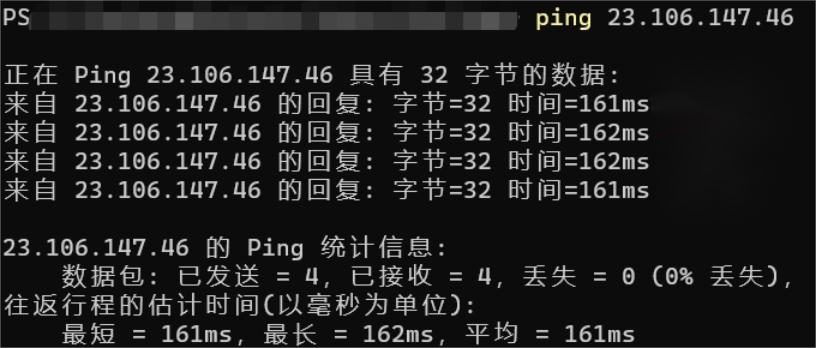
客户端应向服务器发送10次基于udp的ping。因为UDP是不可靠的协议，所以从客户端发送到服务器的数据包可能在网络中丢失。因此，客户端不能无限期地等待ping消息的回复。客户端一般会设置一个超时时间，如果在超时时间内没有收到回复，客户端程序会认为数据包在网络传输期间丢失。关于超时时间的设置，请自行查找Python文档，以了解如何在数据报套接字上设置超时值。

具体来说，你的ping客户端程序应该：

1）使用UDP发送ping消息；

2）超时时间设置成5秒；

3）按照windows标准ping程序的格式进行输出，在控制台输出ping的结果，如下图所示（不需要实现TTL）：

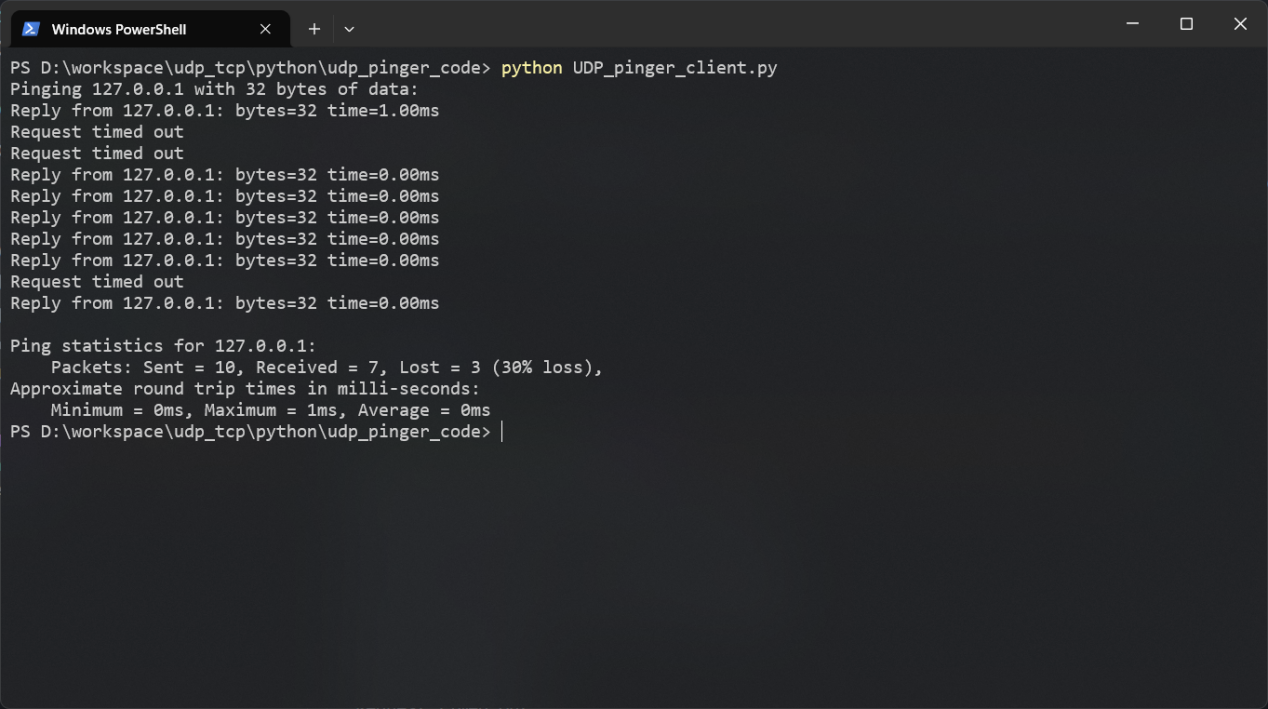


**图5：你的ping客户端程序输出示例**

**提交内容：**

1、完整的客户端代码，保存为“udp\_pinger\_code/UDP\_pinger\_client.py”；

2、你所编写的ping客户端程序运行效果，将截图放到此处。



## 课后练习题：

1）在“1、简单UDP编程”你所编写的代码中，服务端一定会收到客户端发送一次的数据吗？为什么？

2）为什么TCP数据传输会出现粘包问题？常见的粘包问题解决方案有哪些？分析这些方案的优缺点。你在本次作业中又是如何处理这种粘包问题的？

3）尝试把第一题服务端（UDPServer\_Q1.py）的接受缓冲区设置成5字节,即recv(5)，然后运行，观察会发生什么事情？为什么（请结合TCP和UDP的区别进行说明）？**提示：TCP基于流，UDP基于报文。**

## 参考阅读：

1、[硬核图解TCP的粘包问题](https://segmentfault.com/a/1190000039691657)

2、[Python网络编程](https://www.runoob.com/python3/python3-socket.html)

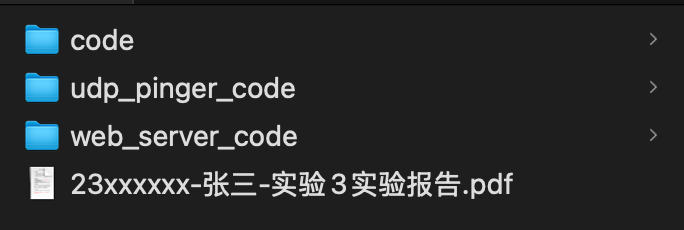
3、[python socket网络编程之粘包问题详解](https://www.cnblogs.com/wj-1314/p/8953148.html)

## 六、提交要求

1、实验报告：完成所有需要回答的部分（已使用**红字**指出），适当配合截图加以说明。你的回答应尽可能条理清晰，截图请缩放至适当大小；

2、提交截止时间：2025/6/5 中午11:59:59

3、提交方式：将该实验报告**转换为PDF，重命名为“学号-姓名-实验3实验报告”**，将**三个代码文件夹（“code”，“udp\_pinger\_code”，“web\_server\_code”）和实验报告放在一起，压缩为一个zip文件**，上传至坚果云，如下图所示。



4、提交链接：[实验课作业统一提交链接](https://send2me.cn/-O_UnRwR/SGa8nwAPlRDPQQ)。注意**正确填写坚果云界面中要求的姓名、学号等字段**，系统会自动对 zip 压缩包进行重命名。