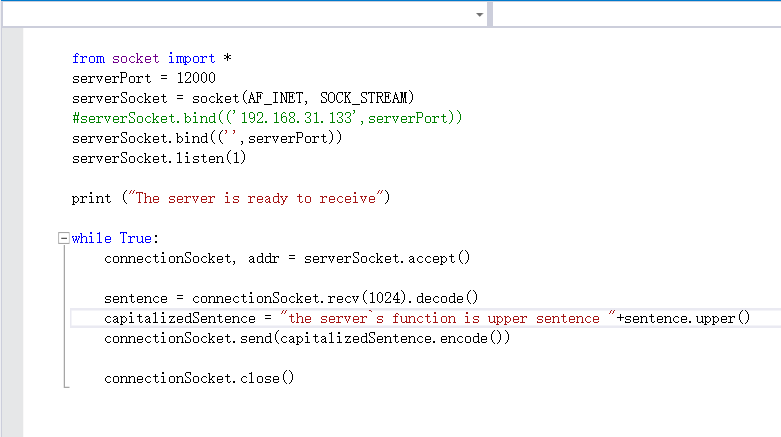
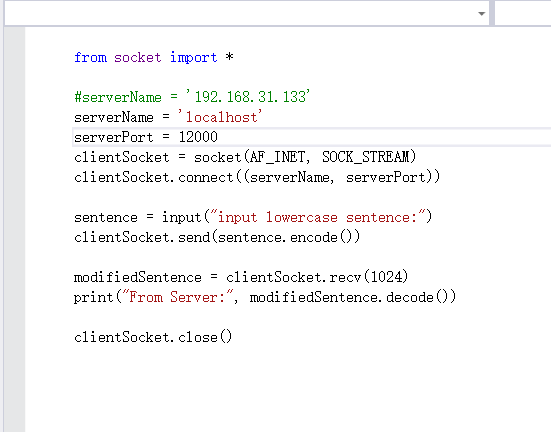
1.tcp应用小程序：

server.py源码：

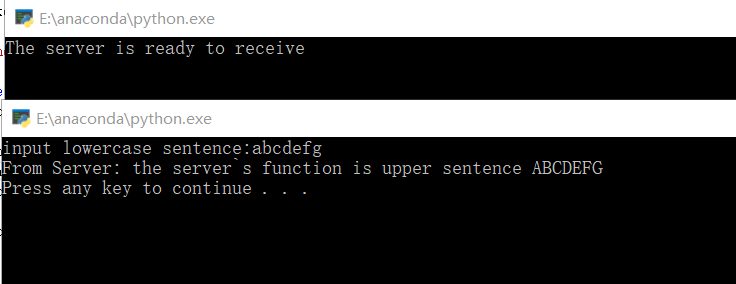


client.py源码：



此处需要先运行server.py打开服务器的socket在运行client.py打开客户的socket

运行截图：



服务器返回提示字符串“the server`s function is upper sentence”+客户向服务器传送的字符串的大写形式

2.习题

P1.

1. 源端口号：客户A发起的Telnet会话中的socket端口号，目的端口号：23号。
2. 源端口号：客户B发起的Telnet会话中的socket端口号，目的端口号：23号。
3. 源端口号：23号，目的端口号：客户A发起的Telnet会话中的socket端口号。
4. 源端口号：23号，目的端口号：客户B’发起的Telnet会话中的socket端口号。
5. 是可能的。
6. 从A向S发送的报文段的源端口号和从B向S发送的报文段的源端口号一定不同。

P3.

01010011

+ 01100110

= 10111001

+ 01110100

= 00101101

进位为1

有溢出, 回卷为00101110

反码为：11010001

使用反码的好处：计算检验和的方法简单，将收到的比特求得的和与检验和相加，若全为1，则没有出错。

1比特的差错可以检测出来，2比特的差错不一定能检测出来，如第一个和第二个比特的最后一位一个变为0，一个变为1，和不变。

P6.

考虑如下过程：发送方向接收方发送序列号为0的分组，接收方收到并返回ACK，但该ACK传输过程中出现位错误，即corrupt，这时接收方期待收到序列号为1的分组，而发送方由于ACK发生错误，会继续发送序列号为0的分组，而接收方接收到序列号为0的分组会返回NAK，而发送方接收到NAK仍会返回序列号为0的分组，因此造成死锁。

P7.

分组需要序列号是因为接受方需要知道本次接收到的分组是否是已经接受过的分组，而ACK只需要告诉发送方某分组是否已成功接受即可，发送方不需要知道该ACK是否已经被接收过，因此ACK不需要序列号

P42.

超时间隔加倍只能分组丢失重传的等待时间加长，但对于未丢失的，还是会向接收方发送新的分组，因为还有很多未丢失的分组，向接收方发送新分组的速度很可能不会减慢,因此需要引入拥塞控制机制。