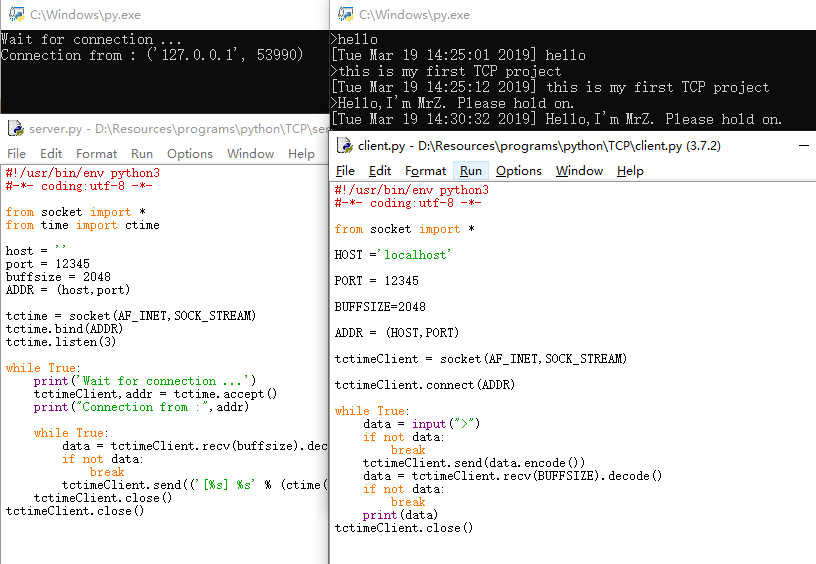
一、使用Python编写的TCP程序：



二、课后习题：

**P1**

假定 A 向 S 的源端口号为 x， B 向 S 的源端口号为 y

a）.

A 向 S 的源端口号为 x， 目的端口号为 23

b）.

B 向 S 的源端口号为 y， 目的端口号为 23

c）.

S 向 A 的源端口号为 23， 目的端口号为 x

d）.

S 向 B 的源端口号为 23， 目的端口号为 y

e）.

x y 可能相同

f）.

不可能

**P2.**

从 B 到 C：

   左边的连接：源端口号 80，源 IP 为 B 的 IP；目的端口 26145，目的 IP 为 C 的 IP；

   右边的连接：源端口号 80，源 IP 为 B 的 IP；目的端口 7532，目的 IP 为 C 的 IP；

从 B 到 A：

   源端口号 80，源 IP 为 B 的 IP；目的端口 26145，目的 IP 为 C 的 IP；

**P4.**

**a).** 00111110  
**b).** 01000000  
**c).** a 中的第一、二字节变为 01011101、01100100

**P6.**

如果 rdt2.1 发送方正处于“等待来自上层的调用0”，接收方处于“等待来自下层的0”，发送方发送序号为 0 的分组，而接收方正确接收并向发送方发送 ACK；此时发送方处于“等待 ACK 或 NAK 0”，接收方处于“等待下层的 1”，如果此 ACK 损坏，发送方重发序号0的分组，而接收方会发送 NAK，这将导致一个死循环；

其实此接收方并没有标注初始状态，如果发送方初始状态为“等待来自上层的 0”，接收方初始为“等待下层的 1”，也会导致上述死锁。

**P39**

λ’in 超过 R/2，会导致更多的丢包以及重传，λout 可能会下降