Tcp Server

```
import socket
import sys

# Create a TCP/IP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Bind the socket to the port
server_address = ('localhost', 10000)
print >>sys.stderr, 'starting up on %s port %s' % server_address
sock.bind(server_address) #bind() is used to associate the socket with the server address.

# Listen for incoming connections
sock.listen(1)

while True:
    # Wait for a connection
    print >>sys.stderr, 'waiting for a connection'
    connection, client_address = sock.accept() #listen() puts the socket into server mode, and accept() waits for an incoming connection. accept() returns an open connection between the server and client, along with the address of the client. The connection is actually a different socket on another port (assigned by the kernel). Data is read from the connection with recv() and transmitted with sendall()
try:
        print >>sys.stderr, 'connection from', client_address

        # Receive the data in small chunks and retransmit it
        while True:
            data = connection.recv(16)
            print >>sys.stderr, 'received "%s"' % data
            if data:
                print >>sys.stderr, 'sending data back to the client'
                connection.sendall(data)
            else:
                print >>sys.stderr, 'no more data from', client_address
                break

    finally:
        # Clean up the connection
        connection.close()
```

Tcp Client

```
import socket
import sys

# Create a TCP/IP socket
sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

# Connect the socket to the port where the server is listening
server_address = ('localhost', 10000)
print >>sys.stderr, 'connecting to %s port %s' % server_address
sock.connect(server_address)# Instead of binding to a port and listening, it uses connect() to attach the socket directly to the remote address.
try:

    # Send data
    message = 'This is the message.  It will be repeated.'
    print >>sys.stderr, 'sending "%s"' % message
    sock.sendall(message)  #data can be sent through the socket with sendall() and received with recv(), just as in the server

    # Look for the response
    amount_received = 0
    amount_expected = len(message)

    while amount_received < amount_expected:
        data = sock.recv(16)
        amount_received += len(data)
        print >>sys.stderr, 'received "%s"' % data

finally:
    print >>sys.stderr, 'closing socket'
    sock.close()
```