This project from the textbook’s website gives the following skeleton code to build off of for the web server:

Skeleton Python Code for the Web Server

#import socket module

from socket import \*

serverSocket = socket(AF\_INET, SOCK\_STREAM)

#Prepare a sever socket

#Fill in start

#Fill in end

while True:

#Establish the connection

print ‘Ready to serve…’

connectionSocket, addr = #Fill in start     #Fill in end

try:

message = #Fill in start     #Fill in end

filename = message.split()[1]

f = open(filename[1:])

outputdata = #Fill in start     #Fill in end

#Send one HTTP header line into socket

#Fill in start

#Fill in end

#Send the content of the requested file to the client

for i in range(0, len(outputdata)):

connectionSocket.send(outputdata[i])

 connectionSocket.close()

except IOError:

#Send response message for file not found

#Fill in start

#Fill in end

#Close client socket

#Fill in start

#Fill in end

serverSocket.close()

I used the TCP server example in the book (page 167) to help me edit this and I came up with this code:

#import socket module

from socket import \*

serverSocket= socket(AF\_INET, SOCK\_STREAM)#creates socket

#Prepare a sever socket

#Fill in start

serverPort = 7000 #sets port

serverSocket.bind((”, serverPort)) #associates socket with this port

serverSocket.listen(1) #tells socket to listen for requests

#Fill in end

while True:

#Establish the connection

print ‘Ready to serve…’

connectionSocket, addr = serverSocket.accept() #creates a socket specifically for this client

try:

message =connectionSocket.recv(1024) #receives message from client

print message

filename = message.split()[1]

f = open(filename[1:]) #opens file and reads the contents

outputdata =f.read()

#Send one HTTP header line into socket

#Fill in start

connectionSocket.send(‘\nHTTP/1.x 200 OK\n’) #sends a 200 OK header line

#Fill in end

#Send the content of the requested file to the client

for i in range(0, len(outputdata)):

connectionSocket.send(outputdata[i]) #outputs all of the data in the file

connectionSocket.close() #closes the socket for this client

print ‘File Recieved’

except IOError:

#Sendresponse message for file not found

#Fill in start

connectionSocket.send(‘\n404 File Not Found\n’) #sends an error message to be printed on the page

#Fill in end

#Close client socket

#Fill in start

connectionSocket.close() #closes the socket for the client

#Fill in end

serverSocket.close() #closes the server socket

This project was fairly straightforward. Most of the code was given, and most was easy to derive knowing the TCP server code. This code creates a socket, defines the port number, associates the port with this socket, and tells the server to listen for incoming requests.

Then, in the while, it prints a statement declaring that it is ready. The server then creates another socket for this particular client. Then, it tries to receive the request, open the file, and reads the data from the file.  I had to look up the http header lines to know how the header lines are formatted, and found out that I just needed to follow the header line (200 OK) with HTTP/1.x, then I added a line to have the connectionSocket send a 200 OK header line to state that the request succeeded. The outputdata is then sent out and the connectionSocket closes.

If the try does not work, the code runs the exception instead because there is an error. The connectionSocket prints a 404 NOT FOUND message, which prints to the page, and the socket closes.

Then the serverSocket closes.