Bijay Regmi (210913032)

Lab 9

Network Programming [TCP Socket Programming]

Internetworking and Sockets. Implement the sample TCP client/ server programs found in the Python Library Reference documentation on the socket module and get them to work. Set up the server and then the client.

You decide the server is too boring. Update the server so that it can do much more, recognizing the following commands:

Date: Server will return its current date/timestamp, that is, time.ctime().

Os: Get OS information (os.name).

ls: Give a listing of the current directory. (Hints: os.listdir() lists a directory, os.curdir is the current directory.)

Extra Credit: Accept ls dir and return dir's file listing.

You do not need a network to do this assignment—your computer can communicate with itself. Be aware that after the server exits, the binding must be cleared before you can run it again. You might experience "port already bound" errors. The operating system usually clears the binding within 5 minutes, so be patient.

Source Code:

Server:

```
import socket

import datetime

from datetime import date

import os

import platform


s =socket.socket()

print ("Socket successfully created")

port = 12345

s.bind(('', port))
```

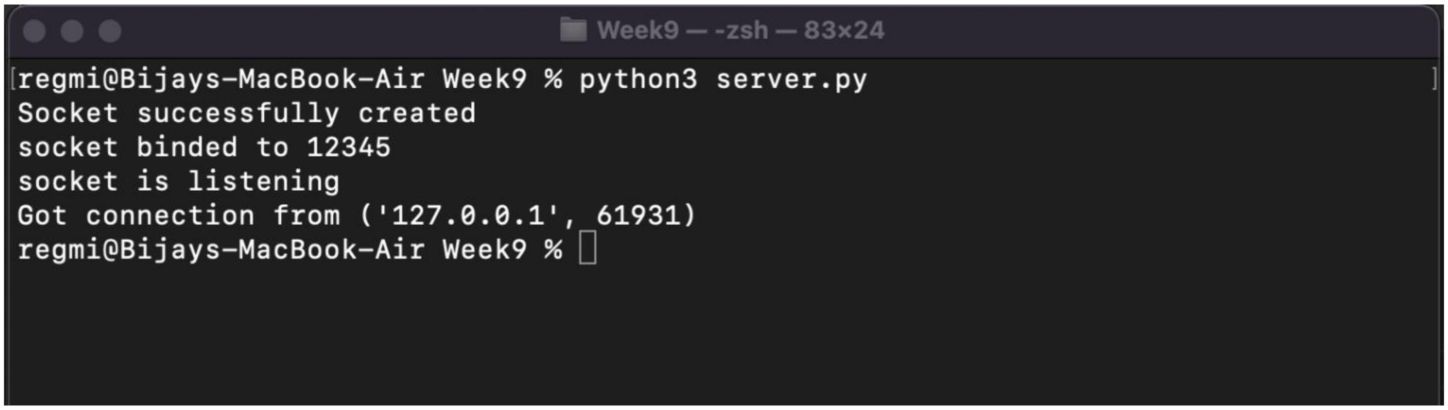
```
print ("socket binded to %s" %(port))
s.listen(5)
print ("socket is listening")
today = date.today()
dt = datetime.datetime.now()
date_str = dt.ctime()
os_name = os.name
platfrom = platform.system()
dir_cur = os.getcwd()
while True:
    c, addr = s.accept()
    print ('Got connection from', addr )
    c.send(today.strftime("%d/%m/%Y").encode())
    c.send(date_str.encode())
    c.send(os_name.encode())
    c.send(platfrom.encode())
    c.send(dir_cur.encode())
    c.close()
    break
```

Client:

```
import socket
from datetime import date
s = socket.socket()
port = 12345
s.connect(('127.0.0.1', port))
print ("\n",s.recv(1024).decode())
print ("\n",s.recv(1025).decode())
s.close()
```

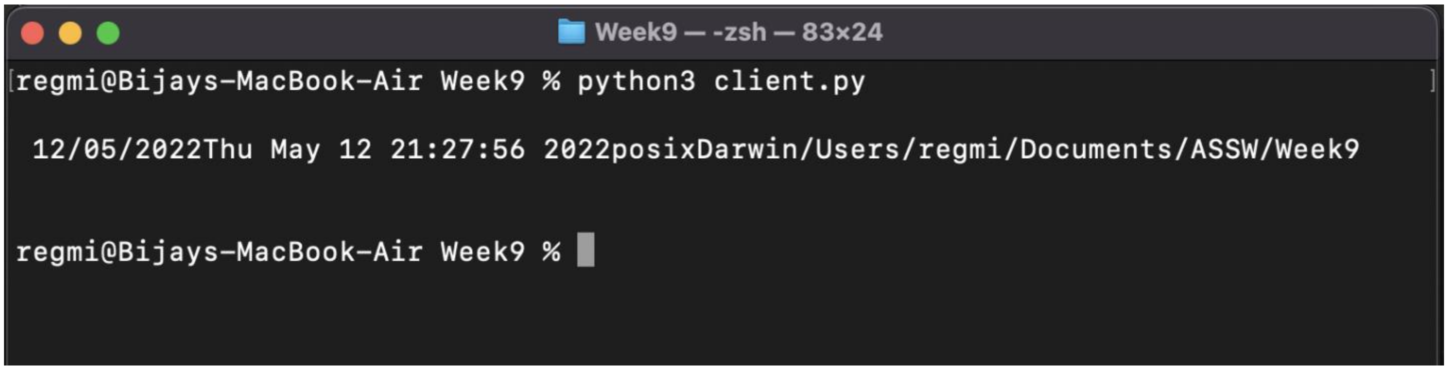
Output:

Server:



```
regmi@Bijays-MacBook-Air Week9 % python3 server.py
Socket successfully created
socket binded to 12345
socket is listening
Got connection from ('127.0.0.1', 61931)
regmi@Bijays-MacBook-Air Week9 %
```

Client:



```
regmi@Bijays-MacBook-Air Week9 % python3 client.py

 12/05/2022Thu May 12 21:27:56 2022posixDarwin/Users/regmi/Documents/ASSW/Week9


regmi@Bijays-MacBook-Air Week9 %
```

Additional Questions:

Q.1) Half-Duplex Chat. Create a simple, half-duplex chat program. By half-duplex, we mean that when a connection is made and the service starts, only one person can type. The other participant must wait to get a message before being prompted to enter a message. Once a message is sent, the sender must wait for a reply before being allowed to send another message. One participant will be on the server side; the other will be on the client side.

Source Code:

Server:

```
from socket import *

server_port = 9999

server_socket = socket(AF_INET,SOCK_STREAM)

server_socket.bind(('',server_port))

server_socket.listen(1)

print ("Welcome: The server is now ready to receive")

connection_socket, address = server_socket.accept()

while True:

    sentence = connection_socket.recv(2048).decode()

    print('>> Client : ',sentence)

    message = input(">> Your Message : ")

    connection_socket.send(message.encode())

    if(message == 'q'):

        connection_socket.close()
```

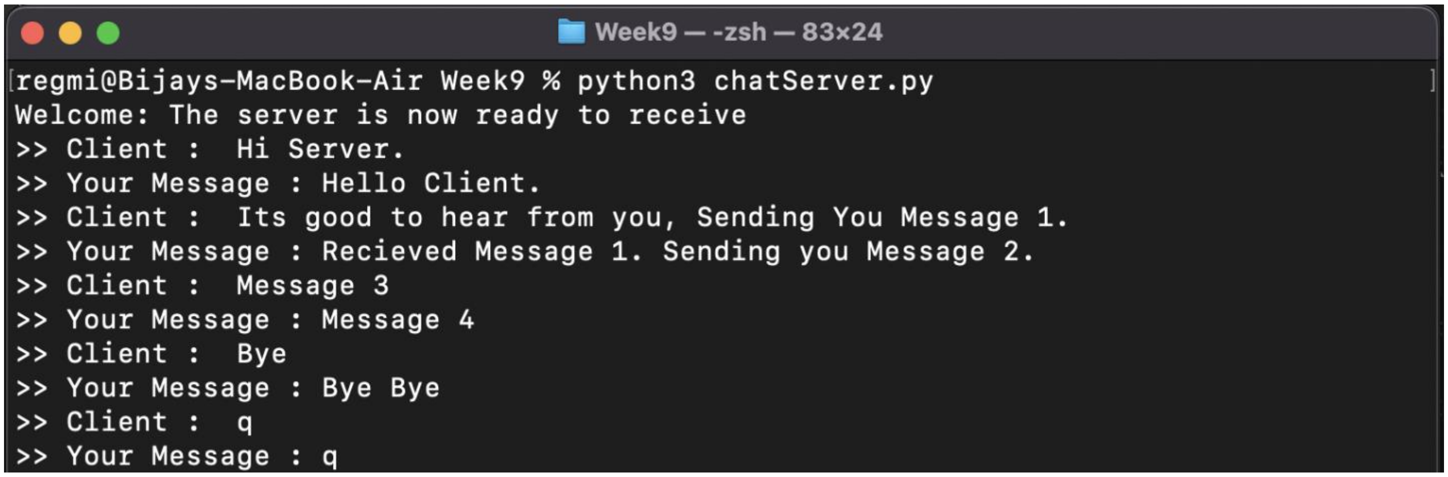
Client:

```
from socket import *

server_name = 'localhost'

server_port = 9999

client_socket = socket(AF_INET, SOCK_STREAM)

client_socket.connect((server_name,server_port))
```

```
while True:
    sentence = input(">> Your Message : ")
    client_socket.send(sentence.encode())
    message = client_socket.recv(2048)
    print (">> Server : ", message.decode())
    if(sentence == 'q'):
        client_socket.close()
```

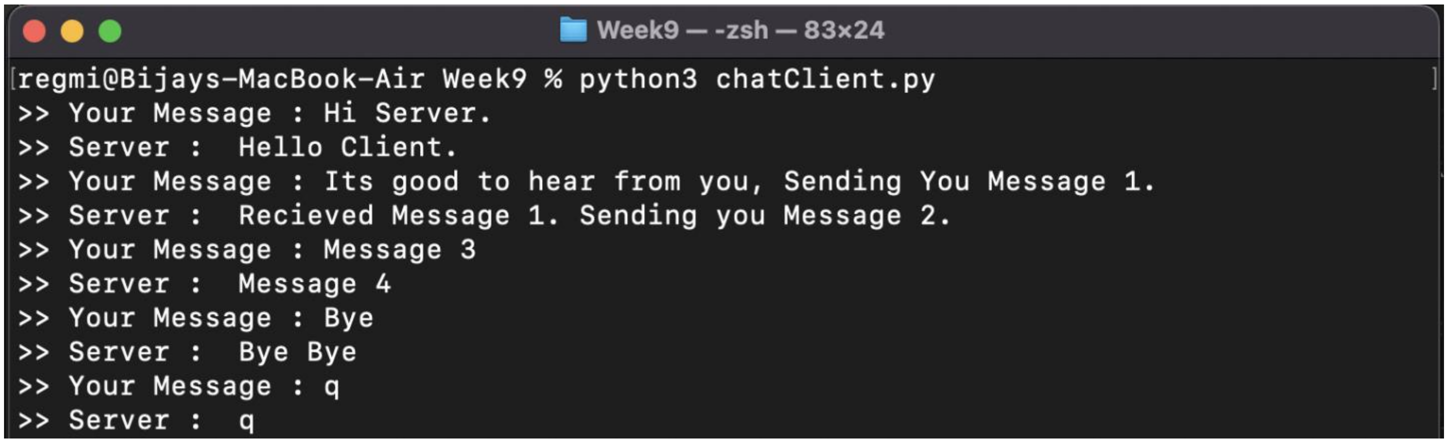
Output:

Server:



```
Week9 — -zsh — 83×24
[regmi@Bijays-MacBook-Air Week9 % python3 chatServer.py
Welcome: The server is now ready to receive
>> Client :  Hi Server.
>> Your Message : Hello Client.
>> Client :  Its good to hear from you, Sending You Message 1.
>> Your Message : Recieved Message 1. Sending you Message 2.
>> Client :  Message 3
>> Your Message : Message 4
>> Client :  Bye
>> Your Message : Bye Bye
>> Client :  q
>> Your Message : q
```

Client:



```
Week9 — -zsh — 83×24
[regmi@Bijays-MacBook-Air Week9 % python3 chatClient.py
>> Your Message : Hi Server.
>> Server :  Hello Client.
>> Your Message : Its good to hear from you, Sending You Message 1.
>> Server :  Recieved Message 1. Sending you Message 2.
>> Your Message : Message 3
>> Server :  Message 4
>> Your Message : Bye
>> Server :  Bye Bye
>> Your Message : q
>> Server :  q
```