

**Network Architecture-I**

Group Chat Program – Phase 2

Name : Ravindra Reddy Chennapareddy

Student ID: 16296266

Email ID : rcr3n@mail.umkc.edu

**Department of Computer Science**

**University of Missouri, Kansas City**

**Part II. Group Chatting Program:**

Develop a simple chat program (similar to google hangout and skype chat),and show the screenshots of the execution of the below. Extend the first program to chat client-server program following these steps.

1) A chat server will accept a single client connection and display everything the client types. If the client user types ‘exit’, both client and server will end the program.

**CLIENT CODE:**

# To import the socket module

import socket

# To get the Hostname of our Local system

Host\_name = socket.gethostname()

# To print our Local Hostname

print(Host\_name)

# To get the IP of our Local system based on our Hostname

IP\_Address = socket.gethostbyname(Host\_name)

# To print the IP Address of our Local Host

print(IP\_Address)

# To initiate port number above 1024

port = 8080

# To create socket object called 'client'

client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

# To connect to the server

client.connect((IP\_Address, port))

while True:

    message = str(

        input("Do you want to send  a message to the server?(y/n) : ")).lower()

    if(message == "y"):

        # To enter the message to be sent to server

        input\_message = input("Enter your message : ")

        # To send the message to server

        client.send(input\_message.encode())

    else:

        client.send("exit".encode())

        # To close the connection with server

        client.close()

        # Print statement

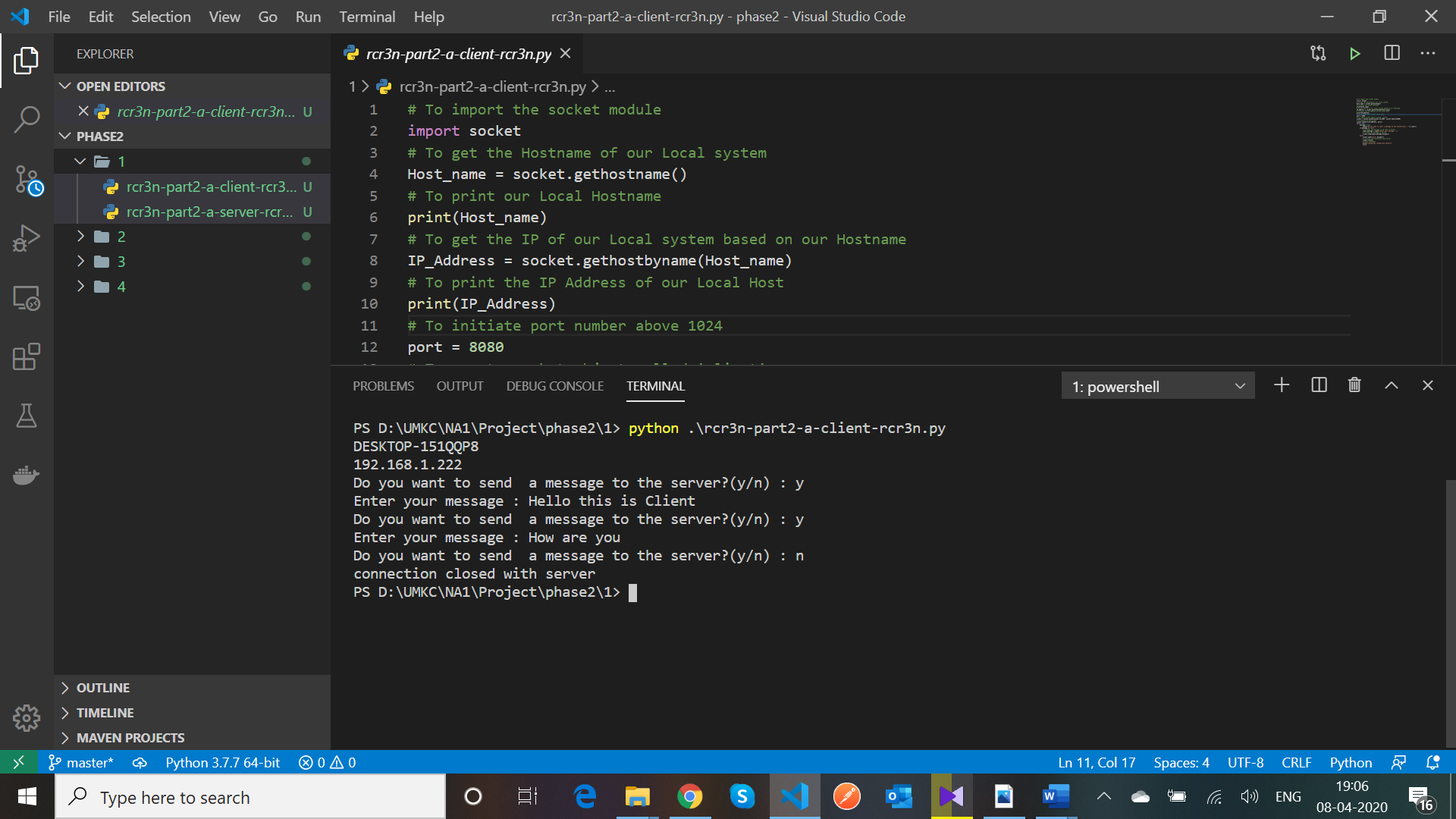
        print("connection closed with server")

        break

**CLIENT CODE EXPLANATION:**

* First we need to import the socket and then create a TCP socket object.
* Then the client connects to the server using the connect () method by providing the server’s IP address and the Host number.
* If the client wants to send a message to the server then the input from the terminal will be “y”. Then it prompts us enter the message to be sent to client.
* If the client doesn’t want to send a message to the server then the input from the terminal will be “n”. Then a default message “exit” will be sent to the server and the connection will be closed with the server.

**CLIENT OUTPUT:**



**SERVER CODE EXPLANATION:**

* First we need to import the socket and then create a TCP socket object.
* Then we need to bind the server with an IP address and Host number.
* Then the server accepts an incoming connection from the client using accept () method.
* Once the connection is established then it receives data from the client.
* If the data received from the client is “exit” then it will close the connection with the client and prints the connection with client is closed.
* If the data from the client is other than “exit” then it will print the data on the server terminal.

**SERVER CODE:**

# To import the socket module

import socket

# To get the Hostname of our Local system

Host\_name = socket.gethostname()

# To print our Local Hostname

print(Host\_name)

# To get the IP of our Local system based on our Hostname

IP\_Address = socket.gethostbyname(Host\_name)

# To print the IP Address of our Local Host

print(IP\_Address)

# To initiate port number above 1024

port = 8080

# To create socket object called 'server'

server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

# To bind the IP Address and Port number to the server

server.bind((IP\_Address, port))

# To configure the number of clients the server can listen to simultaneously

server.listen(1)

print("Server waiting for connection from the client on port:", port)

# To accept new connection from the client

conn, address = server.accept()

# To print the address of the connected client to server

print("Connection established with: " + str(address))

while True:

    # To receive data stream from the connected client

    data = conn.recv(1024).decode()

    if(data == "exit"):

        # To print the data received from the client

        print("Message received from client: " + str(data))

        # To close the connection with server

        conn.close()

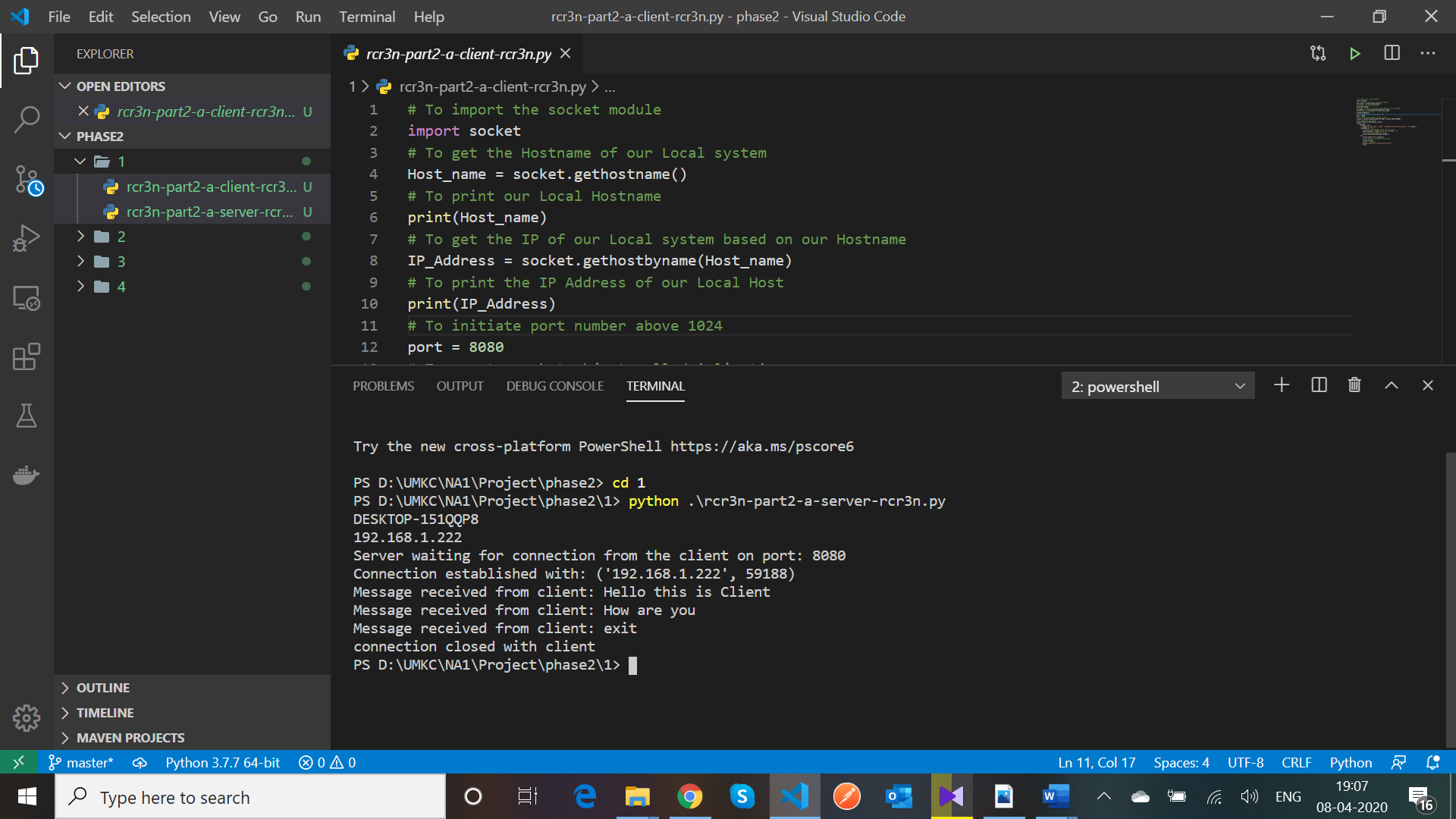
        print("connection closed with client")               # Print statement

        break

    else:

        print("Message received from client: " + str(data))  # Print statement

**SERVER OUTPUT:**



2) A server now remains ‘open’ for additional connection once a client quits. The server can handle at most one connection at a time.

**CLIENT CODES:**

**Client – 1**

import socket

SERVER = "127.0.0.1"

PORT = 4000

client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client.connect((SERVER, PORT))

client.sendall(bytes("This is from Client 1", 'UTF-8'))

while True:

    in\_data = client.recv(1024)

    print("From Server :", in\_data.decode())

    out\_data = input()

    client.sendall(bytes(out\_data, 'UTF-8'))

    if out\_data == 'bye':

        break

client.close()

**Client – 2**

import socket

SERVER = "127.0.0.1"

PORT = 4000

client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client.connect((SERVER, PORT))

client.sendall(bytes("This is from Client 2", 'UTF-8'))

while True:

    in\_data = client.recv(1024)

    print("From Server :", in\_data.decode())

    out\_data = input()

    client.sendall(bytes(out\_data, 'UTF-8'))

    if out\_data == 'bye':

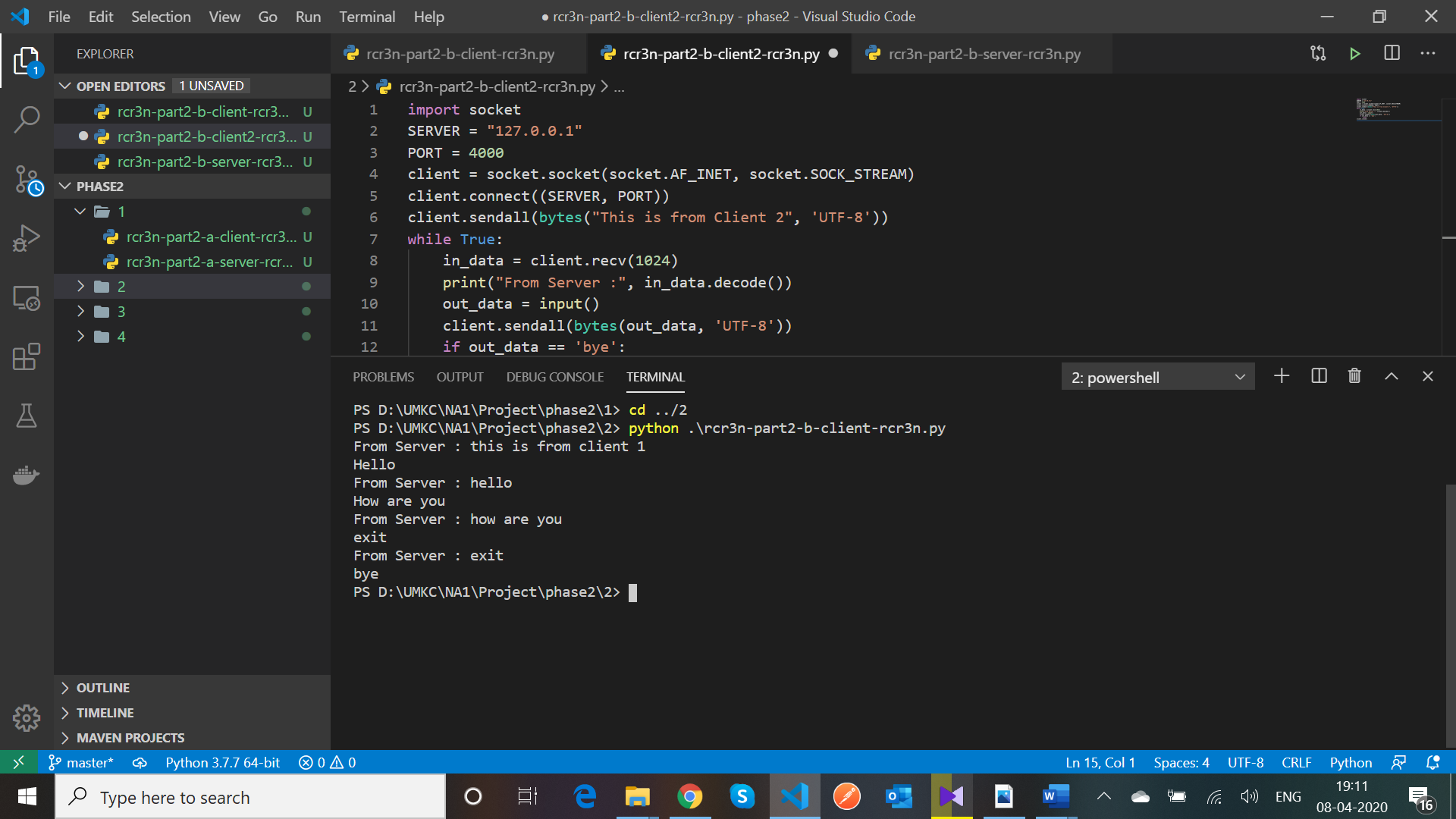
        break

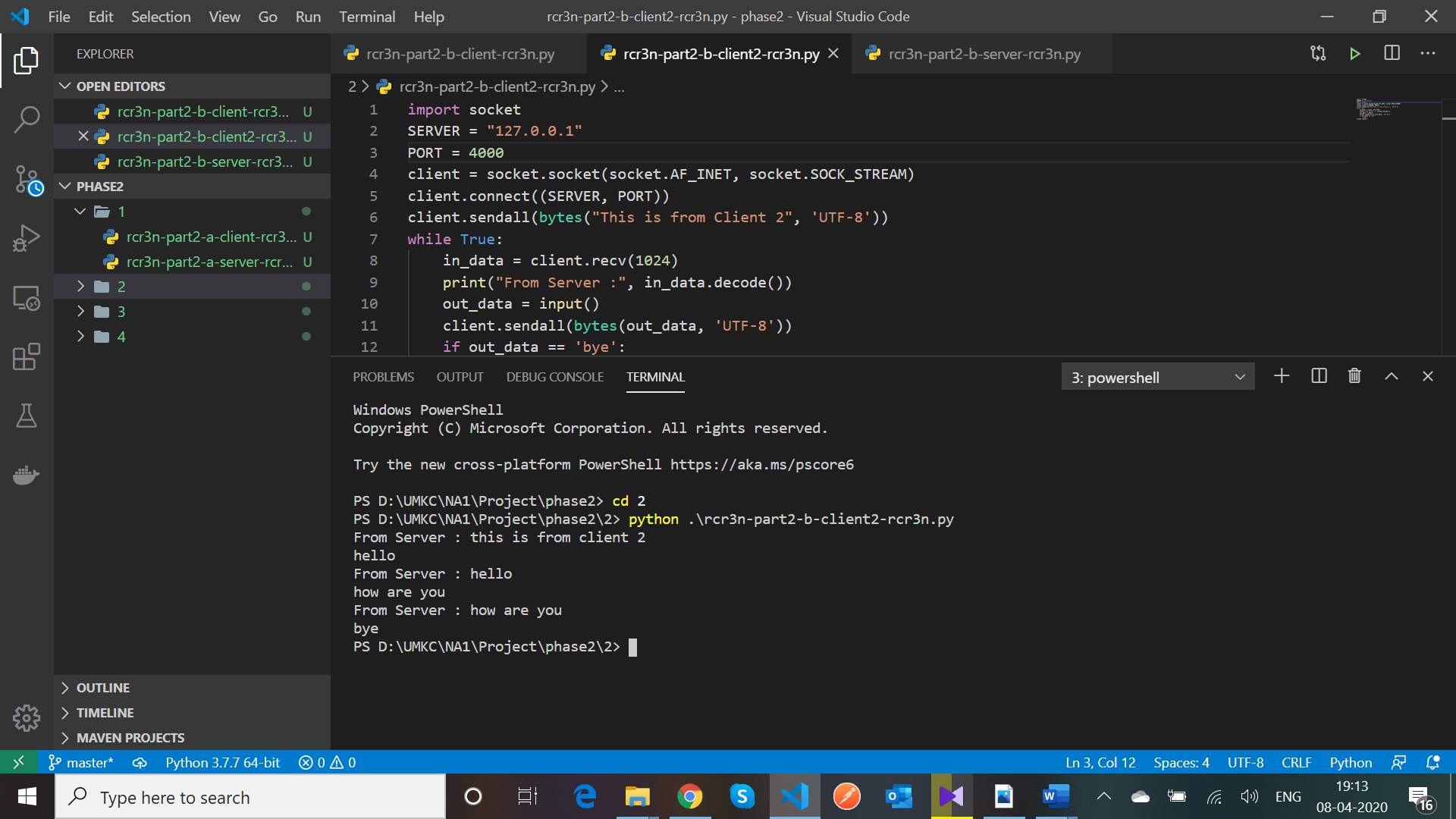
client.close()

**CLIENT CODES EXPLANATION:**

* First we need to import the socket and then create a TCP socket object.
* Then the client connects to the server using the connect () method by providing the server’s IP address and the Host number.
* The client can send the message to the server and gets the same message as response from server and if the client sends “bye” then the server closes the connection for the client and will display a message as “Connection closed”.

**CLIENT OUTPUTS:**





**SERVER CODE:**

import socket

HOST = ''

PORT = 4000

def connect():

    sock = socket.socket()

    server = sock.bind((HOST, PORT))

    sock.listen(1)

    print("Server is Listening...")

    conn, addr = sock.accept()

    return conn

def send\_messages(c):

    while True:

        msg = str(c.recv(1024).decode())

        msg = msg.lower()

        if msg == "bye":

            print(msg)

            c.send(str.encode("Server connection is closed"))

            c.close()

            c = connect()

            send\_messages(c)

            # break

        elif msg == "hello from client-ravi":

            print(msg)

            c.send(str.encode("Hello from Server"))

        else:

            print(msg)

            c.send(str.encode(msg))

def main():

    c = connect()

    send\_messages(c)

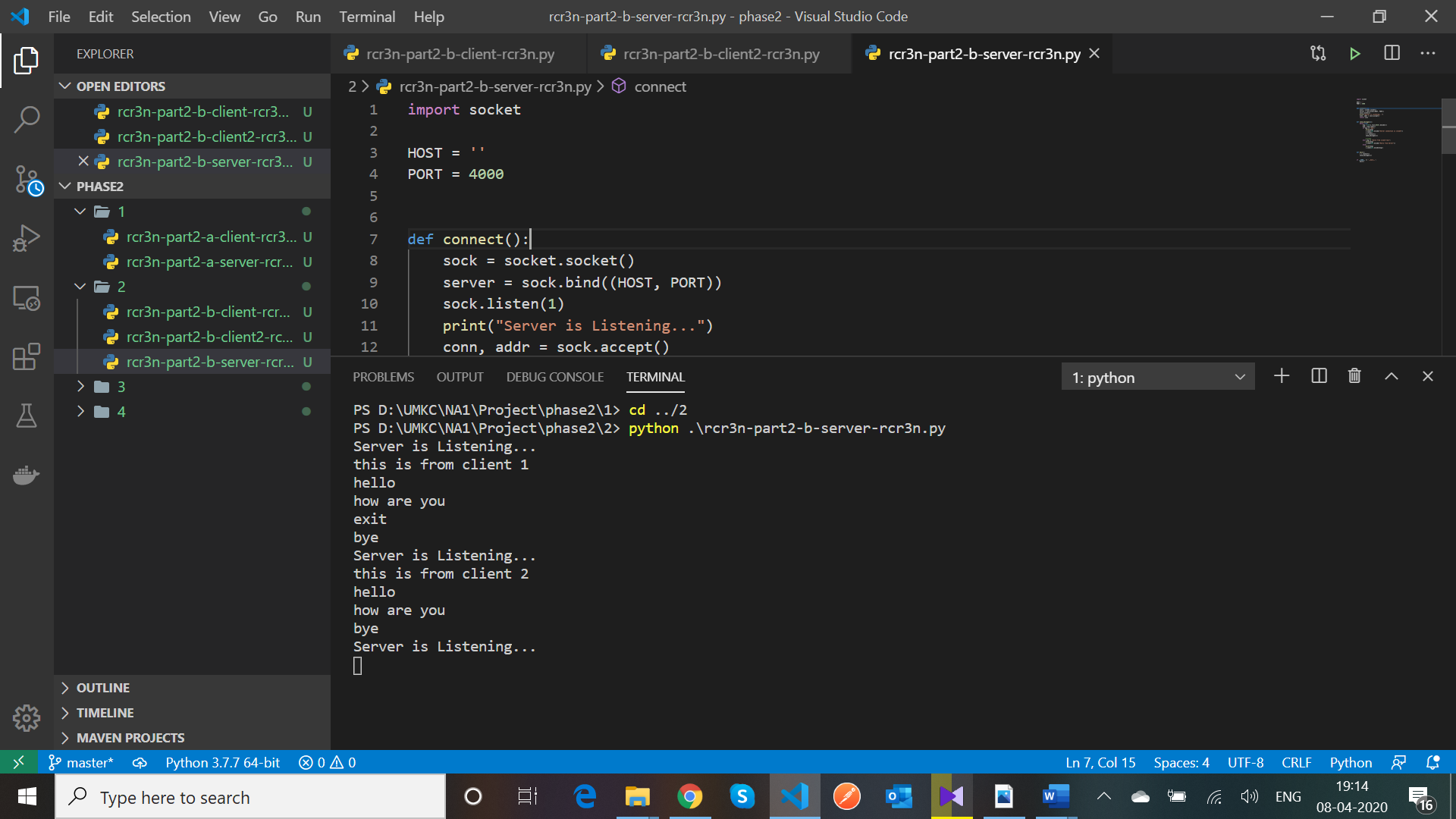
if \_\_name\_\_ == '\_\_main\_\_':

    main()

**SERVER CODE EXPLANATION:**

* First we need to import the socket and then create a TCP socket object.
* Then we need to bind the server with an IP address and Host number.
* Then the server accepts an incoming connection from the client using accept () method.
* Once the connection is established then it receives data from the client.
* If the data received from the client is “bye” then it will close the connection with the client and prints the connection with client is closed. The server is again now open for connection from new client
* If the data from the client is other than “bye” then it will print the data on the server terminal.

**SERVER OUTPUT:**



3) A server now can handle multiple clients at the same time. The output from all the connected clients will appear on the server’s screen.

**CLIENT CODES EXPLANATION:**

* First we need to import the socket and then create a TCP socket object.
* Then the client connects to the server using the connect () method by providing the server’s IP address and the Host number.
* The client can send the message to the server and gets the same message as response from server and if the client sends “bye” then the server closes the connection for the client and will display a message as “Connection closed”.

**CLIENT CODES:**

**Client – 1**

import socket

SERVER = "127.0.0.1"

PORT = 8080

client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client.connect((SERVER, PORT))

client.sendall(bytes("This is from Client 1", 'UTF-8'))

while True:

    in\_data = client.recv(1024)

    print("From Server :", in\_data.decode())

    out\_data = input()

    client.sendall(bytes(out\_data, 'UTF-8'))

    if out\_data == 'bye':

        break

client.close()

**Client – 2**

import socket

SERVER = "127.0.0.1"

PORT = 8080

client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client.connect((SERVER, PORT))

client.sendall(bytes("This is from Client 2", 'UTF-8'))

while True:

    in\_data = client.recv(1024)

    print("From Server :", in\_data.decode())

    out\_data = input()

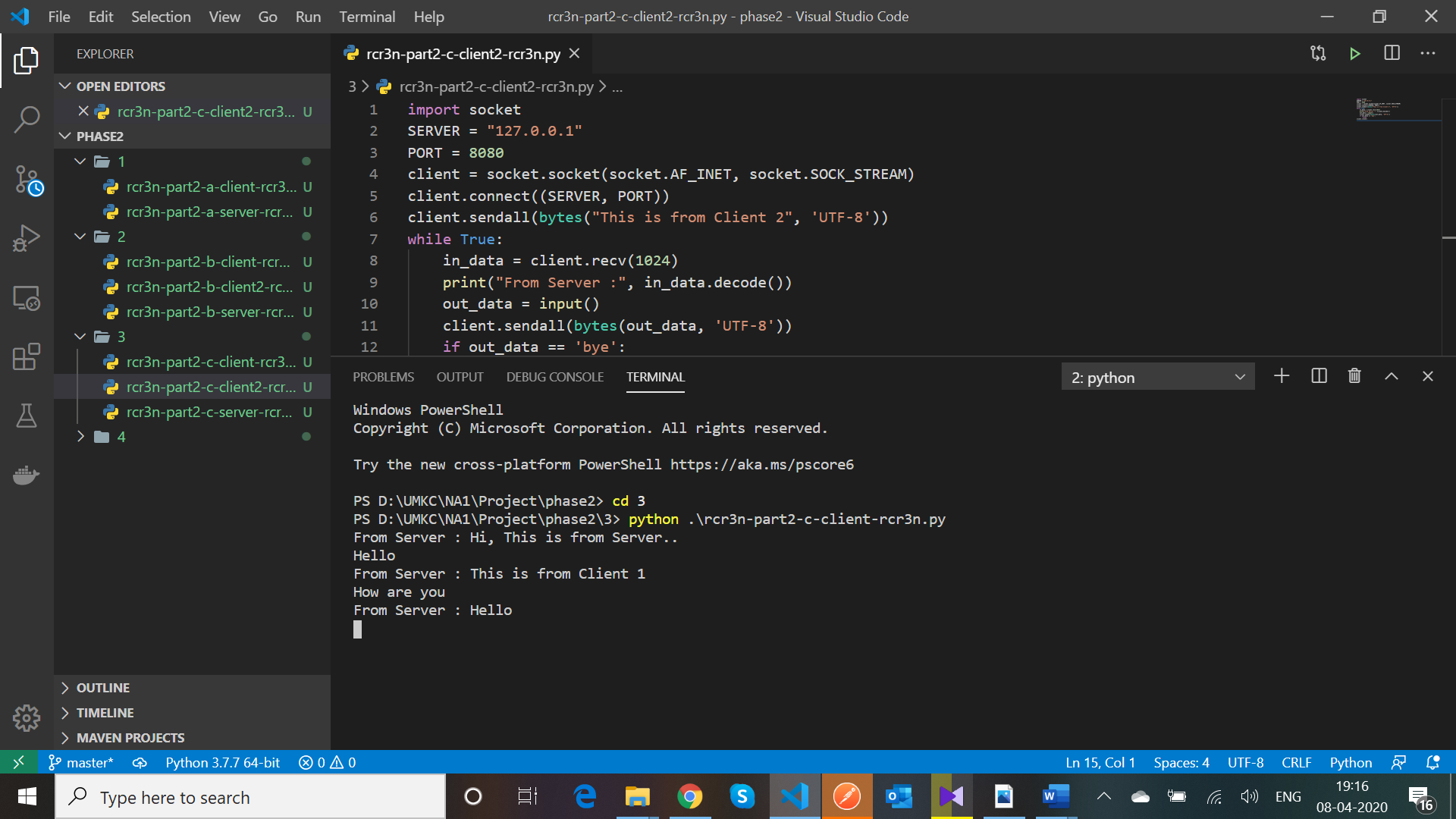
    client.sendall(bytes(out\_data, 'UTF-8'))

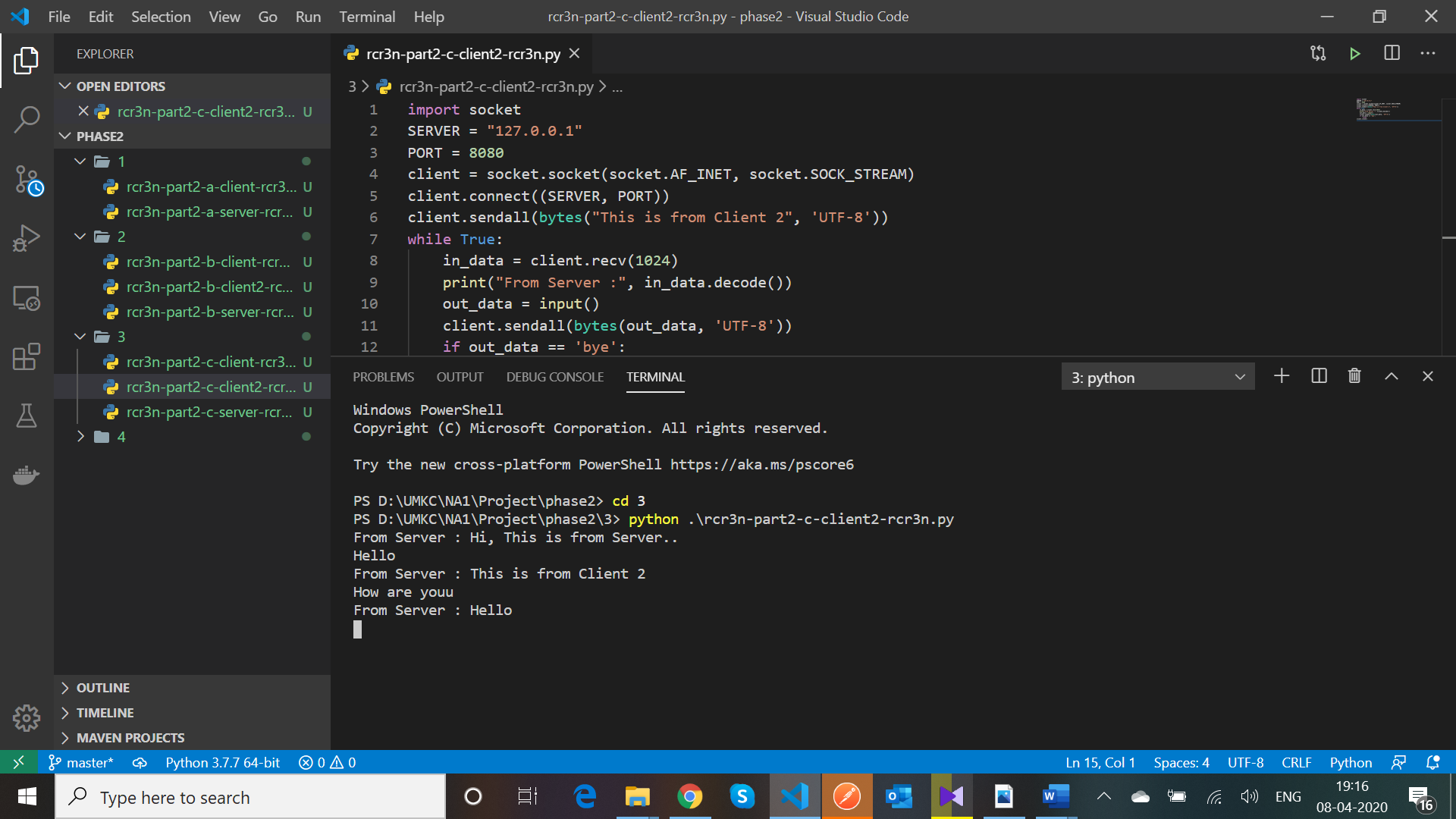
    if out\_data == 'bye':

        break

client.close()

**CLIENT CODE OUTPUTS:**





**SERVER CODE:**

import socket

import threading

class ClientThread(threading.Thread):

    def \_\_init\_\_(self, clientAddress, clientsocket):

        threading.Thread.\_\_init\_\_(self)

        self.csocket = clientsocket

        print("New connection added: ", clientAddress)

    def run(self):

        print("Connection from : ", clientAddress)

        self.csocket.send(bytes("Hi, This is from Server..", 'utf-8'))

        msg = ''

        while True:

            data = self.csocket.recv(2048)

            msg = data.decode()

            if msg == 'bye':

                break

            print("from client", msg)

            self.csocket.send(bytes(msg, 'UTF-8'))

        print("Client at ", clientAddress, " disconnected...")

        print("Waiting for new connection....\n")

LOCALHOST = "127.0.0.1"

PORT = 8080

server = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

server.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)

server.bind((LOCALHOST, PORT))

print("Server started")

print("Waiting for client request..")

server.listen(1)

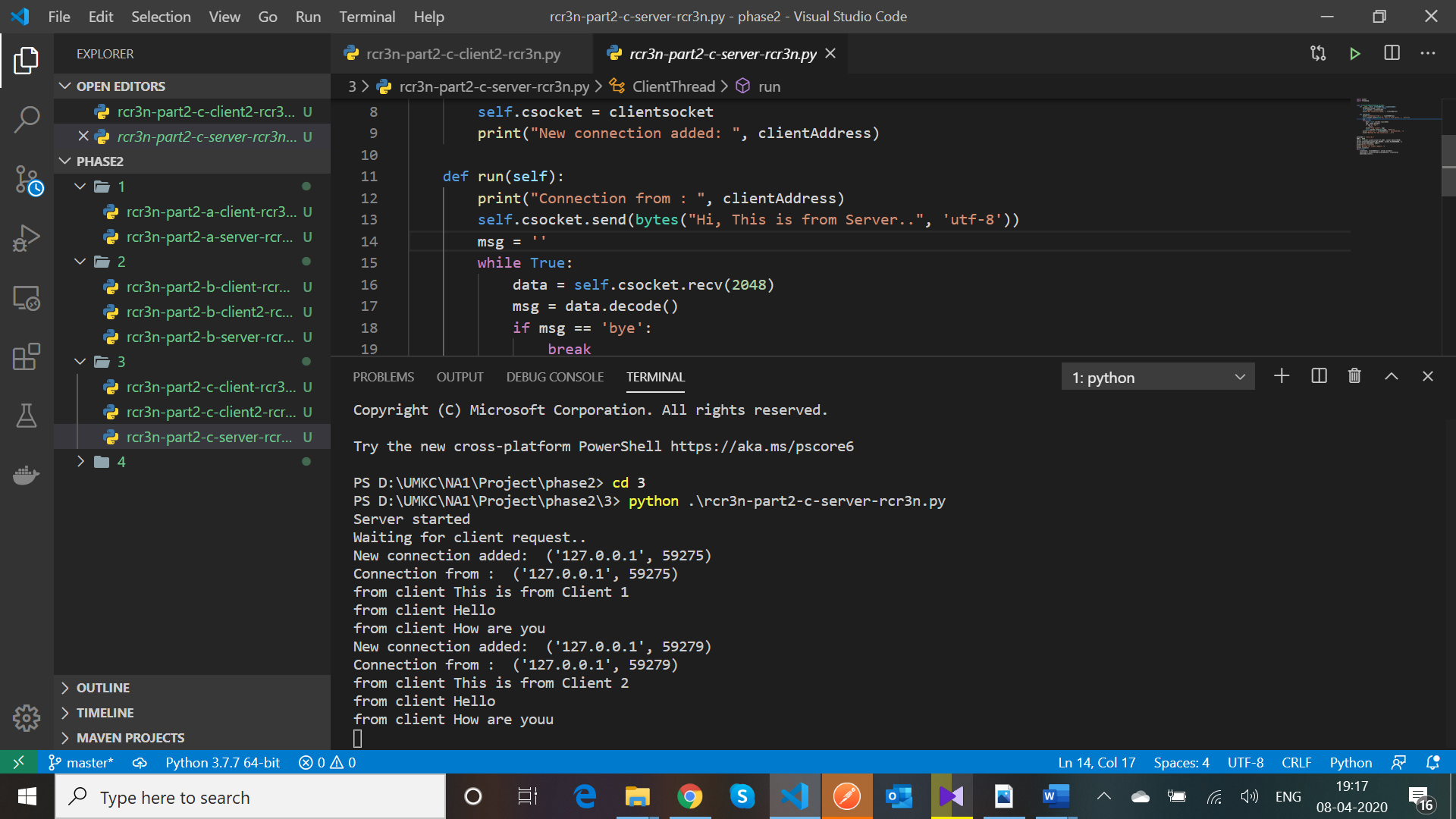
while True:

    clientsock, clientAddress = server.accept()

    newthread = ClientThread(clientAddress, clientsock)

    newthread.start()

**SERVER CODE OUTPUT:**



**SERVER CODE EXPLANATION:**

* First we need to import the socket and then create a TCP socket object.
* Then we need to bind the server with an IP address and Host number.
* Then the server accepts an incoming connection from the client using accept () method.
* Once the connection is established then it receives data from the client.
* If the data received from the client is “exit” then it will close the connection with the client and prints the connection with client is closed. The server is again now open for connection from new client
* If the data from the client is other than “exit” then it will print the data on the server terminal.
* Here, the server handles the multiple clients simultaneously and prints the data from the clients on the server terminal.

4) A server replies next 3 days temperature of Kansas city ( eg 30 C/45 F), when client sends “Weather” otherwise server echoes same message.

**CLIENT CODE:**

import socket

SERVER = "127.0.0.1"

PORT = 4000

client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client.connect((SERVER, PORT))

client.sendall(bytes("This is from Client 1", 'UTF-8'))

while True:

    in\_data = client.recv(1024)

    print("From Server :", in\_data.decode())

    out\_data = input()

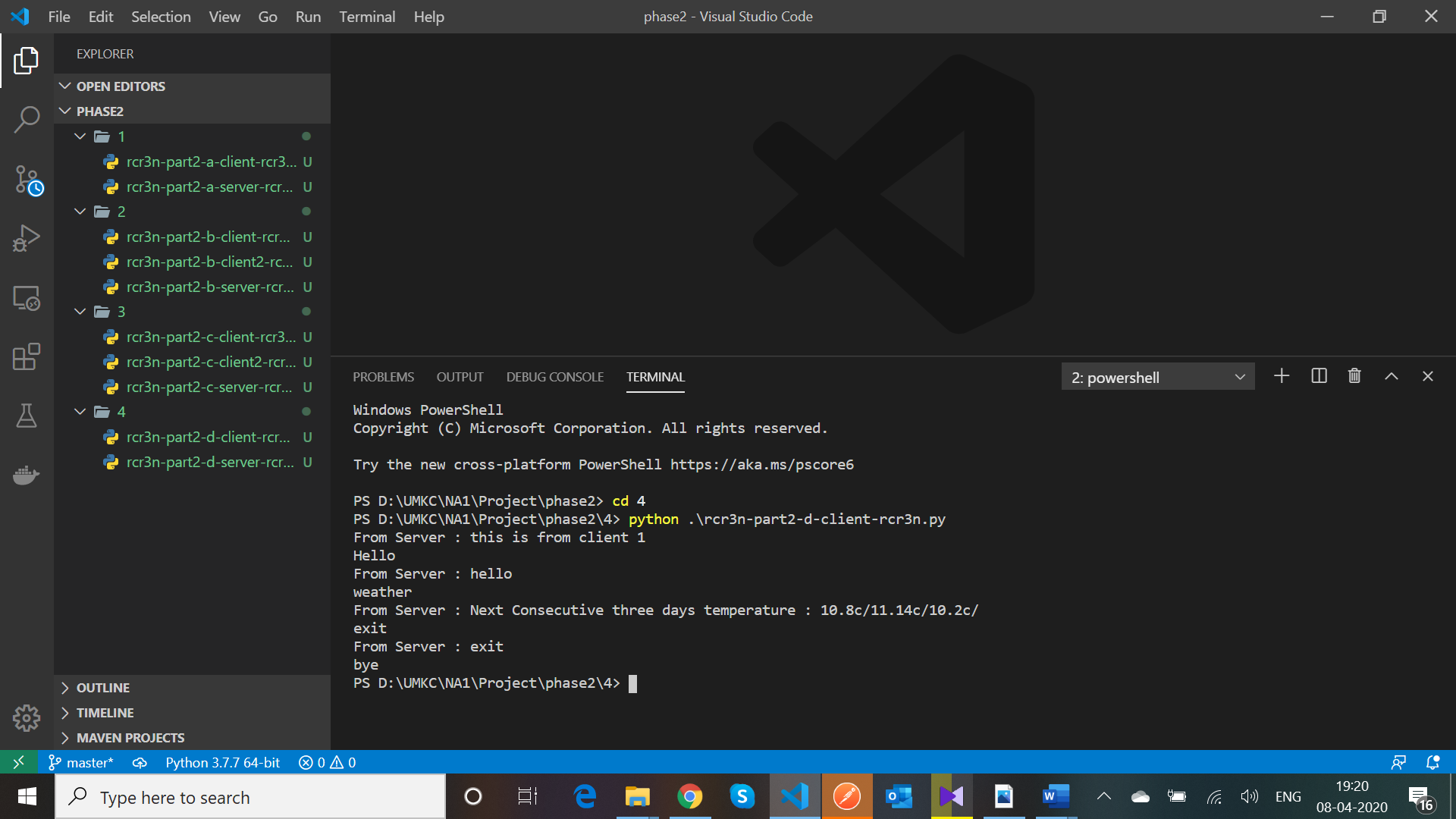
    client.sendall(bytes(out\_data, 'UTF-8'))

    if out\_data == 'bye':

        break

client.close()

**CLIENT CODE OUTPUT:**



**SERVER CODE:**

import socket

import json

import requests

HOST = ''

PORT = 4000

def connect():

    sock = socket.socket()

    server = sock.bind((HOST, PORT))

    sock.listen(1)

    print("Server is Listening...")

    conn, addr = sock.accept()

    return conn

def send\_messages(c):

    while True:

        msg = str(c.recv(1024).decode())

        msg = msg.lower()

        if msg == "bye":

            print(msg)

            c.send(str.encode("Server connection is closed"))

            c.close()

            c = connect()

            send\_messages(c)

        elif msg == "weather":

            response = requests.get(

                'https://api.darksky.net/forecast/4ab909be9e7658b9de4274c3ff28c8cd/37.8267,-122.4233?units=si&exclude=currently,flags,hourly,minutely')

            json\_response = response.json()

            repository = json\_response['daily']['data']

            message = ''

            for i in range(3):

                message += str(repository[i]['temperatureLow']) + 'c/'

            message = 'Next Consecutive three days temperature : ' + message

            c.send(str.encode(message))

        else:

            print(msg)

            c.send(str.encode(msg))

def main():

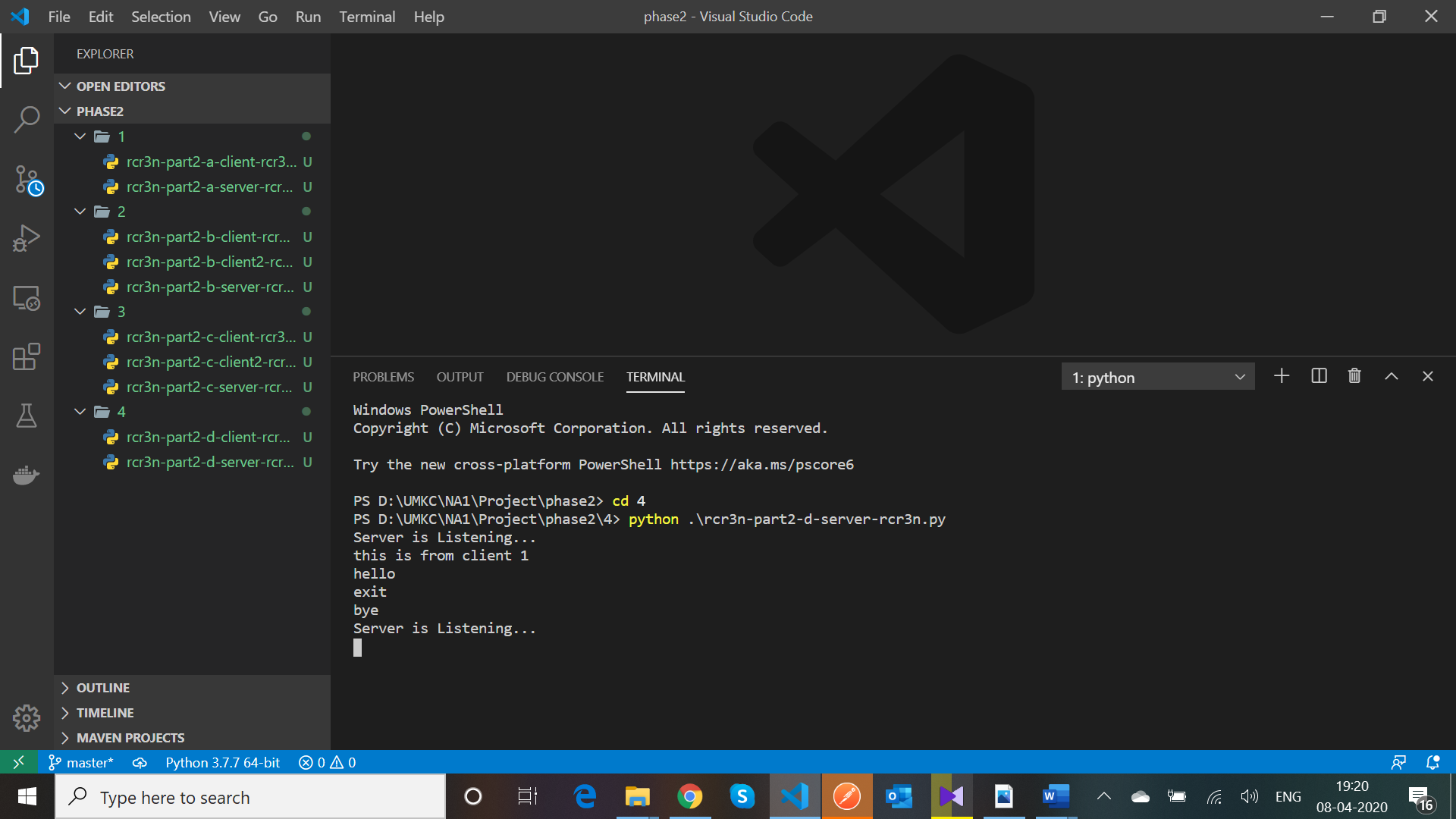
    c = connect()

    send\_messages(c)

if \_\_name\_\_ == '\_\_main\_\_':

    main()

**SERVER CODE OUTPUT:**



**SERVER CODE EXPLANATION:**

* First we need to import the socket and then create a TCP socket object.
* Then we need to bind the server with an IP address and Host number.
* Then the server accepts an incoming connection from the client using accept () method.
* Once the connection is established then it receives data from the client.
* If the data received from the client is “exit” then it will close the connection with the client and prints the connection with client is closed. The server is again now open for connection from new client
* If the data from the client is other than “exit” then it will print the data on the server terminal.
* If the data from client is “Weather”, then using the request method we make a call to the Darksky API to get the weather information of the Kansas city and will format the response to send the next three days temperature information to the client.