# Computer Network Lab
# CEN-593

# Mini Project

**Q. Write a socket program to implement TCP client and Server such that server should be able to send text to client and client check that the received number is composite or prime.**

## Team Members

20BCS027 - Harsh Raina

20BCS060 - Kovid Sharma

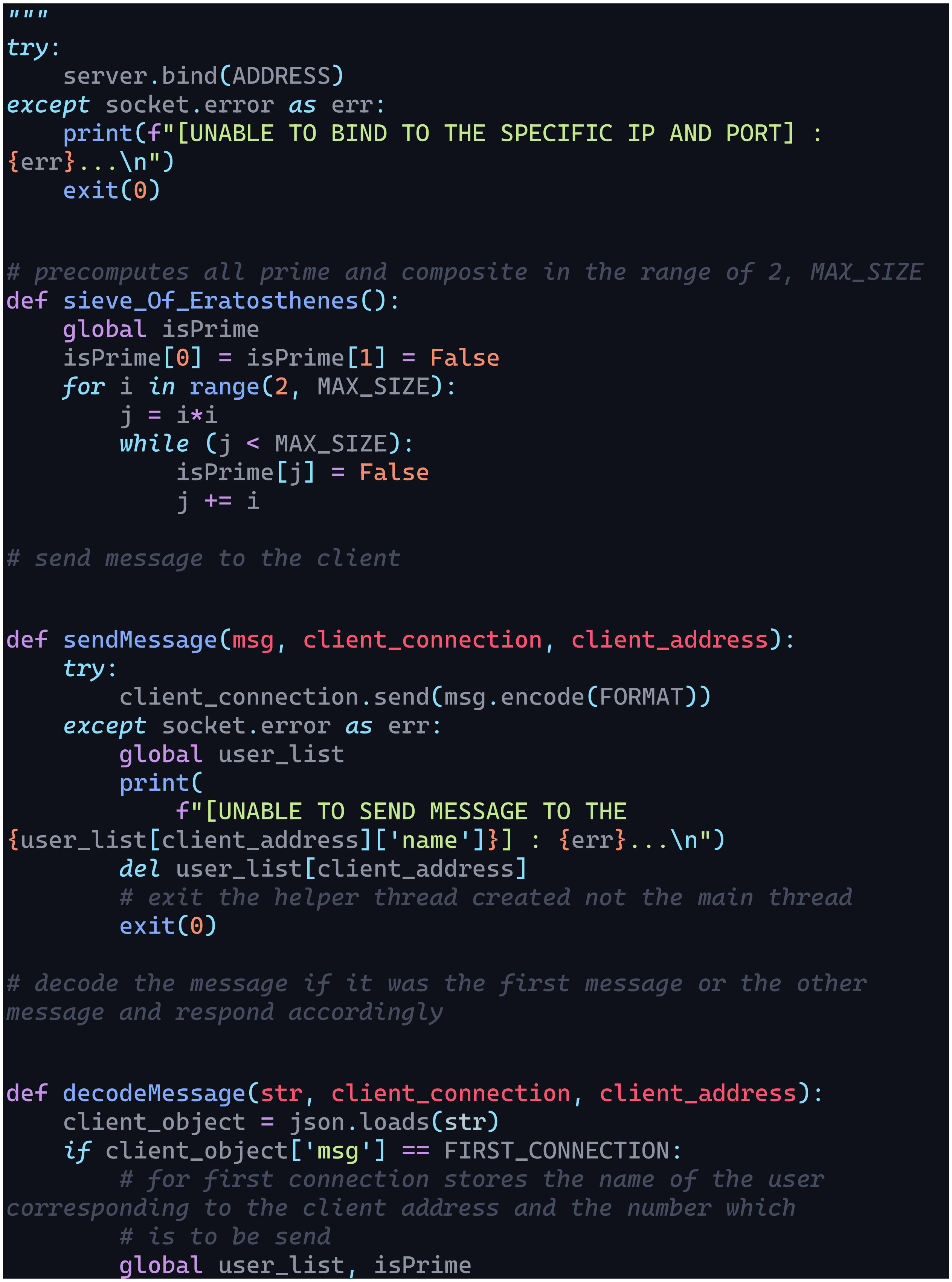
20BCS061 - Kshitiz Kumar

20BCS064 - Mohammed Haziq

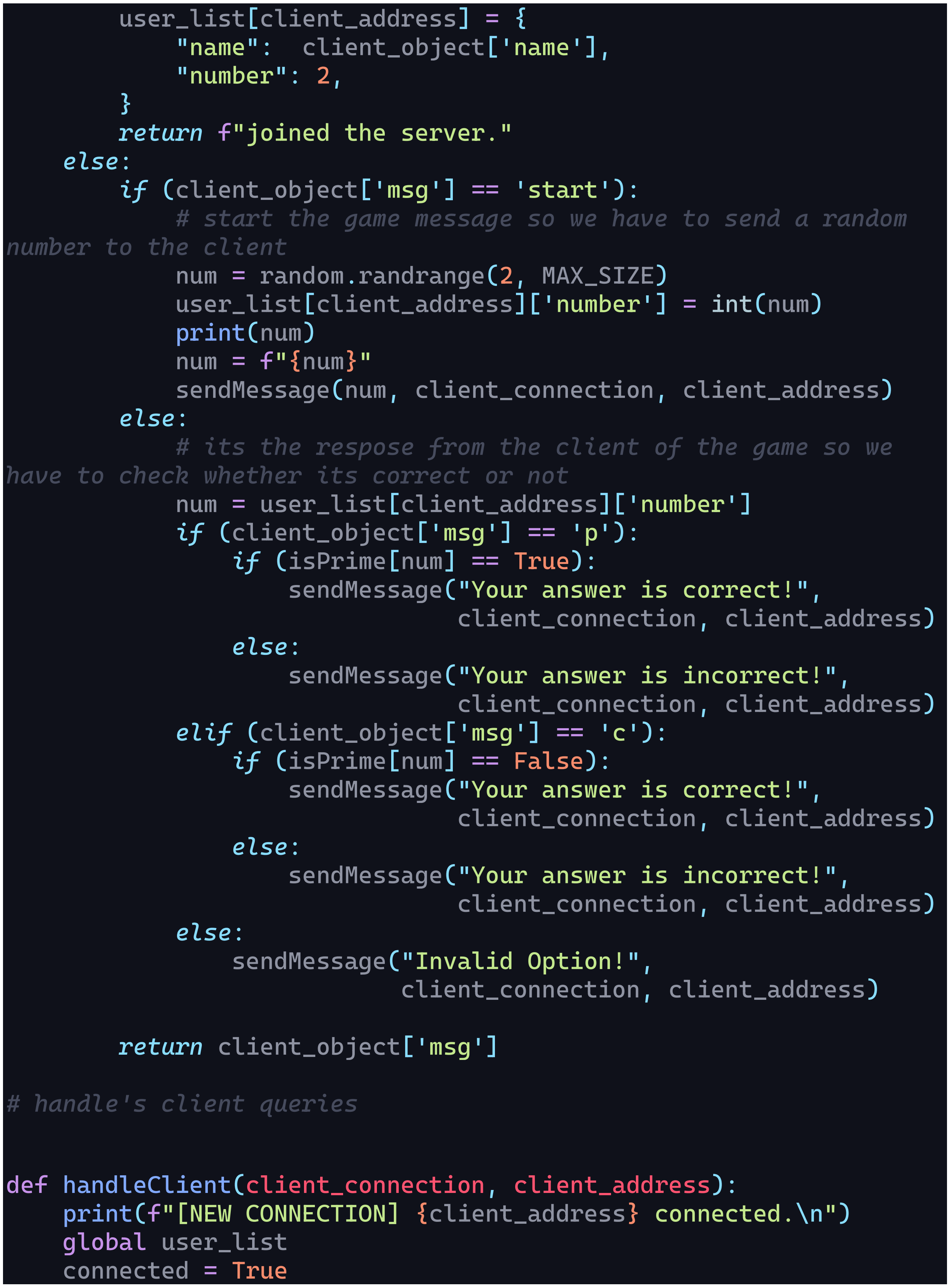
20BCS070 - Vicky Gupta (Group Leader)

## Server’s Code :-

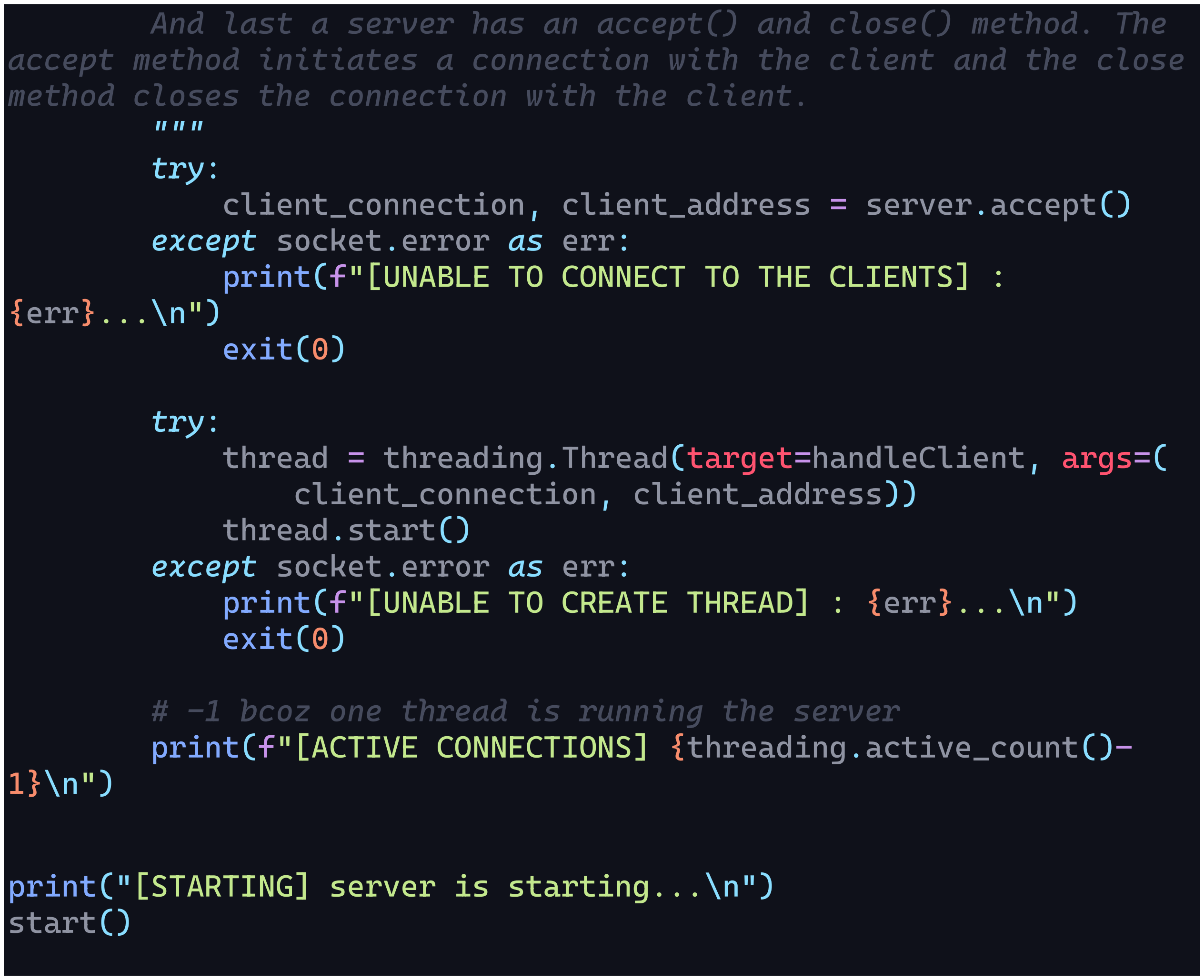
```python
import random
import socket
import threading
import json

print("\033c")

PORT = 4000
# size of data in bytes that can go in one packets
HEADER = 1024
FORMAT = "utf-8"
MAX_CLIENT = 2
DISCONNECT_MESSAGE = "!DISCONNECTED!"
FIRST_CONNECTION = "!FIRST_CONNECTION!"
SERVER = socket.gethostbyname(socket.gethostname())
ADDRESS = (SERVER, PORT)
MAX_SIZE = 1000001

# stores the client information like username
user_list = {}

# store the information that number is prime or composite
# True means prime and False means composite
isPrime = [True]*MAX_SIZE

"""
Here we made a socket instance and passed it two parameters. The
first parameter is AF_INET and the second one is SOCK_STREAM.
AF_INET refers to the address-family ipv4. The SOCK_STREAM means
connection-oriented TCP protocol.
"""
try:
    server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
except socket.error as err:
    print(f"[UNABLE TO CREATE SOCKET] : {err}...\n")
    exit(0)

"""
A server has a bind() method which binds it to a specific IP and
port so that it can listen to incoming requests on that IP and
port.
```



```
"""
try:
    server.bind(ADDRESS)
except socket.error as err:
    print(f"[UNABLE TO BIND TO THE SPECIFIC IP AND PORT] :
{err}...\n")
    exit(0)


# precomputes all prime and composite in the range of 2, MAX_SIZE
def sieve_Of_Eratosthenes():
    global isPrime
    isPrime[0] = isPrime[1] = False
    for i in range(2, MAX_SIZE):
        j = i*i
        while (j < MAX_SIZE):
            isPrime[j] = False
            j += i

# send message to the client


def sendMessage(msg, client_connection, client_address):
    try:
        client_connection.send(msg.encode(FORMAT))
    except socket.error as err:
        global user_list
        print(
            f"[UNABLE TO SEND MESSAGE TO THE
{user_list[client_address]['name']}] : {err}...\n")
        del user_list[client_address]
        # exit the helper thread created not the main thread
        exit(0)

# decode the message if it was the first message or the other
message and respond accordingly


def decodeMessage(str, client_connection, client_address):
    client_object = json.loads(str)
    if client_object['msg'] == FIRST_CONNECTION:
        # for first connection stores the name of the user
corresponding to the client address and the number which
        # is to be send
        global user_list, isPrime
```



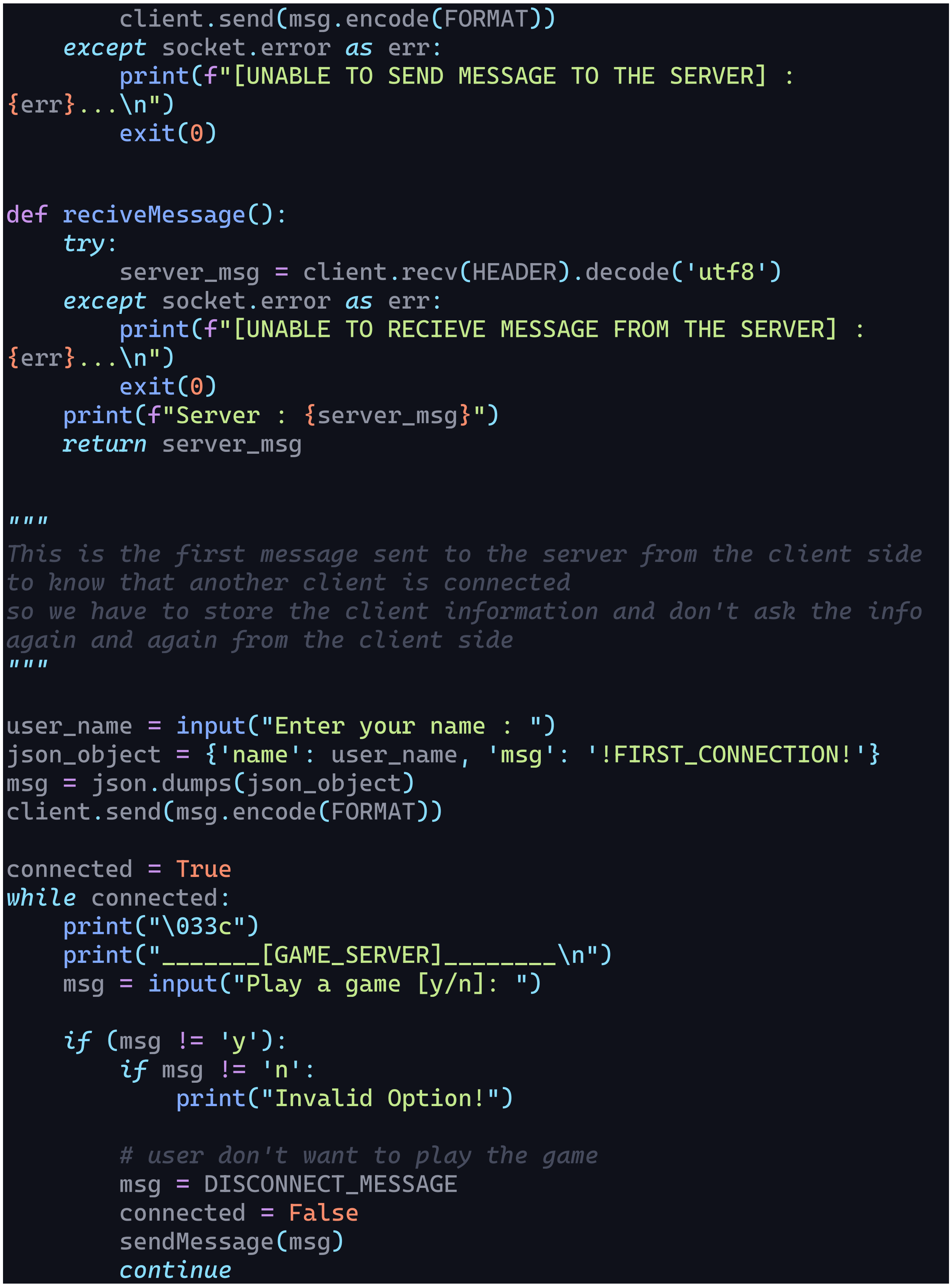
```
        user_list[client_address] = {
            "name":  client_object['name'],
            "number": 2,
        }
        return f"joined the server."
    else:
        if (client_object['msg'] == 'start'):
            # start the game message so we have to send a random
number to the client
            num = random.randrange(2, MAX_SIZE)
            user_list[client_address]['number'] = int(num)
            print(num)
            num = f"{num}"
            sendMessage(num, client_connection, client_address)
        else:
            # its the respose from the client of the game so we
have to check whether its correct or not
            num = user_list[client_address]['number']
            if (client_object['msg'] == 'p'):
                if (isPrime[num] == True):
                    sendMessage("Your answer is correct!",
                                client_connection, client_address)
                else:
                    sendMessage("Your answer is incorrect!",
                                client_connection, client_address)
            elif (client_object['msg'] == 'c'):
                if (isPrime[num] == False):
                    sendMessage("Your answer is correct!",
                                client_connection, client_address)
                else:
                    sendMessage("Your answer is incorrect!",
                                client_connection, client_address)
            else:
                sendMessage("Invalid Option!",
                            client_connection, client_address)

        return client_object['msg']

# handle's client queries


def handleClient(client_connection, client_address):
    print(f"[NEW CONNECTION] {client_address} connected.\n")
    global user_list
    connected = True
    while connected:
        # reciveing response from client
        try:
            str = client_connection.recv(HEADER).decode(FORMAT)
        except socket.error as err:
            print(
                f"[UNABLE TO RECIVE MESSAGE FROM THE
{user_list[client_address]['name']}] : {err}...\n")
            del user_list[client_address]
            # exit the helper thread created not the main thread
            exit(0)

        if len(str) == 0:
            continue

        msg = decodeMessage(str, client_connection,
client_address)
        if msg == DISCONNECT_MESSAGE:
            # disconnect the client from the server if message is
!DISCONNECTED!
            connected = False
            print(f"{user_list[client_address]['name']} is offline
now.")
            continue

        print(f"{user_list[client_address]['name']} : {msg}")

    # removing the client from the list after he/she get
disconnected
    del user_list[client_address]
    client_connection.close()


def start():
    """
    A server has a listen() method which puts the server into
listening mode. This allows the server to listen to incoming
connections.
    """
    server.listen(MAX_CLIENT)
    sieve_Of_Eratosthenes()
    print(f"[LISTENING]  server is listening on {SERVER}\n")
    connected = True
    while connected:
        """
```



```
        And last a server has an accept() and close() method. The
accept method initiates a connection with the client and the close
method closes the connection with the client.
        """
        try:
            client_connection, client_address = server.accept()
        except socket.error as err:
            print(f"[UNABLE TO CONNECT TO THE CLIENTS] :
{err}...\n")
            exit(0)

        try:
            thread = threading.Thread(target=handleClient, args=(
                client_connection, client_address))
            thread.start()
        except socket.error as err:
            print(f"[UNABLE TO CREATE THREAD] : {err}...\n")
            exit(0)

        # -1 bcoz one thread is running the server
        print(f"[ACTIVE CONNECTIONS] {threading.active_count()-
1}\n")


print("[STARTING] server is starting...\n")
start()
```

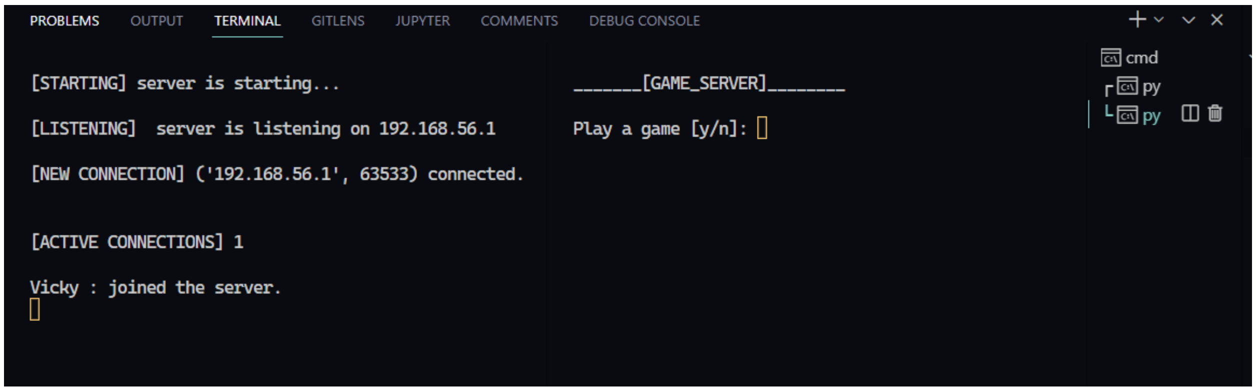
## Client’s Code:-

```python
import socket
import json

print("\033c")

PORT = 4000
# size of data in bytes that can go in one packets
HEADER = 1024
FORMAT = "utf-8"
DISCONNECT_MESSAGE = "!DISCONNECTED!"

# get the server ip address

SERVER = socket.gethostbyname(socket.gethostname())
ADDRESS = (SERVER, PORT)


"""
Here we made a socket instance and passed it two parameters. The
first parameter is AF_INET and the second one is SOCK_STREAM.
AF_INET refers to the address-family ipv4. The SOCK_STREAM means
connection-oriented TCP protocol.
"""

try:
    client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
except socket.error as err:
    print(f"[UNABLE TO CREATE SOCKET] : {err}...\n")
    exit(0)
try:
    # coneecting the client to the server
    client.connect(ADDRESS)
except socket.error as err:
    print(f"[UNABLE TO CONNECT TO THE SERVER] : {err}...\n")
    exit(0)


# function to send message to the server
def sendMessage(msg):
    json_object = {'msg': msg}
    msg = json.dumps(json_object)
    try:
```



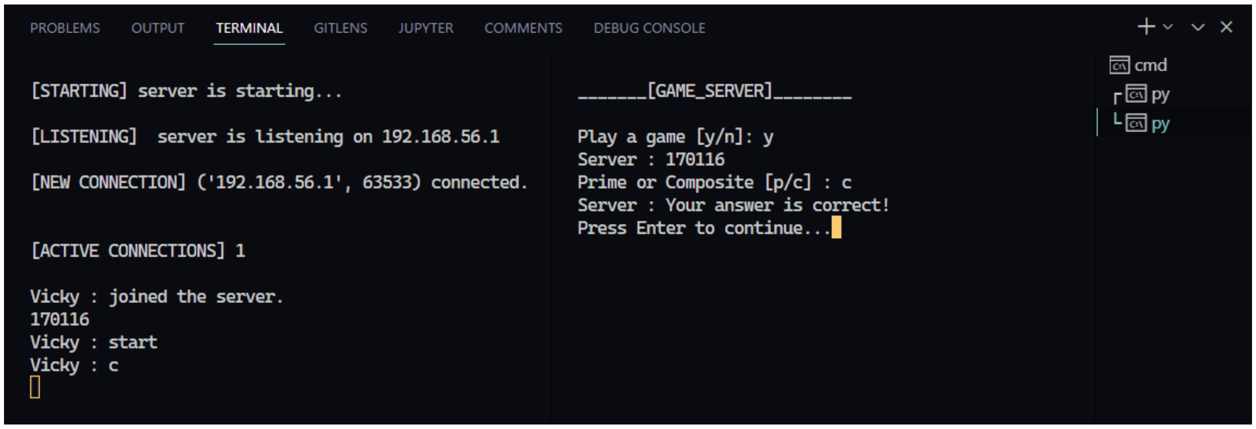
```python
        client.send(msg.encode(FORMAT))
    except socket.error as err:
        print(f"[UNABLE TO SEND MESSAGE TO THE SERVER] :
{err}...\n")
        exit(0)


def reciveMessage():
    try:
        server_msg = client.recv(HEADER).decode('utf8')
    except socket.error as err:
        print(f"[UNABLE TO RECIEVE MESSAGE FROM THE SERVER] :
{err}...\n")
        exit(0)
    print(f"Server : {server_msg}")
    return server_msg


"""
This is the first message sent to the server from the client side
to know that another client is connected
so we have to store the client information and don't ask the info
again and again from the client side
"""

user_name = input("Enter your name : ")
json_object = {'name': user_name, 'msg': '!FIRST_CONNECTION!'}
msg = json.dumps(json_object)
client.send(msg.encode(FORMAT))

connected = True
while connected:
    print("\033c")
    print("_______[GAME_SERVER]________\n")
    msg = input("Play a game [y/n]: ")

    if (msg != 'y'):
        if msg != 'n':
            print("Invalid Option!")

        # user don't want to play the game
        msg = DISCONNECT_MESSAGE
        connected = False
        sendMessage(msg)
        continue
    # client response to the server that user wants to play the
game
    sendMessage('start')

    # server response to the client with a question
    server_msg = reciveMessage()

    # client response to the server to answer the question
    msg = input("Prime or Composite [p/c] : ")
    sendMessage(msg)

    # server response to the client wheter the answer is correct
or not
    server_msg = reciveMessage()

    input("Press Enter to continue...")

# closing the connection fromt the server
print("Connection Closed!")
client.close()
```

# Output :-



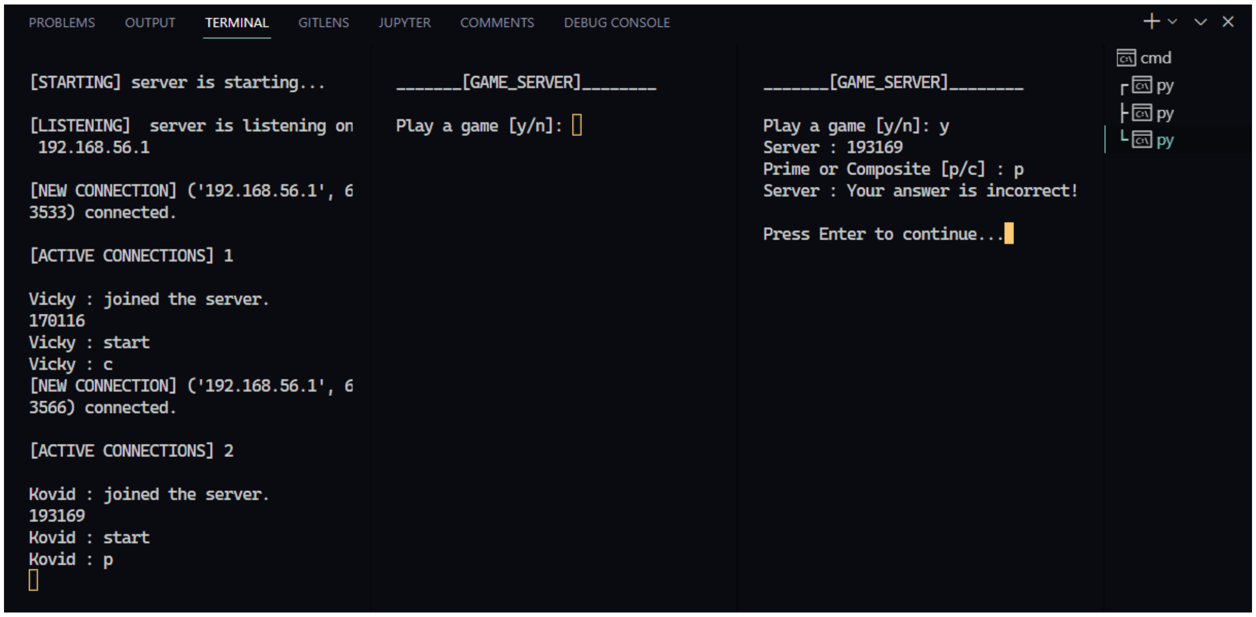
```
[STARTING] server is starting...

[LISTENING]  server is listening on 192.168.56.1

[NEW CONNECTION] ('192.168.56.1', 63533) connected.


[ACTIVE CONNECTIONS] 1

Vicky : joined the server.
```

```
_______[GAME_SERVER]________

Play a game [y/n]:
```



```
[STARTING] server is starting...

[LISTENING]  server is listening on 192.168.56.1

[NEW CONNECTION] ('192.168.56.1', 63533) connected.


[ACTIVE CONNECTIONS] 1

Vicky : joined the server.
170116
Vicky : start
Vicky : c
```

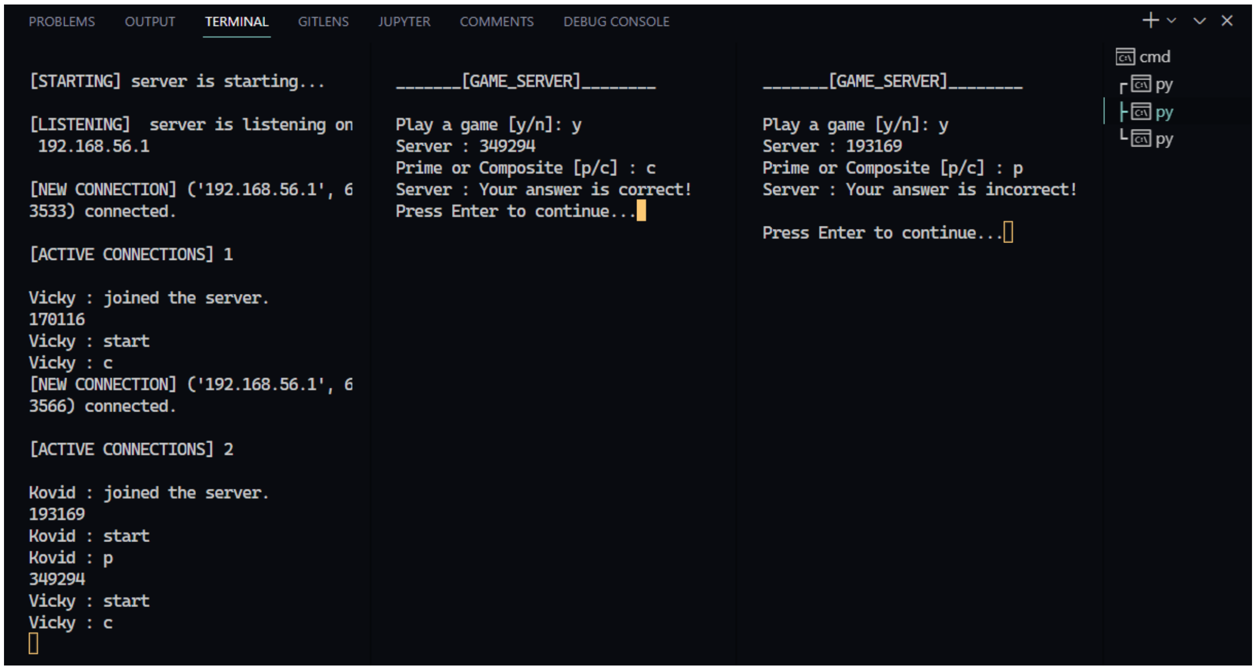
```
_______[GAME_SERVER]________

Play a game [y/n]: y
Server : 170116
Prime or Composite [p/c] : c
Server : Your answer is correct!
Press Enter to continue...
```



```
[STARTING] server is starting...

[LISTENING]  server is listening on
 192.168.56.1

[NEW CONNECTION] ('192.168.56.1', 6
3533) connected.

[ACTIVE CONNECTIONS] 1

Vicky : joined the server.
170116
Vicky : start
Vicky : c
[NEW CONNECTION] ('192.168.56.1', 6
3566) connected.

[ACTIVE CONNECTIONS] 2

Kovid : joined the server.
193169
Kovid : start
Kovid : p
```

```
_______[GAME_SERVER]________

Play a game [y/n]:
```

```
_______[GAME_SERVER]________

Play a game [y/n]: y
Server : 193169
Prime or Composite [p/c] : p
Server : Your answer is incorrect!

Press Enter to continue...
```

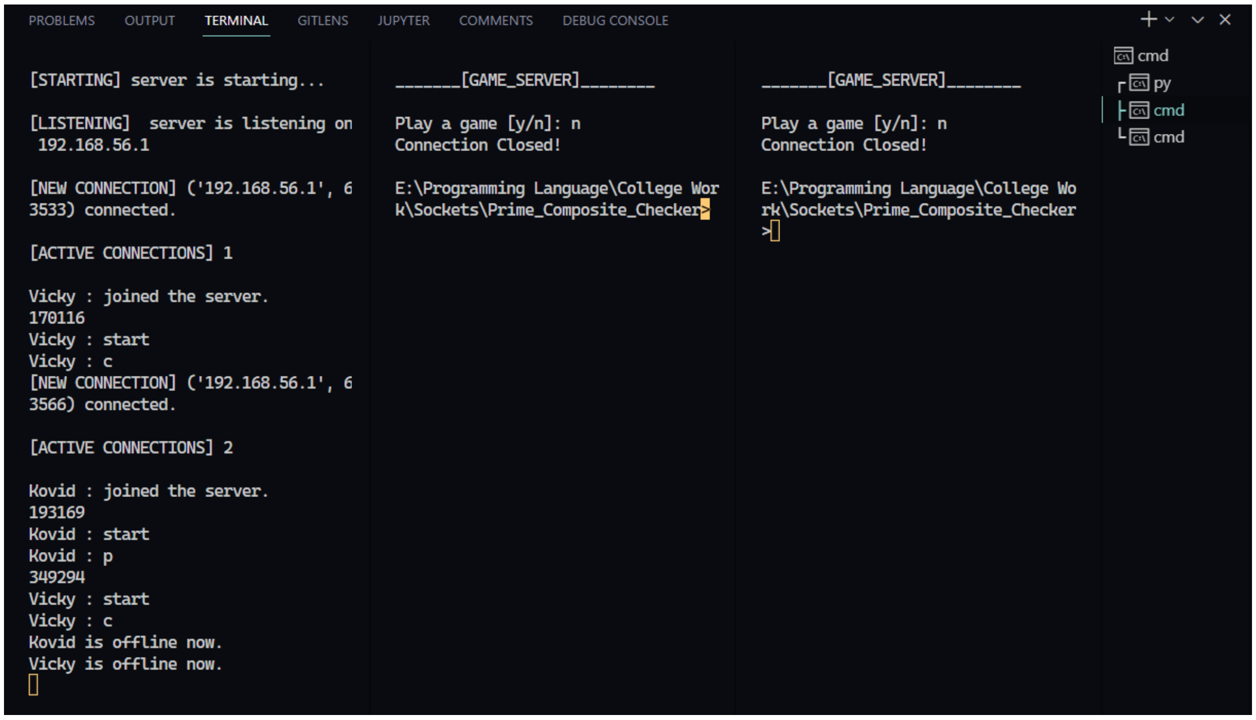


PROBLEMS OUTPUT TERMINAL GITLENS JUPYTER COMMENTS DEBUG CONSOLE

```
[STARTING] server is starting...

[LISTENING]  server is listening on
 192.168.56.1

[NEW CONNECTION] ('192.168.56.1', 6
3533) connected.

[ACTIVE CONNECTIONS] 1

Vicky : joined the server.
170116
Vicky : start
Vicky : c
[NEW CONNECTION] ('192.168.56.1', 6
3566) connected.

[ACTIVE CONNECTIONS] 2

Kovid : joined the server.
193169
Kovid : start
Kovid : p
349294
Vicky : start
Vicky : c
```

```
_______[GAME_SERVER]________

Play a game [y/n]: y
Server : 349294
Prime or Composite [p/c] : c
Server : Your answer is correct!
Press Enter to continue...
```

```
_______[GAME_SERVER]________

Play a game [y/n]: y
Server : 193169
Prime or Composite [p/c] : p
Server : Your answer is incorrect!

Press Enter to continue...
```



PROBLEMS OUTPUT TERMINAL GITLENS JUPYTER COMMENTS DEBUG CONSOLE

```
[STARTING] server is starting...

[LISTENING]  server is listening on
 192.168.56.1

[NEW CONNECTION] ('192.168.56.1', 6
3533) connected.

[ACTIVE CONNECTIONS] 1

Vicky : joined the server.
170116
Vicky : start
Vicky : c
[NEW CONNECTION] ('192.168.56.1', 6
3566) connected.

[ACTIVE CONNECTIONS] 2

Kovid : joined the server.
193169
Kovid : start
Kovid : p
349294
Vicky : start
Vicky : c
Kovid is offline now.
Vicky is offline now.
```

```
_______[GAME_SERVER]________

Play a game [y/n]: n
Connection Closed!

E:\Programming Language\College Wor
k\Sockets\Prime_Composite_Checker>
```

```
_______[GAME_SERVER]________

Play a game [y/n]: n
Connection Closed!

E:\Programming Language\College Wo
rk\Sockets\Prime_Composite_Checker
>
```