The code implements a simple UDP (User Datagram Protocol) client-server application that facilitates message exchange between multiple clients. The server listens for messages on a specified port and can broadcast messages to all clients on the local network. Clients can send messages and request to view messages sent by others. Here's a general overview of how the code functions:

1. **Server Setup**:

- The server is configured to listen on port 5051.
- It uses a broadcast IP (192.168.1.255) to send messages to all clients in the network.
- The server socket is set up to handle UDP packets, which are connectionless and suitable for broadcasting messages.

2. **Client Initialization**:

- Each client is prompted to input their first and last name, which is then combined to form the client_name.
- This name is used to identify messages sent by the client and to filter out the client's own messages from the displayed list of received messages.

3. **Message Handling**:

- **Receiving Messages**:
- The server continuously listens for incoming messages using a separate thread.
- Upon receiving a message, it decodes and logs the message along with the sender's information and timestamp.
- The server also prints a notification of received messages from other clients.

- **Sending Messages**:

- Clients can enter messages which are then sent to the server.
- Each message is formatted with the sender's first name, last name, and the message content, separated by semicolons (;).

4. **Displaying Messages**:

- Clients have the option to view all received messages.
- Messages are listed with a unique index, and clients can select a specific message to view its details.

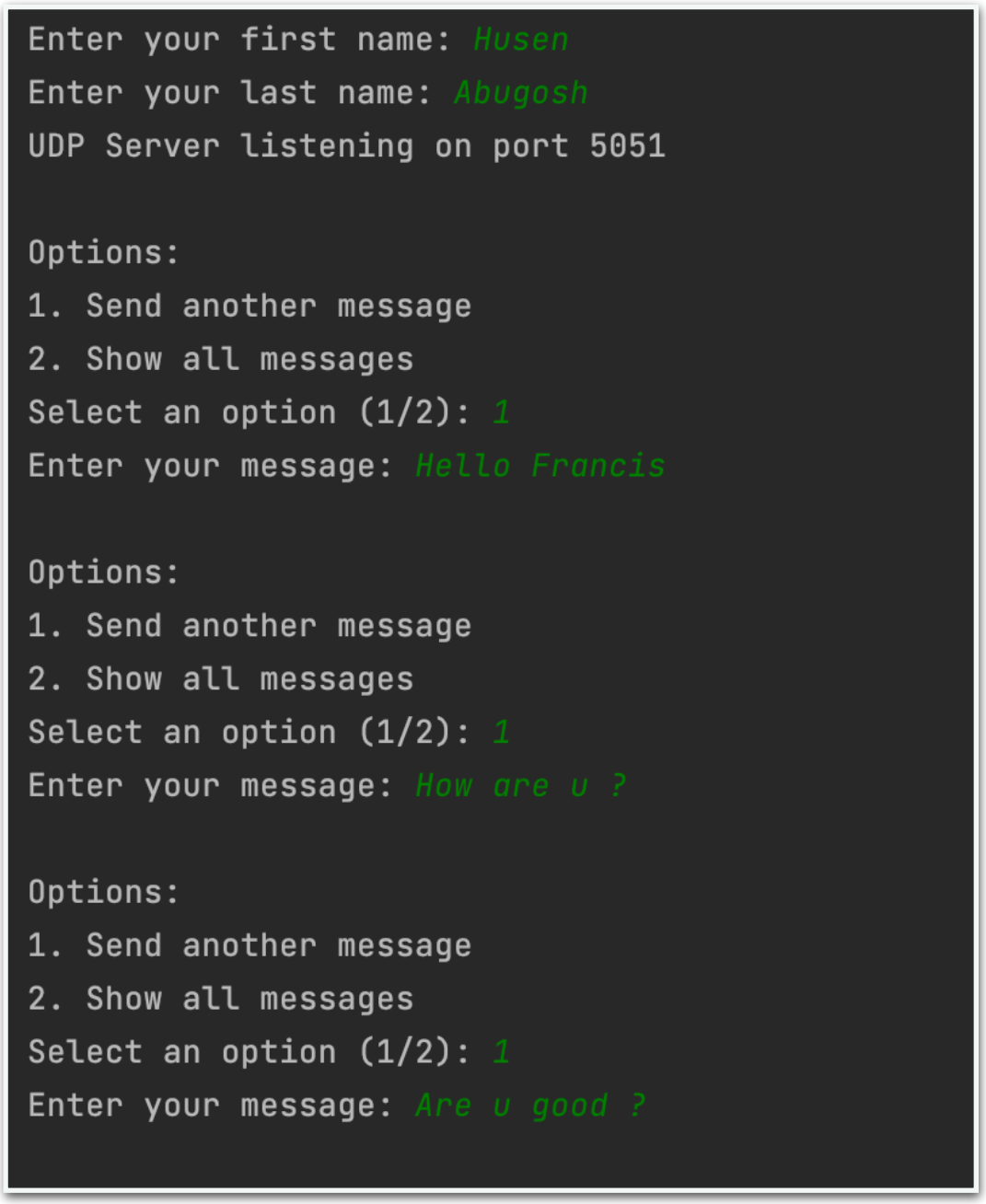
5. **Concurrency**:

- The server uses threading to handle message reception in the background while allowing the main program to continue running, enabling simultaneous sending and receiving of messages.

```python
import socket
import datetime
import threading

# Server settings:
server_port = 5051
buffer_size = 1024
broadcast_ip = '192.168.1.255'

# to get and save the user names :
client_name = None
if client_name is None:
    first_name = input("Enter your first name: ")
    last_name = input("Enter your last name: ")
    client_name = f"{first_name} {last_name}"

clients = {}  # <--this a  Dictionary to keep track of client addresses and last messages
received_messages = []  # <--  a list to store the messages

# Create a UDP socket
server_socket = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)  # <-- to  creates a UDP socket.
server_socket.setsockopt(socket.SOL_SOCKET, socket.SO_BROADCAST, 1)  # <-- to allow broadcasting.
server_socket.bind(('', server_port))  # <-- to binds the socket to the specified port.

print("UDP Server listening on port", server_port)


#Prints a list of received messages, excluding messages from the client itself.
def display_received_messages():
    if received_messages:
        print("\nList of received messages:")
        for idx, (sender, time, msg) in enumerate(received_messages, start=1):
            if sender != client_name:
                print(f"-{idx} received a message from {sender} at {time}")
    else:
        print("\nNo messages to display.")


def rec():
    # Receive message from any client
    message, client_address = server_socket.recvfrom(buffer_size)
    decoded_message = message.decode()
    current_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")

    # Update client's last message and timestamp
    sender_info = decoded_message.split(';')
    if len(sender_info) == 3:
        sender = f"{sender_info[0]} {sender_info[1]}"
        if client_name != sender:
            print(f"\nReceived message from {sender} at {current_time}")
        received_messages.append((sender, current_time, sender_info[2]))
        rec()

# creating a new thread to run the rec function, allowing the server to receive messages concurrently
with other tasks:
t1 = threading.Thread(target=rec)
t1.start()


#to show the options menu:
while True:

    print("\nOptions:\n1. Send another message\n2. Show all messages")
    option = input("Select an option (1/2): ")

    if option == '1':

        message = input("Enter your message: ")
        send_message = f"{first_name};{last_name};{message}"
        server_socket.sendto(send_message.encode(), (broadcast_ip, server_port)) #<-- to send it via
UDP to the broadcast address.

    elif option == '2':
        display_received_messages()
        choice = input("Enter line number followed by 'D' to display the message (e.g., 1D): ")
        if choice.endswith('D') and choice[:-1].isdigit():
            index = int(choice[:-1]) - 1
            if 0 <= index < len(received_messages):
                print(
                    f"Message from {received_messages[index][0]} at {received_messages[index][1]}:
{received_messages[index][2]}")
            else:
                prin"Invalid line number.")
    else:
        print("Invalid option. Please select 1 or 2.")
```

Key Features:

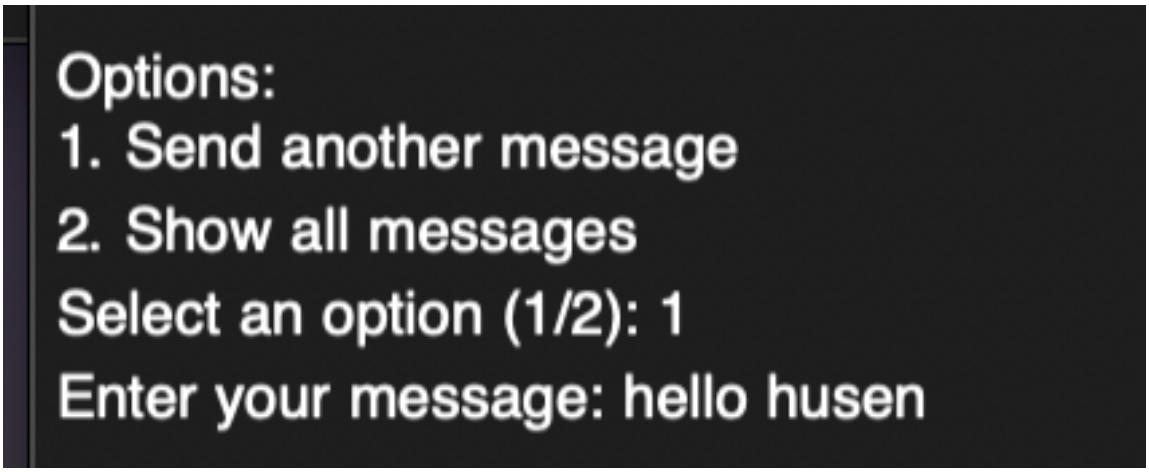
- **Broadcast Communication**: Uses UDP broadcasting to send messages to all clients in the local network.
- **Concurrency**: Utilizes threading to handle incoming messages concurrently with other operations.
- **Message Filtering**: Ensures clients do not see their own messages in the received list.
- **User Interaction**: Provides a simple interface for clients to send messages and view received messages.



The first user Husen send his messages to all peers

```
Enter your first name: Francis
Enter your last name: Miadi
UDP Server listening on port 5051

Options:
1. Send another message
2. Show all messages
Select an option (1/2):
Received message from Husen Abugosh at 2024-05-10 19:46:27

Received message from Husen Abugosh at 2024-05-10 19:46:33

Received message from Husen Abugosh at 2024-05-10 19:46:49
2

List of received messages:
-1 received a message from Husen Abugosh at 2024-05-10 19:46:27
-2 received a message from Husen Abugosh at 2024-05-10 19:46:33
-3 received a message from Husen Abugosh at 2024-05-10 19:46:49
Enter line number followed by 'D' to display the message (e.g., 1D): 1D
Message from Husen Abugosh at 2024-05-10 19:46:27: Hello Francis
```

The second user Francis received the messages from Husen and he can see the details of the message by chose the line with D char



The second user Francis chose to send message to the all peers

```
Options:
1. Send another message
2. Show all messages
Select an option (1/2):
Received message from Francis Miadi at 2024-05-10 19:51:38
2

List of received messages:
-1 received a message from Francis Miadi at 2024-05-10 19:51:38
Enter line number followed by 'D' to display the message (e.g., 1D): -1
Enter line number followed by 'D' to display the message (e.g., 1D):

Options:
1. Send another message
2. Show all messages
Select an option (1/2): 2

List of received messages:
-1 received a message from Francis Miadi at 2024-05-10 19:51:38
Enter line number followed by 'D' to display the message (e.g., 1D): 1D
Message from Francis Miadi at 2024-05-10 19:51:38: hello husen
```

The first user Husen received Francis message and chose to showed it up

The screenshots demonstrate the following interactions:

1. **Client Initialization**:

- Users input their first and last names.
- The server acknowledges its readiness to listen on the designated port.

2. **Sending Messages**:

- Clients enter messages which are then broadcasted to the network.
- Messages like "Hello Francis," "How are u?", and "Are u good?" are shown being sent.

3. **Receiving Messages**:
- The server receives messages from clients and displays them, excluding the sender's own messages.

4. **Viewing Messages**:
- Clients can list all received messages and view details of specific messages by entering the corresponding index followed by 'D'.