

**Faculty of Engineering and Technology**

**Department of Electrical and Computer Engineering**

**ENCS3320-Computer Networks**

**Task -UDP Client-Server Trivia Game Using Socket Programming**

**Prepared by:** Lara Abu Asfour 1221484

Hala Sabobeh 1220322

Tasneem Shelleh 1220439

**Instructor:** Dr. Ahmed Shawahna,

Dr. Ibrahim Nemer

**Section:** Lara Sec 1

Hala & Tasneem Sec 4

**Date:** 1/12/2024

# **Abstract**

Imagine playing a trivia game where multiple players connect to a server, answer questions, and compete for the top score all in real time. This is made possible using UDP, a fast and lightweight protocol designed for quick communication. Unlike other protocols like TCP, which ensures every packet of data arrives safely, UDP focuses on speed, making it perfect for games where every second counts. The server acts as the game host, sending questions to all players and keeping track of their answers and scores. Players, as clients, receive the questions, submit their responses, and wait for updates from the server. This task shows how UDP and socket programming come together to create a fun and interactive multiplayer experience, while balancing speed and reliability in a distributed environment.

**Table of Contents**

[**Abstract** 2](#_Toc186179429)

[**Table of figures** 3](#_Toc186179430)

[**Codes & Discussion:** 4](#_Toc186179431)

[ **The server code:** 4](#_Toc186179432)

[ **Server code discussion:** 6](#_Toc186179433)

[ **The client code:** 7](#_Toc186179434)

[ **Client code discussion:** 7](#_Toc186179435)

[**Results and Discussion:** 8](#_Toc186179436)

[- **The first example:** 8](#_Toc186179437)

[- **The second example:** 9](#_Toc186179438)

[- **More than one round example:** 10](#_Toc186179439)

[ **Discussion:** 11](#_Toc186179440)

[**Conclusion** 12](#_Toc186179441)

# **Table of figures**

[Figure 1: server code 4](#_Toc186179407)

[Figure 2: server code 4](#_Toc186179408)

[Figure 3: server code 4](#_Toc186179409)

[Figure 4: server code 5](#_Toc186179410)

[Figure 5: server code 5](#_Toc186179411)

[Figure 6: server code 5](#_Toc186179412)

[Figure 7: the client code 7](#_Toc186179413)

[Figure 8: the client code 7](#_Toc186179414)

[Figure 9: the output of server Ex1 8](#_Toc186179415)

[Figure 10: the output of client1 Ex1 8](#_Toc186179416)

[Figure 11: the output of client2 EX1 8](#_Toc186179417)

[Figure 12: output of server EX2 9](#_Toc186179418)

[Figure 13: output of client1 EX2 9](#_Toc186179419)

[Figure 14: output of client2 EX2 9](#_Toc186179420)

[Figure 15: output of server EX3 10](#_Toc186179421)

[Figure 16:output of client2 EX3 10](#_Toc186179422)

## **Codes & Discussion:**

### **The server code:**

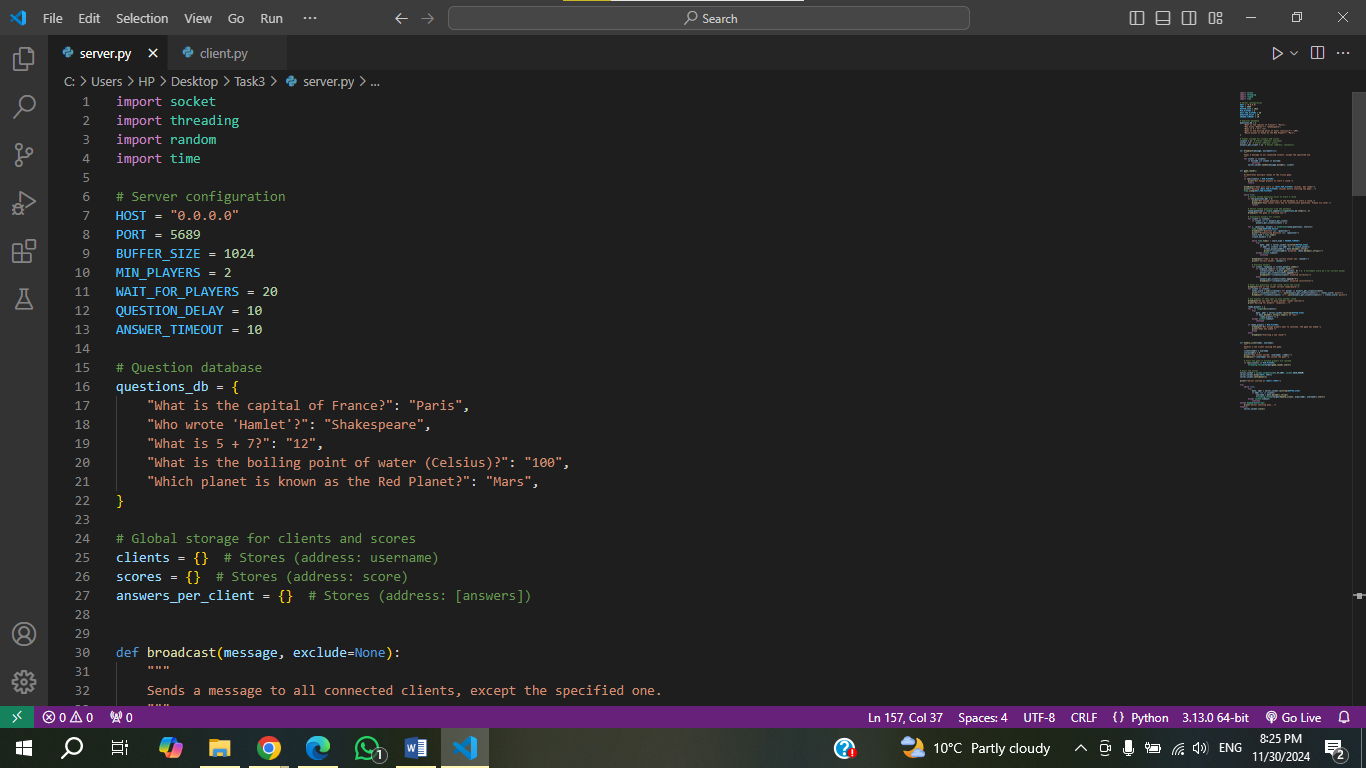


Figure : server code

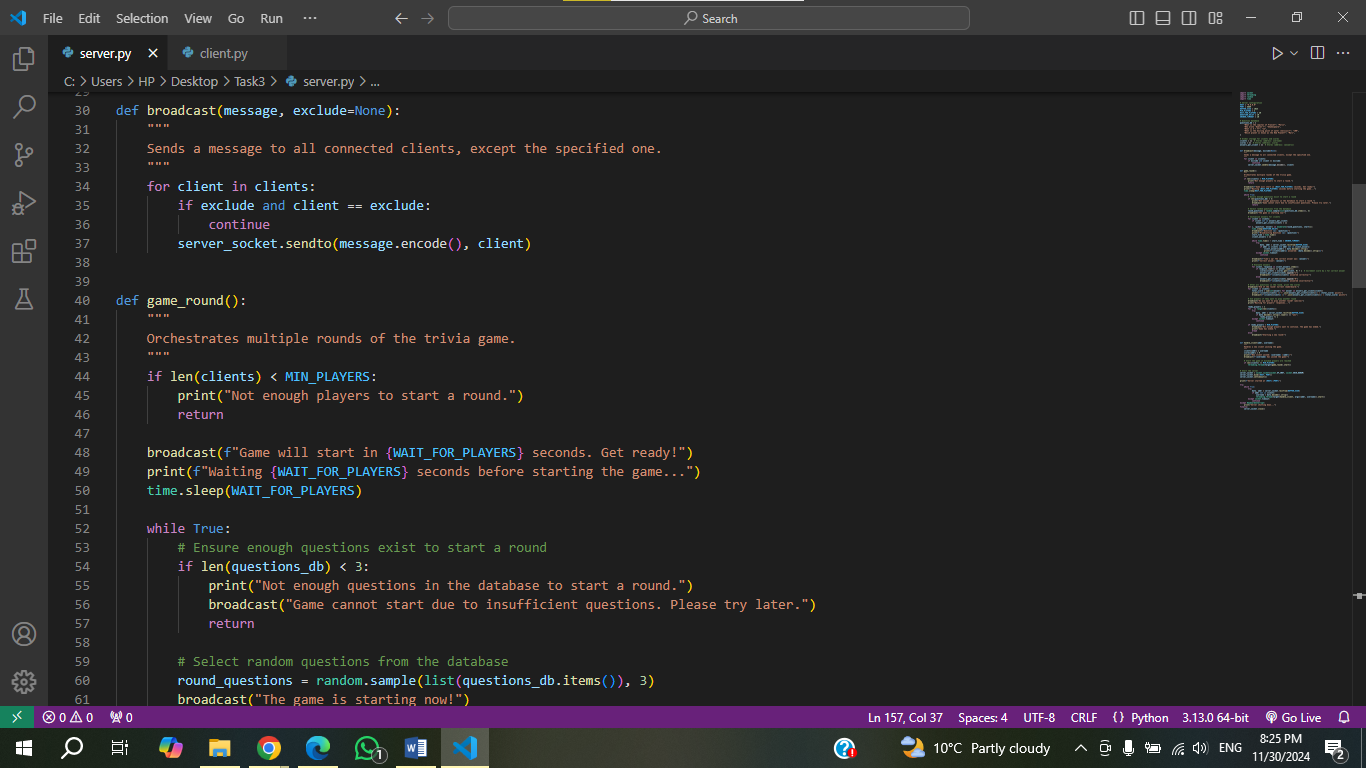


Figure : server code

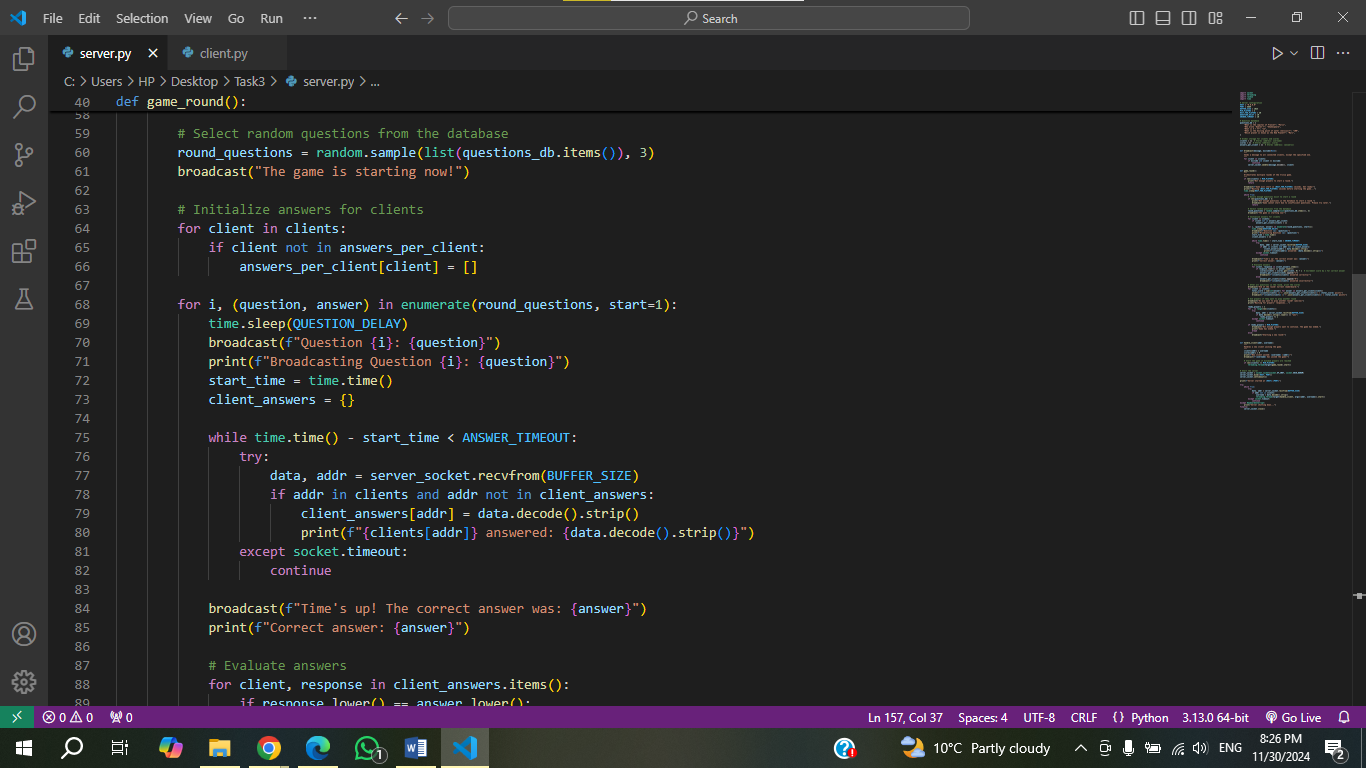


Figure : server code

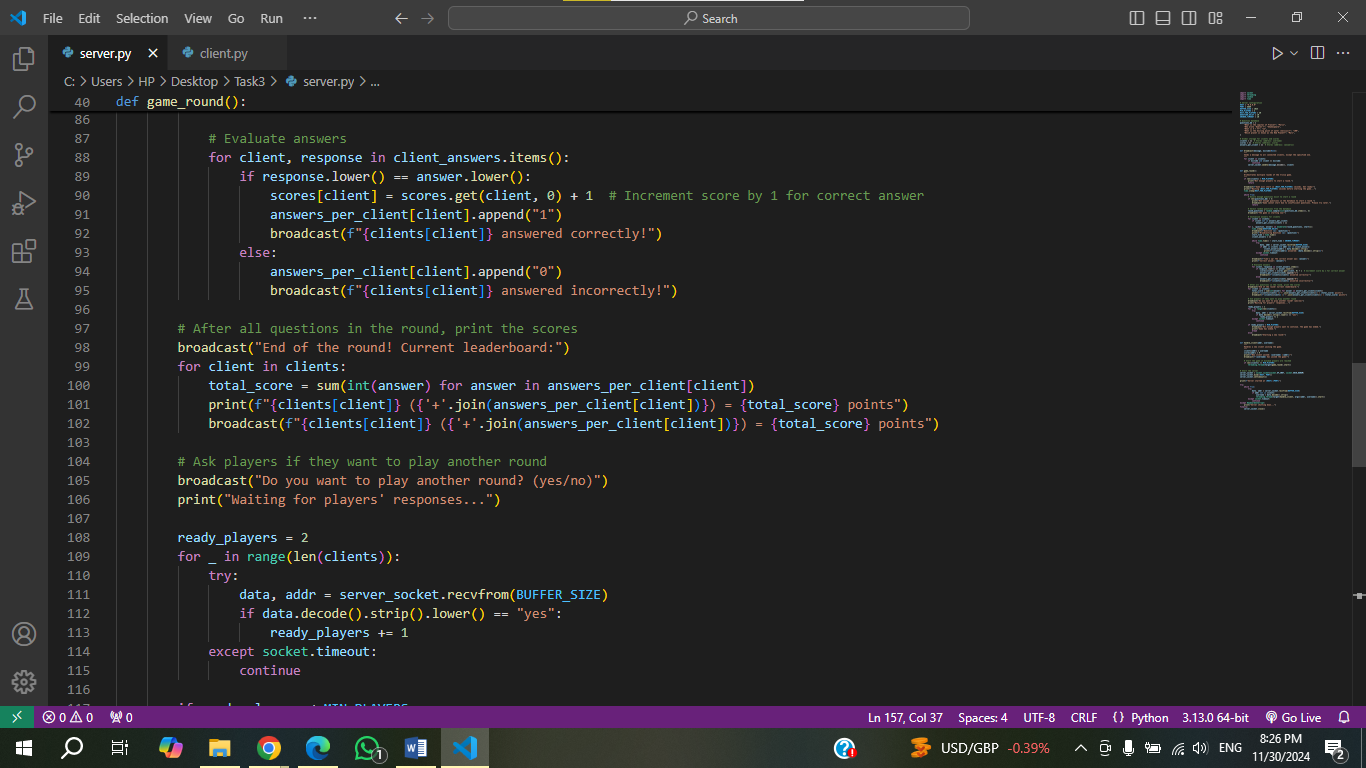


Figure : server code

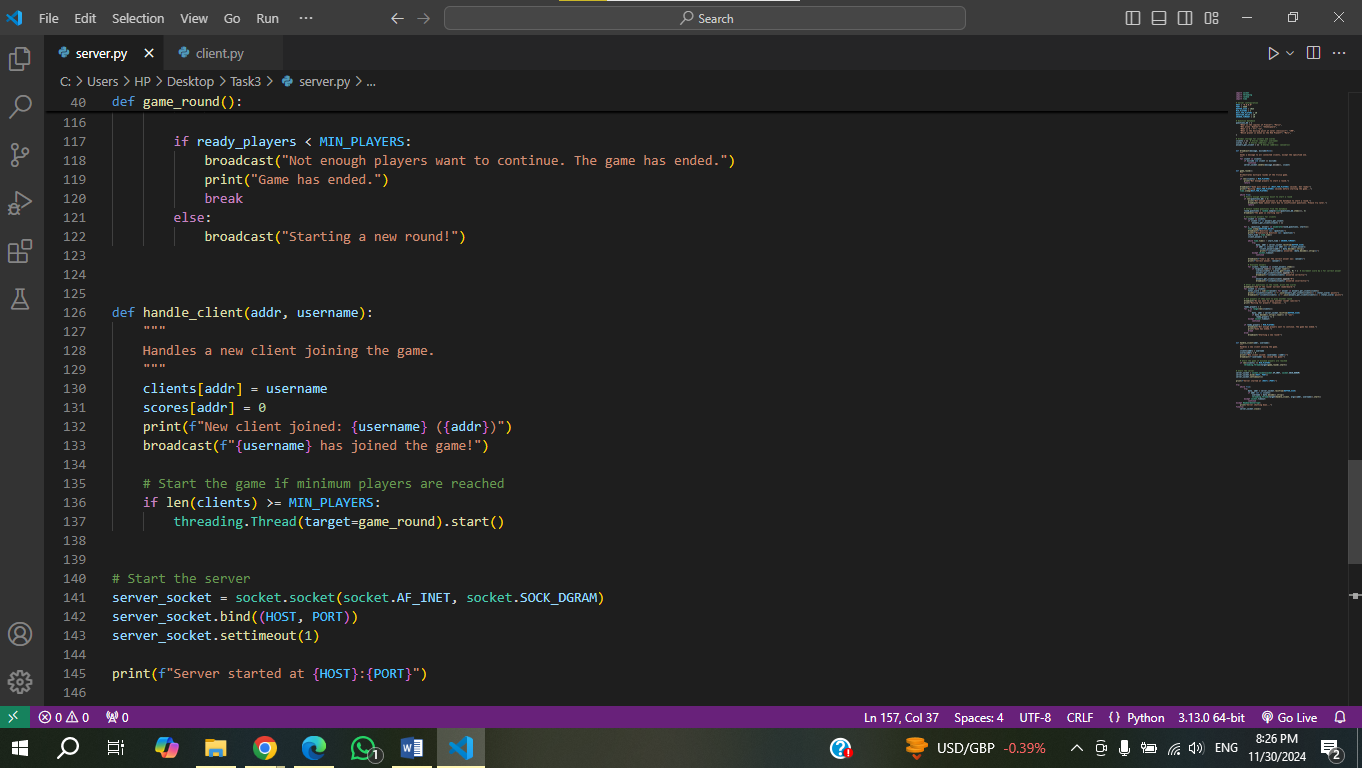


Figure : server code

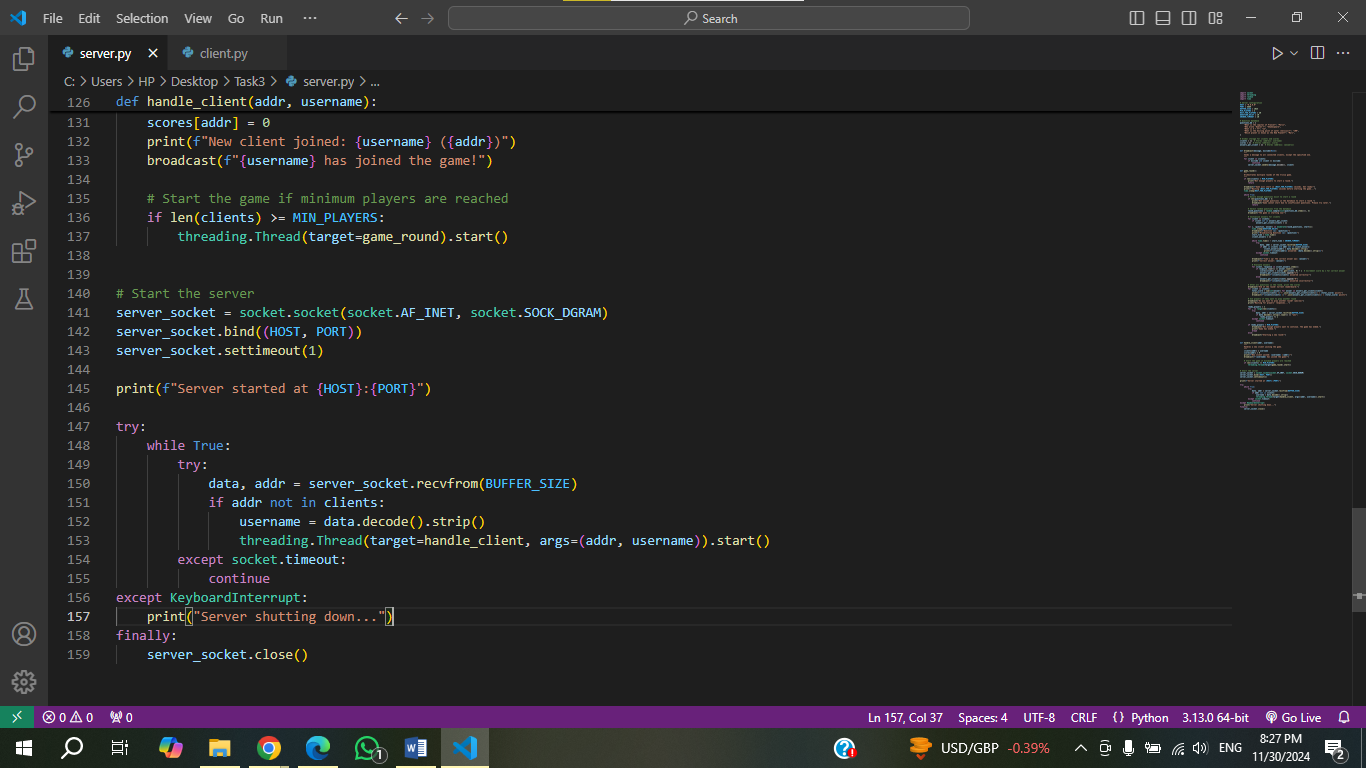


Figure : server code

### **Server code discussion:**

This Python code creates a multiplayer trivia game server using UDP. The server listens for players to join the game by connecting on port 5689, where each player is asked to provide a username. Once at least two players are connected, the server kicks off the game by randomly selecting three trivia questions from a pre-made list and sending them to all players. Each player has a set amount of time to answer, and the server collects their responses, checks if they’re correct, and awards points. After each round, the server shows the current scores and asks if the players want to continue with another round. If not, enough players want to keep going, the game ends. The server uses threading to manage multiple players at once, ensuring the game runs smoothly for everyone.

### **The client code:**

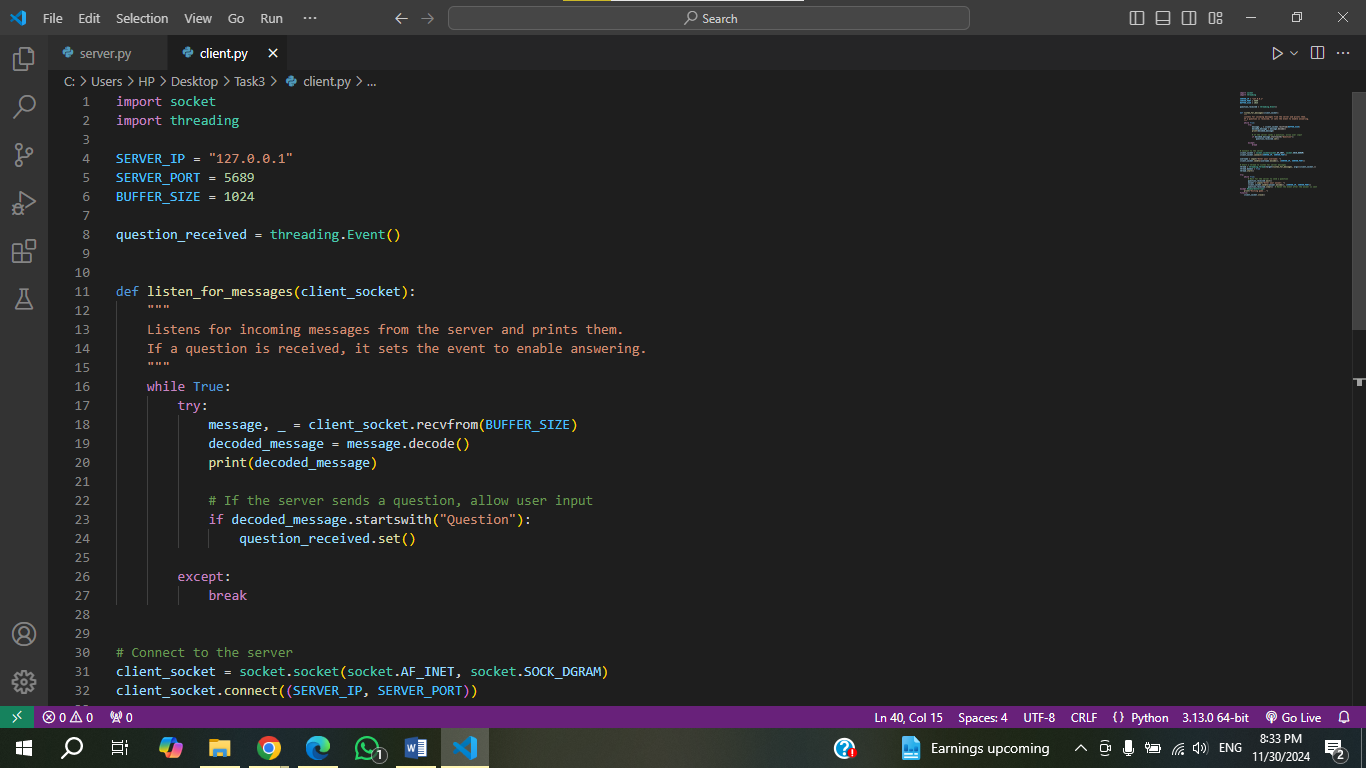


Figure : the client code

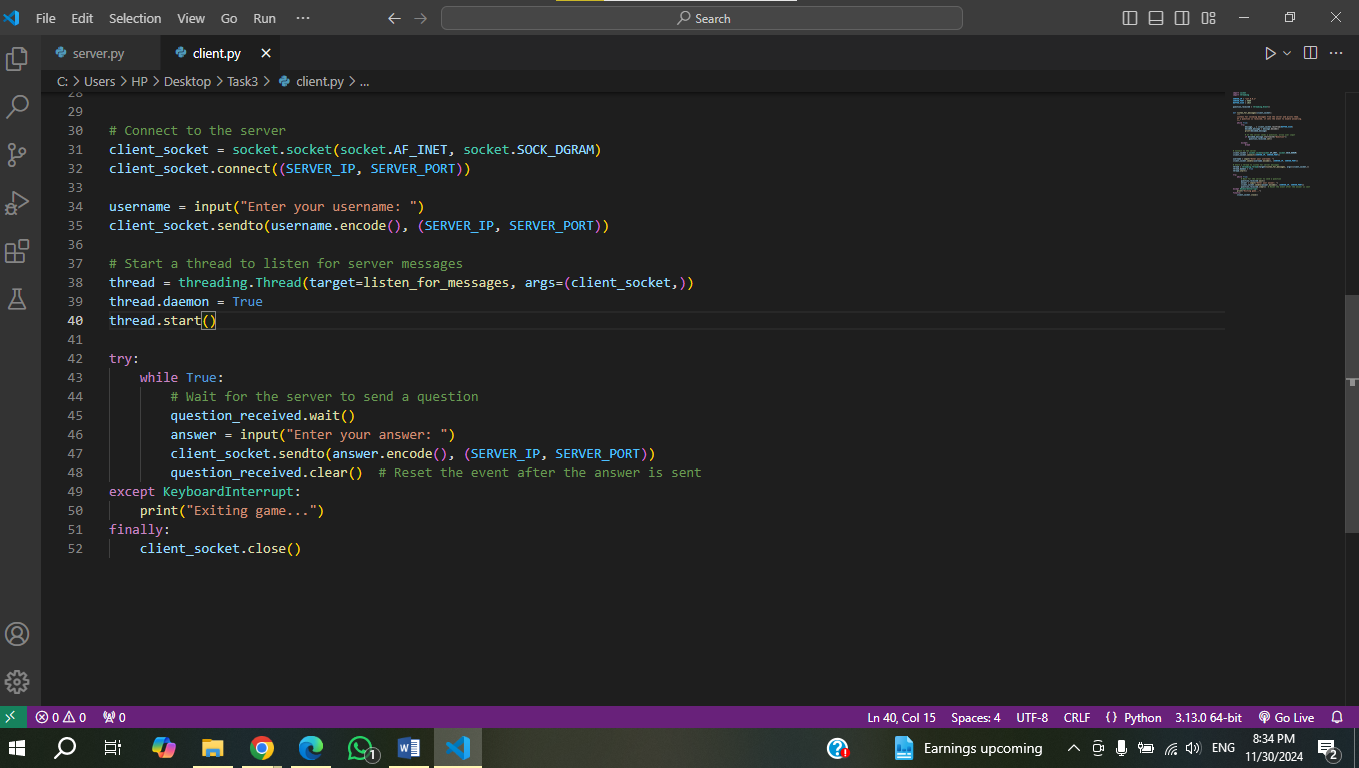


Figure : the client code

### **Client code discussion:**

This Python code is for the client side of a trivia game, where the client connects to the server using UDP. After the client connects, it asks the user to enter a username, which is then sent to the server. A separate thread listens for messages from the server and prints them out. When the server sends a trivia question, it lets the user know that they can now answer by triggering an event. The client then waits for the question, prompts the user to type their answer, and sends the response back to the server. After each answer, the event is reset so the user can only answer one question at a time. The game continues until the user decides to exit.

## **Results and Discussion:**

### **The first example:**

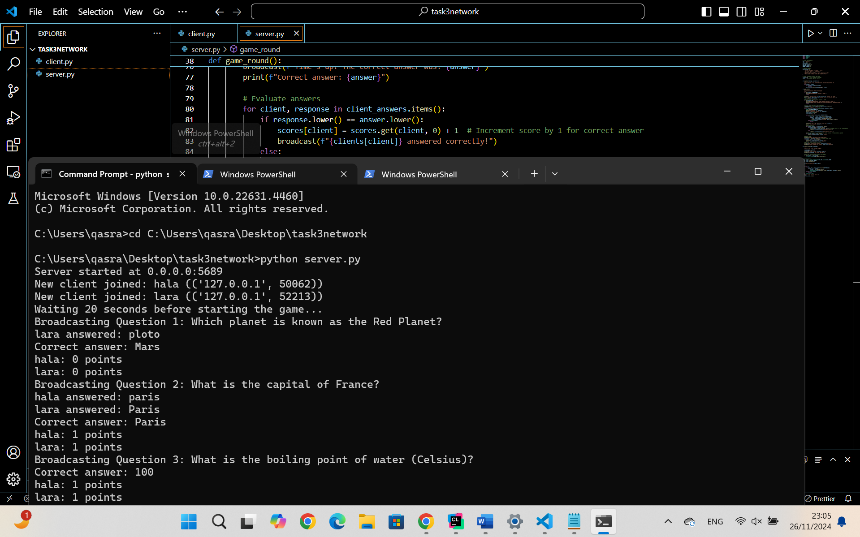


Figure : the output of server Ex1

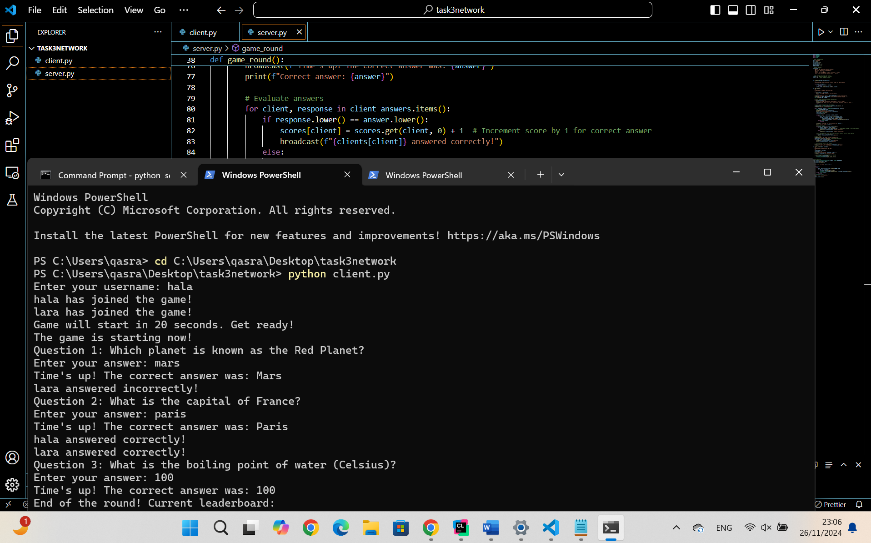


Figure : the output of client1 Ex1

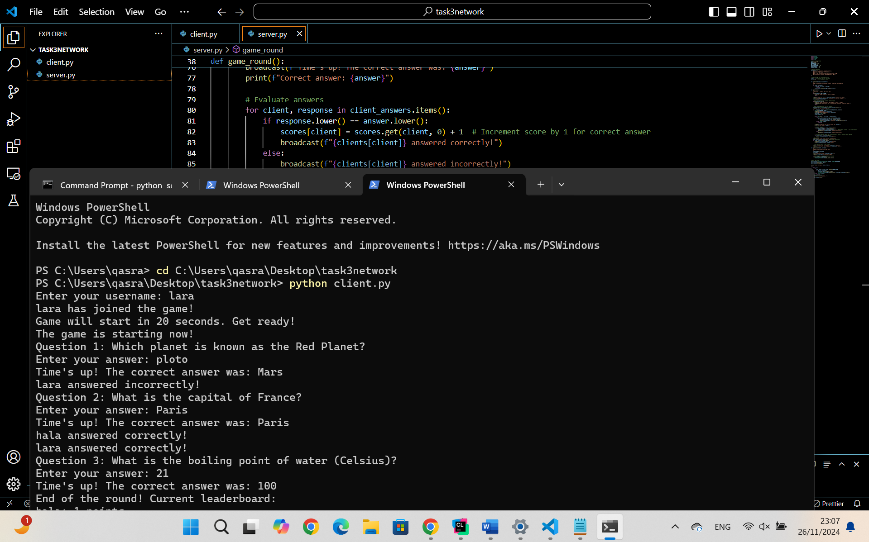


Figure : the output of client2 EX1

### **The second example:**

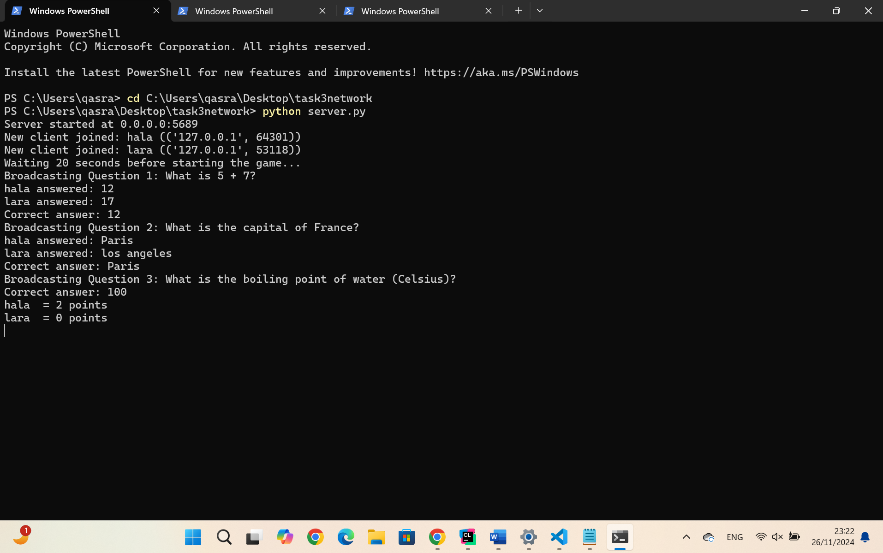


Figure : output of server EX2

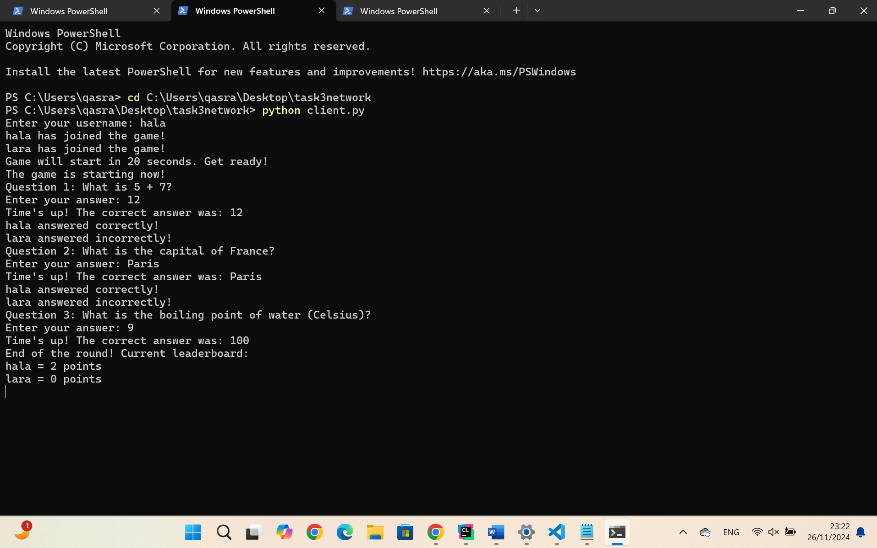


Figure : output of client1 EX2

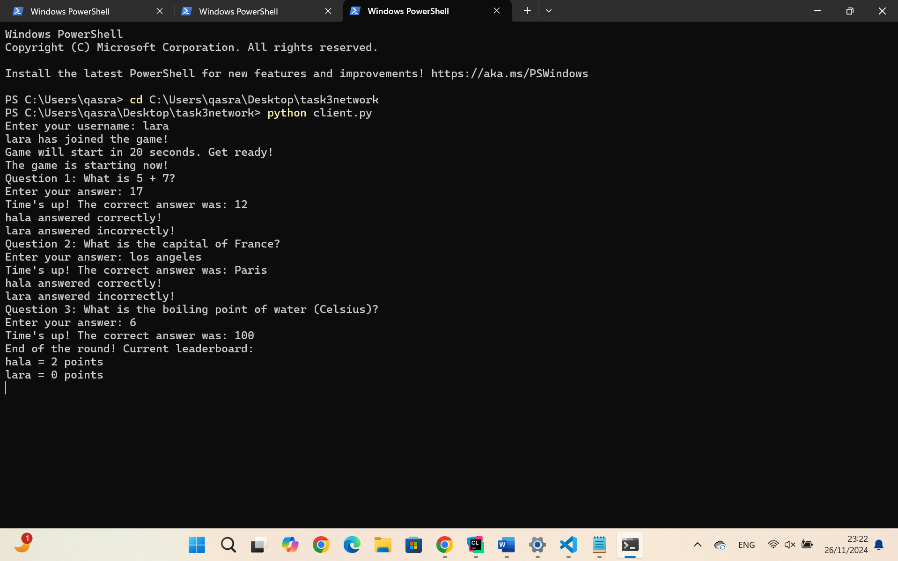


Figure : output of client2 EX2

### **More than one round example:**

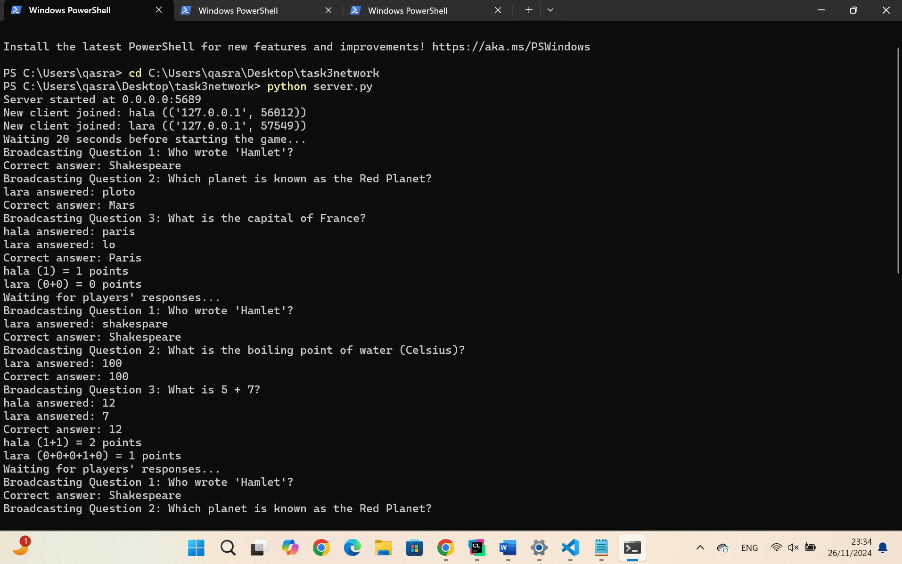


Figure : output of server EX3

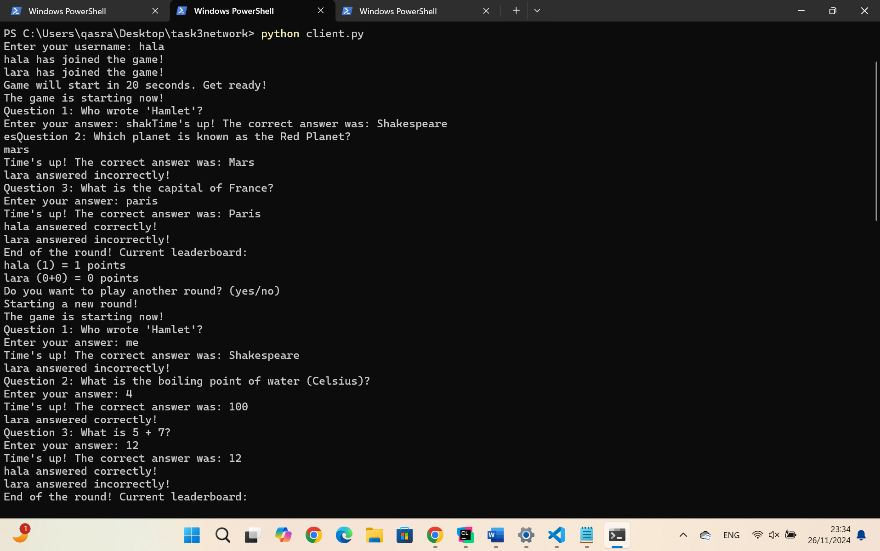


Figure 16: output of client1 EX3

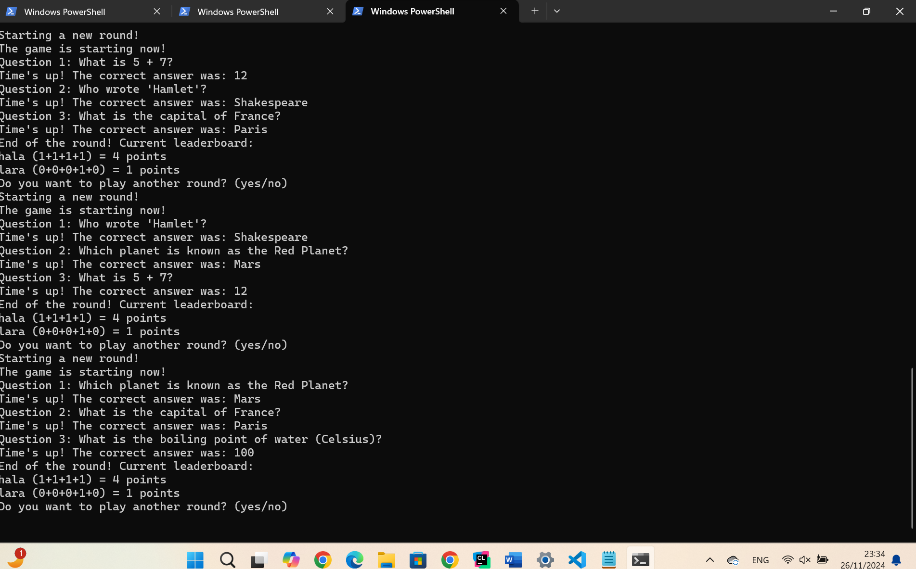


Figure :output of client2 EX3

### **Discussion:**

When the server starts, it will let everyone know that it’s up and running, waiting for players on port 5689. As players join, it will show their username and IP address. Once enough players are in, the server will announce that the game is starting and send out the first question. It will print the question for everyone to see, and as players respond, the server will display each answer, along with whether it was right or wrong. At the end of each round, the server will share the scores, showing how each player did. If players want to continue, it will ask if they’re ready for another round, but if not, enough players are interested, the game will end. When the server shuts down, it will let everyone know it’s closing. On the client side, when a player connects, they’ll see a message confirming they’re connected to the server. As the game progresses, the client will display messages about the game starting, the questions being asked, and updates to the leaderboard. The client will prompt the player for an answer when a question arrives, and once the answer is submitted, it will confirm that the answer was sent. At the end of the round, the client will show the updated scores and any disconnection messages if the game is over or the server shuts down.

# **Conclusion**

In conclusion, UDP makes real-time trivia games fast and exciting by prioritizing speed over perfect reliability. With the server hosting questions and tracking scores, and players submitting answers in real-time, the game stays interactive and engaging. This setup shows how clever use of technology can bring people together for a fun and competitive multiplayer experience.