

**INSTITUTO TECNOLOGICO DE CHILPANCINGO**

**Cómputo En La Nube Y Grandes Datos**

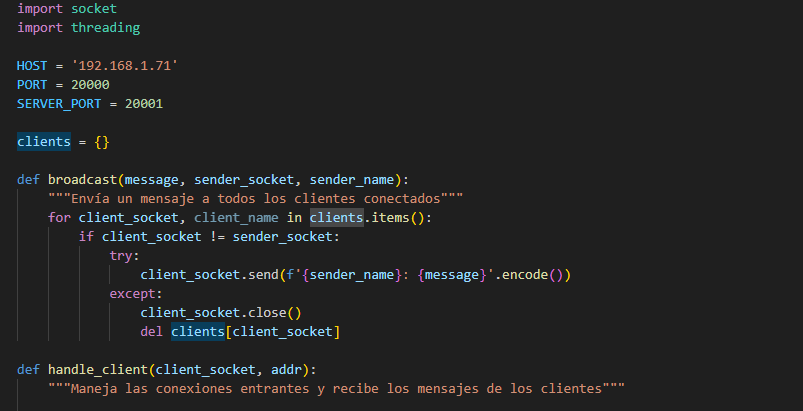
Practica Sockets TCP y UDP Cooperativo

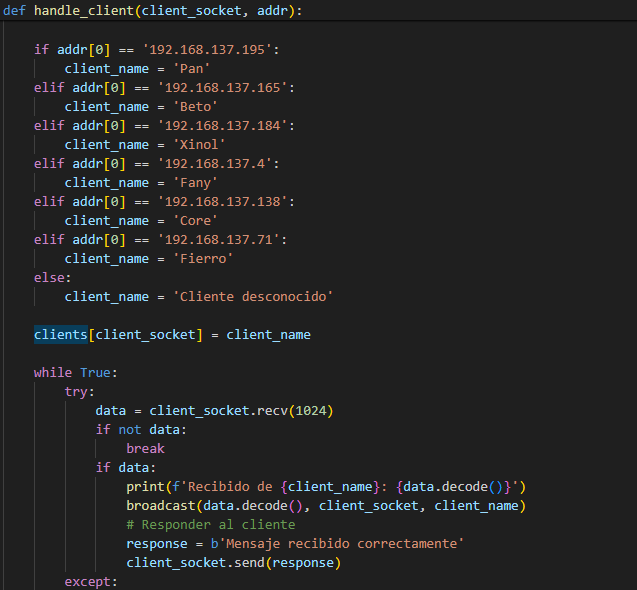
**Alumno:**

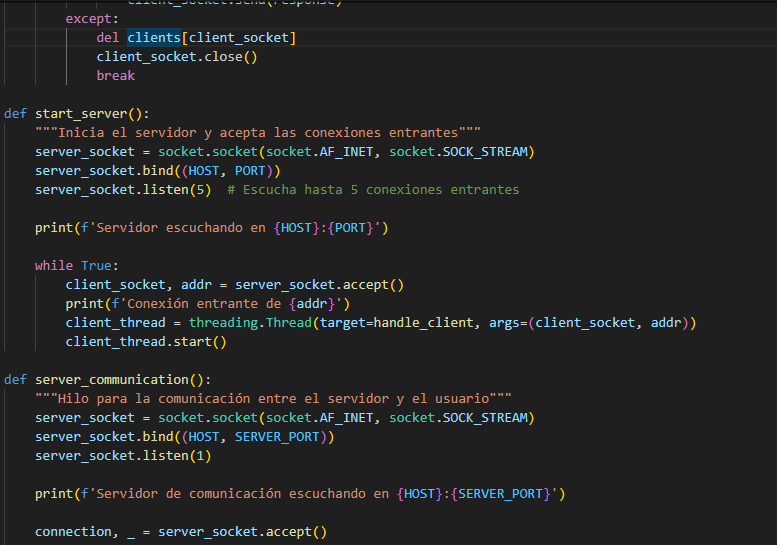
Fierro Monroy Isaac Antonio

Lunes 13 de mayo de 2024

# Sockets TCP





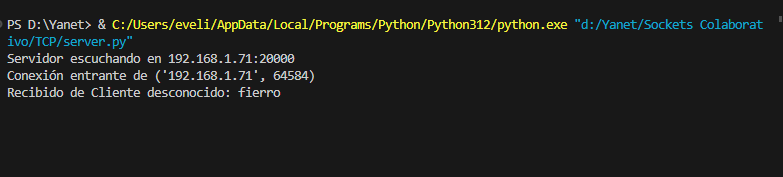


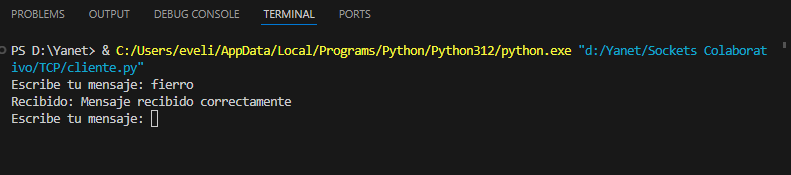


Este código crea un servidor de chat que puede manejar múltiples clientes y permite la comunicación bidireccional entre el servidor y los usuarios conectados. Cada cliente puede enviar mensajes al servidor, que luego se retransmiten a todos los demás clientes. Además, el servidor puede enviar mensajes a todos los clientes.

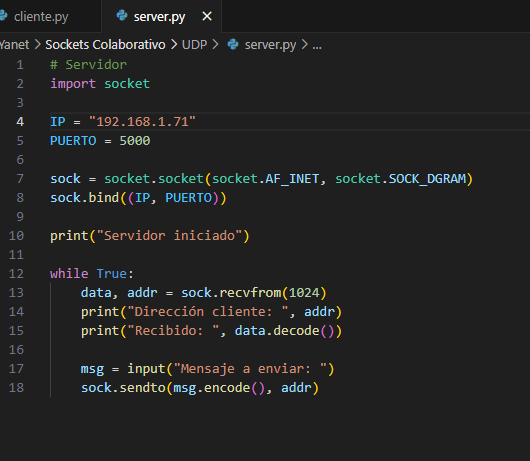


Este cliente básico permite a un usuario enviar mensajes al servidor y recibir respuestas del mismo. Puedes ejecutar múltiples instancias de este cliente para simular múltiples usuarios comunicándose con el servidor.

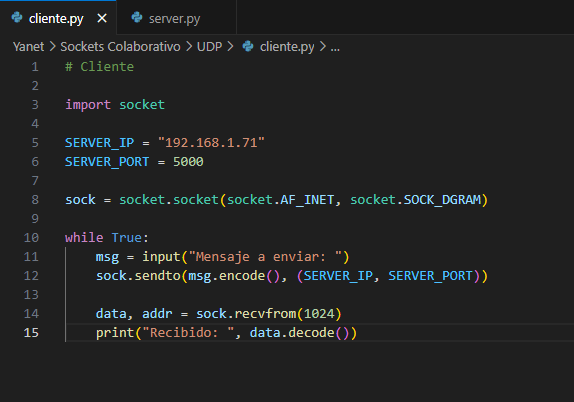




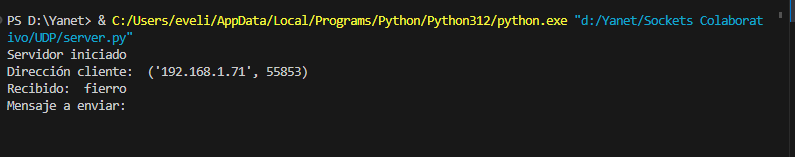
# Sockets UDP

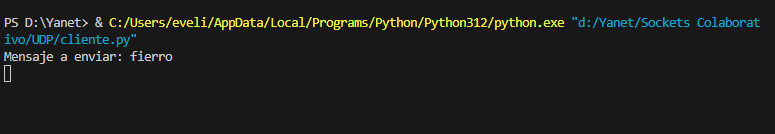


Este servidor UDP escucha continuamente mensajes entrantes de los clientes, imprime los mensajes recibidos y envía una respuesta al cliente que envió el mensaje original. Es importante tener en cuenta que UDP es un protocolo sin conexión, lo que significa que no hay una conexión persistente entre el servidor y el cliente, y cada paquete enviado es independiente.



Este cliente UDP envía mensajes al servidor y espera recibir una respuesta. Una vez que recibe una respuesta del servidor, imprime el mensaje recibido en la consola. Este proceso se repite continuamente mientras el cliente esté en ejecución.





# CONCLUSION

Los fragmentos de código presentados son ejemplos simples de comunicación utilizando sockets en Python, específicamente implementando la comunicación mediante el protocolo TCP/IP.

* Servidor TCP/IP: Este fragmento de código crea un servidor que espera conexiones entrantes de clientes. Una vez que se establece una conexión, el servidor puede recibir mensajes del cliente, procesarlos y enviar respuestas de vuelta al cliente.
* Cliente TCP/IP: Por otro lado, el fragmento de código del cliente se conecta al servidor y permite al usuario enviar mensajes al servidor. Después de enviar un mensaje, el cliente espera recibir una respuesta del servidor, que luego muestra en la consola.

Ambos fragmentos de código están diseñados para operar de forma continua, lo que significa que el servidor permanece a la espera de conexiones y el cliente permanece interactivo, permitiendo al usuario enviar mensajes al servidor y recibir respuestas.