

**INSTITUTO TECNOLOGICO DE CHILPANCINGO**

**Cómputo En La Nube Y Grandes Datos**

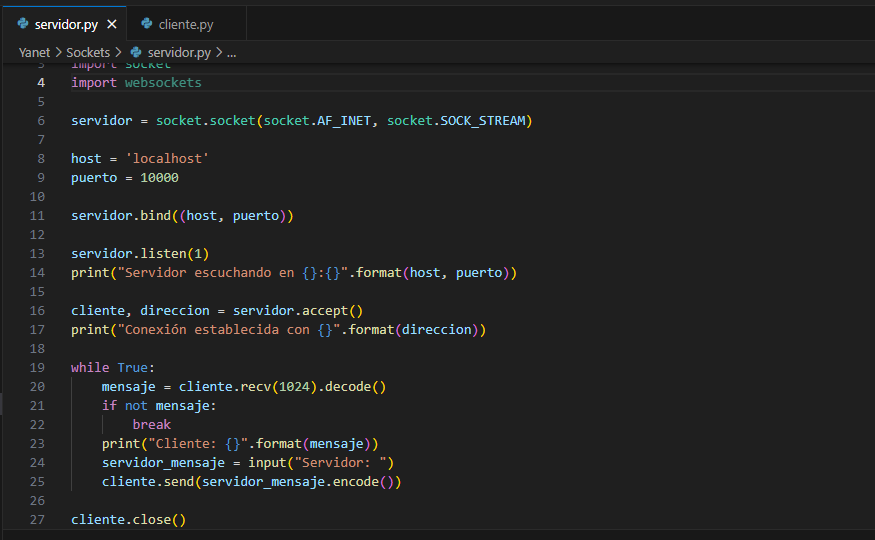
Practica Sockets TCP y UDP Local

**Alumno:**

Fierro Monroy Isaac Antonio

Lunes 13 de mayo de 2024

# Sockets TCP



Crea un socket TCP utilizando el módulo socket.

Enlaza el socket al host y puerto especificados.

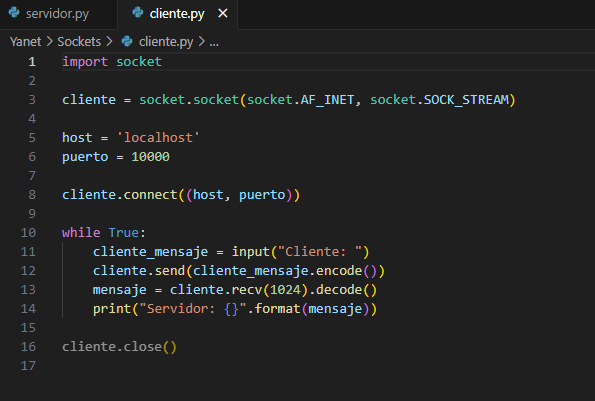
Escucha conexiones entrantes utilizando el método listen().

Acepta una conexión entrante utilizando el método accept().

Entra en un bucle infinito donde espera recibir mensajes del cliente.

Recibe mensajes del cliente, los imprime y envía mensajes de vuelta al cliente.

Cierra la conexión con el cliente al salir del bucle.



Crea un socket TCP utilizando el módulo socket.

Se conecta al servidor especificado utilizando el método connect().

Entra en un bucle infinito donde espera la entrada del usuario para un mensaje.

Envía los mensajes ingresados por el usuario al servidor y espera recibir respuestas.

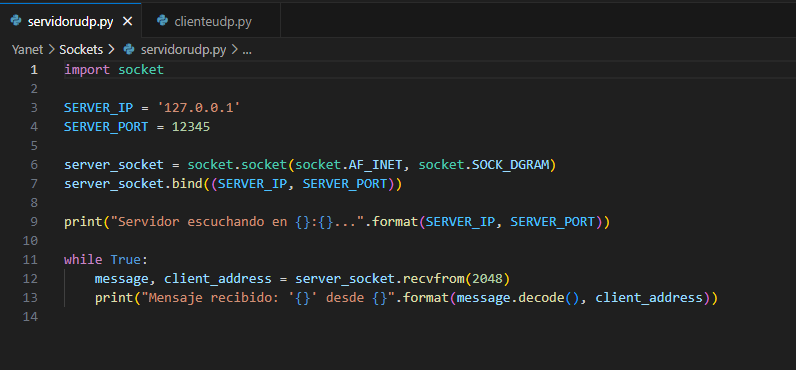
Imprime las respuestas del servidor en la consola.

Cierra la conexión con el servidor al salir del bucle.





# Sockets UDP

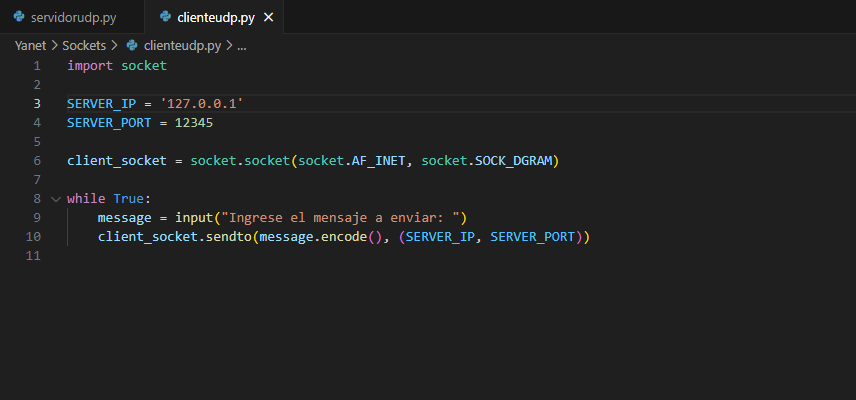


El servidor crea un socket usando socket.socket() con la familia de direcciones socket.AF\_INET y el tipo de socket socket.SOCK\_DGRAM, que es para comunicación de datagramas UDP.

Luego, el servidor enlaza el socket al SERVER\_IP y al SERVER\_PORT especificados.

Entra en un bucle infinito donde espera recibir mensajes de los clientes.

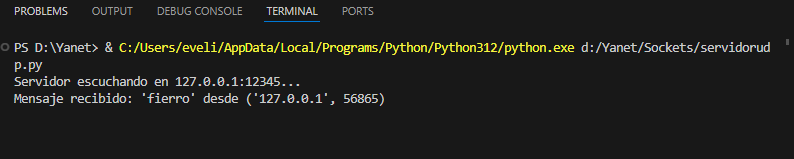
Cuando recibe un mensaje, lo decodifica y muestra el mensaje junto con la dirección del cliente que envió el mensaje.

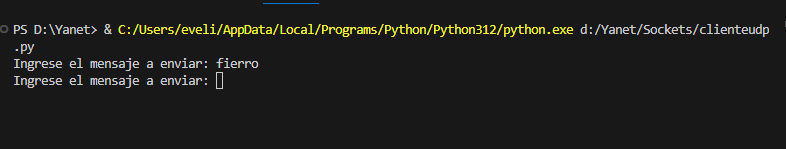


El cliente también crea un socket UDP similar al servidor.

Entra en un bucle infinito donde espera la entrada del usuario para un mensaje.

Cuando el usuario ingresa un mensaje, lo codifica y lo envía al servidor en la dirección SERVER\_IP y el puerto SERVER\_PORT.





# 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.