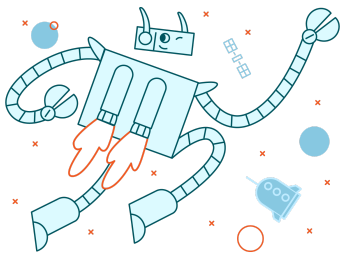
UDP Server 📞



Goal

Develop a Python UDP server that listens for incoming datagrams and responds to clients with the message "Hello".

Background

UDP (User Datagram Protocol) is a lightweight, connectionless protocol used for sending messages (datagrams) over a network. Unlike TCP, UDP does not establish a persistent connection between the client and server. In this exercise, you'll create a UDP server that listens for incoming datagrams and responds to clients with a "Hello" message.

Tasks

1. Write a UDP server that does the following:
    - Creates a UDP socket using `socket.socket()` with `socket.SOCK_DGRAM` as the socket type.
    - Binds the socket to all interfaces ('0.0.0.0') and port 5000 using `socket.bind()`.
    - Enters a loop to continuously listen for incoming datagrams using `socket.recvfrom()`.
    - When a datagram is received, the server extracts the client's address (IP and port) and the message from the received data.
    - Sends the message "Hello " + thte client's message (echo) back to the client using `socket.sendto()`, specifying the client's address.
    - Continues to listen for incoming datagrams until the server is manually closed (e.g., by pressing Ctrl+C).

2. Run the UDP server from the command line with the command `python udp_hello_server.py`.

3. Test the behavior of the server by running the UDP client from the previous exercise and sending messages to the server. The server should respond with "Hello" for each message received.

Example

Here's an example of how the UDP server should work:

```
$ python udp_hello_server.py
Server is listening on localhost:5000
Received message from ('127.0.0.1', 60000): Hello, server!
Sent response to ('127.0.0.1', 60000): Hello Hello, server!
```

To Submit

- The `udp_server.py` file containing the complete code for the UDP server.

Make sure to test your UDP server with the UDP client from the previous exercise to ensure it functions as expected.

Note: UDP is a connectionless protocol, so you don't need to establish a persistent connection like in TCP. The server simply listens for incoming datagrams, processes them, and sends a response back to the client using the client's address obtained from the received datagram.