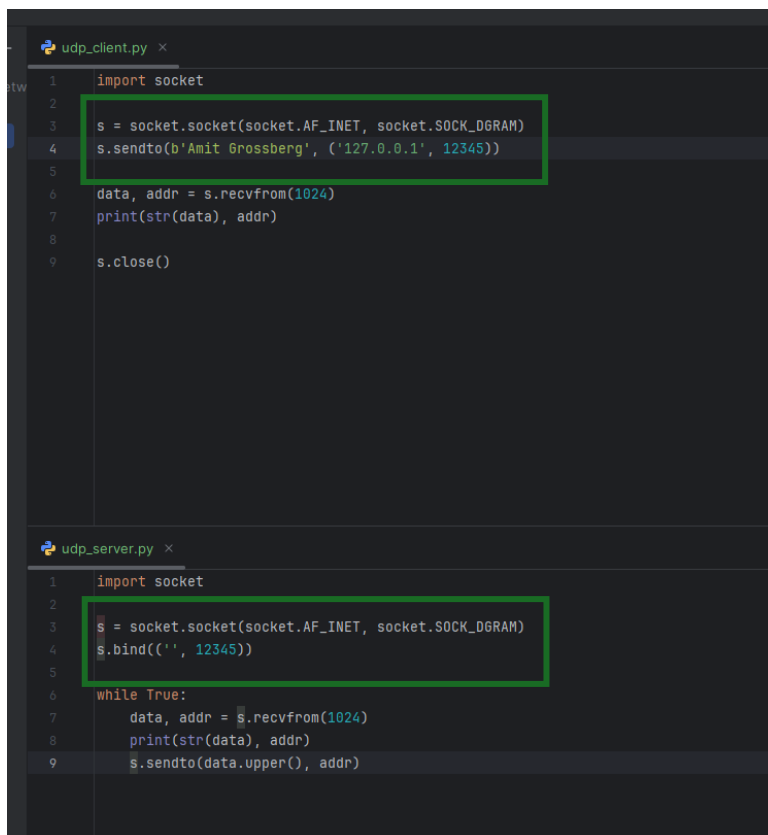


# Networks Assignment 1 – Part 1

## 1. Code Explanation

**udp\_server.py (Server)** – The server imports the socket library to create an IPv4 UDP socket using SOCK\_DGRAM and AF\_INET. It binds to all available addresses (") on port 12345. The server then enters an infinite loop where it receives data via s.recvfrom, prints the data and the sender's address, converts the received string to uppercase, and sends it back to the sender.

**udp\_client.py (Client)** – The client imports the socket library and creates a socket like the server did. It uses s.sendto to transmit the string "Amit Grossberg" to the Loopback IP 127.0.0.1 (localhost) at destination port 12345. The script waits for a response (s.recvfrom), prints the received data and the server address, and then closes the socket (s.close()).



```
udp_client.py x
1 import socket
2
3 s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
4 s.sendto(b'Amit Grossberg', ('127.0.0.1', 12345))
5
6 data, addr = s.recvfrom(1024)
7 print(str(data), addr)
8
9 s.close()

udp_server.py x
1 import socket
2
3 s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
4 s.bind(('', 12345))
5
6 while True:
7     data, addr = s.recvfrom(1024)
8     print(str(data), addr)
9     s.sendto(data.upper(), addr)
```

## 2. Traffic Analysis

In Wireshark, we observe the capture of the two packets.

Packet 1: Client -> Server

This is the first packet sent, generated directly by the command `s.sendto(b'Amit Grossberg', ('127.0.0.1', 12345))` in the client code.

The image shows a Wireshark packet capture of a UDP packet on a loopback interface. The packet is from 127.0.0.1 to 127.0.0.1, port 53002 to 12345. The Ethernet II header shows source and destination MAC as 00:00:00:00:00:00. The IP header shows source and destination as 127.0.0.1. The UDP header shows source port 53002 and destination port 12345. The payload is 14 bytes of data.

**Layer 2 (Ethernet):** Source/Destination MAC addresses are 00:00:00:00:00:00. This is because of Loopback traffic (sending data to 127.0.0.1), as the packet never leaves the physical network interface, making real MAC addresses unnecessary.

**Layer 3 (Internet Protocol - IP):**

- Source IP: 127.0.0.1

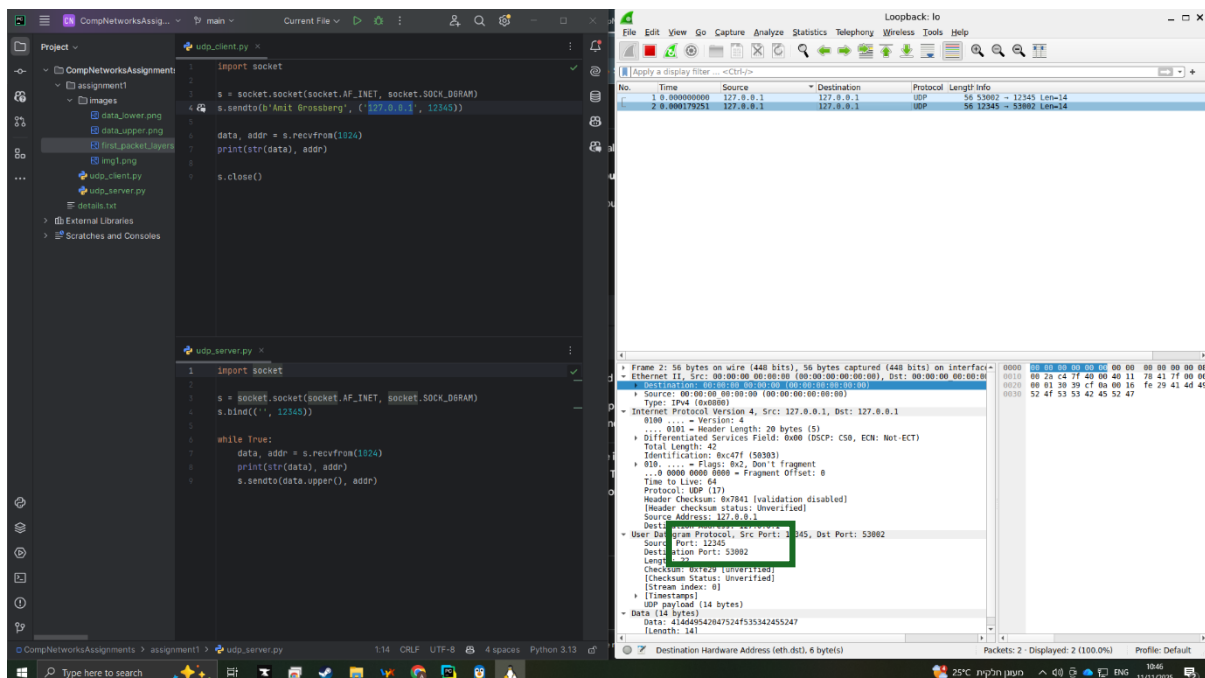
## Layer Analysis:

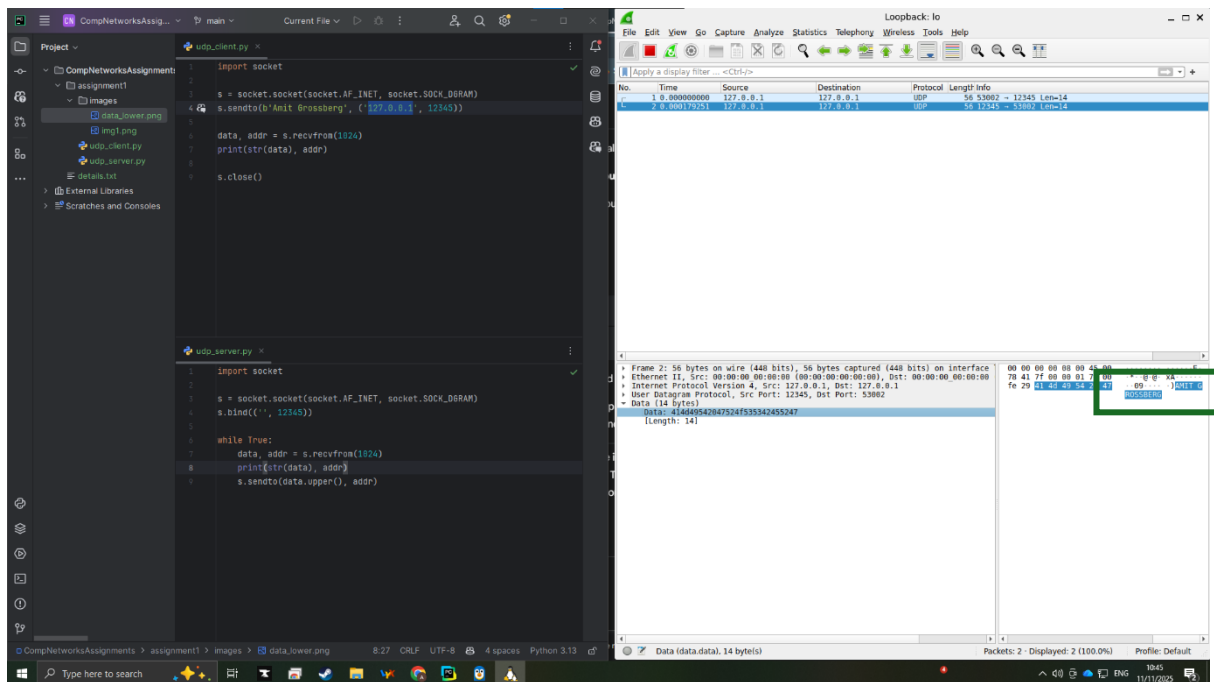
- Layer 2 (Ethernet): Source/Destination MAC addresses are 00:00:00:00:00:00. This is because of Loopback traffic (sending data to 127.0.0.1), as the packet never leaves the physical network interface, making real MAC addresses unnecessary.
- Layer 3 (Internet Protocol - IP):
  - Source IP: 127.0.0.1

- Destination IP: 127.0.0.1 (defined in the client's sendto command).
- Layer 4 (User Datagram Protocol - UDP):
  - Source Port: 56300, This port was given randomly by the client's OS to receive the response.
  - Destination Port: 12345. Defined in the client's sendto command, matching the port the server is bound to.
- Layer 7 (Data):
  - Payload: Amit Grossberg. This matches exactly the string sent by the client.

## Packet 2: Server -> Client

This packet is generated by the server code in response to the first packet, specifically by the command `s.sendto(data.upper(), addr)`.





### Layer Analysis:

- Layer 2 (Ethernet): Source/Destination MAC addresses remain 00:00:00:00:00:00 for the same Loopback reasons.
- Layer 3 (Internet Protocol - IP):
  - Source IP: 127.0.0.1.
  - Destination IP: 127.0.0.1 (the client). The server learned this address from the first packet via the addr variable.
- Layer 4 (User Datagram Protocol - UDP):
  - Source Port: 12345.
  - Destination Port: 56300, The server extracted this port from the addr variable in the incoming packet.
- Layer 7 (Data):
  - Payload: AMIT GROSSBERG. This shows the required logic: the server received "Amit Grossberg", executed data.upper(), and transmitted the uppercase result.