**🔹 Ejercicio 1**

**Servidor (udp\_servidor1.py)**

* s = socket.socket(...) → crea socket UDP.
* s.bind(("", puerto)) → lo asocia a un puerto.
* while True: ... s.recvfrom(1024) → recibe datagramas en bucle.
* print(...) → muestra mensaje recibido y dirección.

👉 **Resumen**: Servidor básico que escucha y muestra los mensajes.

**Cliente (udp\_cliente1.py)**

* input(...) → pide texto al usuario.
* s.sendto(...) → envía cada línea al servidor.
* Se corta si el usuario escribe FIN.

👉 **Resumen**: Cliente que envía texto al servidor.

**🔹 Ejercicio 2**

**Servidor con pérdidas (udp\_servidor2\_simula\_perdidas.py)**

* random.randint(0,1) → decide 50% perder el paquete.
* Si se pierde → print("Simulando paquete perdido").
* Si no → muestra mensaje normalmente.

👉 **Resumen**: Servidor que a veces “ignora” mensajes para simular pérdidas.

**Cliente con numeración (udp\_cliente2\_numera\_mensajes.py)**

* contador = 1 → inicia numeración.
* mensaje = f"{contador}: {linea}" → añade número al inicio.
* contador += 1 → incrementa.

👉 **Resumen**: Cliente que numera los mensajes enviados.

**🔹 Ejercicio 3**

**Servidor con OK (udp\_servidor3\_con\_ok.py)**

* Igual al servidor con pérdidas.
* Si procesa mensaje → s.sendto("OK".encode(...), direccion).

👉 **Resumen**: Servidor que responde con “OK” cuando recibe algo.

**Cliente con espera de OK (udp\_cliente3\_espera\_ok.py)**

* s.settimeout(0.5) → espera máx. 0.5 segundos.
* try ... recvfrom(...) → intenta recibir “OK”.
* except socket.timeout: → si no llega, muestra error.

👉 **Resumen**: Cliente que espera confirmación y detecta si no llega.

**🔹 Ejercicio 4**

**Cliente con reintentos (udp\_cliente4\_reintenta.py)**

* Igual al anterior, pero con reenvíos.
* timeout = 0.2 → empieza con 200 ms.
* Si no llega OK → timeout \*= 2.
* Si timeout > 2.0 → asume que el servidor cayó y termina.

👉 **Resumen**: Cliente que reintenta enviar el mismo mensaje hasta recibir OK o superar 2s.

**🔹 Ejercicio 5 (opcional, mejorado)**

**Servidor mejorado (udp\_servidor5\_mejorado.py)**

* id\_str, contenido = mensaje.split("|", 1) → separa ID y mensaje.
* if id in procesados: → detecta duplicados.
* Si es duplicado → reenvía “OK id” pero **no repite acción**.
* Si es nuevo → lo procesa y guarda el ID.

👉 **Resumen**: Servidor que evita ejecutar duplicados, confirma con el ID del mensaje.

**Cliente mejorado (udp\_cliente5\_mejorado.py)**

* id\_msg = random.randint(1000,9999) → genera ID único.
* mensaje = f"{id\_msg}|{linea}" → envía ID + texto.
* Espera exactamente "OK {id\_msg}".
* Si no llega, reintenta con timeout creciente.

👉 **Resumen**: Cliente que envía mensajes identificados y solo acepta la confirmación correcta.

⚡ En resumen:

* **Ej1**: Comunicación básica.
* **Ej2**: Se simulan pérdidas y se numeran mensajes.
* **Ej3**: Confirmación con “OK” y timeout.
* **Ej4**: Reintentos automáticos con timeout creciente.
* **Ej5**: Robustez añadida → IDs únicos, confirmación exacta, y servidor que evita duplicados.