

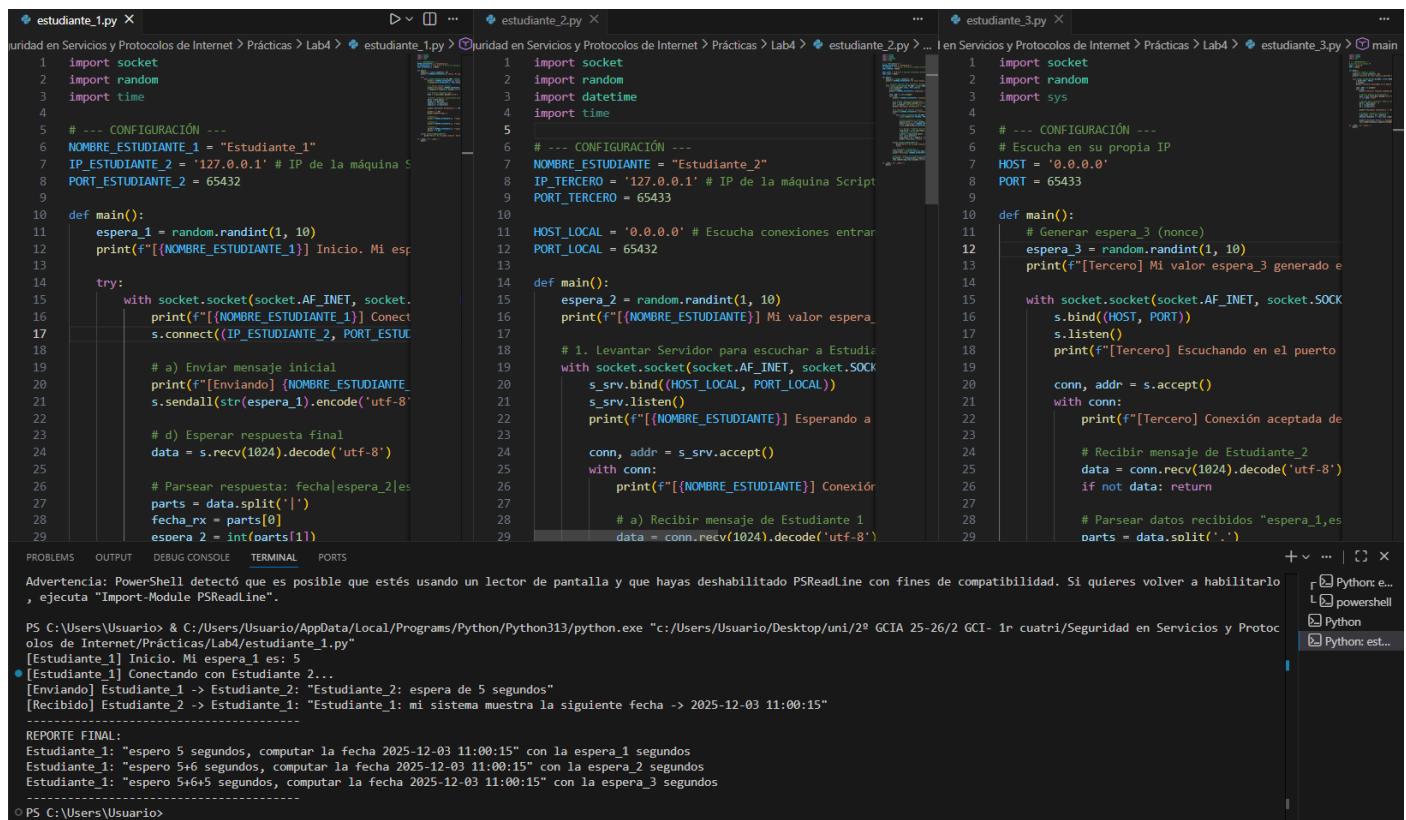
PRÁCTICA #4

DATOS DEL ESTUDIANTE

Nombre y apellidos	María Victoria Maldonado Bao
E-mail	mvictoriamb0425@uma.es
Grupo (A / B)	A
Fecha	3/12/2025

EJERCICIO 1. COMUNICACIÓN CLIENTE-SERVIDOR TCP SIN TLS

He hecho 3 scripts, uno para cada estudiante, y este es el resultado:



```
estudiante_1.py
import socket
import random
import time

# --- CONFIGURACIÓN ---
NOMBRE_ESTUDIANTE_1 = "Estudiante_1"
IP_ESTUDIANTE_2 = '127.0.0.1' # IP de la máquina Script
PORT_ESTUDIANTE_2 = 65432

def main():
    espera_1 = random.randint(1, 10)
    print(f"[{NOMBRE_ESTUDIANTE_1}] Inicio. Mi espero {espera_1} segundos")
    try:
        with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
            print(f"[{NOMBRE_ESTUDIANTE_1}] Conectando con Estudiante_2...")
            s.connect((IP_ESTUDIANTE_2, PORT_ESTUDIANTE_2))

            # a) Enviar mensaje inicial
            print(f"[Enviando] {NOMBRE_ESTUDIANTE_1}")
            s.sendall(str(espera_1).encode('utf-8'))

            # d) Esperar respuesta final
            data = s.recv(1024).decode('utf-8')
            parts = data.split('|')
            fecha_rx = parts[0]
            espera_2 = int(parts[1])
            print(f"[{NOMBRE_ESTUDIANTE_1}] Recibido {fecha_rx} y {espera_2} segundos")

            # Parsear respuesta: fecha|espera_2|es
            parts = data.split('|')
            fecha_rx = parts[0]
            espera_2 = int(parts[1])
            print(f"[{NOMBRE_ESTUDIANTE_1}] Recibido {fecha_rx} y {espera_2} segundos")
    except Exception as e:
        print(f"[{NOMBRE_ESTUDIANTE_1}] Error: {e}")

print(f"[{NOMBRE_ESTUDIANTE_1}] Finalizado")
```

```
estudiante_2.py
import socket
import random
import datetime
import time

# --- CONFIGURACIÓN ---
NOMBRE_ESTUDIANTE_2 = "Estudiante_2"
IP_TERCERO = '127.0.0.1' # IP de la máquina Script
PORT_TERCERO = 65433

HOST_LOCAL = '0.0.0.0' # Escucha conexiones entrantes
PORT_LOCAL = 65432

def main():
    espera_2 = random.randint(1, 10)
    print(f"[{NOMBRE_ESTUDIANTE_2}] Mi valor espera_2 generado es {espera_2} segundos")
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind((HOST_LOCAL, PORT_LOCAL))
        s.listen()
        print(f"[{NOMBRE_ESTUDIANTE_2}] Esperando a conectar...")
        conn, addr = s.accept()
        with conn:
            print(f"[{NOMBRE_ESTUDIANTE_2}] Conexión aceptada de {addr}")
            data = conn.recv(1024).decode('utf-8')
            if not data:
                return
            print(f"[{NOMBRE_ESTUDIANTE_2}] Recibido mensaje de Estudiante_1: {data}")
            data = conn.recv(1024).decode('utf-8')
            if not data:
                return
            print(f"[{NOMBRE_ESTUDIANTE_2}] Recibido datos recibidos '{data}'")
            parts = data.split('.')
            fecha_rx = parts[0]
            espera_1 = int(parts[1])
            print(f"[{NOMBRE_ESTUDIANTE_2}] Recibido {fecha_rx} y {espera_1} segundos")
            conn.sendall(str(datetime.datetime.now()).encode('utf-8'))
            conn.sendall(str(espera_2).encode('utf-8'))
```

```
estudiante_3.py
import socket
import random
import sys

# --- CONFIGURACIÓN ---
HOST = '0.0.0.0'
PORT = 65433

def main():
    # Generar espera_3 (nonce)
    espera_3 = random.randint(1, 10)
    print(f"[Tercero] Mi valor espera_3 generado es {espera_3} segundos")
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.bind((HOST, PORT))
        s.listen()
        print(f"[Tercero] Escuchando en el puerto {PORT}")
        conn, addr = s.accept()
        with conn:
            print(f"[Tercero] Conexión aceptada de {addr}")
            data = conn.recv(1024).decode('utf-8')
            if not data:
                return
            print(f"[Tercero] Recibido mensaje de Estudiante_2: {data}")
            if data == str(espera_3):
                print(f"[Tercero] Espera_3 coincide con el valor recibido")
            else:
                print(f"[Tercero] Espera_3 no coincide con el valor recibido")
            conn.sendall(str(espera_3).encode('utf-8'))
```

EJERCICIO 2. CERTIFICADOS

Los archivos .py requeridos están adjuntos en la tarea.

