Project 1: TCP Socket Programming and Packet Capture

Objectives

In this project you will:

- Learn Python socket programming by building a client and server.
- Capture and analyze network traffic using both `tshark` (CLI) and Wireshark (GUI).
- Relate application-layer behavior (echo messages) to transport-layer packets.
- Prepare for future projects that build upon socket programming.

Assigned Port Number

Each student is assigned a **unique TCP port number** (starting from 30000).

- The list of port numbers is published on Canvas.
- **Your server must run on your assigned port number.**
- When submitting your project, **explicitly state the port number you used** in your report.
- Submissions without the correct port number will not receive credit.

Part 1: Understanding and Modifying the Code

1. Start from the provided `proj.py`. Understand how the combined server and client threads work.
2. Remove the two `time.sleep(5)` statements and observe what changes.
3. Split the code into two separate files: `server.py` and `client.py`.
4. Modify the server so that it **reverses the string** sent by the client and **swaps the case** of all characters before returning the response.
    - Example: Client sends `HELLO` → Server returns `olleh`.

5. Extend the client to read input lines from `in-proj.txt`, send them to the server, and write the server's response to `out-proj.txt`.

Part 2: Capturing Traffic

You may use either **tshark (CLI)** or **Wireshark (GUI)**. Both produce `.pcap` files that can be analyzed later.

Important: Please set the capture interface to `any`.
This way you can see traffic from both the loopback (local) and external (remote) interfaces, and understand the difference between them.

Capture 1 (both client and server on rlab5, loopback traffic visible via `any`)

- **tshark**:

```
tshark -i any -f "tcp port <ASSIGNED_PORT>" -w proj1_part1.pcap
```

- **Wireshark**:
  1. Start Wireshark on rlab5.
  2. Select interface `any`.
  3. Set capture filter: `tcp port <ASSIGNED_PORT>`.
  4. Start capturing, then stop after the program finishes.
  5. Save as `proj1_part1.pcap`.

Capture 2 (client on rlab5, server on another ilab machine)

- **tshark**:

```
tshark -i any -f "host <SERVER_IP> and tcp port <ASSIGNED_PORT>" -w
proj1_part2.pcap
```

- **Wireshark**:
  1. Start Wireshark, choose interface `any`.
  2. Set capture filter: `host <SERVER_IP> and tcp port <ASSIGNED_PORT>`.
  3. Run your client and server programs.
  4. Stop capture after messages are exchanged.
  5. Save as `proj1_part2.pcap`.

Part 3: Analyzing Traffic

You can analyze captures using either `tshark` or Wireshark GUI.

- **tshark examples**:
  - Summary statistics:

    ```
    tshark -r proj1_part1.pcap -q -z io,stat,0
    ```

  - List TCP packets:

    ```
    tshark -r proj1_part2.pcap -Y 'tcp' -T fields       -e frame.number
    -e ip.src -e ip.dst -e tcp.srcport -e tcp.dstport -e tcp.len | head
    ```

  - Find the welcome message packet:

    ```
    tshark -r proj1_part1.pcap -Y 'tcp contains "Welcome to CS 352!"'
    -T fields -e frame.number -e tcp.len
    ```

- **Wireshark GUI tips**:
  - Use **Follow → TCP Stream** to see the client-server exchange.
  - Inspect the **three-way handshake** (SYN, SYN-ACK, ACK).
  - Right-click packets to view payloads and application data.
  - Use display filters:
    - `tcp.port == <ASSIGNED_PORT>`
    - `tcp contains "Welcome to CS 352!"`

What to Hand In

Submit a single zipped folder containing:

1. `client.py`
2. `server.py`
3. `proj1_part1.pcap` (rlab5 loopback capture, interface = any)
4. `proj1_part2.pcap` (cross-machine capture, interface = any)
5. `report.pdf` (your answers to analysis questions, including port number used and team details)

Report Guidelines

Your report (`report.pdf`) must include:

1. **Team details**: names and NetIDs of both members, and the assigned server port number.
2. **Collaboration**: note with whom you collaborated (if anyone), and on what aspects.
3. **Analysis questions**:
   - Protocol basics (transport protocol, ports, interfaces).
   - Application-layer messages (which packet carries "Welcome to CS 352!", client message/reply packets, etc.).
   - Packet sizes and total byte counts.

Grading Rubric

Component	**Points**
Correct implementation of client & server (Steps 1–5)	40
Proper use of assigned port number	5
Successful packet captures (`.pcap` files)	20
Report correctness & completeness (analysis questions)	30
Code clarity and submission format	5
Total	**100**