Server Structure (server.py)

1. Startup and Initial Setup
    - The server attempts to determine its own IP using a temporary socket. It connects to a distant address to discern its network IP address.
    - A list of servers is loaded from a CSV file, servers.csv.
2. Connection Management
    - For each server (line in the CSV corresponding to the server's current IP), a thread is started to listen on the specified port and accept incoming connections.
3. Task Reception and Management
    - When a connection is established, the server reads incoming messages, breaks them down into tasks (extracting a list of numbers), and stores them in an incoming queue.
4. Long-Term Management
    - The server periodically examines the incoming queue to move tasks to a ready queue, based on a scoring strategy that minimizes a function of the task's length divided by a timestamp.
5. Short-Term Management
    - Tasks in the ready queue are processed one by one, calculating either the maximum value or the average of the numbers, depending on the option specified by the client.
6. Returning Results
    - After processing, the result is sent back to the corresponding client.

Client Structure (client.py)

1. Connecting to the Server
    - Clients connect to the server at the specified address and port, send a task as a list of numbers, and an option (max or mean).
2. Receiving the Response
    - After sending the data, the client waits for the server's response, receives it, displays it, and closes the connection.

Example of servers.csv

- A file containing IP addresses and ports for setting up the servers.

Points of Attention and Potential Improvements

1. Security and Reliability
    - Data transmitted in plain text via TCP can be subject to interception.
2. Performance and Scalability
    - The method of selecting tasks in the queue and multithread processing can be optimized to better handle a larger number of simultaneous connections or tasks.
3. System Resources

- Intensive use of threads can be resource-demanding. Explore alternatives like asynchronous operations or thread pools.