Understanding the RL Agent Code

1 Problem Definition: Reaching a Goal State

The reinforcement learning (RL) agent starts at **state 0** and aims to reach **state 10**. It has two possible actions:

- 0: Move left ($s \to s - 1$)
- 1: Move right ($s \to s + 1$)

The reward system is defined as:

- Reaching state 10: **+10 reward**
- Any other state: **-1 penalty** (to encourage faster learning)

2 Mathematical Representation: Q-Learning

At any state s, the agent takes an action a, transitions to s', and receives a reward R. The Q-value update rule is given by:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[R + \gamma \max_{a'} Q(s',a') - Q(s,a) \right] \tag{1}$$

where:

- $\alpha = 0.1$ is the **learning rate**, controlling how fast the agent updates its knowledge.
- $\gamma = 0.9$ is the **discount factor**, determining the importance of future rewards.
- $\max_{a'} Q(s', a')$ represents the best future reward the agent can expect from state s'.

3 Q-Table Initialization and Learning

The agent maintains a Q-table $Q[s][a]$ to store expected rewards for each state-action pair. Initially, all values are set to zero. The agent learns by exploring different actions over multiple episodes.

4 Exploration vs. Exploitation: Epsilon-Greedy Strategy

To balance learning and decision-making, the agent follows an ϵ-greedy approach:

- With probability $\epsilon = 0.1$, the agent **chooses a random action** (exploration).
- Otherwise, it selects the action with the highest Q-value (exploitation).
- Over time, ϵ **decreases**, making the agent rely more on learned values.

5 Training Process: Running an Episode

Each episode follows these steps:

1. Start at **state 0**.
2. Choose an action using the **ϵ-greedy policy**.
3. Take the action, receive **reward** and transition to **next state**.
4. Update the **Q-table** using the Q-learning formula.
5. Repeat until reaching **goal (state 10)** or exceeding the step limit.
6. Send episode results to the **MCP server** to share with other agents.
7. If an agent finds a better solution, update the **global best** steps.

6 Multi-Agent Collaboration

- The system runs **3 agents in parallel**.
- Each agent communicates through **socket messages**.
- If one agent finds a better path, **all agents learn from it**.

7 Stopping Conditions

Training stops when:

- An agent finds the **optimal solution** (10 steps).
- The exploration rate (ϵ) reduces to a small value.

8 Conclusion

This approach enables efficient learning by:

- Using Q-learning to update knowledge over time.
- Balancing **exploration and exploitation**.
- Allowing **multi-agent collaboration** to speed up training.

This RL framework can be extended to more complex environments and multi-agent decision-making scenarios.