Exercise: Smart Traffic Light Controller using Q-Learning

Objective

In this exercise, you'll design a Smart Traffic Light System that learns when to switch lights (Green/Red) based on real-time traffic conditions using Q-Learning. Your AI agent will balance reducing waiting time for vehicles, saving energy, and maintaining safety.

Background

A city intersection has a traffic light that can either stay green or turn red. Depending on how heavy the traffic is, the system must learn the best decision — whether to keep the light green for more vehicles or turn it red to allow cross traffic. Reinforcement Learning allows the agent to learn these rules automatically using rewards and penalties.

Instructions

Step 1: Import Required Libraries

Import the required Python libraries:
- NumPy → to handle numerical operations
- random → to simulate unpredictable traffic conditions

Print a confirmation message once imports are successful.

Step 2: Define States and Actions

Define discrete traffic states to represent how busy the road is:
0 = Empty road
1 = Light traffic
2 = Moderate traffic
3 = Heavy traffic
4 = Very heavy traffic

Define possible actions:
- 'GREEN' → Keep the light green
- 'RED' → Turn the light red

Print both lists to verify.

Step 3: Initialize the Q-Table and Hyperparameters

Create a Q-table initialized with zeros (rows = states, columns = actions). Define hyperparameters:
- alpha → Learning rate (e.g., 0.1)
- gamma → Discount factor (e.g., 0.9)
- epsilon → Exploration rate (e.g., 0.2)

- episodes → Number of training episodes (e.g., 300)
Explain in a comment what each parameter controls.

Step 4: Design the Reward Function

Create logic for assigning rewards:

Traffic Level	Action	Reward	Meaning
Heavy traffic	GREEN	+10	Good: clears congestion
Heavy traffic	RED	-10	Bad: creates jams
Empty road	RED	+5	Saves energy
Empty road	GREEN	-5	Wastes power
Moderate traffic	Either	+1	Neutral

Write a get_reward(traffic, action) function that returns the correct reward.

Step 5: Define Environment Dynamics

Simulate how traffic changes each step:
- Traffic can randomly increase, decrease, or stay constant.

Create a function next_traffic(current) that randomly changes traffic by ±1 level and uses np.clip() to keep it between 0 and 4.

Step 6: Train the Agent

Implement the Q-learning training loop:
1. For each episode, start from a random traffic level.
2. Choose an action using ε-greedy strategy.
3. Get next traffic level and reward.
4. Update Q-table using the Q-learning formula:
   $Q(s,a) = Q(s,a) + \alpha [r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
5. Print a message when training completes.

Step 7: Test the Learned Traffic Light Controller

Ask the user to enter a starting traffic level (0–4). Simulate 10 steps of decision-making:
Each step:
- Choose the best action (GREEN/RED)
- Print traffic level and action
- Simulate next traffic condition.

Step 8: Analyze the Behavior

After testing, discuss:
- Does the system keep light GREEN for heavy traffic?
- Does it turn RED when the road is empty?
- What happens if ε is increased or decreased?
- How does Q-table change over time?