**Reinforcement Learning Basics**

**Problem: 01**  
Consider a **2D Grid World** where an agent moves in a 3×3 grid. The agent starts at the top-left (0,0) and must reach the bottom-right (2,2). The possible actions are:

* **0**: Move Left
* **1**: Move Right
* **2**: Move Up
* **3**: Move Down

The environment follows these rules:

* The agent cannot move outside the grid.
* The goal state (2,2) gives a reward of **+1**.
* All other moves give a reward of **-0.1** to encourage faster completion.
* The episode ends when the agent reaches (2,2).

**Task:**

1. Implement the environment.
2. Create a loop where an agent moves randomly in the grid.
3. Print the state, action taken, and reward at each step.

This problem introduces the **agent**, **state space**, **action space**, **rewards**, and **episodes** in a way that makes reinforcement learning concepts intuitive. You can later expand it by implementing Q-learning or deep RL methods.

**Problem: 02**  
Consider a **Taxi Environment**, where a taxi moves on a **5×5 grid** to pick up and drop off a passenger. The taxi starts at a random position, and the passenger is located at a random position. The task is to pick up the passenger, navigate to the destination, and drop them off.

**Environment Rules:**

1. The taxi can move in four directions:
   * **0**: Move Left
   * **1**: Move Right
   * **2**: Move Up
   * **3**: Move Down
2. The taxi can perform two additional actions:
   * **4**: Pick up passenger
   * **5**: Drop off passenger
3. The taxi receives the following rewards:
   * **-1** for each move
   * **+10** for successfully dropping off the passenger at the correct destination
   * **-5** for trying to pick up or drop off incorrectly
4. The episode ends when the passenger is dropped at the correct destination.

**Task:**

1. Implement the environment.
2. Create a loop where an agent moves randomly in the grid.
3. Print the state, action taken, and reward at each step.

This problem introduces:

* **State space** (Taxi location, passenger location, destination)
* **Action space** (Move, pick up, drop off)
* **Rewards** (Encourages correct drop-offs and penalizes incorrect actions)
* **Episodes** (End when passenger is successfully dropped off)

This is a **simplified version** of the OpenAI Gym **Taxi-v3** environment. You can later extend it by adding obstacles, reinforcement learning policies, or training an RL agent using Q-learning.

**Problem: 03**

**Q-learning implementation for the Personalized Quiz Tutor, including training and user interaction.**

**1. Define the Q-Learning Environment**

* The states are Beginner, Intermediate, and Advanced.
* The actions are Easy, Medium, and Hard questions.
* The reward system:
  + +10 points for a correct answer.
  + -5 points for an incorrect answer.

**2. Train the Q-Learning Agent**

* The agent explores different question difficulties for each skill level.
* The Q-table updates values based on feedback.

**3. User Interaction**

* The agent asks questions based on learned Q-values.
* The user answers, and the agent adjusts difficulty dynamically

**Problem 04:**

Optimizing boiler efficiency is a critical task in power plants, as it directly impacts fuel consumption, emissions, and operational costs. The goal of this project is to develop a Reinforcement Learning (RL)-based agent that learns the best combination of input parameters to maximize boiler efficiency.

The agent will be trained on synthetic data representing various boiler input settings and their corresponding efficiencies. Once trained, the agent will suggest the optimal settings based on a given efficiency target set by the user.

**Define the Environment**

The environment simulates a boiler system where the agent controls critical parameters. We define:

* State: The current boiler conditions (efficiency and settings).
* Action: Adjusting input parameters like coal flow, air supply, flue gas settings, etc.
* Reward: Higher reward for achieving better efficiency while maintaining safe operational limits.

**Define the Parameters**

Key operational parameters considered:

1. Coal Input (kg/hr) – Amount of coal supplied to the boiler.
2. Primary Air Flow (m³/hr) – Air used for coal combustion.
3. Secondary Air Flow (m³/hr) – Additional air to improve combustion efficiency.
4. Spray Water Flow (kg/hr) – Used to control steam temperature.
5. Flue Gas Exit Temperature (°C) – Higher values indicate heat loss.
6. Excess Oxygen (%) – Ensures complete combustion but too much reduces efficiency.
7. Boiler Efficiency (%) – Target variable to be optimized.

**Generate Synthetic Data**

1. Since real boiler data may not be available, we will generate synthetic data. The data will include randomized values for the above parameters while ensuring realistic constraints

**Train a Q-learning Agent**

1. Initialize a Q-table where the agent stores actions and their impact on efficiency.
2. Define an exploration strategy (ε-greedy) to balance learning vs. exploitation.
3. Simulate interactions where the agent takes an action (adjusts a setting), receives a new efficiency, and updates the Q-table.
4. Train until convergence where the agent learns the best settings for any given efficiency request.

**Use the Agent for Decision-Making**

Once trained, the agent should be able to:

* Take a target efficiency from the user.
* Retrieve the best possible settings from the Q-table.
* Suggest optimal values for coal input, airflow, flue gas temp, etc