# Cat and Mouse Game using Q-Learning

## Overview

The Cat and Mouse Game project demonstrates the application of the Q-Learning algorithm in a simplified grid-based environment. This project is designed to help visualize how an agent (the cat) learns to navigate through a grid to reach a goal (the mouse) while avoiding obstacles (dogs). The visual representation is built using Tkinter, providing an interactive and engaging way to observe the learning process of the agent.

Let's dive into each segment of the code one by one.

## Environment: GridWorld

### Initialization

The `GridWorld` class defines a 6x6 grid environment with the following attributes:

- **grid_size**: Size of the grid (6x6).
- **state**: Current position of the agent, initialized at the top-left corner (0, 0).
- **goal_state**: Position of the mouse, set at the bottom-right corner (5, 5).
- **obstacles**: List of coordinates representing obstacles (dogs) in the grid.
- **actions**: Possible actions the agent can take ('up', 'down', 'left', 'right').

### Methods

- **reset()**: Resets the agent's position to the initial state.
- **step(action)**: Takes an action and updates the agent's position. Returns the new state, reward, and a boolean indicating if the episode is done.
- **render()**: Returns a grid representation where:
  - -1 represents the cat's position.
  - 1 represents the mouse's position.
  - 2 represents obstacles.

## Agent: Q-Learning

### Q-Learning Algorithm

Q-Learning is a model-free RL algorithm that aims to learn the value of taking a particular action in a given state. The agent uses a Q-table to store Q-values, which are updated based on the agent's experiences.

### QLearningAgent Class

- **Attributes**:

    - **env**: The environment the agent interacts with.
    - **alpha**: Learning rate.
    - **gamma**: Discount factor.
    - **epsilon**: Exploration rate.
    - **epsilon_decay**: Rate at which exploration rate decays.
    - **epsilon_min**: Minimum exploration rate.
    - **q_table**: Table storing Q-values for state-action pairs.
- **Methods**:
    - **choose_action(state)**: Chooses an action based on the epsilon-greedy policy.
    - **update_q_table(state, action, reward, next_state)**: Updates the Q-value based on the TD error.
    - **train(num_episodes)**: Trains the agent by running multiple episodes and updating the Q-table.

## Visualization: RLVisualizer

### Tkinter Interface

The RLVisualizer class extends Tkinter's `Tk` class to create a graphical representation of the GridWorld environment.

- **Attributes**:
    - **agent**: The Q-learning agent.
    - **env**: The GridWorld environment.
    - **canvas**: Tkinter canvas to draw the grid.
    - **images**: Images for the cat, mouse, sad mouse, sad cat, and dog.
    - **labels**: Labels to display current and cumulative rewards.
- **Methods**:
    - **render_grid(final_state=None)**: Renders the grid with the cat, mouse, and obstacles.
    - **update_grid()**: Updates the grid in real-time as the agent moves and learns.

## Running the Project

1. **Initialize Environment and Agent**:

   ```
   env = GridWorld(grid_size=6)
   agent = QLearningAgent(env)
   ```

2. **Train the Agent**:

   ```
   agent.train(num_episodes=500)
   ```

3. **Run the Visualizer**:

   ```
   app = RLVisualizer(agent, env)
   app.mainloop()
   ```

## Q-Learning Algorithm Details

Q-Learning follows a simple update rule:

$$q^{new}(s,a) = (1-\alpha)\underbrace{q(s,a)}_{\text{old value}} + \alpha\overbrace{\left(R_{t+1} + \gamma \max_{a'} q(s',a')\right)}^{\text{learned value}}$$

Where:

- α\alphaα is the learning rate.
- γ\gamma is the discount factor.
- R is the reward received after taking action A in state S.
- S′ is the next state after taking action A.

## Conclusion

This Cat and Mouse Game demonstrates the application of Q-learning in a simple grid environment. By visualizing the agent's learning process, we can better understand how reinforcement learning algorithms work and how the agent improves over time. The project highlights the importance of exploration and the impact of rewards on the agent's behavior.