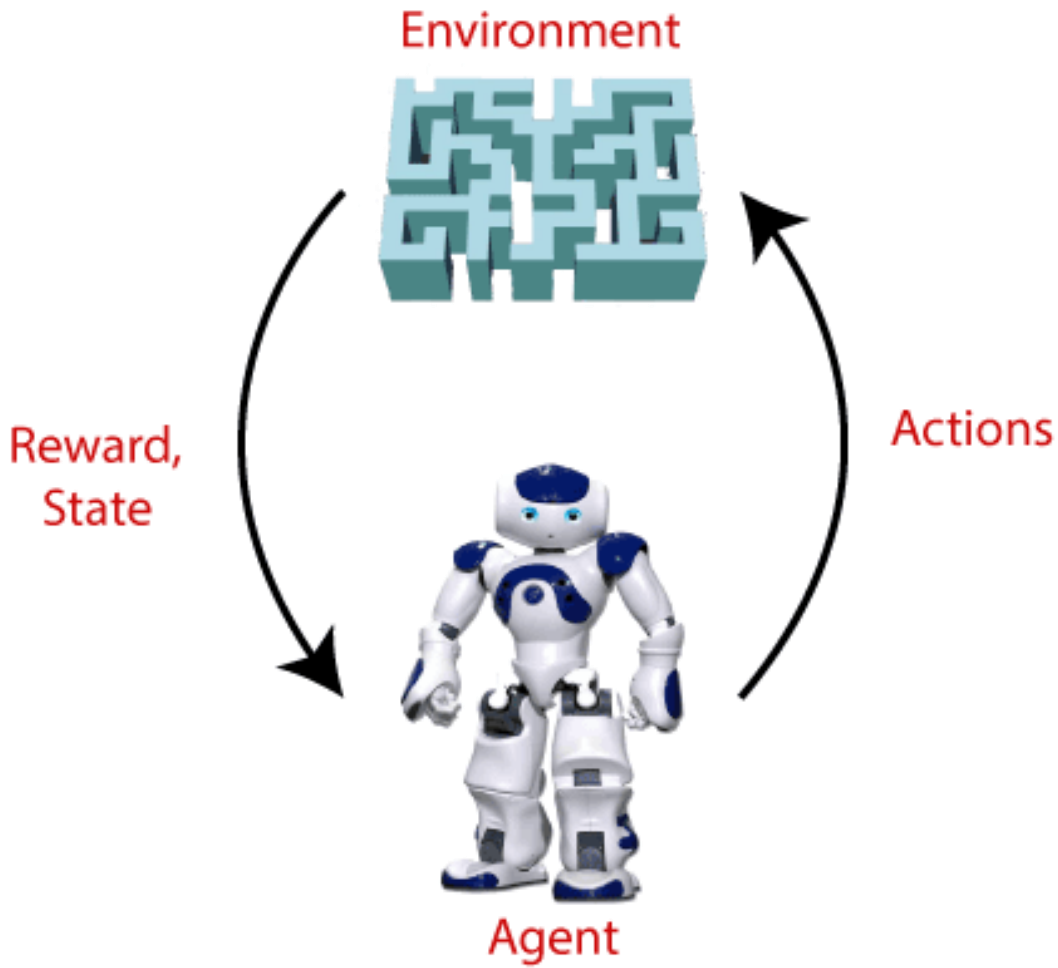
WEEK 1

What is Reinforcement Learning?

- **Agent:** The learner or decision-maker.
- **Environment:** Everything the agent interacts with.
- **State:** A specific situation in which the agent finds itself.
- **Action:** All possible moves the agent can make.
- **Reward:** Feedback from the environment based on the action taken.



How Reinforcement Learning Works?

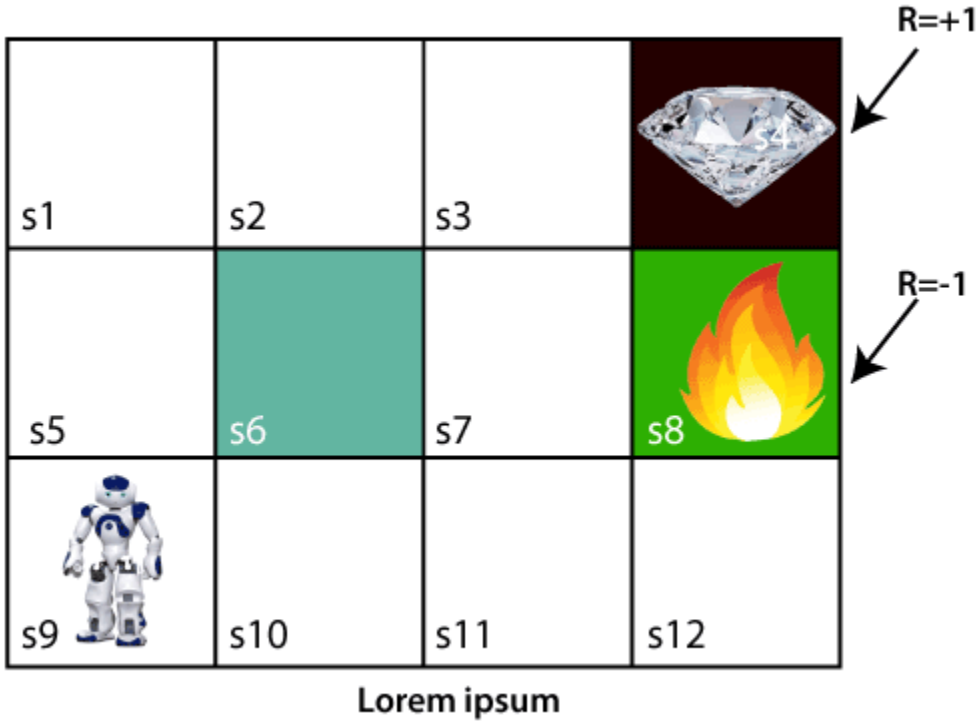
- RL operates on the principle of learning optimal behavior through trial and error.
- The agent takes actions within the environment
- Receives rewards or penalties
- Adjusts its behavior to maximize the cumulative reward.

How Reinforcement Learning Works?

- **Policy:** A strategy used by the agent to determine the next action based on the current state.
- **Reward Function:** A function that provides a scalar feedback signal based on the state and action.
- **Value Function:** A function that estimates the expected cumulative reward from a given state.
- **Model of the Environment:** A representation of the environment that helps in planning by predicting future states and rewards.

Example

- **Environment:** It can be anything such as a room, maze, football ground, etc.
- **Agent:** An intelligent agent such as AI robot.



S_6 block, which is a **wall**, S_8 a **fire pit**, and S_4 a **diamond block.**

agent reaches the S_4 block, then get the **+1 reward**

reaches the fire pit, then gets **-1 reward point**

four actions**: move up, move down, move left, and move right.**

any path to reach to the final point - fewer steps.

OpenAI gym Library

Introduction

- Open-source toolkit in Python
- Robust and scalable environment for creating, simulating, and solving RL problems.
- Key Features
  - Wide Range of Environments
  - Standardized API
  - Custom Environment Creation
  - Active Maintenance
  - Integration with RL Libraries

Installation

pip install gymnasium

Basic Usage

```python
import gymnasium as gym

# Create an environment
env = gym.make("CartPole-v1")

# Reset the environment
state, info = env.reset()

done = False
while not done:
    # Take a random action
    action = env.action_space.sample()
    state, reward, done, truncated, info = env.step(action)

env.close()
```

Environments

- **CartPole-v1**: Balance a pole on a moving cart.
- **MountainCar-v0**: Drive a car up a steep hill.
- **Pendulum-v1**: Swing a pendulum upright
- **Taxi-v3**: Pick up and drop off passengers at the right locations.
- **FrozenLake-v1**: Navigate a grid while avoiding holes.
- **Blackjack-v1**: Play a game of blackjack.

Basic concepts

Environment An environment is where the agent interacts to learn behavior. It encapsulates:

- **State (Observation)**: The current situation of the environment.
- **Action**: What the agent can do in the environment.
- **Reward**: Feedback received from the environment after taking an action.
- **Episode**: A sequence of states, actions, and rewards, starting from an initial state and ending in a terminal state.

```
import gymnasium as gym
env = gym.make("CartPole-v1")
```

Action Space

Defines the set of all possible actions an agent can take. It can be:

- **Discrete**: A finite set of actions (e.g., move left or right).
- **Continuous**: Actions with continuous values (e.g., steering an angle).

```
print(env.action_space)
```

Observation Space

Defines the type, structure, and range of observations (or states) the environment provides. It can include:

- Vectors of numbers (e.g., position, velocity).
- Images (e.g., frames in Atari games).

```
print(env.observation_space)
```

Step Function

The step(action) method advances the environment by one timestep based on the action taken by the agent. It returns:

- **State (Observation)**: The new state after the action.
- **Reward**: The immediate reward for the action.
- **Done**: A boolean indicating if the episode is finished.
- **Truncated**: A boolean indicating if the episode was terminated due to a time limit or custom condition.
- **Info**: Additional debugging information.

```
action = env.action_space.sample()
```

```
state, reward, done, truncated, info = env.step(action)
```

Reset Function

Resets the environment to its initial state, starting a new episode.
Returns:

- **Initial State**: The starting observation.
- **Info**: Metadata about the environment state.

```
state, info = env.reset()
```

Reward Function

The reward function provides feedback to the agent for its actions. It guides the agent toward achieving its goal. Rewards can be:

- **Sparse**: Provided only when the goal is achieved.
- **Dense**: Provided for every step, proportional to the progress.

Episodes

An episode begins with reset() and ends when:

- The done flag is True (goal achieved or failure).
- The truncated flag is True (time limit or predefined conditions).

Rendering

Allows visualization of the environment to see what the agent "sees.“

```
env.render()
```

Closing an Environment

Always close the environment after use to free up resources.

```
env.close()
```

Steps

1. **Definition**:

   A step refers to a single interaction between the agent and the environment. During a step:

   - The agent takes an action.
   - The environment responds with the next state, a reward, and information about whether the episode has ended.

2. **Key Characteristics**:

   - Represents a **single timestep** in the environment.
   - Involves the transition from one state to another.
   - Cumulative reward and learning updates occur at each step.
   - Steps accumulate within an episode until it terminates.

3. **Example**: For a CartPole environment:

   - At step 1: The agent moves the cart to the right.
   - At step 2: The agent moves the cart to the left.

Episode

1. **Definition**:

   An episode is a complete sequence of steps, starting from the initial state (reset) and ending when the environment reaches a terminal or truncated state.

   - **Terminal**: Achieving the goal or failing (e.g., the pole falls in CartPole).
   - **Truncated**: Reaching a maximum number of steps or custom-defined criteria.

2. **Key Characteristics**:

   - Represents a **full trial** or run of the environment.
   - Ends with either success, failure, or a predefined condition.
   - Multiple episodes are required to train an agent, as learning happens over time.

3. **Example**: For the CartPole environment:

   - An episode might start with the pole upright.
   - The agent interacts with the environment, balancing the pole through multiple steps.
   - The episode ends when the pole falls or a time limit is reached.

WEEK 1: TUTORIAL ON OPENAI GYMNASIUM

1. Take the tutorial: Getting Started With OpenAI Gym: The Basic Building Blocks
   - https://www.gymlibrary.dev/content/tutorials/
   - https://blog.paperspace.com/getting-started-with-openai-gym/
2. Use the CartPole-v0 environment and write a program to :
   a. Implement the CartPole environment for a certain number of steps
   b. Implement the CartPole environment for a certain number of episodes
   c. Compare and comment on the rewards earned for both approaches.
   d. Plot the cumulative reward of the games and note down the results.

Implement the same considering the “Mountain Car” environment