Reinforcement Learning (RL) revolves around the idea of learning optimal actions in an environment by interacting with it. The goal is to maximize cumulative rewards over time. Mathematically, RL is often described using the following components and formulas:

1. Markov Decision Process (MDP)

An RL problem can be modeled as a Markov Decision Process (MDP), represented by the tuple $\langle S, A, P, R, \gamma \rangle$:

- S: Set of states
- A: Set of actions
- $P(s'|s, a)$: Transition probability from state s to s' given action a
- $R(s, a)$: Reward received when taking action a in state s
- γ: Discount factor, where $0 \leq \gamma \leq 1$

2. Policy π

A policy $\pi(a|s)$ defines the probability of taking action a in state s.

3. Return G_t

The return G_t is the total accumulated reward from time t, and is defined as:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

4. Value Function $V^{\pi}(s)$

The value function represents the expected return starting from state s and following policy π:

$$V^{\pi}(s) = \mathbb{E}^{\pi}\left[G_t | s_t = s\right] = \mathbb{E}^{\pi}\left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | s_t = s\right]$$

5. Action-Value Function $Q^{\pi}(s, a)$

The action-value function gives the expected return when taking action a in state s under policy π:

$$Q^{\pi}(s, a) = \mathbb{E}^{\pi}\left[G_t | s_t = s, a_t = a\right]$$

6. Bellman Equation for $V^{\pi}(s)$

The Bellman equation expresses the recursive relationship of the value function:

$$V^{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s'} P(s'|s, a) \left[R(s, a) + \gamma V^{\pi}(s')\right]$$

7. Optimal Policy π^*

The goal in RL is to find the optimal policy π^* that maximizes the expected return. The optimal value function satisfies:

$$V^*(s) = \max_{a} Q^*(s, a)$$

Example: Cart-Pole Balancing Problem

The Cart-Pole is a classic RL example where an agent needs to balance a pole on a cart by applying forces (actions) to move the cart left or right. The goal is to keep the pole upright for as long as possible.

- **State (S)**: The current position and velocity of the cart, and the angle and angular velocity of the pole.
- **Actions (A)**: Apply force to move the cart left or right.
- **Rewards (R)**: +1 for every time step the pole remains upright, and 0 if the pole falls over.
- **Policy** π: A strategy that maps states (cart and pole's conditions) to actions (which direction to move the cart).

The agent learns through trial and error by interacting with the environment, adjusting its policy to maximize the total accumulated reward over time. This can be done using various algorithms like Q-learning or policy gradient methods.

In summary, Reinforcement Learning is about finding the optimal policy π^* to maximize long-term rewards through the use of value functions and iterative updates based on the environment's feedback.