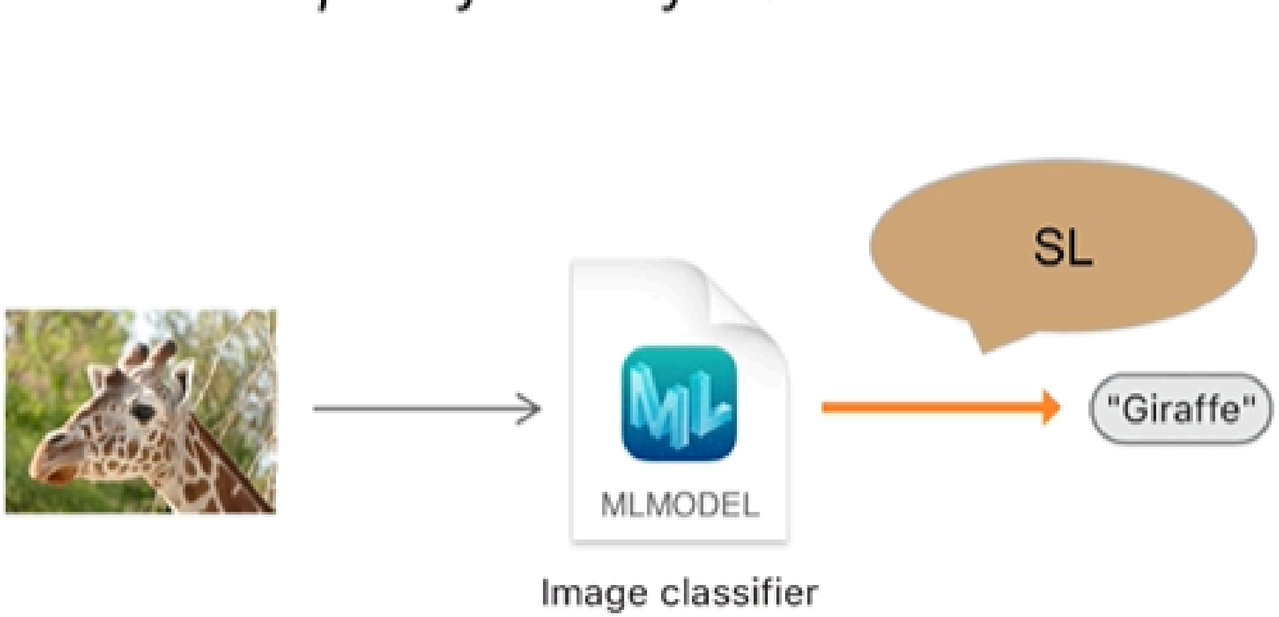
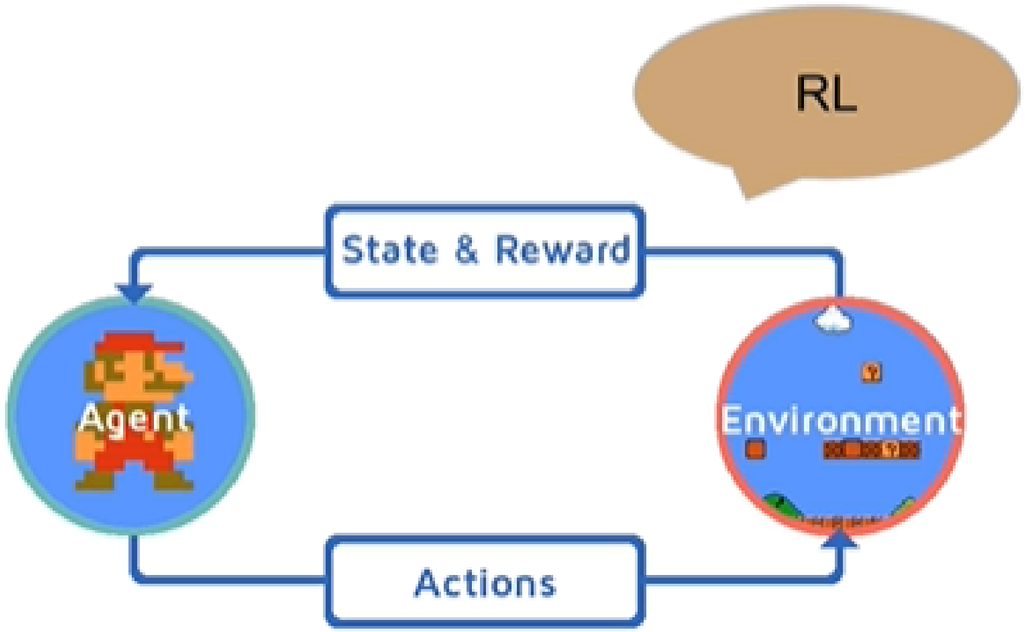
Reinforcement Learning

SL vs. RL

- SL is like a static function
- RL is more like a **loop**
- We want to achieve a **goal**
- It can *plan for the future*





Supervised Driving?

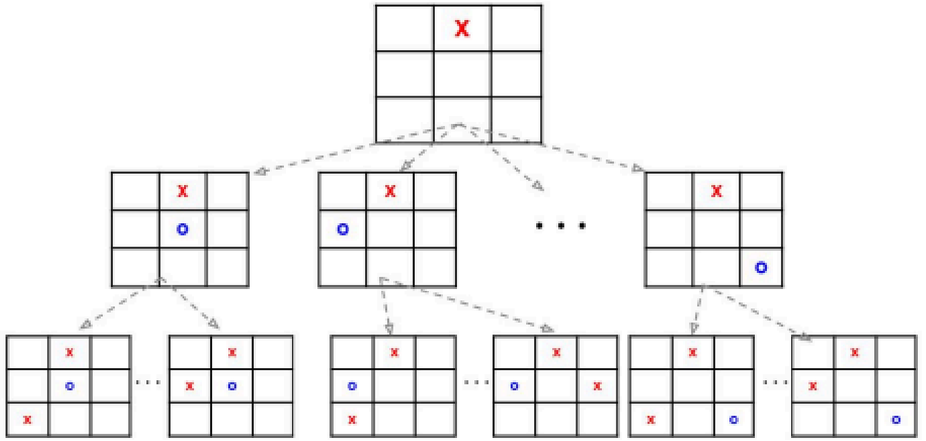
- Given an image, can you give it a target?
  - Steer left / right / brake / accelerate - and by how much?
- Even if you could - how would you label every possible image a car might see?
  - Standard camera: 30 FPS
  - 1 hour trip: 3600s
  - Total frames: 108,000!

Reinforcement Learning Approach

- RL uses *goals* rather than targets
- E.g. Solve a maze
  - You only need the goal: find the maze exit
  - You do not need to label the correct direction for each maze position (that's SL)

Reinforcement Learning

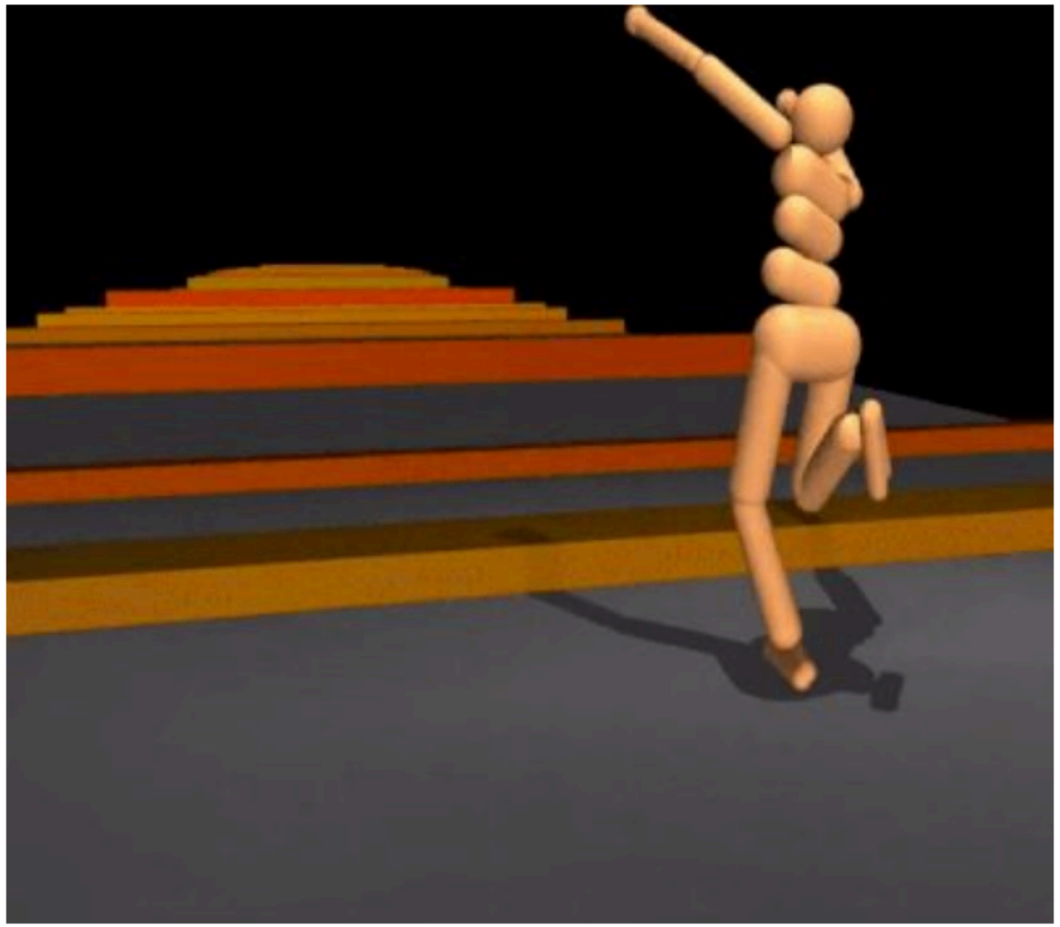
- Learn from evaluation
  - Win gives 1 point
  - Loss gives -1 point
  - Draw gives 0 points



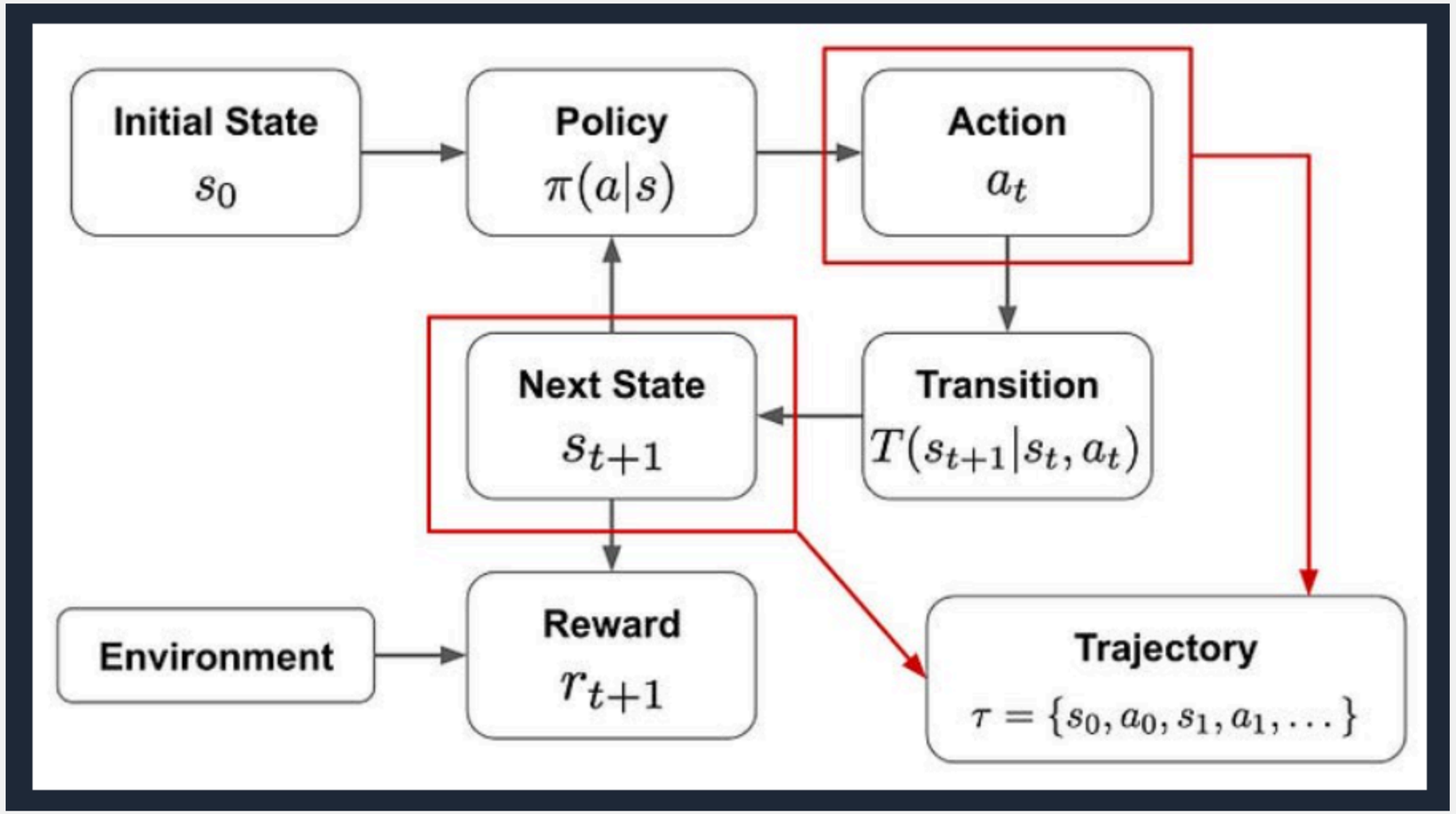
- Learn from repeated play

Examples

Humanoid Control



Reinforcement Learning

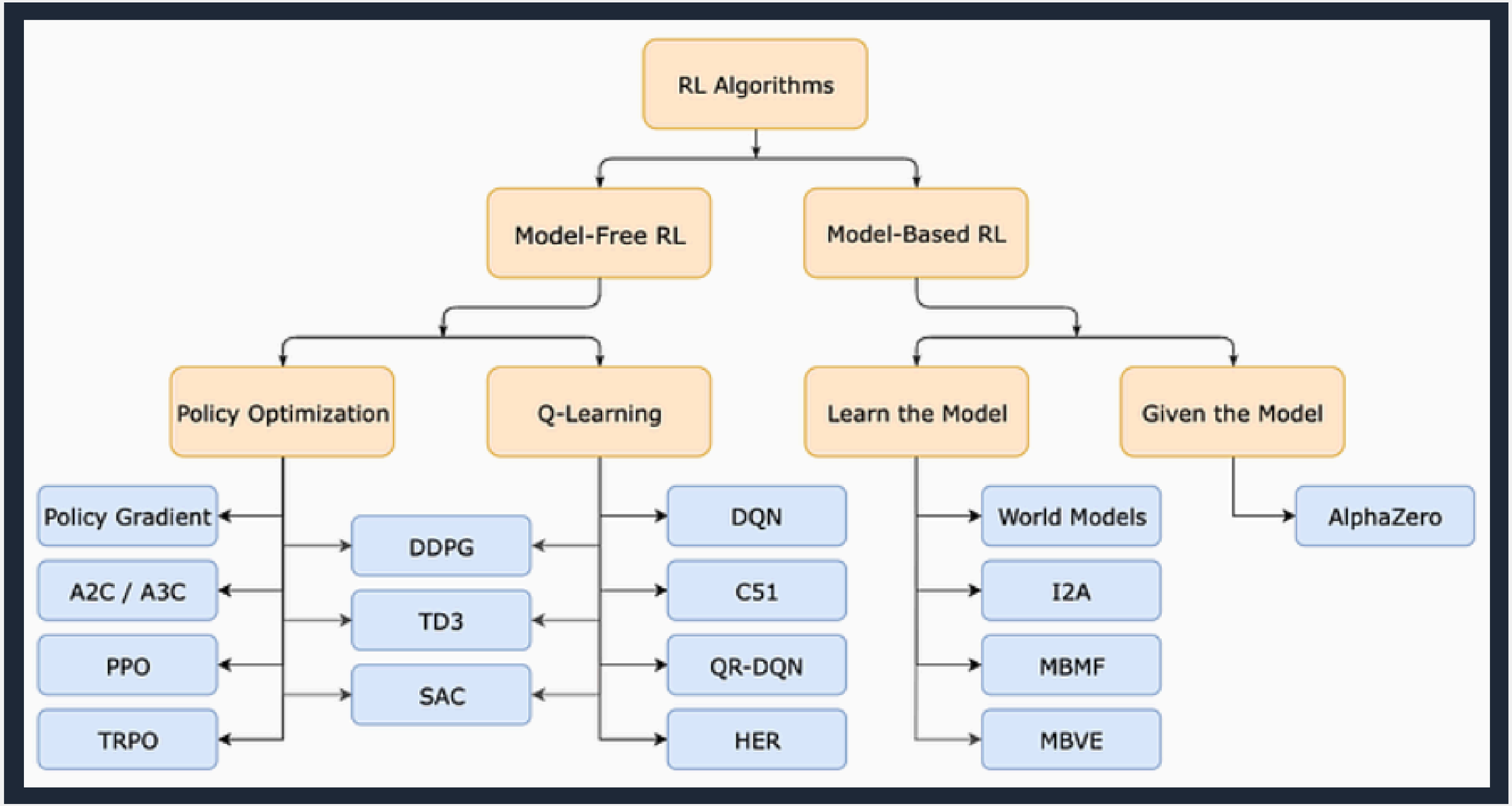


Reinforcement Learning is a type of machine learning where an agent learns to make decisions by interacting with an environment to maximize a reward.

Terms

- Agent : The learner that takes actions.
- Environment : The world where the agent interacts.
- State (s) : The current situation of the agent.
- Action (a) : The choice the agent makes.
- Reward (r) : The feedback from the environment.
- Policy : Probability which the agent uses to determine what action to perform given the current state

Types of Algorithms



Types of Algorithms

- Odds ratio preference optimization (ORPO)
- Group Relative Policy Optimization (GRPO)

Supervised finetuning (cross entropy loss)

- Is the capital of france is paris?

Forward Pass:	Ground Truth:	Back Prop:
Yes (40%) No (20%) Maybe (40%)	Yes (100%)	Yes (60%) No (15%) Maybe (25%)

ORPO (cross entropy loss + odds ratio)

- Is the capital of france is paris?

Forward Pass:

Yes (40%)
No (40%)
Maybe (20%)

Ground Truth:

Yes (100%)

Rejected Ans:

No (100%)

Back Prop:

Yes (40% + 10% + 10%(odds)) : 60%
No (40% - 5 % - 10% (odds)) : 25%
Maybe (20% - 5%) : 15%

GRPO

Example: The model generates these responses:

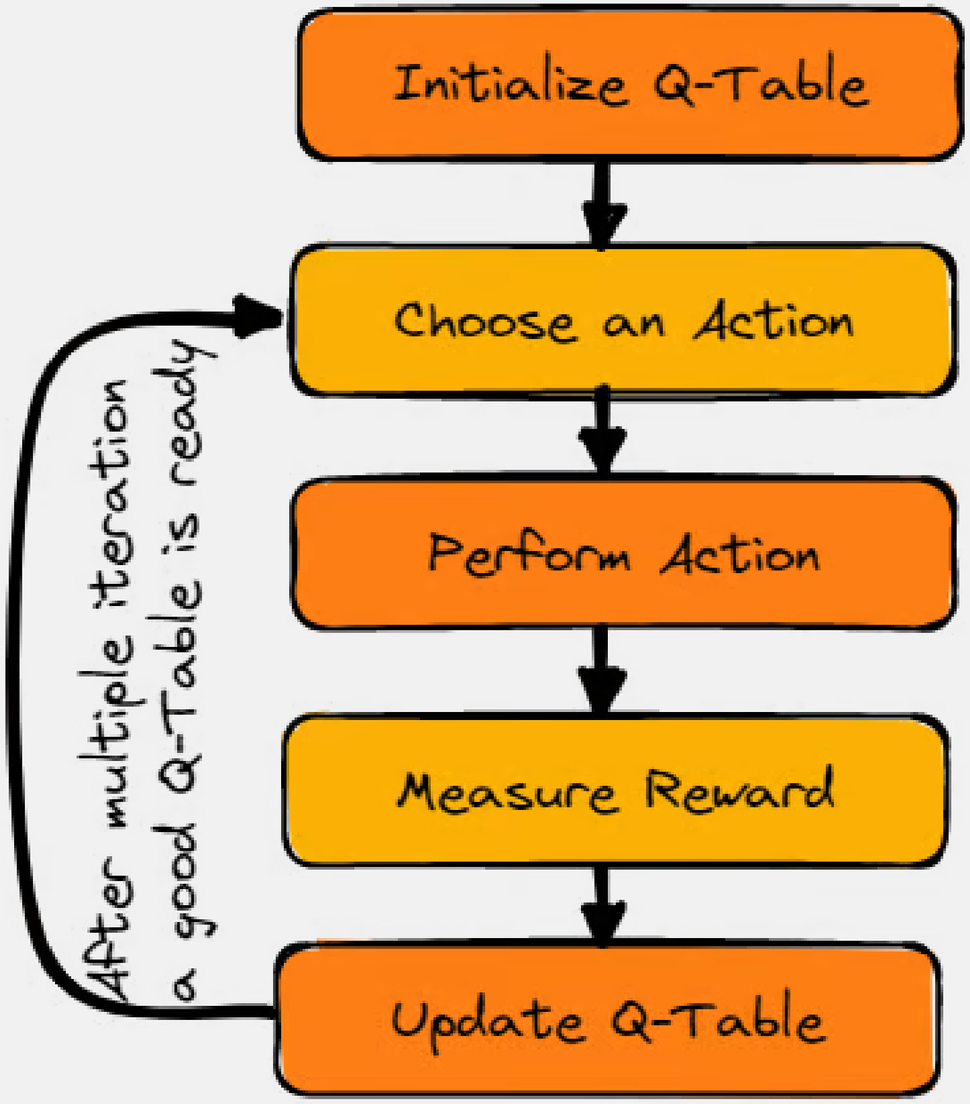
- o1: “The answer is 13.”
- o2: “Thirteen.”
- o3: “It’s 12.”
- o4: “The sum is 13.”

- r1=1.0 (correct and well-formatted).
- r2=0.9 (correct but less formal).
- r3=0.0 (incorrect answer).
- r4=1.0 (correct and well-formatted).

- Mean reward = $(1.0 + 0.9 + 0.0 + 1.0)/4 = 0.725$.
- Standard deviation = 0.453.
- $A_1 = \frac{1.0 - 0.725}{0.453} = 0.61$.
- $A_2 = \frac{0.9 - 0.725}{0.453} = 0.39$.
- $A_3 = \frac{0.0 - 0.725}{0.453} = -1.60$.
- $A_4 = \frac{1.0 - 0.725}{0.453} = 0.61$.

Q Learning

Q-learning is a model-free reinforcement learning algorithm. Here the agent learns to take the best decisions in various situations by interacting with an environment.
It uses trial and error to determine which actions lead to rewards (good outcomes) or penalties (bad outcomes). As the agent explores, it updates a Q-table with Q-values.



A **Q-table** is typically a 2D table, where:

- Rows represent different states (s).
- Columns represent different actions (a).
- Each cell stores the Q-value Q(s,a), which is an estimate of the long-term reward for taking action a in state s.

Q Tables

	→	←	↑	↓
Start	0	0	0	0
Idle	0	0	0	0
Hole	0	0	0	0
End	0	0	0	0

	→	←	↑	↓
Start	0	1	0	0
Idle	2	0	0	3
Hole	0	2	0	0
End	1	0	0	0

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma max_a Q(S_{t+1}, a) - Q(S_t, A_t)]$$

New Q-value estimation

Former Q-value estimation

Learning Rate

Immediate Reward

Discounted Estimate optimal Q-value of next state

Former Q-value estimation

The Q-values represent the expected future rewards for taking a particular action from a given state.