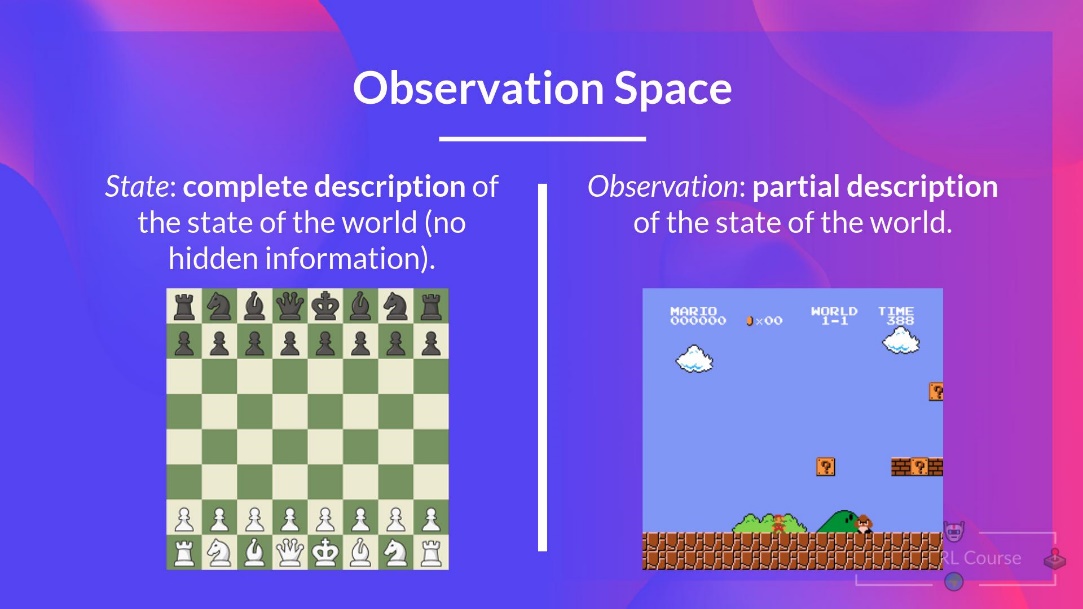
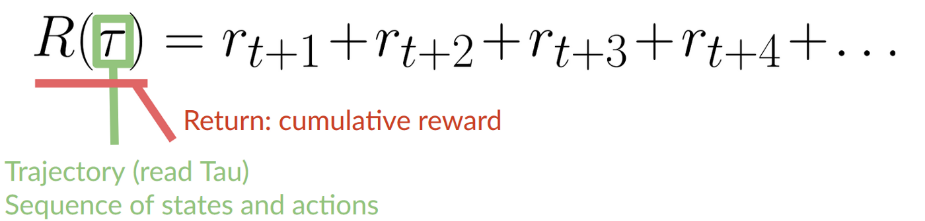
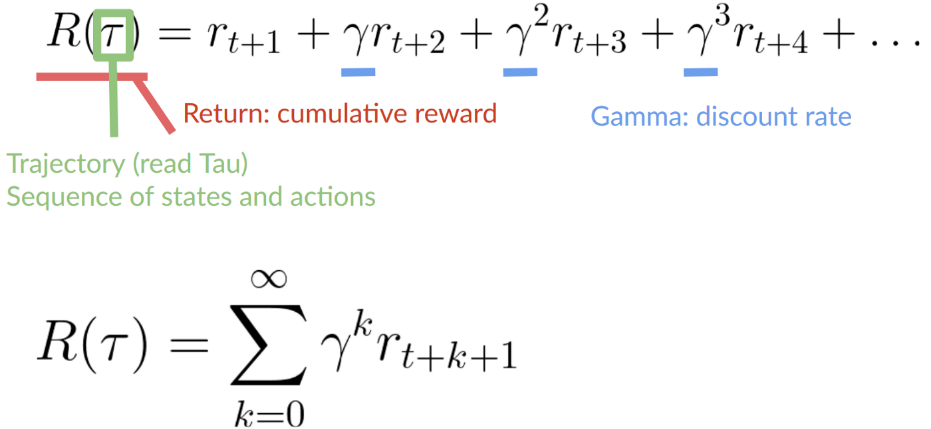
Reinforcement learning

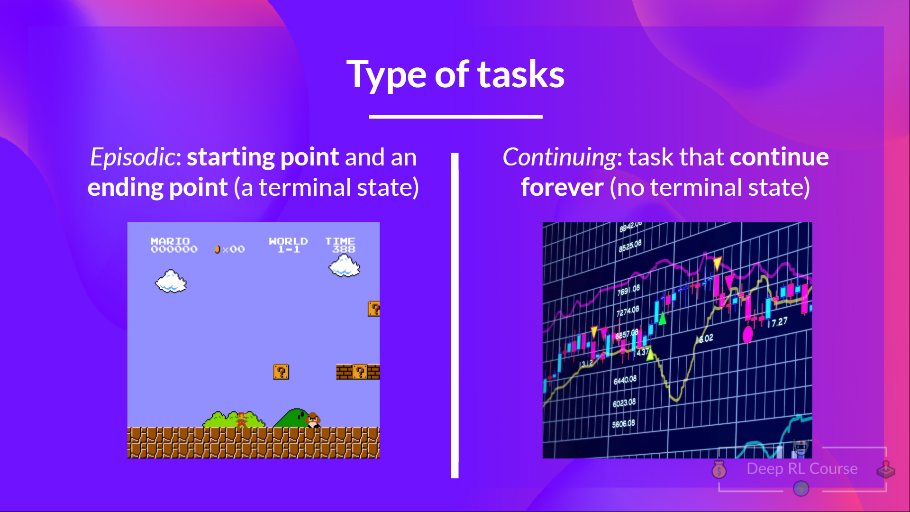
* Ideas comes from natural experiences: learn from the environment by interacting with it.
* Earn rewards or loses (punishments)
* computational approach to learning from actions
* Formal definition*: Reinforcement learning is a framework for solving control tasks (also called decision problems) by building agents that learn from the environment by interacting with it through trial and error and receiving rewards (positive or negative) as unique feedback.*
* Based on ***Reward hypothesis***: *all goals can be described as the****maximization of the expected return****(expected cumulative reward).*
* Learn to take actions that **maximize the expected cumulative reward.**
* **RL process also called “Markov Decision Process”**
* **“**Markov Property implies that our agent needs **only the current state to decide** what action to take and **not the history of all the states and actions** they took before.**”**
* State 🡪 Complete description of the state of the world (no hidden information). Fully observed environment
* E.g.: In chess 🡪 Full information from environment (we see entire chess board) = state
* Observation: Partial description of the state. Partially observed environment.
  + E.g.: In video game: See only part where u are playing



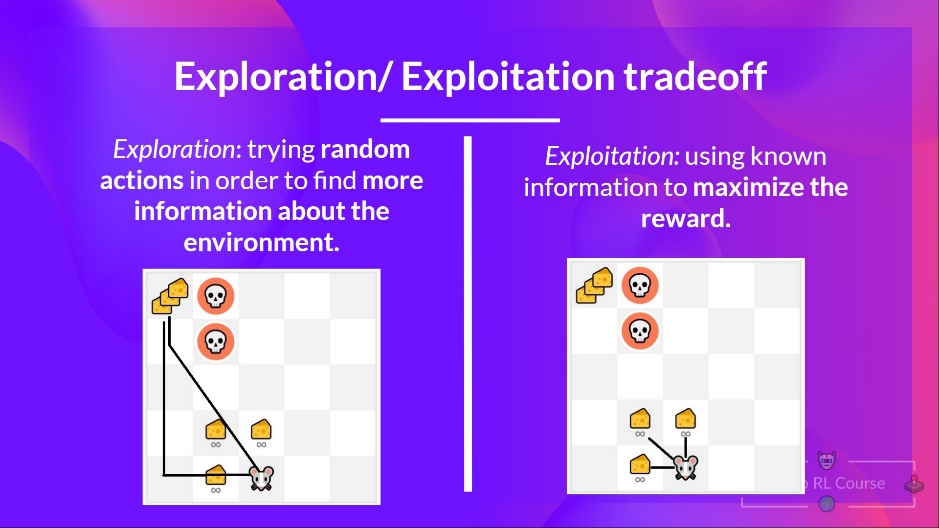
* Action space: Set of all possible actions in an environment.
  + Discrete: Finite set of possible actions (Chess)
  + Continuous: Infinite set of possible actions (self-driving car)
* Reward is only feedback agent gets.
* Cumulative reward = sum of all rewards in the sequence.
* Written as:
* 
* BUT! Rewards that come **sooner are more likely to happen**.
* Far future rewards are less likely 🡪 Discounted
* Reward discounted with gamma: γ 🡪 Defined {0,1}
* Larger γ 🡪 less discount 🡪 Care more about future
* Smaller γ 🡪 more discount 🡪 Care less about future
* Future rewards will be discounted by gamma to the exponent of the time step.
* Higher exponent 🡪 smaller gamma 🡪 more discount for future steps (less reward for far future)



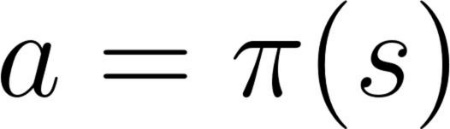
* Task 🡪 Instance of RL problem. 2 Types:
  + **Episodic**: Starting & ending point (terminal state).
    - E.g. Video game: task starts @ new Level, ends **when killed or reached the end of the level.**
  + **Continuing**: Task continues forever (no terminal state)



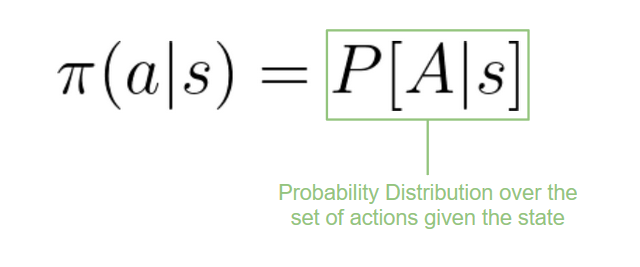
* Exploration / exploitation trade-off 🡪 Need rule to choose:

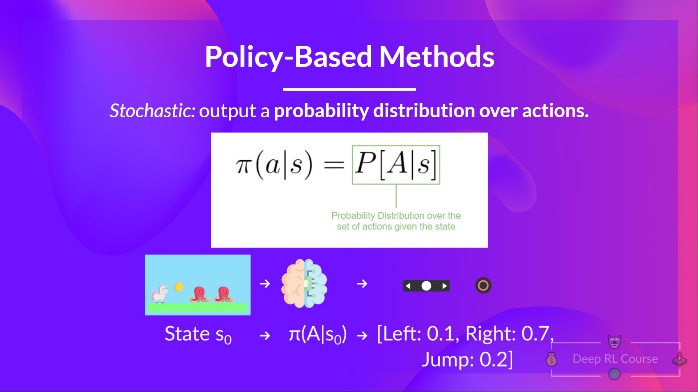


* The Policy π: Function that tells what action to take given the current state (brain of agent)
* Goal: find optimal policy π\* 🡪 **maximizes expected return** when agent acts according to it
* Find π\* 🡪 Training:
  + **Directly**: Teaching agent which action take, given current state: **Policy-based methods**
  + **Indirectly:** Teaching agent which state is more valuable 🡪 Action to reach more valuable states: **Value-based methods**
* **Policy-based methods:** Define mapping from each state 🡪 Best action / **probability distribution over the set of possible actions.**
  + **Deterministic:** Given state *s* 🡪 **always return the same action *a***

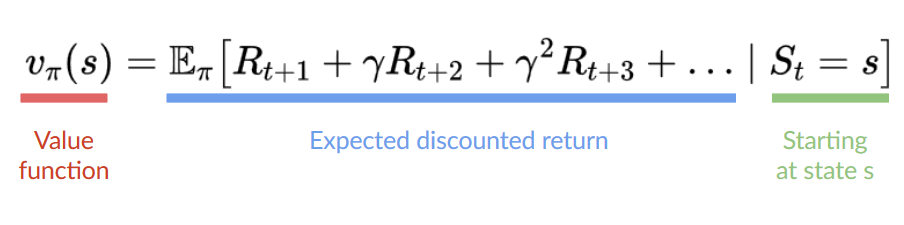


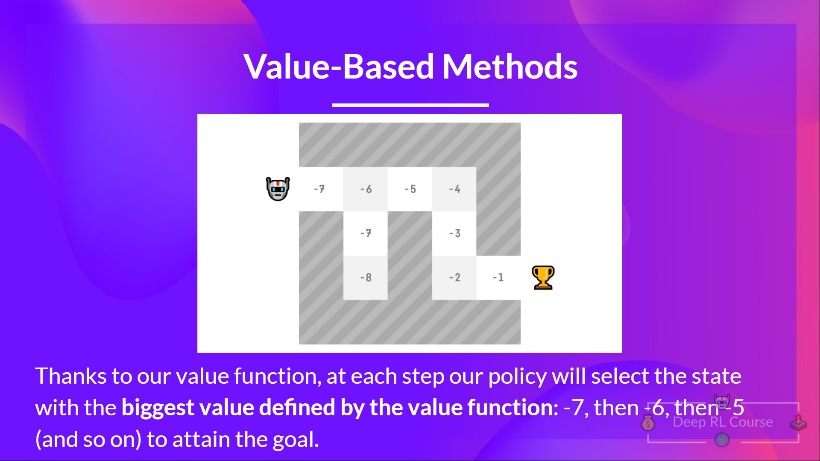
* + **Stochastic: probability distribution over actions**





* **Value-based methods:** Learn a value function 🡪 Maps state to expected value of being in that state.
  + Value of *state:* Expected discounted return agent can get if it starts in that *s* and acts according to policy.
  + “According to policy” 🡪 Policy going to state with higher value.





**The “deep” in RL:**

* Introduce neural networks for RL problems
* E.g. Q-learning vs. deep Q-learning: 