CS143: Artificial Intelligence

Markov Decision Processes (MDPs)

Drake UNIVERSITY

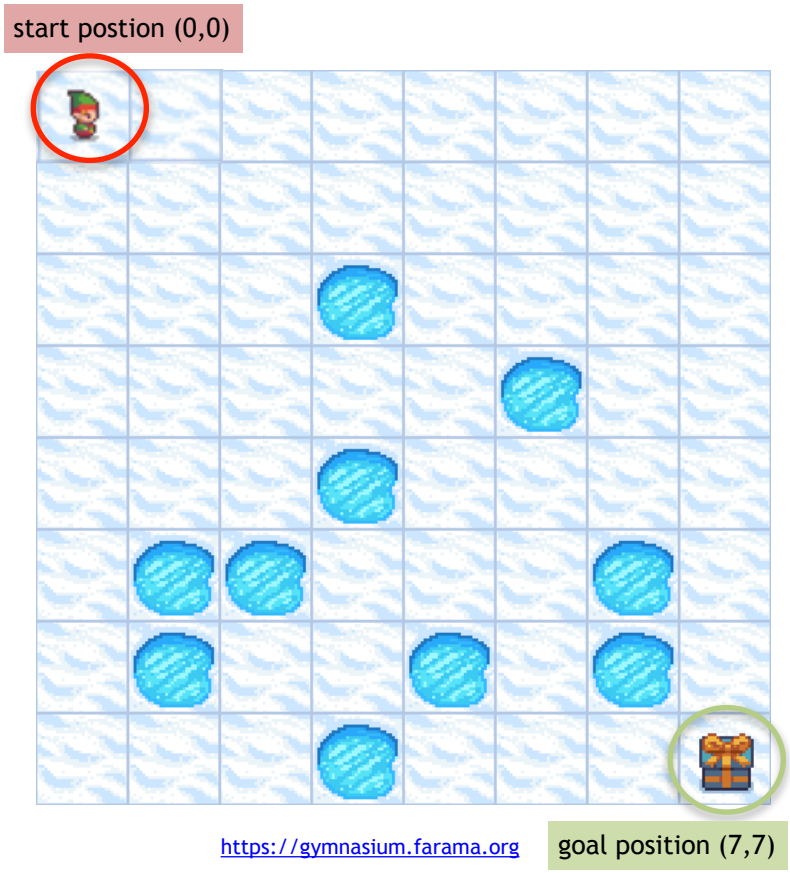
Frozen Lake in a Discrete Grid Space

- Consider an agent living in a frozen lake of 8x8 grid
  - Cross a frozen lake from start to goal without falling into holes
  - Walls block the agent's path

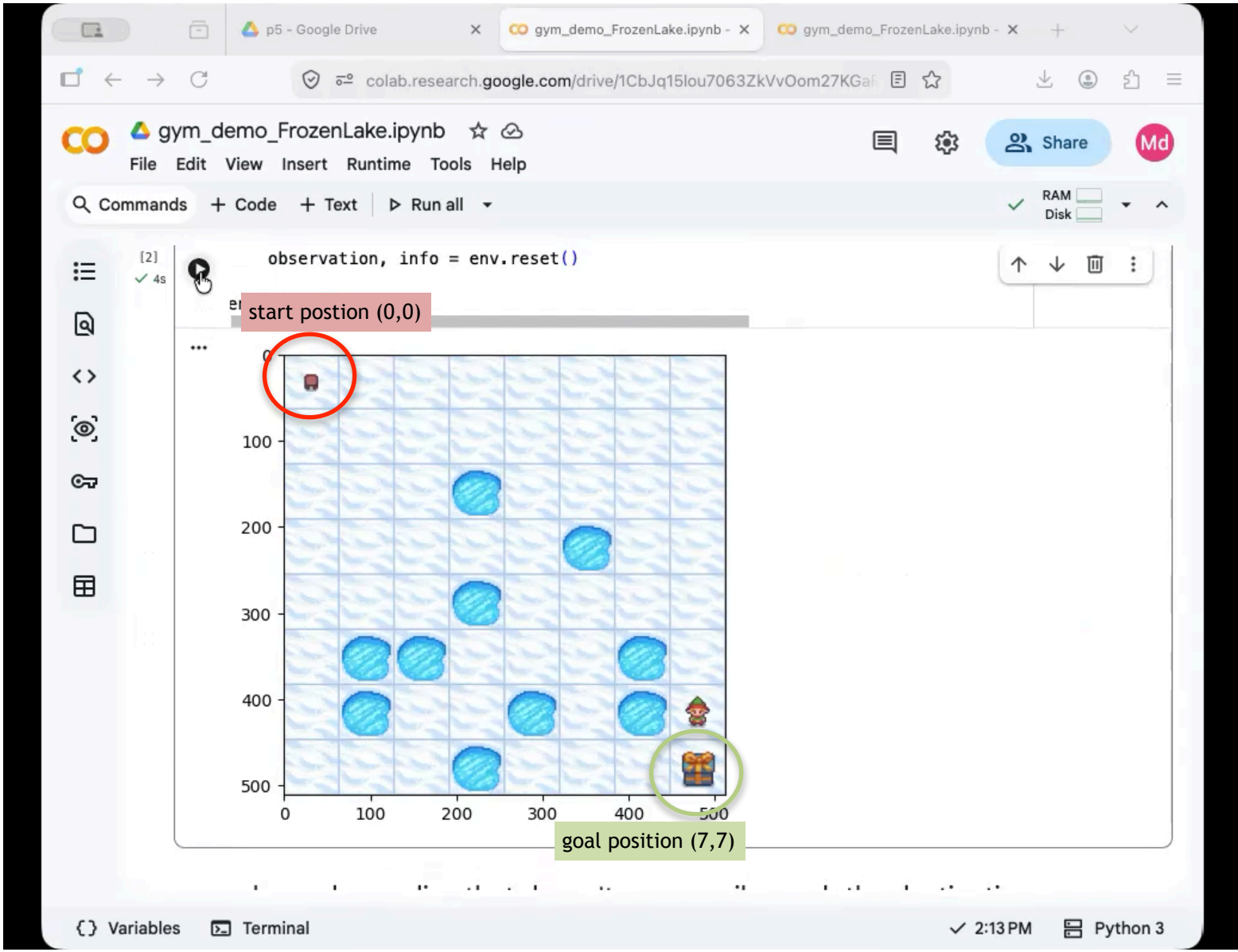
- Movements are stochastic (not fixed or deterministic in nature)
  - **Four actions:** left, down, right, up
  - The slippery lake can make the player slide in unintended directions

- Receives rewards at each time step
  - Positive rewards (less freq.): reach goal (+1)
  - Negative rewards (freq.): reach frozen tile (0)
  - Negative rewards (less freq.): reach hole (0)

- Goal is to maximize the sum of rewards

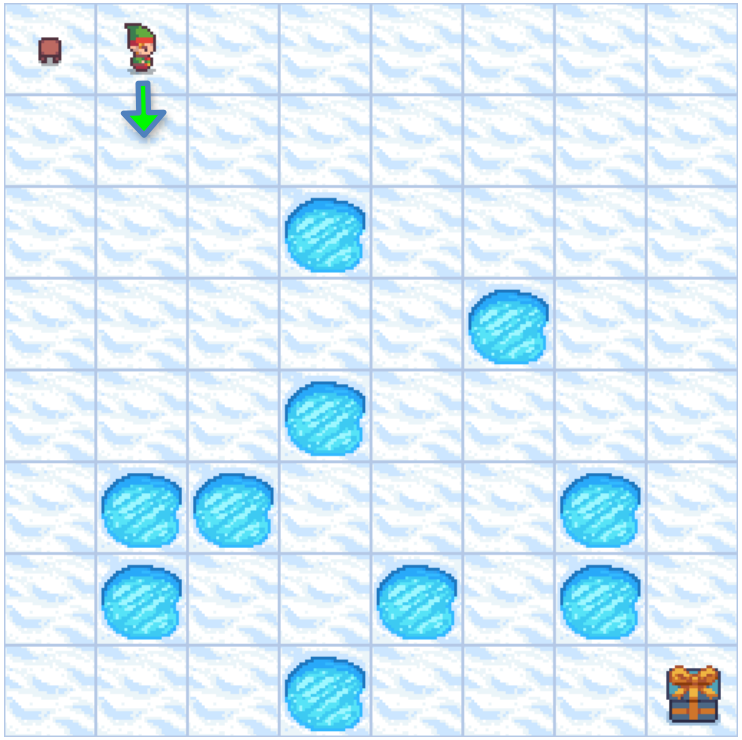


How to Reach the Goal?



Stochastic Actions in the Lake

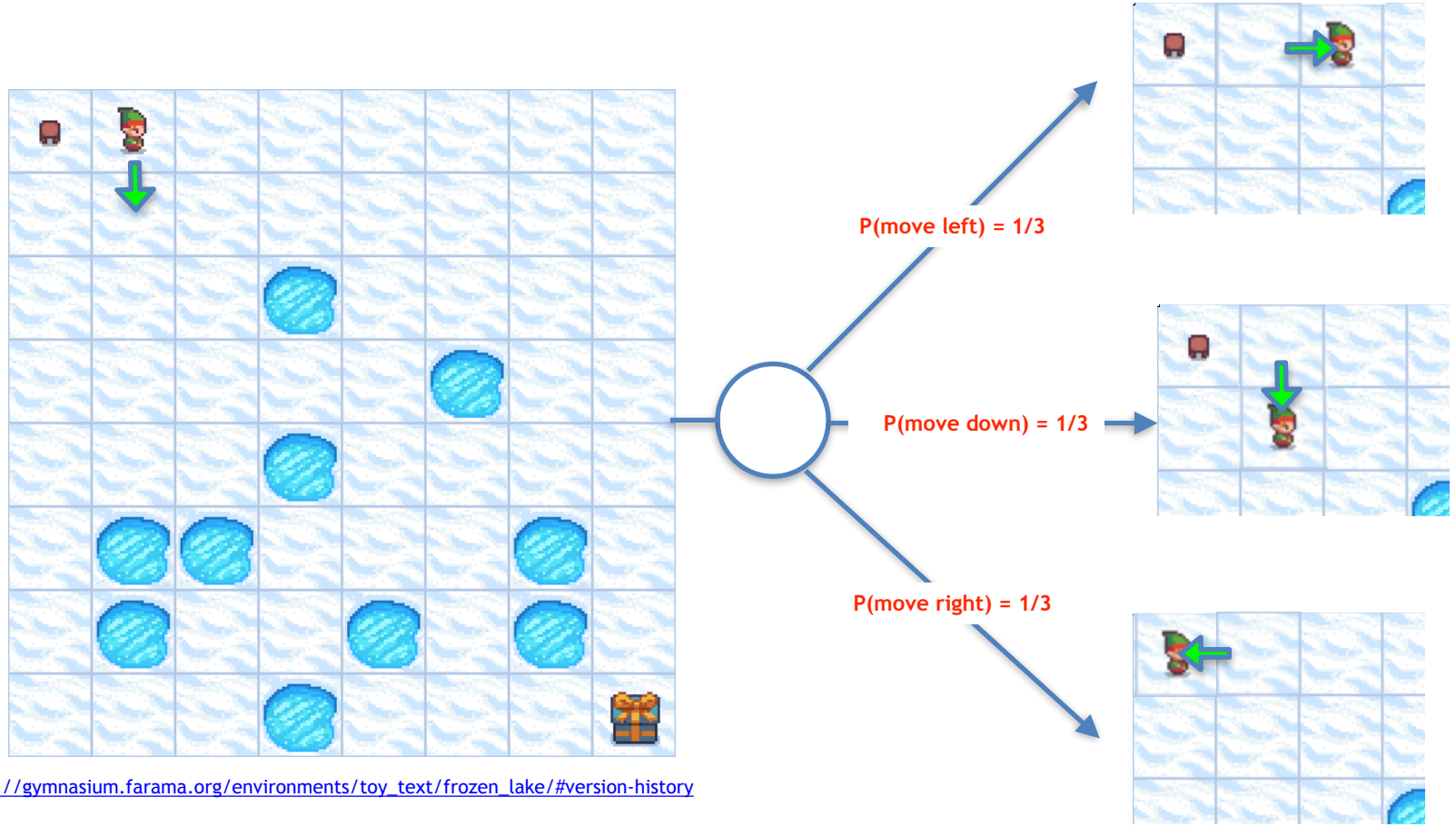
- Movements are stochastic (not fixed or deterministic in nature)
  - Agent moves in intended direction with probability 1/3
  - Also moves in either perpendicular direction with equal probability (1/3 and 1/3)



https://gymnasium.farama.org/environments/toy_text/frozen_lake/#version-history

Stochastic Actions in the Lake

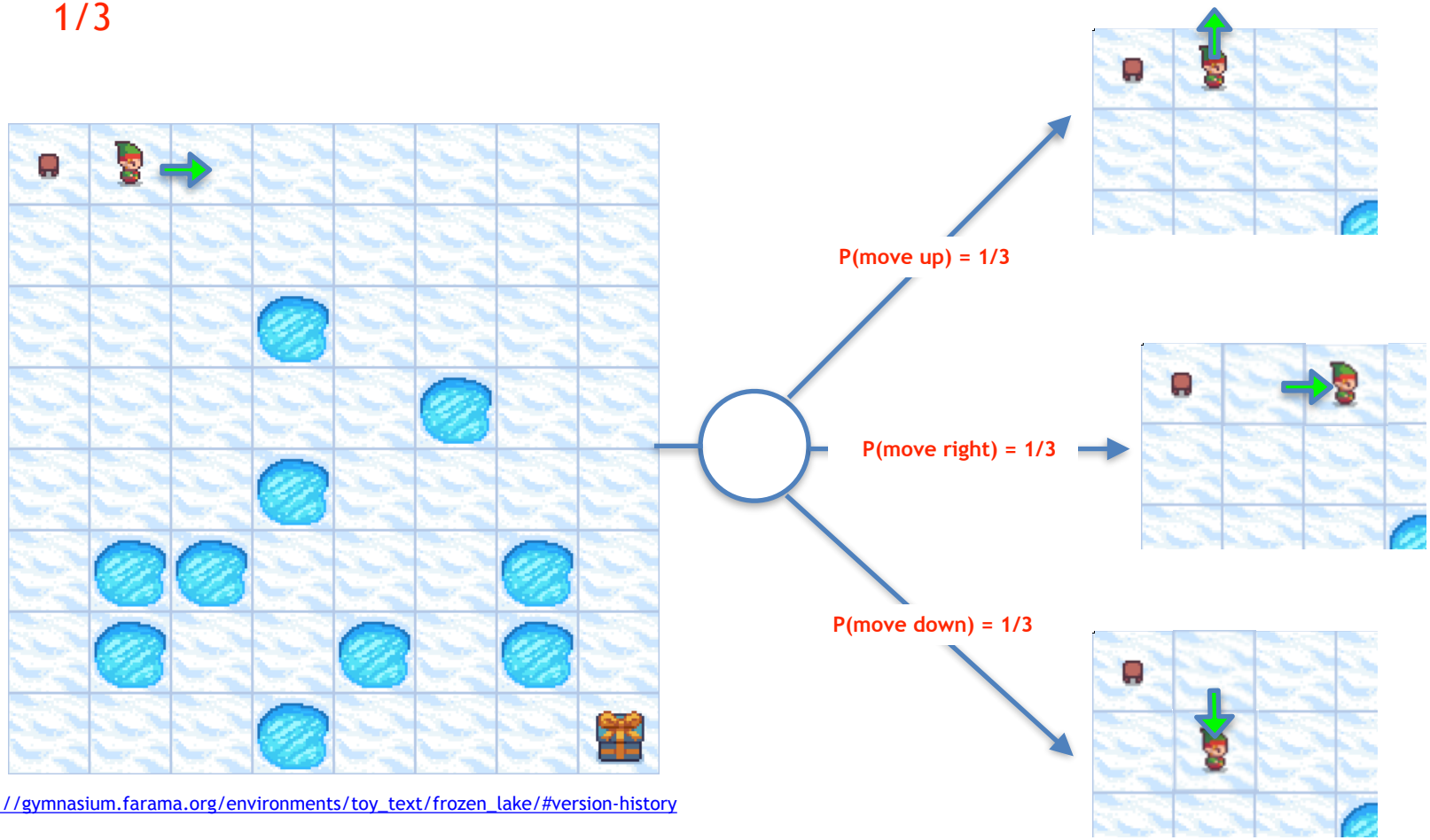
- Movements are stochastic (not fixed or deterministic in nature)
  - For example, agent intends to **move down** with probability 1/3
  - Also moves in either perpendicular directions: **move left** with probability 1/3 and **move right** with probability 1/3



https://gymnasium.farama.org/environments/toy_text/frozen_lake/#version-history

Stochastic Actions in the Lake

- Movements are stochastic (not fixed or deterministic in nature)
  - For example, agent intends to **move right** with probability 1/3
  - Also moves in either perpendicular directions: **move up** with probability 1/3 (actions that would take the agent off the grid leave its location unchanged) and **move down** with probability 1/3



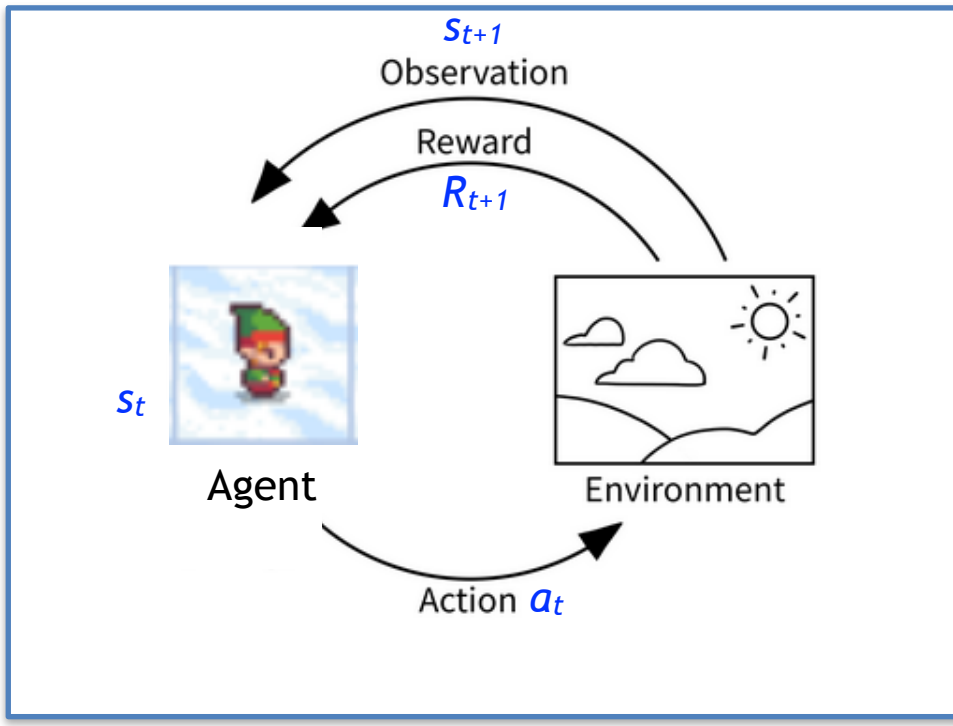
https://gymnasium.farama.org/environments/toy_text/frozen_lake/#version-history

Agent-Environment Interaction

- At each time step t, the agent receives some representation of the environment's state s_t and that basis selects an action a_t
- One time step later ($t+1$), the agent receives a numerical reward R_{t+1} as a consequence of selecting the action a_t and ***transitions*** to a new state s_{t+1}
- The agent and environment interactions gives rise to a trajectory:

$$\{s_0\ a_0\ R_1 \quad s_1\ a_1\ R_2 \quad s_2\ a_2\ R_3 \quad \ldots\ \}$$

- The tuples (s, a, R) are the basic unit of information describing a **reinforcement learning** process.



Markov Decision Process (MDP)

- Agent's actions affect the state it **transitions**/finds itself in.
- In this case, it needs to consider the estimated rewards in terms of the action it chooses and the current state it is in.
- Markov decision processes (MDPs) are appropriate formalizations of this more complex type of situation.
  - In other words, an MDP is the **mathematical formalization** or tool to express an Reinforcement Learning (RL) process.
  - In fact, in the presence of a transition function, an MDP can model any **sequential decision making process.** Reinforcement Learning (RL) is just a sequential decision making process.

What is the significance of term '**Markov**'?

- The agent and environment interactions gives rise to a trajectory:

$\{s_0\ a_0\ R_1 \quad s_1\ a_1\ R_2 \quad s_2\ a_2\ R_3 \ \ldots.\}$

- To make the environment transition function more practical, we add a simplifying assumption that transition to the next state s' only depends on the previous state s and the action a
  - this simplifying property is known as '*Markov Property*'

$$P(s_{t+1} \mid (s_0, a_0), (s_1, a_1), (s_2, a_2), \ldots, (s_t, a_t))$$

$$= P(s_{t+1} \mid (s_0, a_0), (s_1, a_1), (s_2, a_2), \ldots, (s_t, a_t)) \quad \text{'Markov Property'}$$

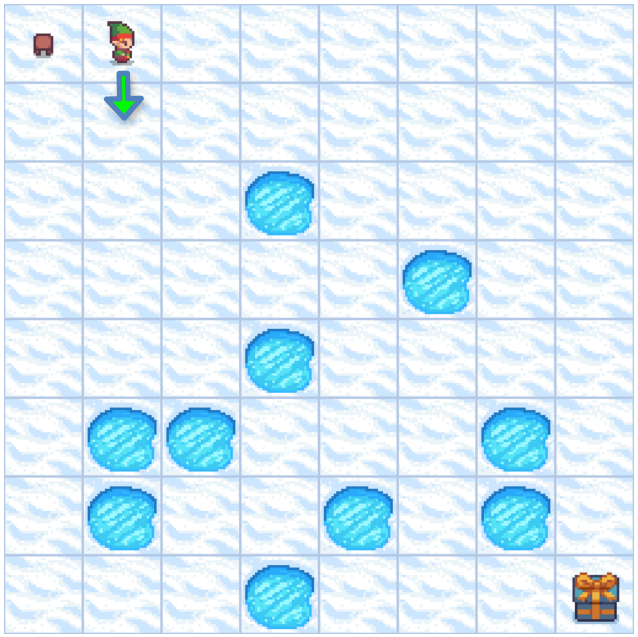
$$\approx P(s_{t+1} \mid s_t, a_t)$$

Other example: Fibonacci function uses Markov property

Let's denote this term (from the next slide) as $P(s' \mid a, s)$

Markov Decision Process (MDP)

- An MDP is defined by the tuple (S, A, R, P)
  - A set of states $s \in S$
  - A set of actions $a \in A$
  - A reward function $R(s, a, s')$
  - A transition function encoding the model or dynamics $P(s' \mid a, s)$



FrozenLake: crossing a frozen lake from start to goal without falling into any holes by walking over the frozen lake.

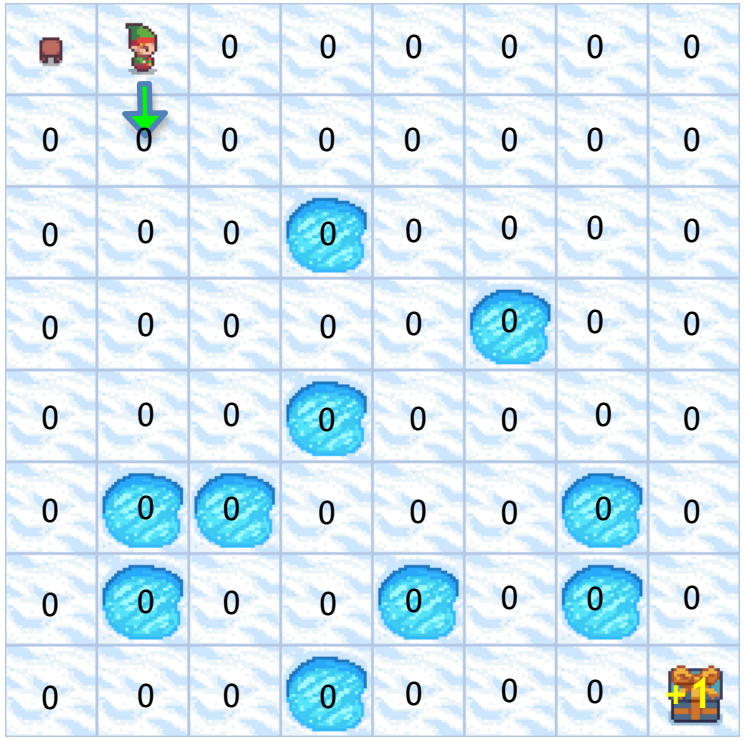
*P(s'|a,s) notation is sometimes substituted with T(s,a,s') in order to denote the transition function. Let's stick to P(s'|a,s) notation as we will be dealing with probabilities into some equations.

MDP for Frozen Lake

- The states are description of the world
  - In Frozen Lake, the state space is 8x8 (2D list) or 64 (1D list). It's fully observable (see the entire content all at once)
  - Or a much simpler state representation could be agent's current position: linear index or (row_index, column_index)
- The actions are the finite number of possible actions (in discrete case)
  - In Frozen Lake, agent can take four actions:
    - move up 
    - move left 
    - move right 
    - move down 

MDP for Frozen Lake

- The reward function encodes the reward agent receives
  - In Frozen Lake, reward function is known
  - $R(s, a, s')$ is the amount of reward the agent receives when it transitions from state s to s' by taking action a
    - reach goal (+1)
    - reach frozen tile (0)
    - reach hole (0)

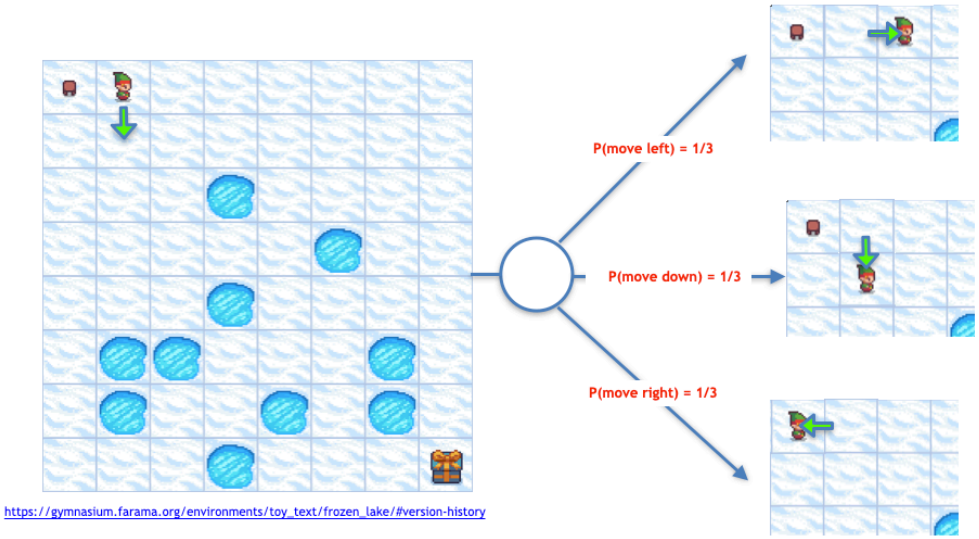


MDP for Frozen Lake

- The transition function encodes the movements in the Lake
  - $P(s'|a,s)$ specify the probability of ending up in state s' if action a is taken (by agent) from state s
  - It should satisfy the probability rule:

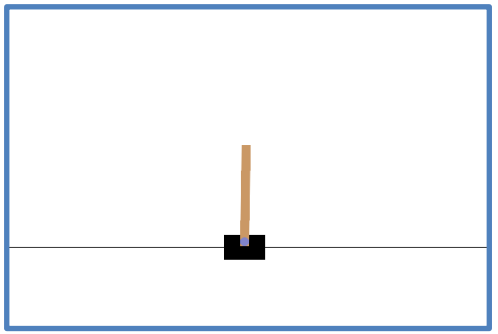
$$\sum_{s' \in states} p(s'|s,a) = 1$$

  - In Frozen Lake, model or dynamics is known
    - Agent moves in intended direction with probability 1/3
    - It also moves in either perpendicular direction with equal probability (1/3 and 1/3)



More Example: MDP for CartPole

- An MDP is defined by the tuple (S, A, R, P)
  - A set of states $s \in S$
  - A set of actions $a \in A$
  - A reward function $R(s, a, s')$
  - A transition function $P(s' \mid a, s)$



CartPole: Keep the pole upright for 200 time steps

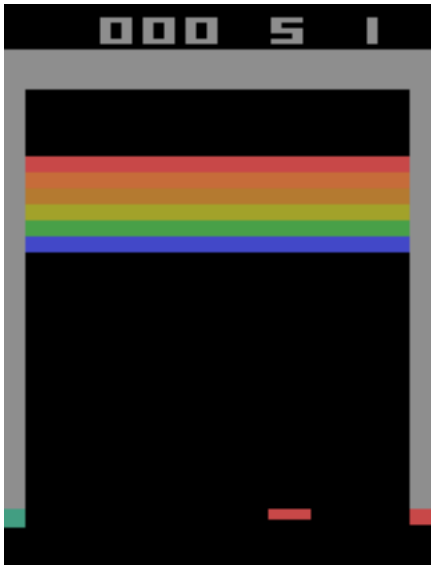
- CartPole MDP:
  - $s \in S$: *An array of of length 4 which represents*

    [cart position, cart velocity, pole angle, pole angular velocity]
  - $a \in A$: *an integer, either 0 to move the cart a fixed distance to the left, or 1 to move the cart a fixed distance to the right*
  - $R(s, a, s')$: *+1 for every time step the pole remains upright, 0 for all other time steps*

https://gymnasium.farama.org/environments/classic_control/cart_pole/

More Example: MDP for Atari Breakout

- An MDP is defined by the tuple (S, A, R, P)
  - A set of states $s \in S$
  - A set of actions $a \in A$
  - A reward function $R(s, a, s')$
  - A transition function $P(s' \mid a, s)$



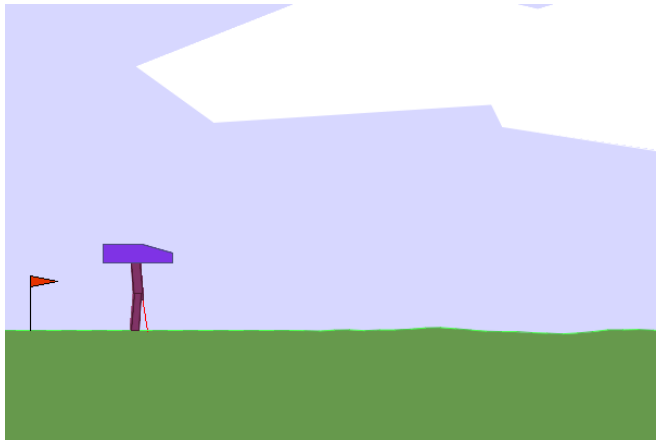
Atari Breakout: Not let the ball drop in the game

- Atari Breakout MDP:
  - $s \in S$: *An RGB image with 160x210 pixels resolution (what we see on the screen)*
  - $a \in A$: *An integer from the set [0,1,2,3] which maps to the game controller actions*
    [no-action, launch the ball, move right, move left]
  - $R(s, a, s')$: *Game score difference between consecutive states*

https://ale.farama.org/environments/breakout/

More Example: MDP for Atari Bipedal Walker

- An MDP is defined by the tuple (S, A, R, P)
  - A set of states $s \in S$
  - A set of actions $a \in A$
  - A reward function $R(s, a, s')$
  - A transition function $P(s' \mid a, s)$



Bipedal Walker: walk to the right without falling

- Atari Breakout MDP:
  - $s \in S$:

    An array of length 24 representing various statistics about the agent

    [hull angle, hull angular velocity, x-velocity, y-velocity, hip 1 joint angle, hip 1 joint speed, Knee 1 joint angle, knee 1 joint speed, leg 1 ground contact, hip 2 joint angle, hip 2 joint speed, Knee 2 joint angle, knee 2 joint speed, leg 2 ground contact, …, 10 lidar readings]
  - $a \in A$:

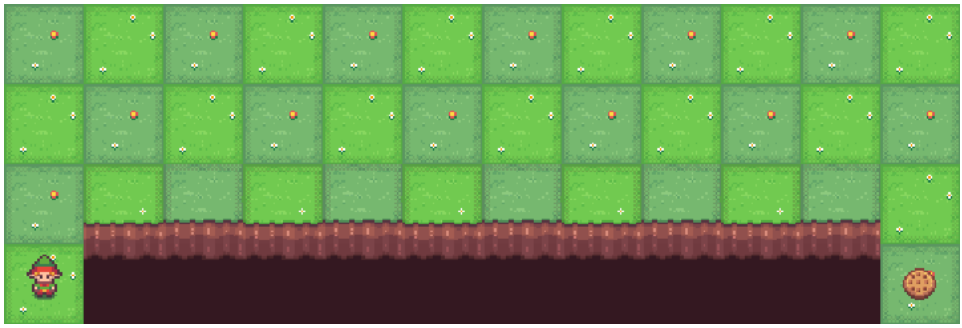
    A vector of 4 floating point numbers each in the interval [-1, +1] which represents [hip$_1$ torque and velocity, knee$_1$ torque and velocity, hip$_2$ torque and velocity, knee$_2$ torque and velocity]
  - $R(s, a, s')$:

    Reward for moving forward to the right, up to maximum +300. Reward -100 if the agent falls. Additionally, there is a small negative reward (movement cost) at every time step proportional to the torque applied

https://gymnasium.farama.org/environments/box2d/bipedal_walker/

More Example: MDP for Cliff Walking

- An MDP is defined by the tuple (S, A, R, P)
  - A set of states $s \in S$
  - A set of actions $a \in A$
  - A reward function $R(s, a, s')$
  - A transition function $P(s' \mid a, s)$



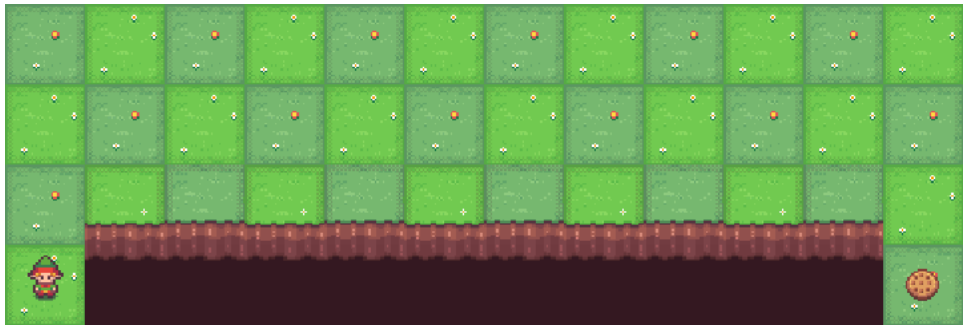
Cliff Walking: crossing a grid world from start to goal while avoiding falling off a cliff

- CartPole MDP:
  - $s \in S$: *An array of of length 4x12 cells which represents state of the map*
  - $a \in A$: *An integer from the set [0,1,2,3] which maps to the game controller actions*
    [up, right, down, left]
  - $R(s, a, s')$: *Each time step incurs -1 reward, unless the player stepped into the cliff, which incurs -100 reward.*

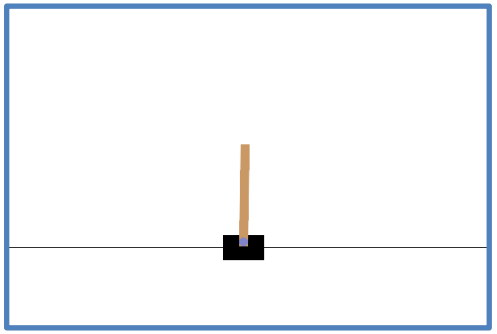
https://gymnasium.farama.org/environments/toy_text/cliff_walking/

Objective of an Agent in RL

- How to formalize the objective of the agent which it intends to **maximize**?

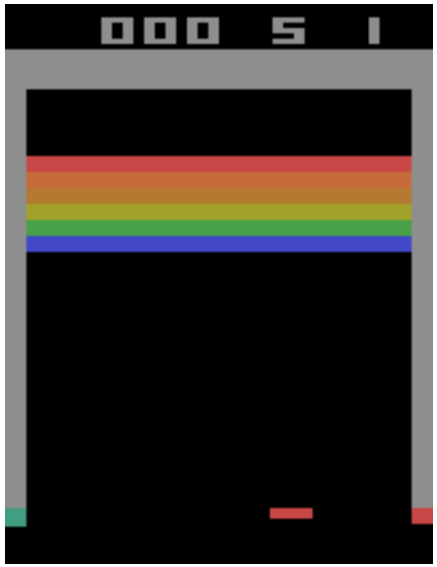


Cliff Walking objective: *crossing a grid world from start to goal while avoiding falling off a cliff*

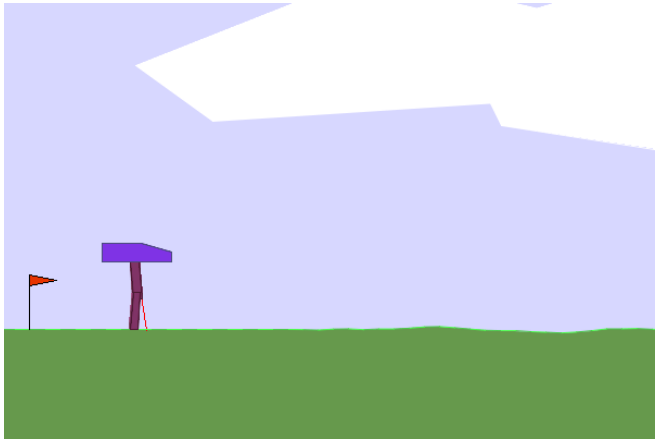


CartPole objective: *Keep the pole upright for 200 time steps*

FrozenLake objective: *Crossing a frozen lake from start to goal without falling into any holes by walking over the frozen lake*



Atari Breakout objective: *Not let the ball drop*



Bipedal Walker objective: *Walk to the right without falling*

Objective of an Agent in RL

- How to formalize the objective of the agent which it intends to **maximize**?

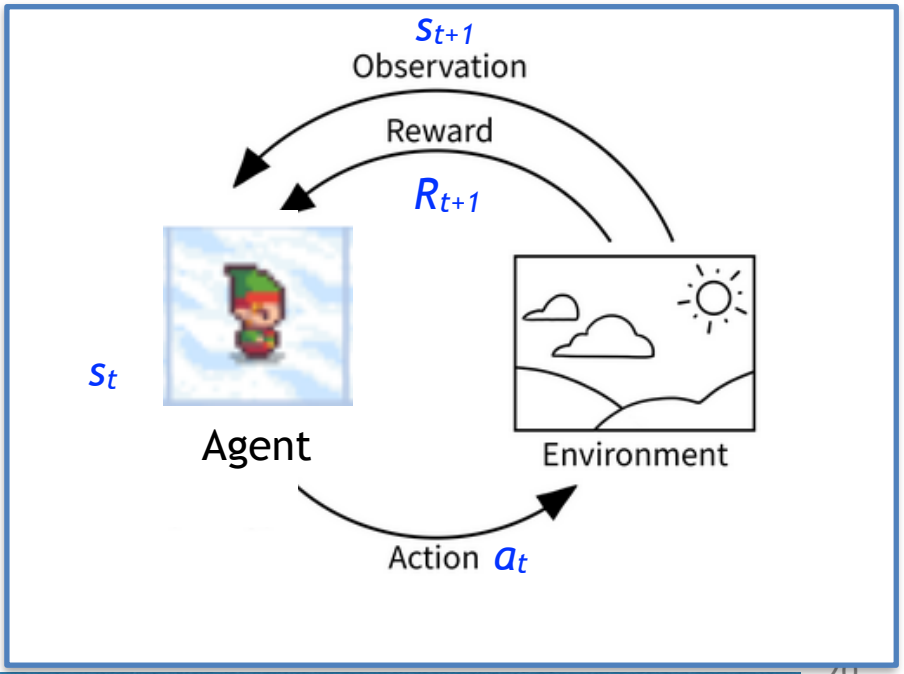
Objective#1: No Discounting

- The agent and environment interactions gives rise to a trajectory. Let's define a trajectory until time step $\underline{T}$ as an episode:

τ episode: $\{s_0\, a_0\, R_1 \quad s_1\, a_1\, R_2 \quad s_2\, a_2\, R_3 \quad \ldots \quad s_T\, a_T\, R_T\}$

- The sum of rewards on this episode is defined as follows:

$$G(\tau) = R_0 + R_1 + R_2 + R_3 + R_T$$



Objective#1: No Discounting

- The agent and environment interactions gives rise to a trajectory. Let's define *M* trajectories (*episodes*) each with *T* time steps:

τ_1 episode^1: $\{s^1_0\, a^1_0\, R^1_1 \quad s^1_1\, a^1_1\, R^1_2 \quad s^1_2\, a^1_2\, R^1_3 \ \ldots\ s^1_T\, a^1_T\, R^1_T\}$

τ_2 episode^2: $\{s^2_0\, a^2_0\, R^2_1 \quad s^2_1\, a^2_1\, R^2_2 \quad s^2_2\, a^2_2\, R^2_3 \ \ldots\ s^2_T\, a^2_T\, R^2_T\}$

. . .

τ_M episode^M: $\{s^M_0\, a^M_0\, R^M_1 \quad s^M_1\, a^M_1\, R^M_2 \quad s^M_2\, a^M_2\, R^M_3 \ \ldots\ s^M_T\, a^M_T\, R^M_T\}$

- The sum of rewards on each of these *M* episodes are defined as follows:

$$G(\tau_1) = R^1_0 + R^1_1 + R^1_2 + R^1_3 + \ldots + R^1_T$$
$$G(\tau_2) = R^2_0 + R^2_1 + R^2_2 + R^2_3 + \ldots + R^2_T$$
$$\ldots = \ldots$$
$$G(\tau_2) = R^M_0 + R^M_1 + R^M_2 + R^M_3 + \ldots + R^M_T$$

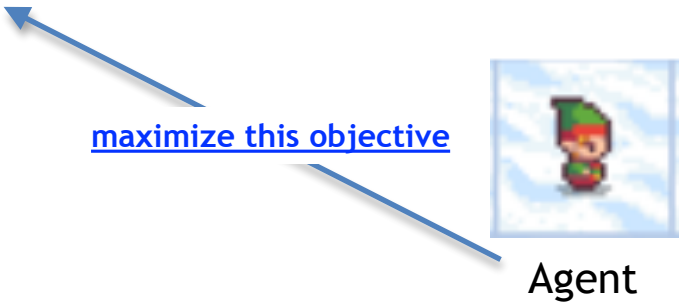
Objective#1: No Discounting

- In RL, the objective is to **maximize** the average rewards *H(.)* over these *M* different episodes is defined as follows:

$$H(\tau) = \frac{1}{M}(G(\tau_1) + G(\tau_2) + G(\tau_3) + \ldots + G(\tau_M))$$

- A simplifying notation of this objective using expectation symbol is as follows:

$$H(\tau) = \mathbb{E}_{\tau \sim \pi}[G(\tau)]$$



Objective of an Agent in RL

- How to formalize the objective of the agent which it intends to **maximize**?

Agent

Objective#1: without discounted rewards

Objective#2: with discounted rewards

discounting is to say that a reward earned in the future should be less than what it would be worth if we received it immediately

Objective#2: With Discounting

- The agent and environment interactions gives rise to a trajectory. Let's define a trajectory until time step T as an episode:

τ episode: $\{s_0\ a_0\ R_1 \quad s_1\ a_1\ R_2 \quad s_2\ a_2\ R_3 \ \ldots\ s_T\ a_T\ R_T\}$

- The sum of discounted rewards on this episode is defined as follows:

$$G(\tau) = R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 R_3 + \gamma^T R_T$$

  - discount rate (gamma) is a parameter which can take a value between 0 and 1
  - The idea behind the discounting is to decay the future rewards exponentially
  - In other words, discounting is to say that a reward earned in the future should be less than what it would be worth if we received it immediately

Objective#2: With Discounting

- The agent and environment interactions gives rise to a trajectory. Let's define a trajectory until time step T as an episode:

τ episode: $\{s_0\ a_0\ R_1 \quad s_1\ a_1\ R_2 \quad s_2\ a_2\ R_3 \ \ldots\ s_T\ a_T\ R_T\}$

- The sum of discounted rewards on this episode is defined as follows:

$$G(\tau) = R_0 + \gamma R_1 + \gamma^2 R_2 + \gamma^3 R_3 + \gamma^T R_T$$

  - discount rate = 0 (live in the moment)
  - discount rate = 1 (save for the future)
  - discount rate = 0.5 (balanced life)

Objective#2: With Discounting

- The agent and environment interactions gives rise to a trajectory. Let's define a trajectory until time step $\underline{T}$ as an episode:

τ_1 episode^1: $\{s^1_0\, a^1_0\, R^1_1 \quad s^1_1\, a^1_1\, R^1_2 \quad s^1_2\, a^1_2\, R^1_3 \;\ldots.\; s^1_T\, a^1_T\, R^1_T\}$

τ_2 episode^2: $\{s^2_0\, a^2_0\, R^2_1 \quad s^2_1\, a^2_1\, R^2_2 \quad s^2_2\, a^2_2\, R^2_3 \;\ldots.\; s^2_T\, a^2_T\, R^2_T\}$

$\ldots$

τ_M episode^M: $\{s^M_0\, a^M_0\, R^M_1 \quad s^M_1\, a^M_1\, R^M_2 \quad s^M_2\, a^M_2\, R^M_3 \;\ldots.\; s^M_T\, a^M_T\, R^M_T\}$

- Incorporating the discounting rate, the sum of rewards on each of these $\underline{M}$ episodes are defined as follows:

$$G(\tau_1) = R^1_0 + \gamma^1 R^1_1 + \gamma^2 R^1_2 + \gamma^3 R^1_3 + \ldots + \gamma^T R^1_T$$
$$G(\tau_2) = R^2_0 + \gamma^1 R^2_1 + \gamma^2 R^2_2 + \gamma^3 R^2_3 + \ldots + \gamma^T R^2_T$$
$$\ldots = \ldots$$
$$G(\tau_2) = R^M_0 + \gamma^1 R^M_1 + \gamma^2 R^M_2 + \gamma^3 R^M_3 + \ldots + \gamma^T R^M_T$$

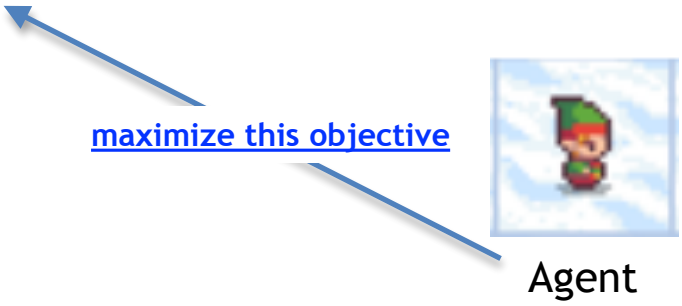
Objective#2: With Discounting

- In RL, the objective is to **maximize** the average rewards *H(.)* over these *M* different episodes is defined as follows:

$$H(\tau) = \frac{1}{M}(G(\tau_1) + G(\tau_2) + G(\tau_3) + \ldots + G(\tau_M))$$

- A simplifying notation of this objective using expectation symbol is as follows:

$$H(\tau) = \mathbb{E}_{\tau \sim \pi}[G(\tau)]$$



MDP Control Loop

Given an environment `env` in Gymnasium platform and an agent `agent`

```
1.  for episode=1,2,…,M do
2.      state = env.reset()
3.      agent.reset()
4.      for t=0,2,…,T do
5.          action = agent.act(state)
6.          next_state, reward = env.step(action)
7.          agent.update(action, next_state, reward)
8.          state = next_state
9.          if env.done() then
10.             break
```

Agent

different reinforcement learning approaches are applied inside this step

Coding Activity: Hands-On Experience with the Gymnasium Framework

- Jump to Blackboard, and let's explore the components of an MDP through a concrete problem using Gymnasium
- Let's familiarize ourselves with Gymnasium.