Fundamentals of reinforcement learning

REINFORCEMENT LEARNING WITH GYMNASIUM IN PYTHON

Fouad Trad
Machine Learning Engineer

datacamp

Reinforcement learning

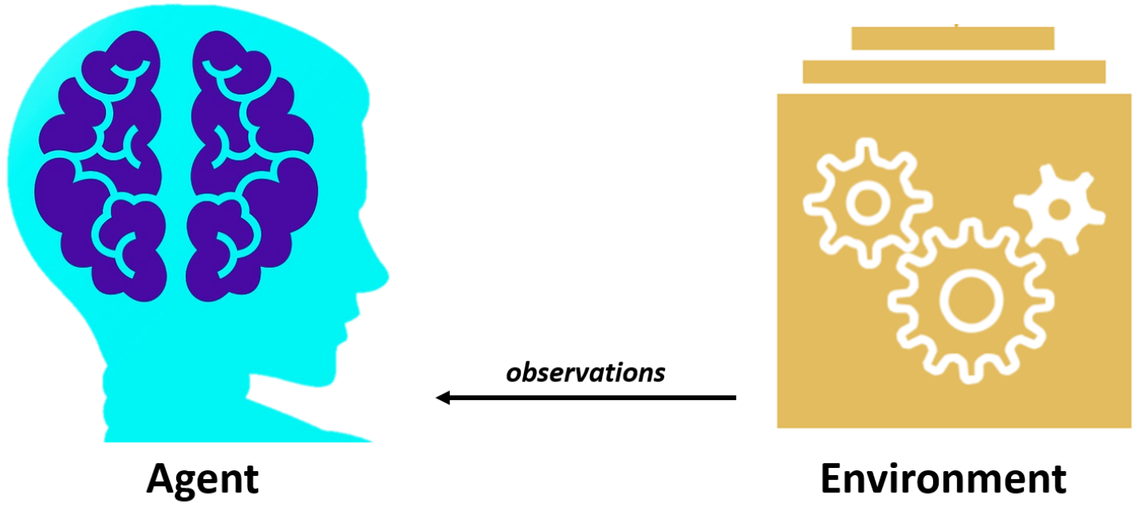
- Agent learns through trial and error

Agent

Environment

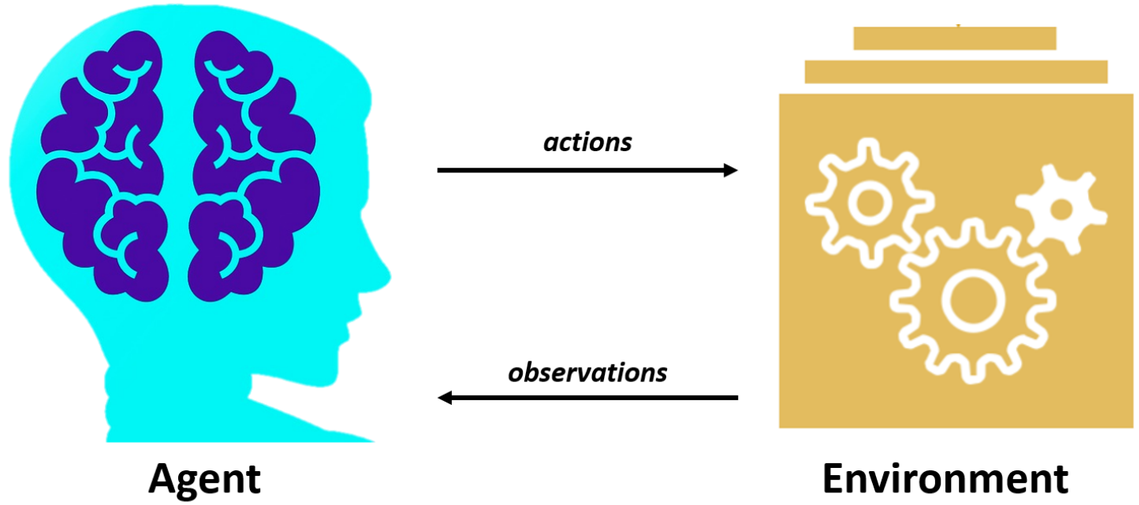
Reinforcement learning

- Agent learns through trial and error



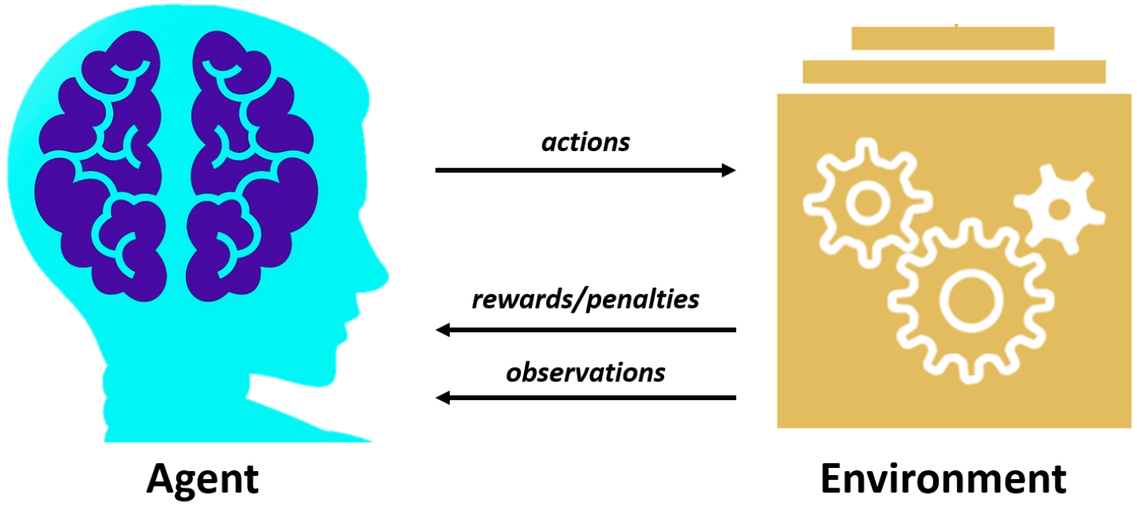
Reinforcement learning

- Agent learns through trial and error



Reinforcement learning

- Agent learns through trial and error
- Agent receives:
  - Rewards for good decisions
  - Penalties for bad decisions
- **Goal**: maximize positive feedback over time



RL as training a pet

RL vs. other ML types

	Supervised Learning
Data Type	Labeled data
Main Objective	Predict outcomes based on input data
Suitability	Classification, regression

RL vs. other ML types

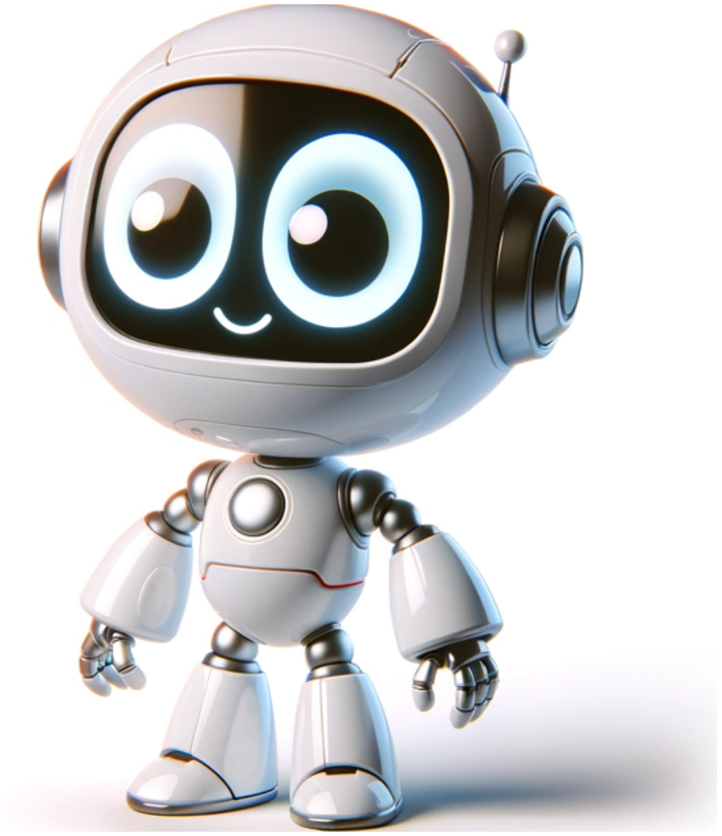
	Supervised Learning	*Unsupervised Learning*
Data Type	Labeled data	Unlabeled data
Main Objective	Predict outcomes based on input data	Discover underlying patterns or associations in data
Suitability	Classification, regression	Clustering, association analysis

RL vs. other ML types

	Supervised Learning	*Unsupervised Learning*	*Reinforcement Learning (RL)*
Data Type	Labeled data	Unlabeled data	No predefined training data
Main Objective	Predict outcomes based on input data	Discover underlying patterns or associations in data	Make decisions that maximize reward from the environment
Suitability	Classification, regression	Clustering, association analysis	Decision-making tasks

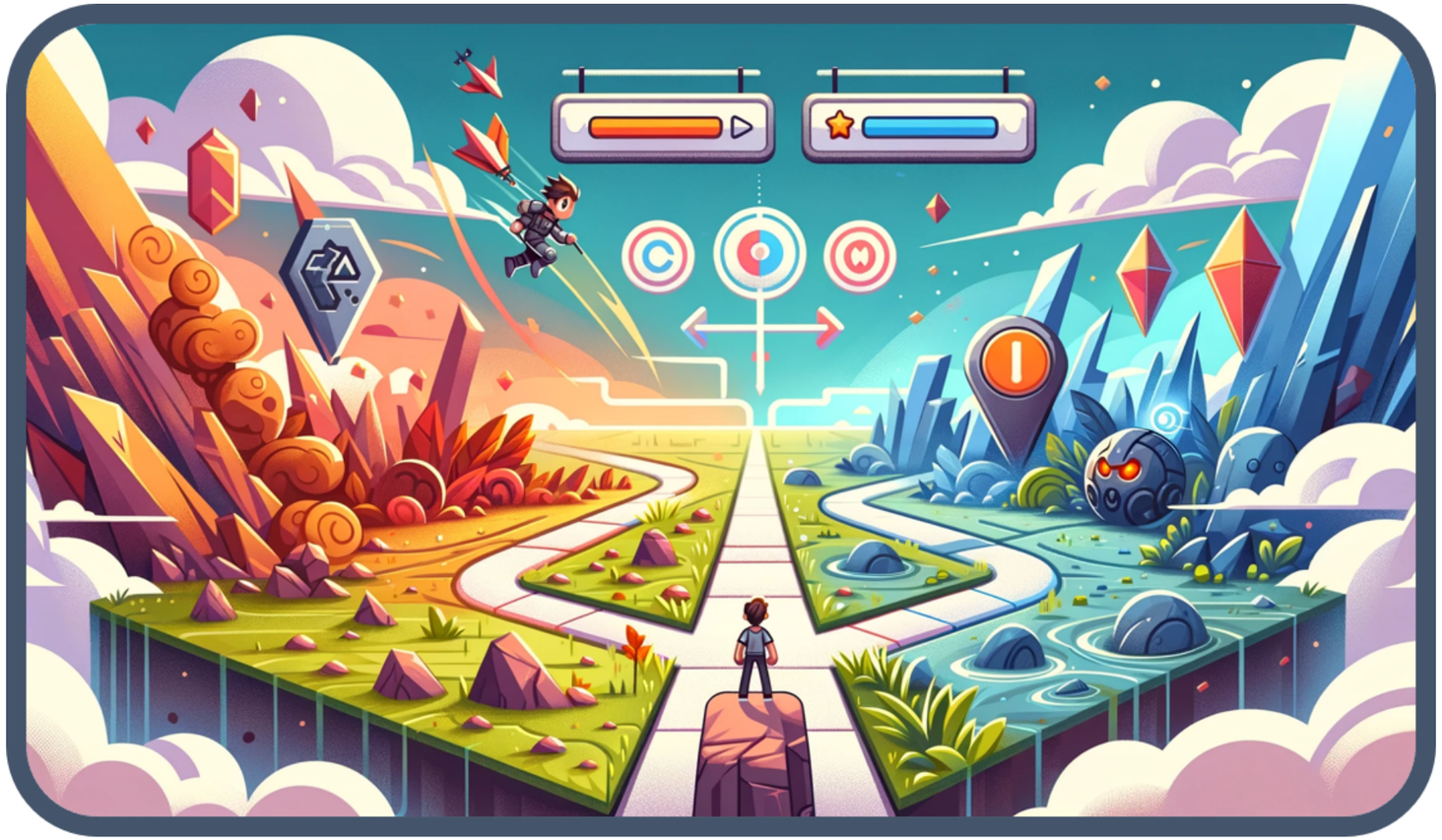
When to use RL?

- Sequential decision-making
  - Decisions influence future observations
- Learning through rewards and penalties
  - No direct supervision



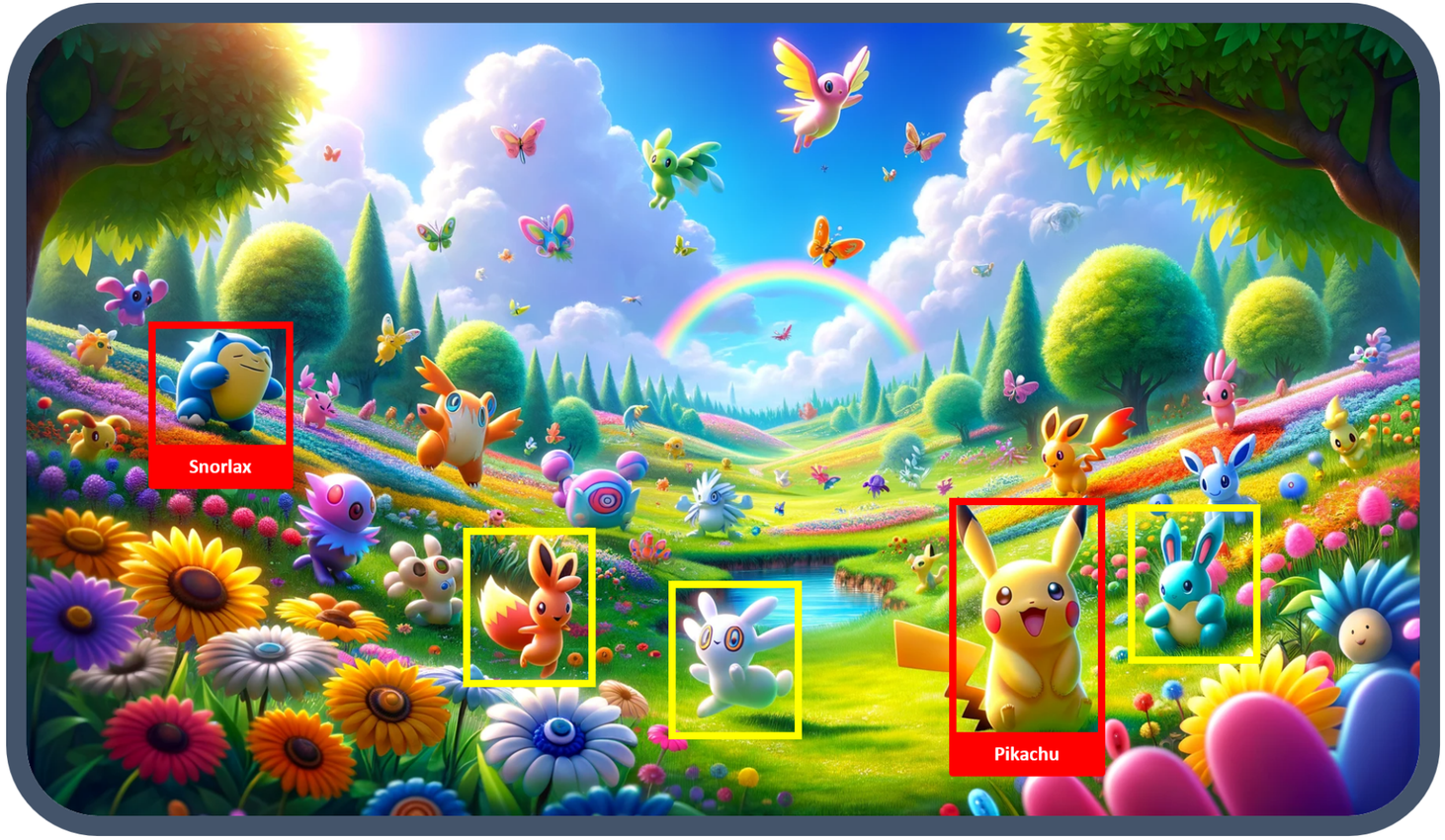
Appropriate for RL: playing video games

- Player makes sequential decisions
- Receives points and loses lives depending on actions



Inappropriate for RL: in-game object recognition

- No sequential decision-making
- No interaction with an environment



RL applications

Robotics

- Robot walking
- Object manipulation

RL applications

Robotics

- Robot walking
- Object manipulation

Finance

- Optimizing trading and investment
- Maximize profit

RL applications

Autonomous Vehicles

- Enhancing safety and efficiency
- Minimizing accident risks

RL applications

Autonomous Vehicles

- Enhancing safety and efficiency
- Minimizing accident risks

Chatbot development

- Enhancing conversational skills
- Improving user experiences

What's next?

In this course we will:

- Understand RL foundations and principles
- Identify, frame, and solve RL problems
- Application with Gymnasium

Let's practice!

REINFORCEMENT LEARNING WITH GYMNASIUM IN PYTHON

Navigating the RL framework

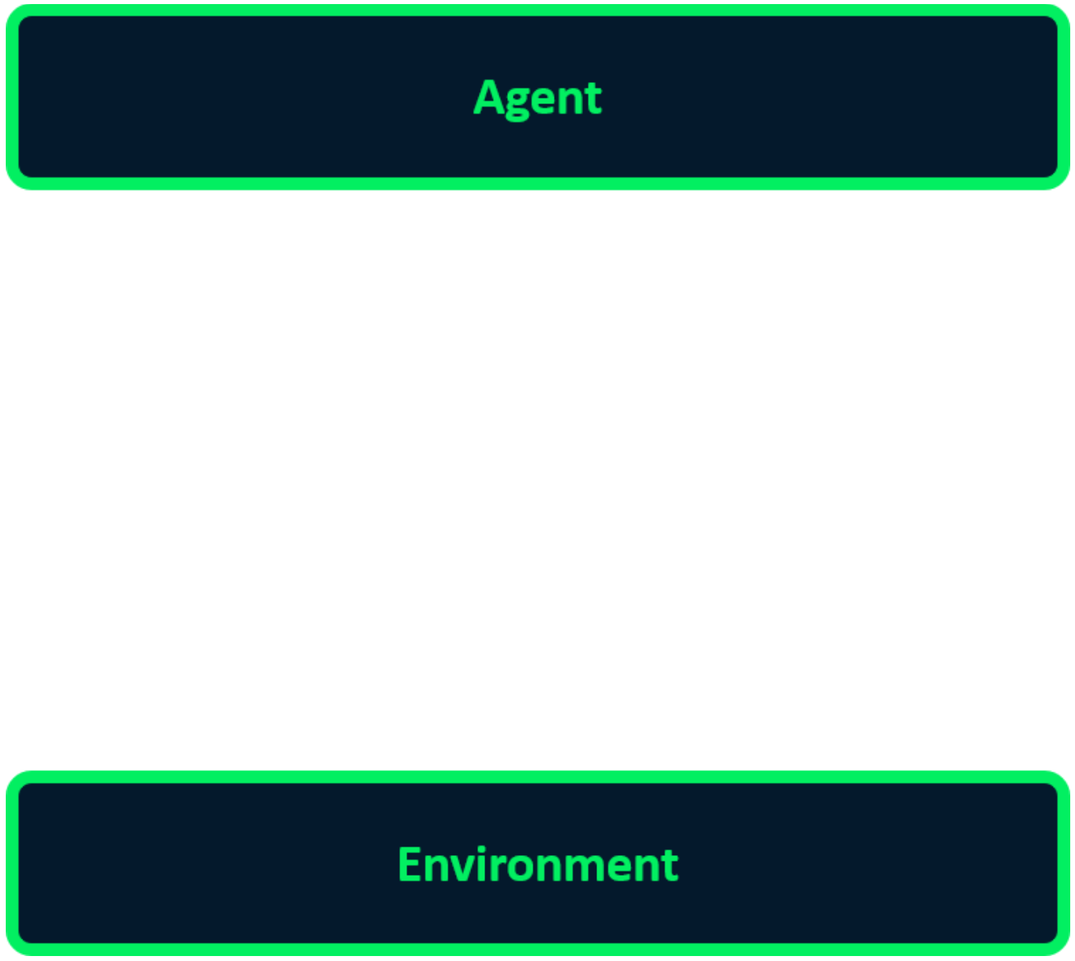
REINFORCEMENT LEARNING WITH GYMNASIUM IN PYTHON

Fouad Trad
Machine Learning Engineer

datacamp

RL framework

RL framework

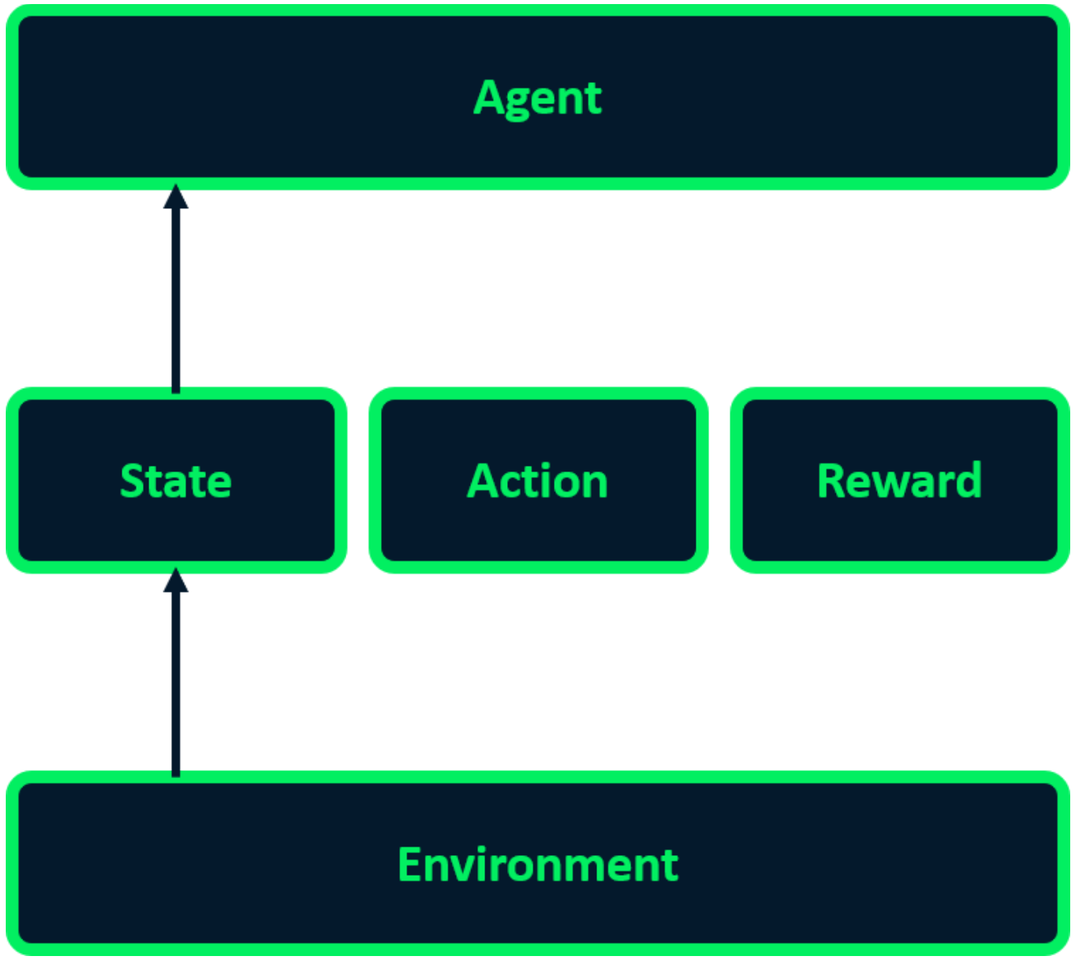


RL framework

- **Agent:** learner, decision-maker
- **Environment:** challenges to be solved

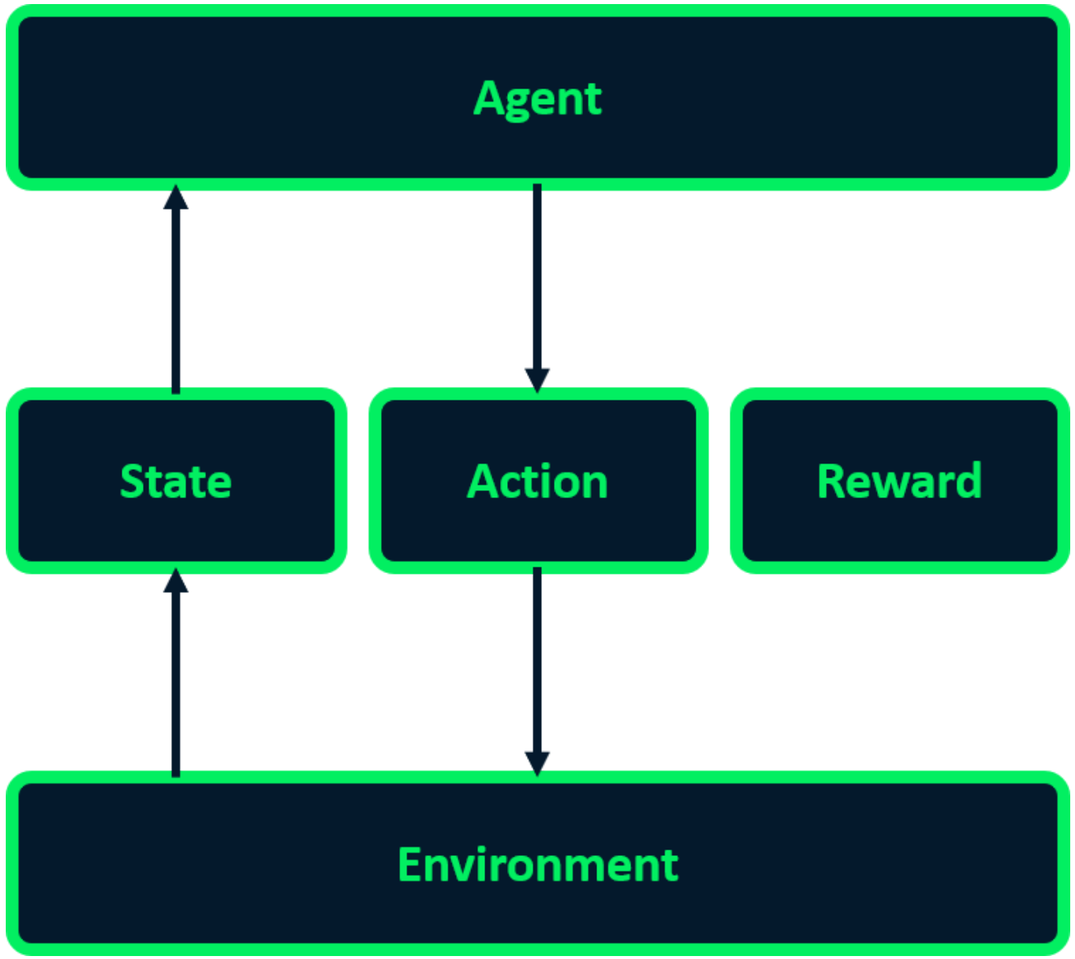
RL framework

- **Agent:** learner, decision-maker
- **Environment:** challenges to be solved
- **State:** environment snapshot at given time



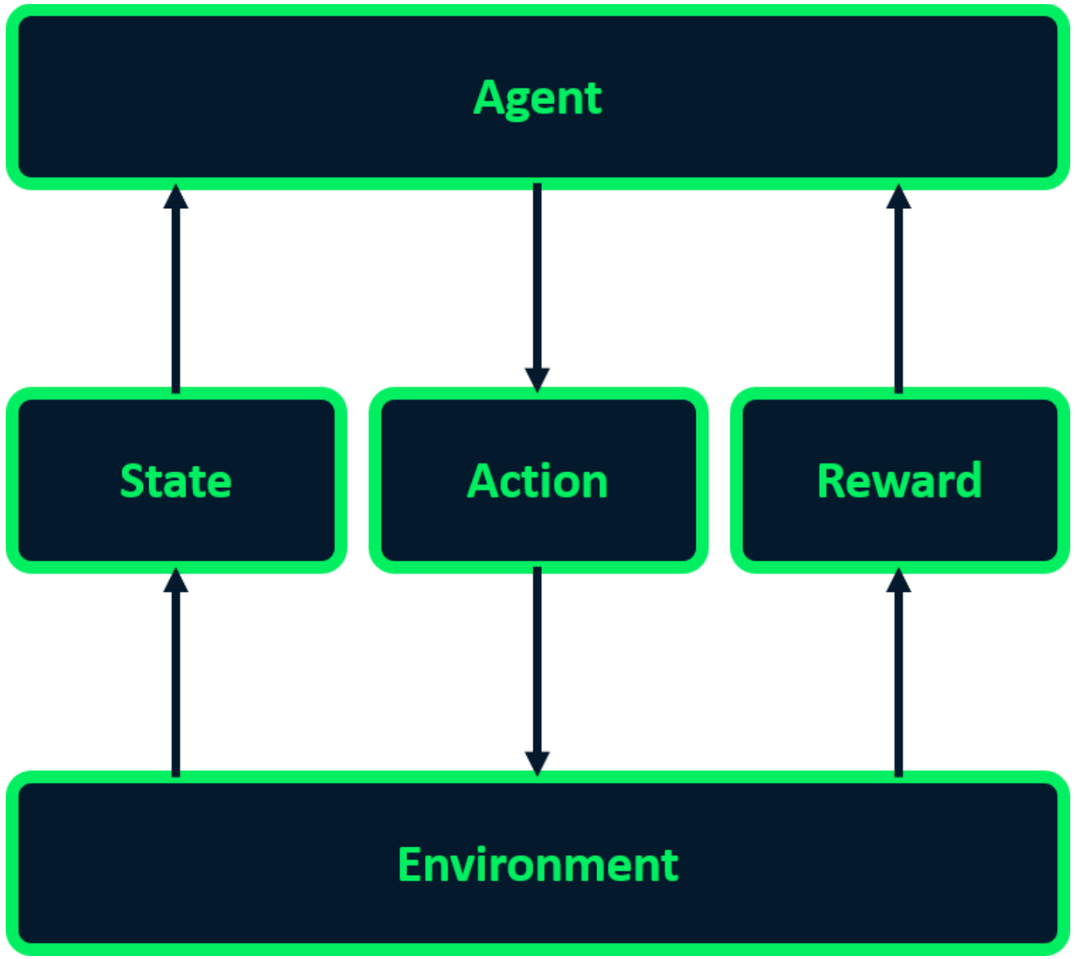
RL framework

- **Agent:** learner, decision-maker
- **Environment:** challenges to be solved
- **State:** environment snapshot at given time
- **Action:** agent's choice in response to state



RL framework

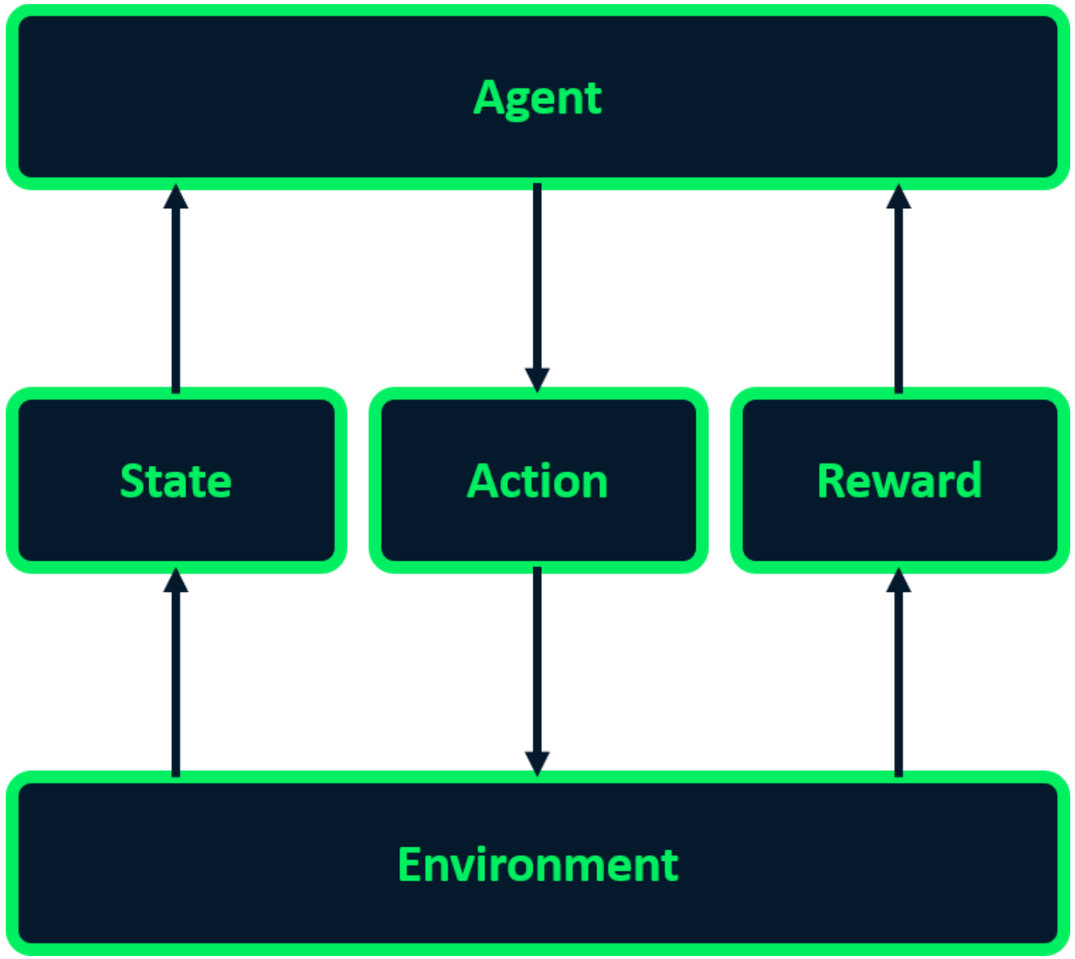
- **Agent:** learner, decision-maker
- **Environment:** challenges to be solved
- **State:** environment snapshot at given time
- **Action:** agent's choice in response to state
- **Reward:** feedback for agent action



RL interaction loop

```
env = create_environment()
state = env.get_initial_state()

for i in range(n_iterations):
    action = choose_action(state)
    state, reward = env.execute(action)
    update_knowledge(state, action, reward)
```



Episodic vs. continuous tasks

Episodic tasks

- Tasks segmented in episodes
- Episode has beginning and end
- **Example:** agent playing chess

Continuous tasks

- Continuous interaction
- No distinct episodes
- **Example:** Adjusting traffic lights

Return

- Actions have long term consequences
- Agent aims to maximize total reward over time
- **Return:** sum of all expected rewards

$$\underbrace{Return}_{\text{Sum of expected rewards}} = r_1 + r_2 + r_3 + \cdots + r_n$$

Discounted return

- Immediate rewards are more valuable than future ones
- **Discounted return:** gives more weight to nearer rewards
- Discount factor (γ): discounts future rewards

$$\underbrace{Discounted\ return}_{\text{Sum of discounted expected rewards}} = r_1 + \gamma r_2 + \gamma^2 r_3 + \gamma^3 r_4 + \cdots + \gamma^{n-1} r_n$$

reward discounts

Discount factor

- Between **zero** and **one**
- Balances immediate vs. long-term rewards
  - Lower value → immediate gains
  - Higher value → long-term benefits

Numerical example

```
import numpy as np
expected_rewards = np.array([1, 6, 3])
discount_factor = 0.9
discounts = np.array([discount_factor ** i for i in range(len(expected_rewards))])
print(f"Discounts: {discounts}")
```

```
Discounts: [1.   0.9  0.81]
```

```
discounted_return = np.sum(expected_rewards * discounts)
print(f"The discounted return is {discounted_return}")
```

```
The discounted return is 8.83
```

Let's practice!

REINFORCEMENT LEARNING WITH GYMNASIUM IN PYTHON

Interacting with Gymnasium environments

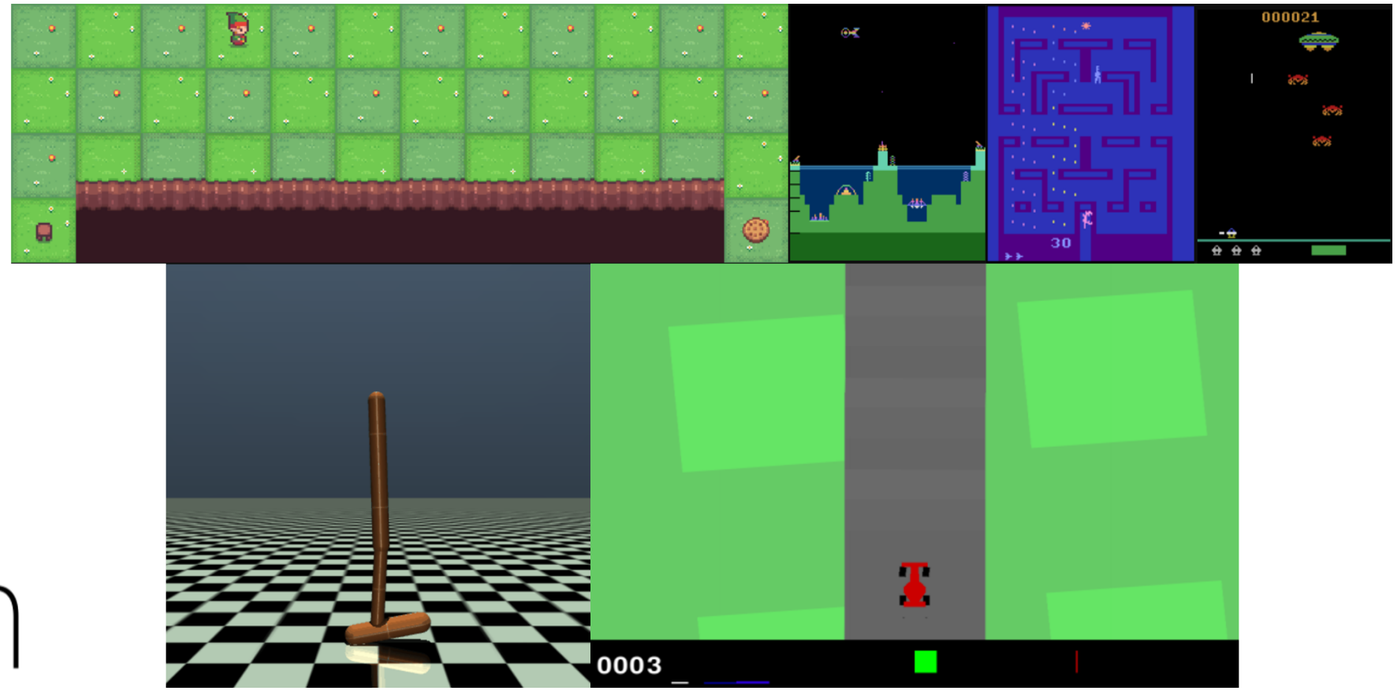
REINFORCEMENT LEARNING WITH GYMNASIUM IN PYTHON

Fouad Trad
Machine Learning Engineer

datacamp

Gymnasium

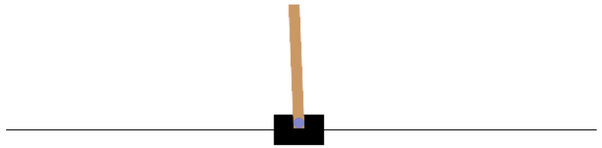
- Standard library for RL tasks
- Abstracts complexity of RL problems
- Provides a plethora of RL environments



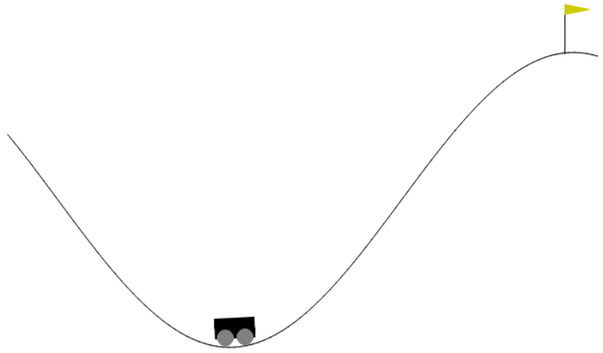
Gymnasium

Key Gymnasium environments

CartPole: Agent must balance a pole on moving cart

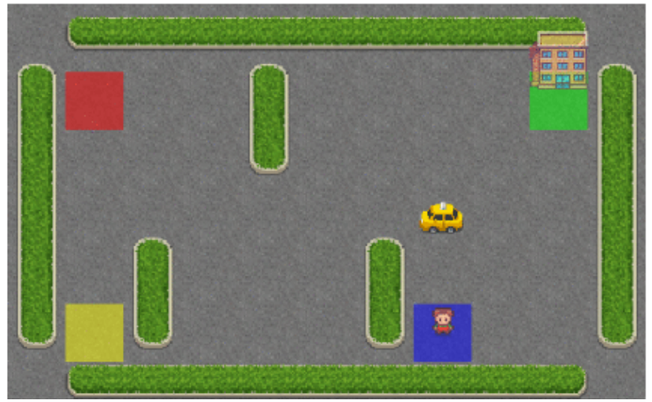


MountainCar: Agent must drive a car up a steep hill



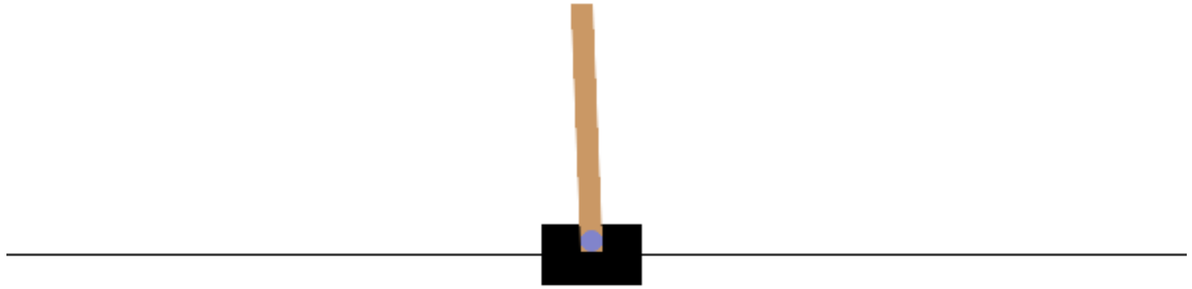
FrozenLake: Agent must navigate a frozen lake with holes

Taxi: Picking up and dropping off passengers



Gymnasium interface

- Unified for all environments
- Includes functions and methods to:
  - Initialize environment
  - Visually represent environment
  - Execute actions
  - Observe outcomes



Creating and initializing the environment

```
import gymnasium as gym


env = gym.make('CartPole', render_mode='rgb_array')
state, info = env.reset(seed=42)
print(state)
```

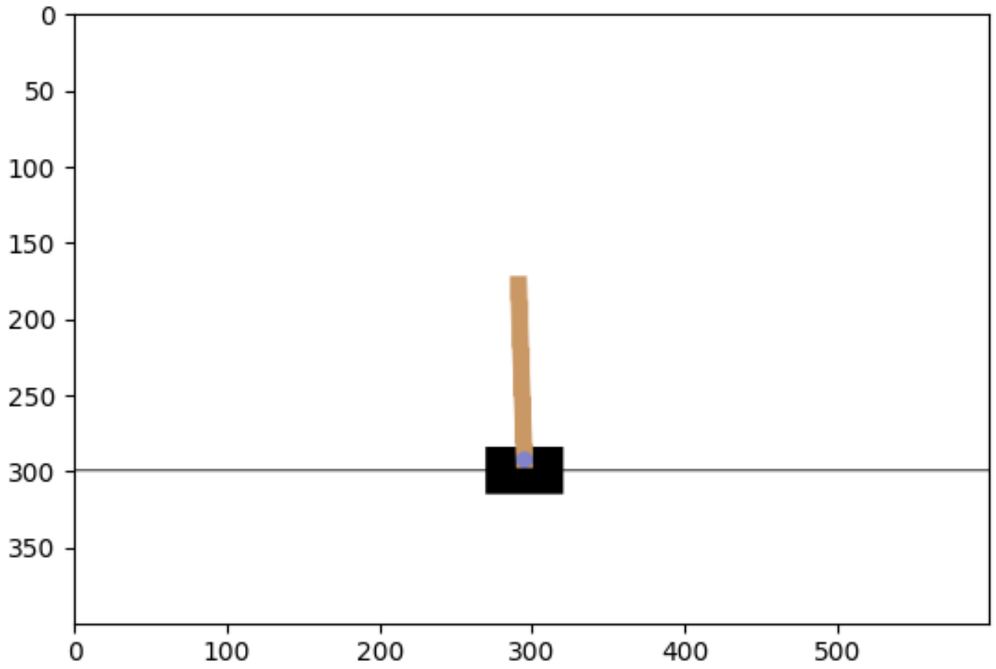
```
[-0.04405273  0.0242996  -0.04377224 -0.01767325]
```

[1] https://gymnasium.farama.org/environments/classic_control/cart_pole/

Visualizing the state

```
import matplotlib.pyplot as plt

state_image = env.render()
plt.imshow(state_image)
plt.show()
```

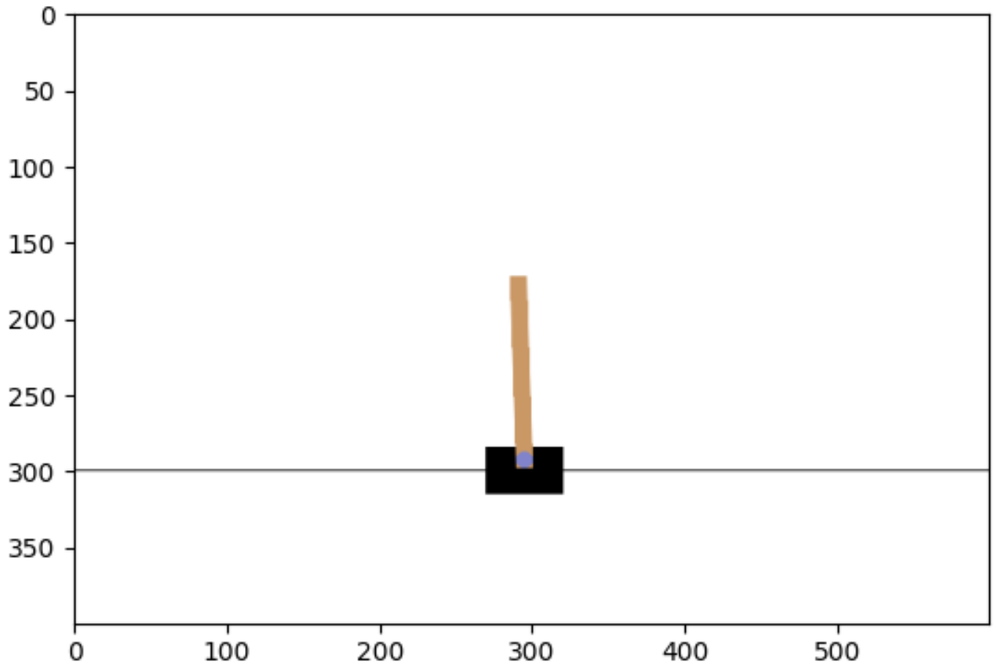


Visualizing the state

```
import matplotlib.pyplot as plt

def render():
  state_image = env.render()
  plt.imshow(state_image)
  plt.show()

# Call function
render()
```



Performing actions

- 0: moving left
- 1: moving right

```
action = 1
state, reward, terminated, truncated, info = env.step(action)
```

Performing actions

- 0: moving left
- 1: moving right

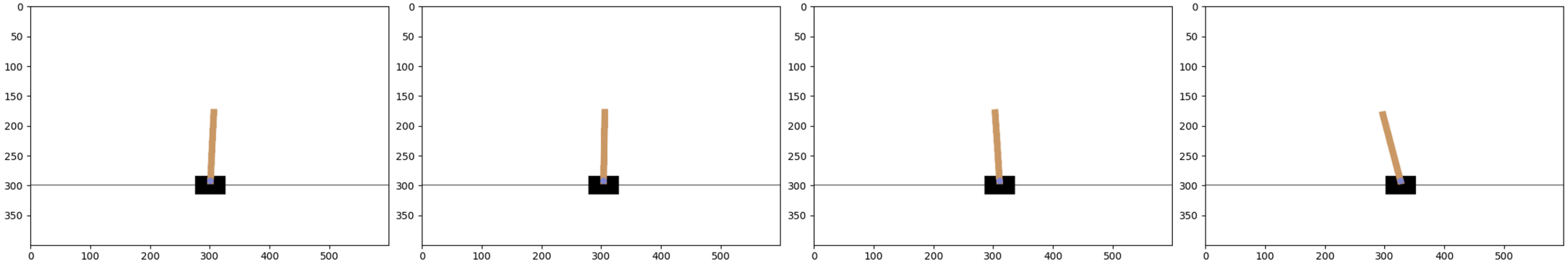
```
action = 1
state, reward, terminated, _, _ = env.step(action)

print("State: ", state)
print("Reward: ", reward)
print("Terminated: ", terminated)
```

```
State:  [-0.04356674  0.22002107 -0.0441257  -0.3238392 ]
Reward:  1.0
Terminated:  False
```

Interaction loops

```
while not terminated:
    action = 1 # Move to the right
    state, reward, terminated, _, _ = env.step(action)
    render()
```



Let's practice!

REINFORCEMENT LEARNING WITH GYMNASIUM IN PYTHON