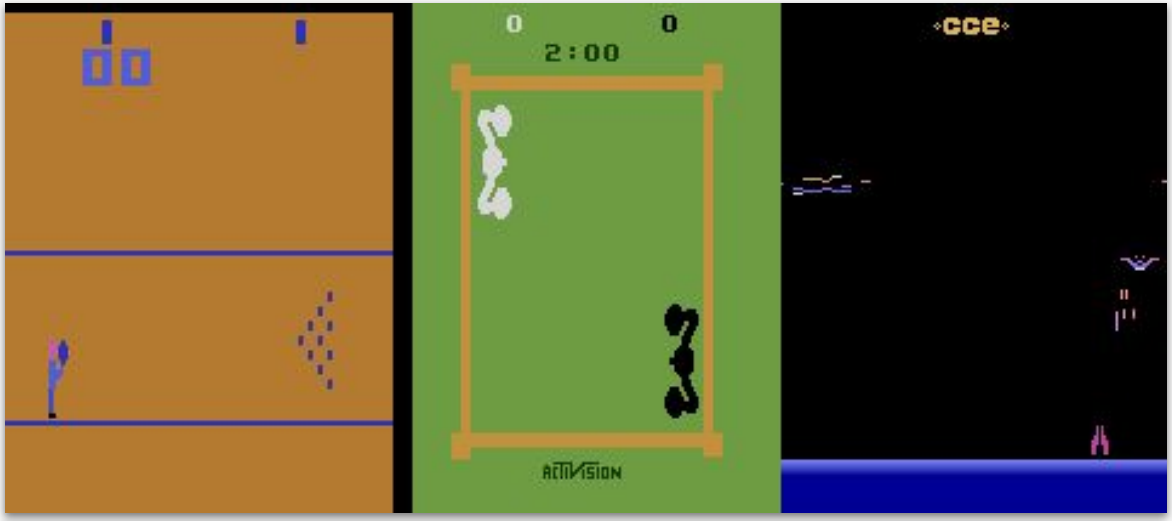
# Aprendizaje por refuerzo en Python

## Una breve introducción

Antonio Manjavacas

*PyDay 2024*

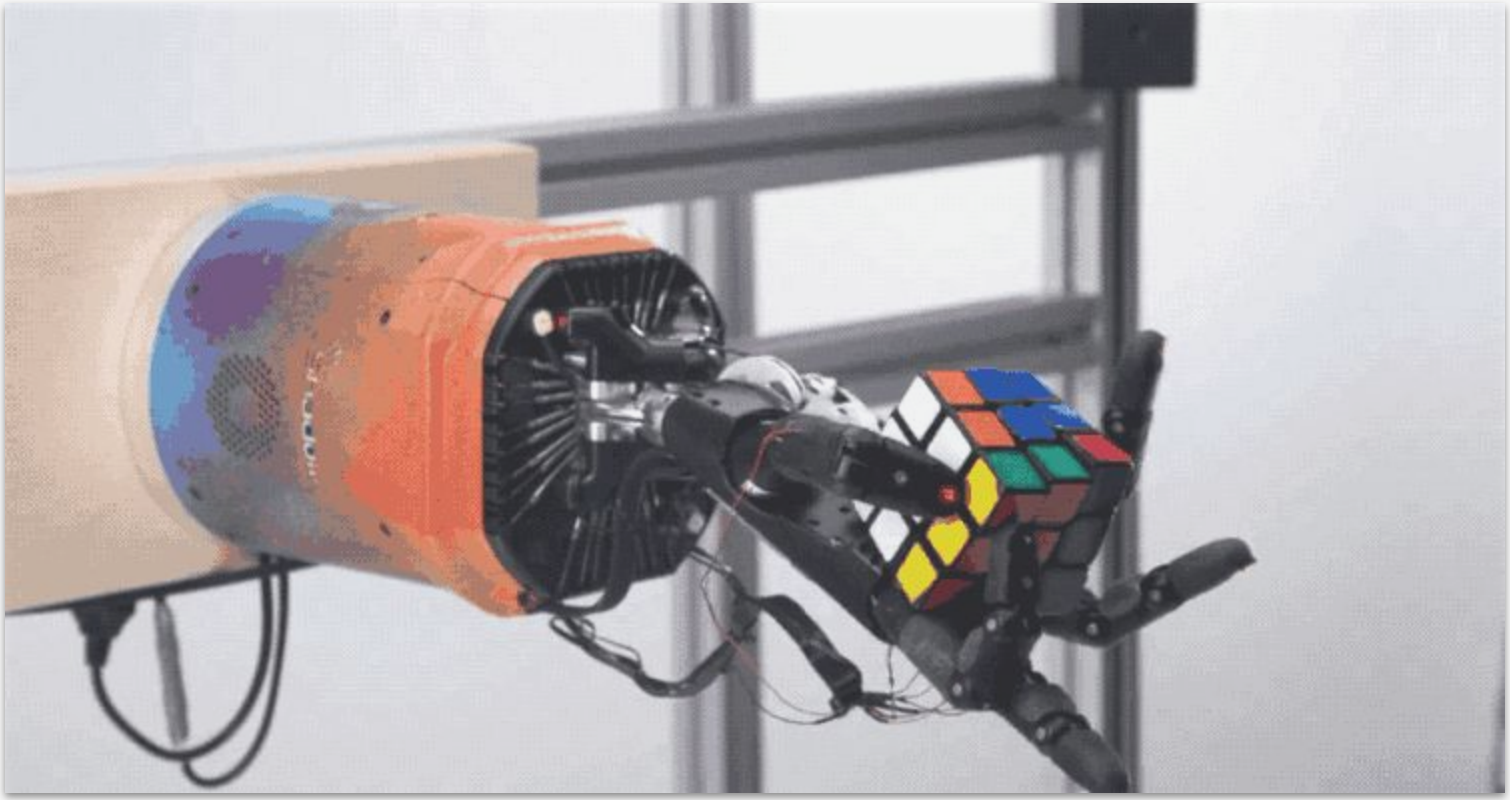


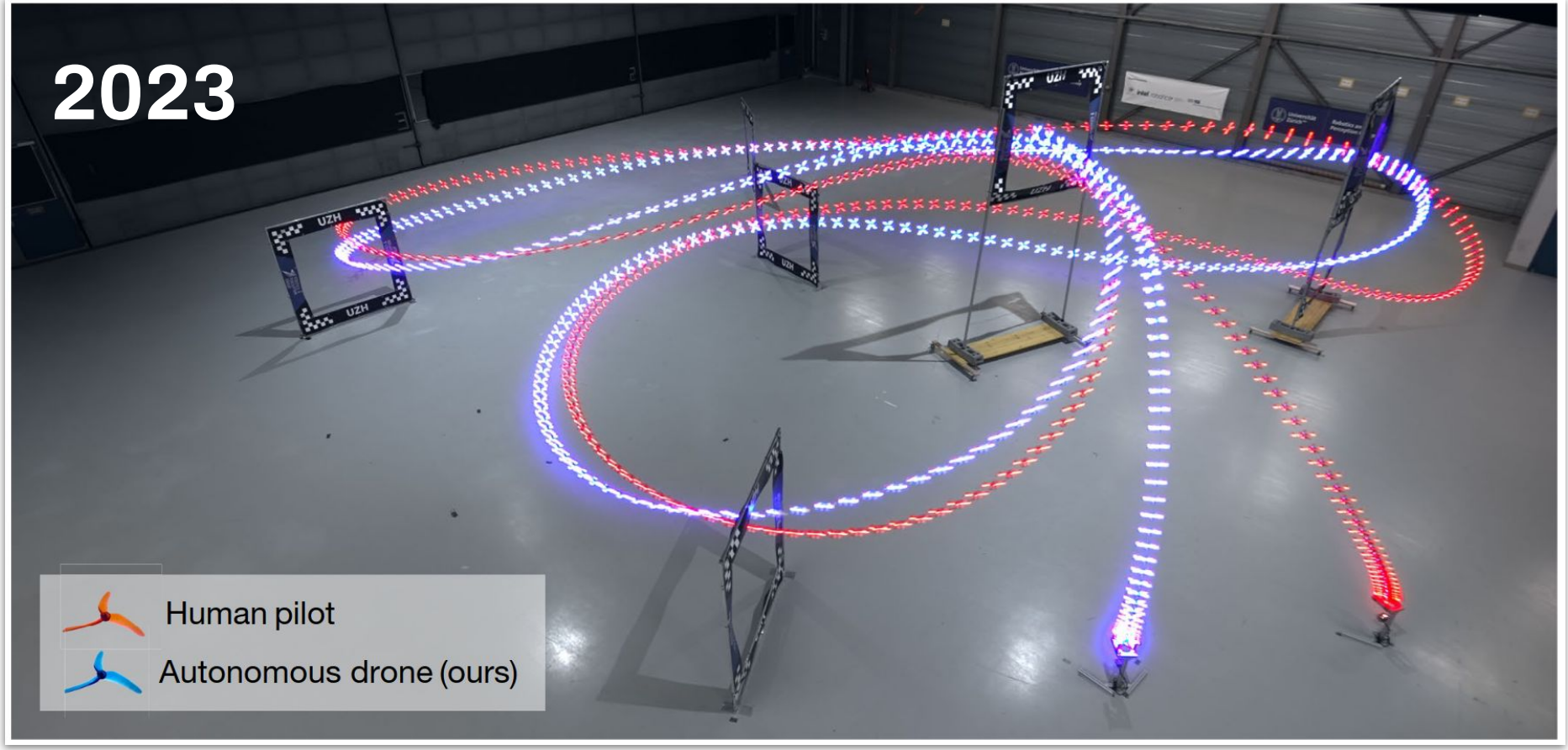


# 2013


ALPHAGO
AlphaGo
Lee Sedol
2016

# 2020



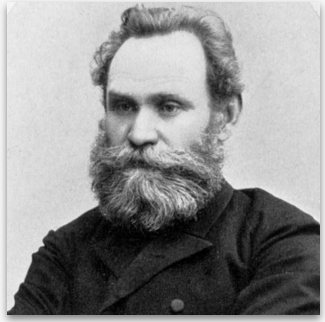

2023
UZH
UZH
Human pilot
Autonomous drone (ours)

# Aprendizaje por refuerzo

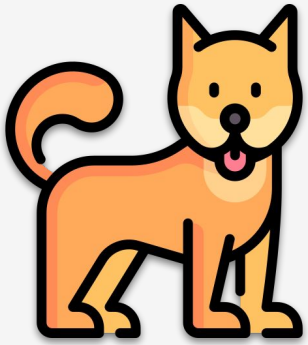
# ¿Aprendizaje por refuerzo?

# Conductismo

*El perro de Pávlov*

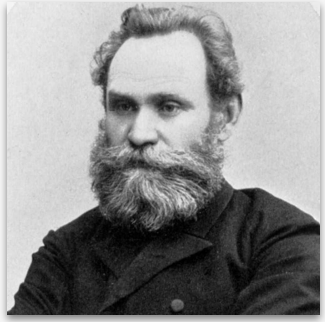


Iván P. Pávlov

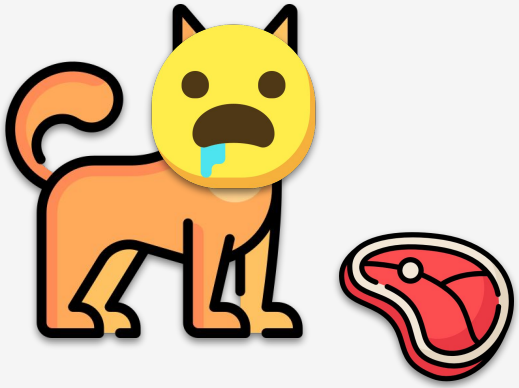


# Conductismo

*El perro de Pávlov*

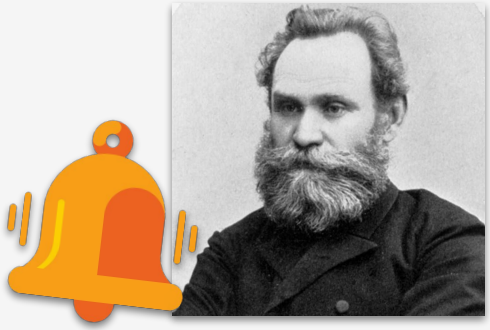


Iván P. Pávlov

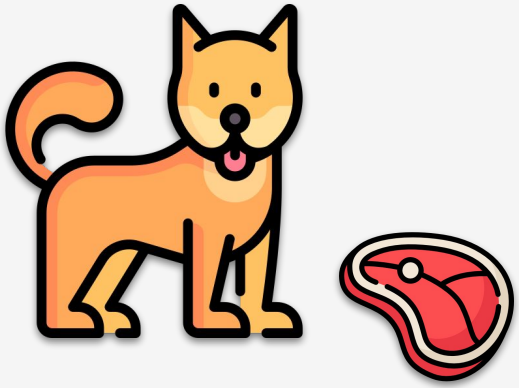


# Conductismo

*El perro de Pávlov*

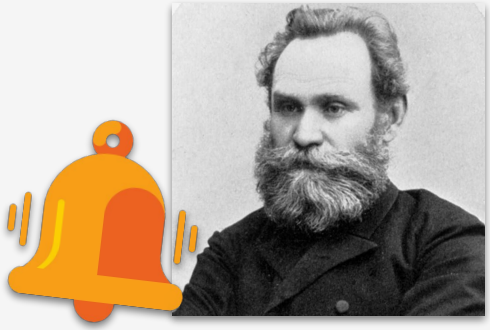


Iván P. Pávlov



# Conductismo

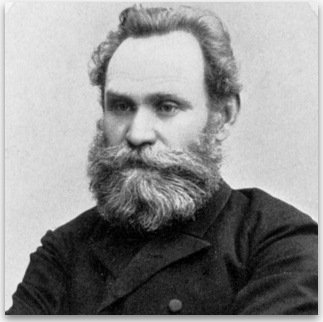
*El perro de Pávlov*



Iván P. Pávlov

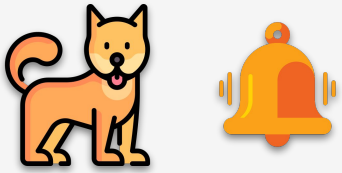
# Conductismo

*El perro de Pávlov*



Iván P. Pávlov

**ESTÍMULOS AMBIENTALES** ➡️ **ASOCIACIÓN** ➡️ **COMPORTAMIENTO**



**Refuerzo**: fortalecimiento de patrones de comportamiento en base a estímulos.

# ¿Pueden las máquinas aprender mediante estímulos?

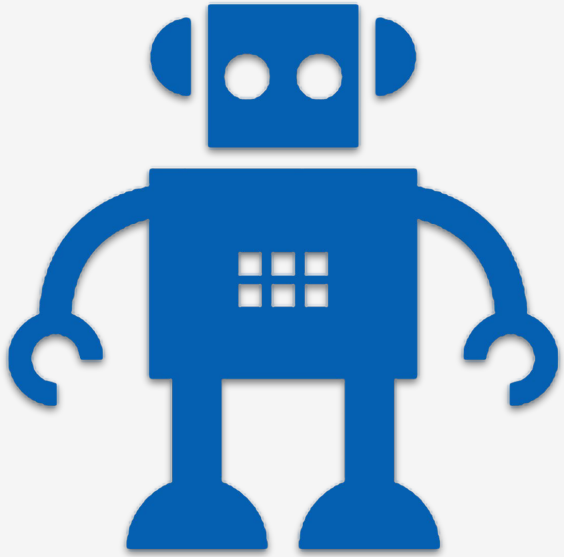
# ¿Pueden las máquinas aprender mediante estímulos?


Richard S. Sutton

Andrew G. Barto

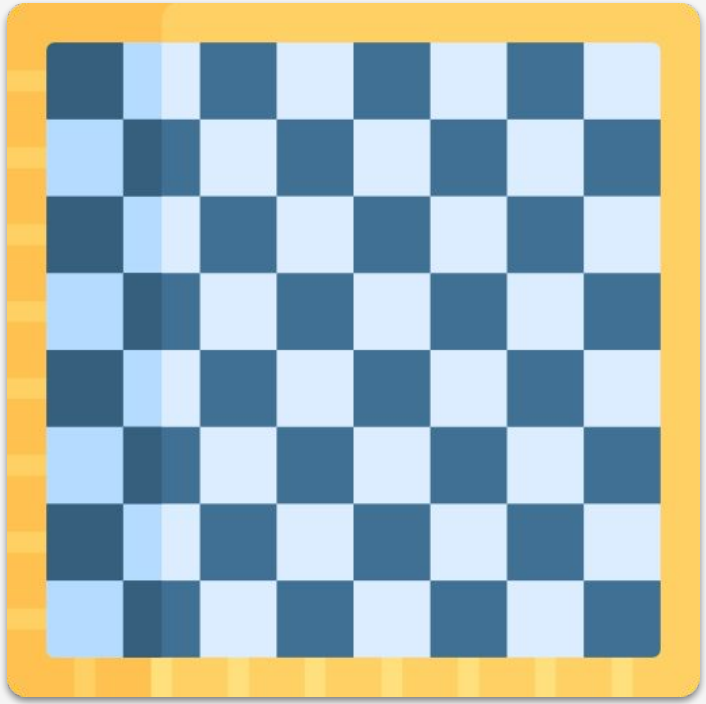
# Agente

- Persigue un **objetivo**.
- Observa el entorno.
- Realiza **acciones**.



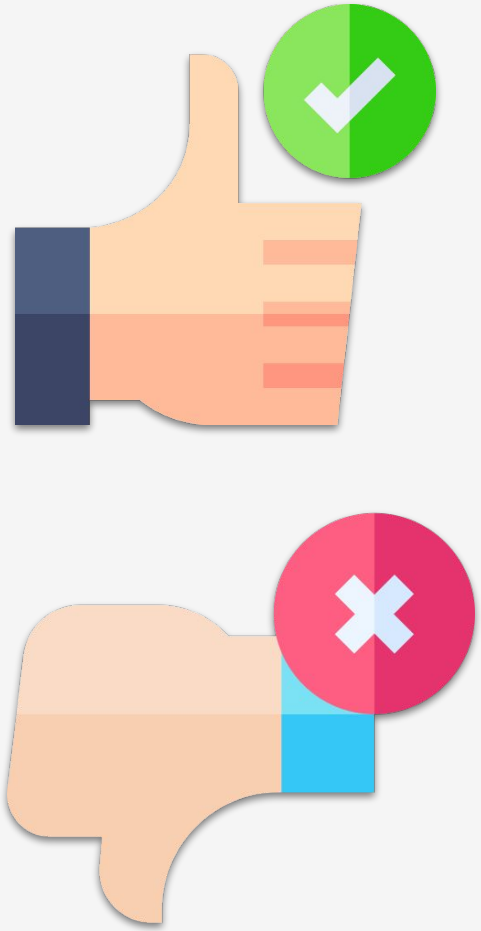
# Entorno

- Sistema en el que el agente se desenvuelve.
- Es dinámico.
- Se define por su **estado**.



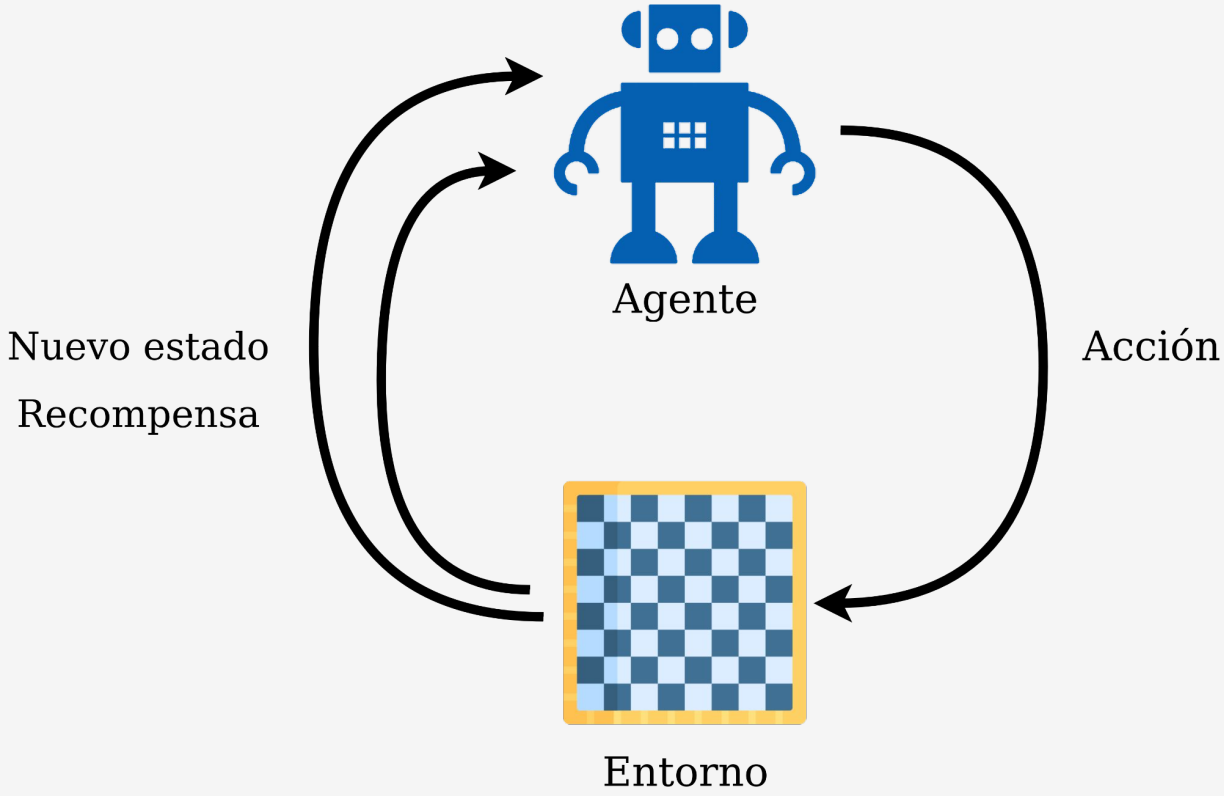
# Recompensa

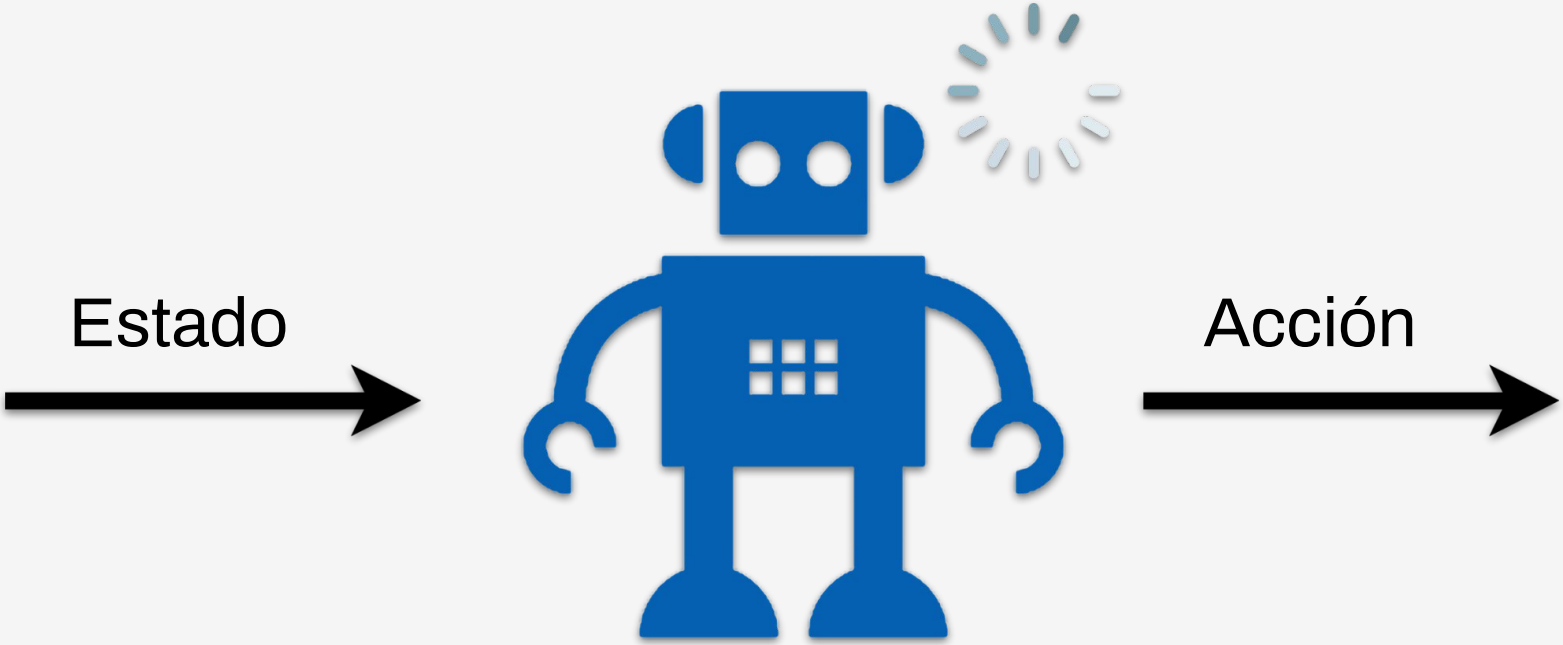
- Indica cómo de buena o mala es una **acción** o **estado**.
- Es el *estímulo* que determina el comportamiento del agente.
- Refuerzo **positivo** o **negativo**.



# Si combinamos estos elementos…

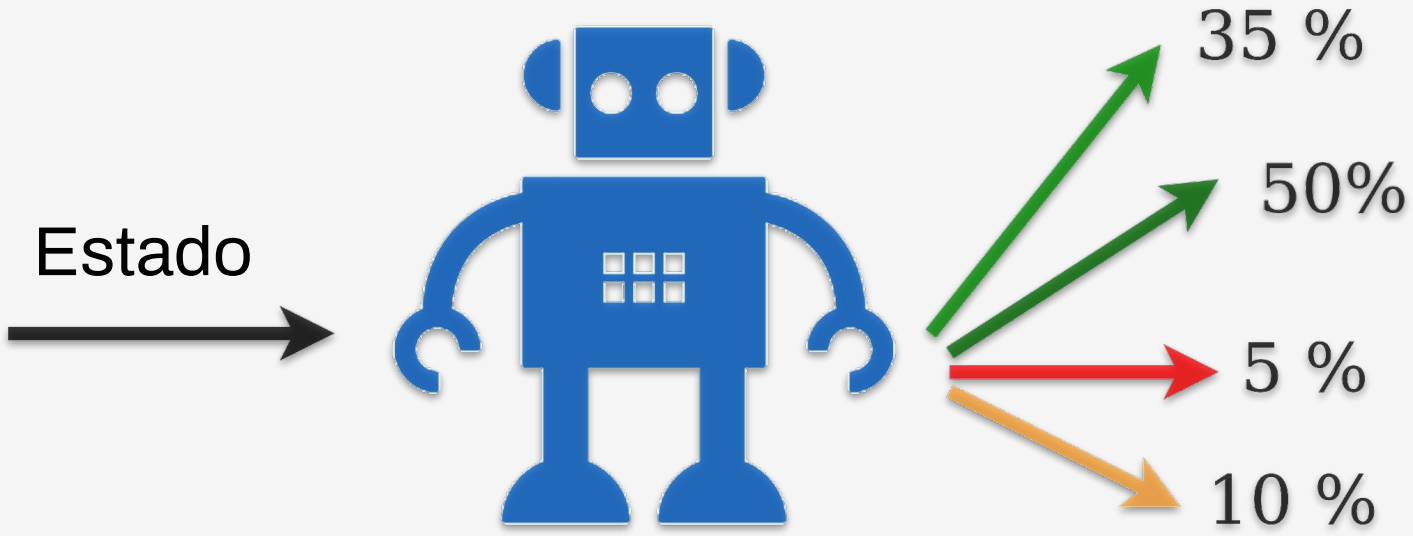
# Proceso de decisión de Markov




Estado
Acción

# Política

- Define el **comportamiento** del agente.
- Correspondencia **estado–acción**.
- Simple *vs*. compleja.


Estado
35 %
50%
5 %
10 %

# Hablemos en *Python*

# *Mountain car*

**Acciones**

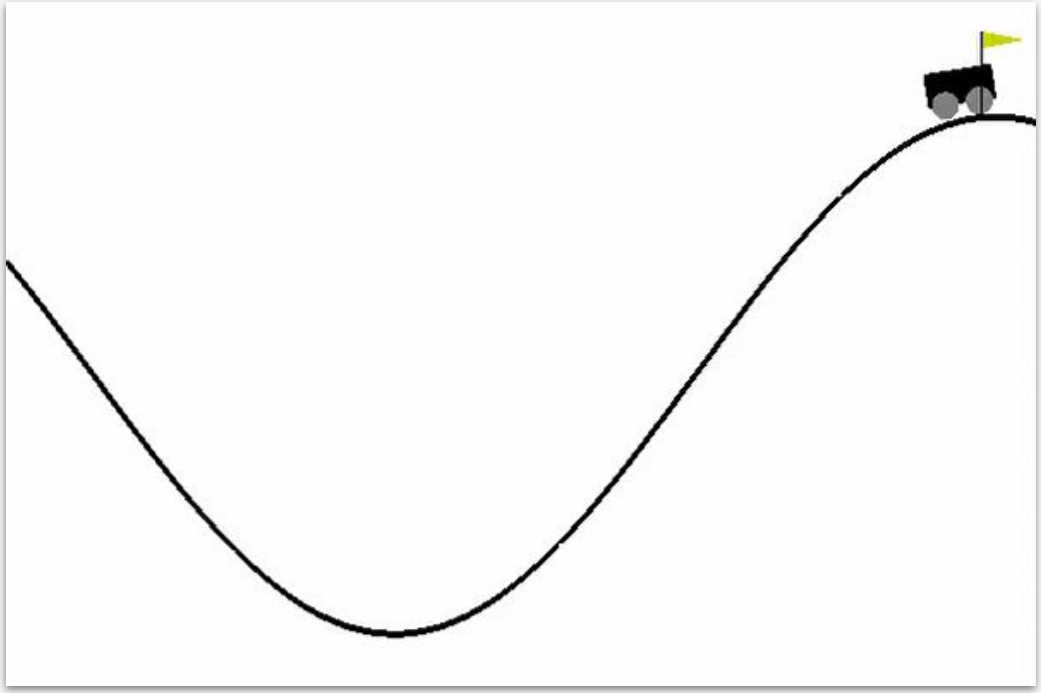
- *Izquierda*, *derecha*, *nada*

**Observación**

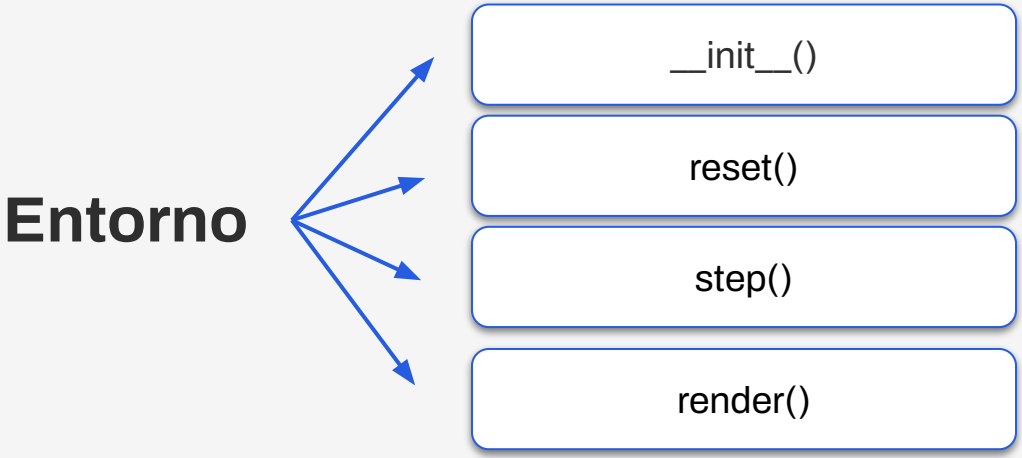
- *Velocidad*
- *Posición*

**Recompensa**

- *–1* por cada instante de tiempo



Gymnasium



Se define el espacio de **observaciones** y **acciones**.

- __init__()
- reset()
- step()
- render()

```python
def __init__(self, …):
    # Datos relevantes para definir el estado del entorno
    self.min_position = -1.2
    self.max_position = 0.6
    self.max_speed = 0.07
    self.goal_position = 0.5
    self.goal_velocity = goal_velocity
    self.force = 0.001
    self.gravity = 0.0025
    # …
    # Espacio de acciones y observaciones
    self.action_space = spaces.Discrete(3)
    self.observation_space = spaces.Box(self.low, self.high, dtype=np.float32)
```

Inicio de un nuevo **episodio**. Se devuelve el **estado inicial**.

- __init__()
- reset()
- step()
- render()

```
def reset(self, …):
    self.state = np.array([self.np_random.uniform(low=low, high=high), 0])

    if self.render_mode == "human":
        self.render() # pygame

    return np.array(self.state, dtype=np.float32)
```

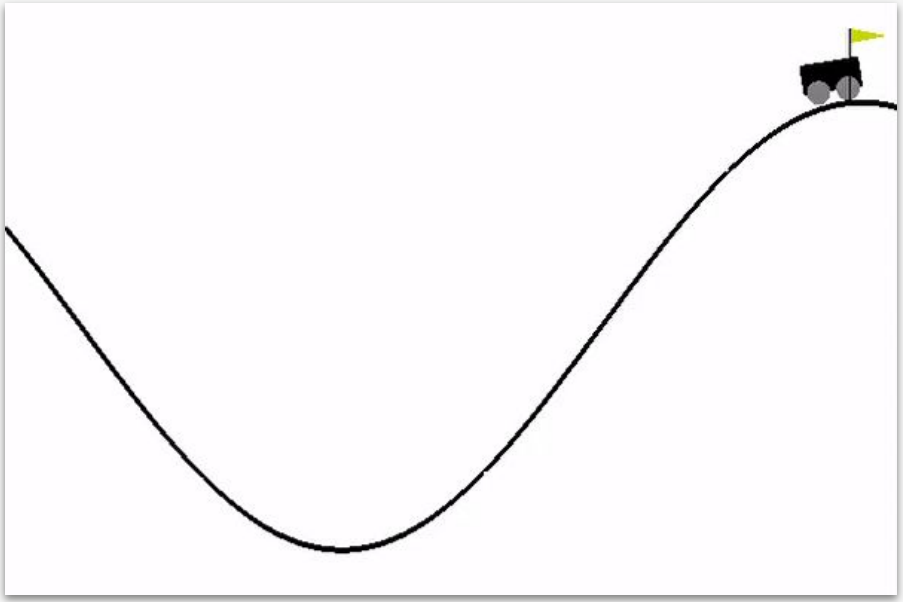
Se realiza una **acción** sobre el entorno.
Se devuelve su **nuevo estado**.

- __init__()
- reset()
- **step()**
- render()

```python
def step(self, action, …):

    position, velocity = self.state
    velocity += (action - 1) * self.force + math.cos(3 * position) * (-self.gravity)
    velocity = np.clip(velocity, -self.max_speed, self.max_speed)
    position += velocity
    position = np.clip(position, self.min_position, self.max_position)

    terminated = bool(
            position >= self.goal_position and velocity >= self.goal_velocity)

    reward = -1.0

    self.state = (position, velocity)

    if self.render_mode == "human":
        self.render()

    return np.array(self.state, dtype=np.float32), reward, terminated, truncated, {}
```

**Representa** el estado actual del entorno.

- __init__()
- reset()
- step()
- render()

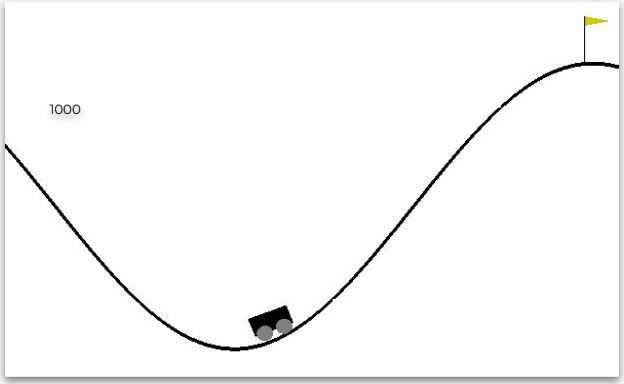


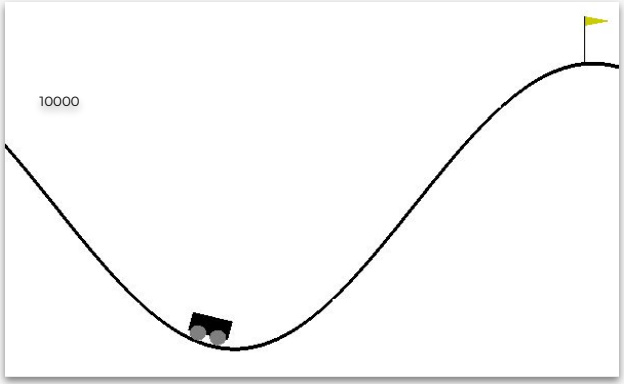
# Agente aleatorio

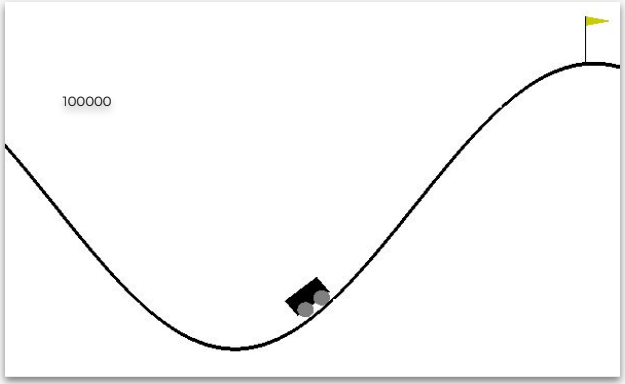
```python
import gymnasium as gym

env = gym.make("MountainCar-v0", render_mode="human")
observation, info = env.reset()

for _ in range(1000):
    # random agent
    action = env.action_space.sample()
    observation, reward, terminated, truncated, info = env.step(action)

    if terminated or truncated:
        observation, info = env.reset()

env.close()
```

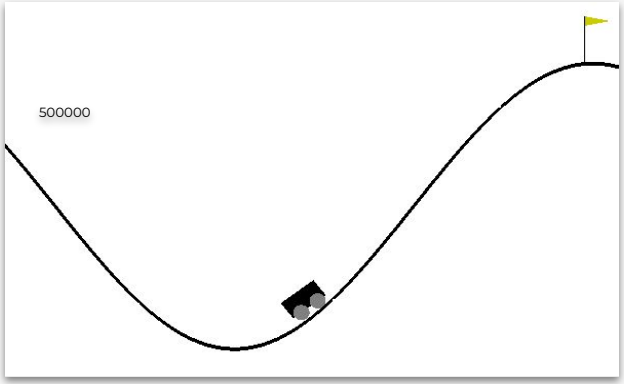
# Agente
## “inteligente”

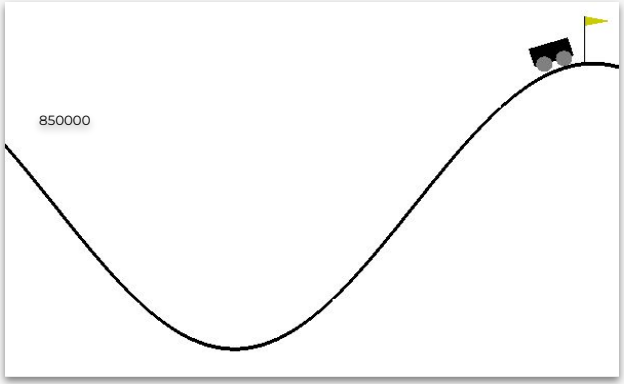
```python
import gymasium as gym

from stable_baselines3 import PPO

env = gym.make('MountainCar-v0')

model = PPO('MlpPolicy', env, verbose=1)

model.learn(total_timesteps=1000)

model.save('model')

model = PPO.load('model.zip')

obs = env.reset()
for i in range(100):
    action = model.predict(obs, deterministic=True)
    obs, reward, done, info = env.step(action)
    env.render()
    if done:
        obs = env.reset()

env.close()
```

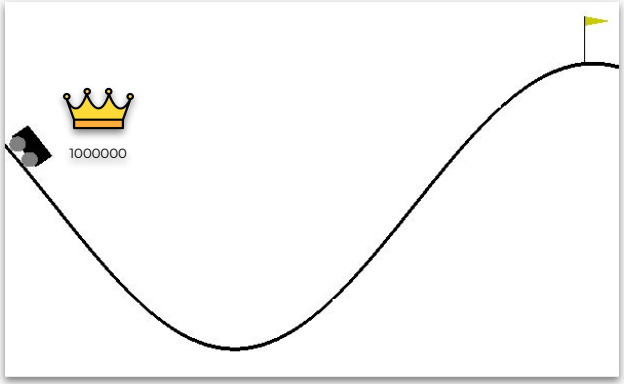

1000


10000


100000


500000


850000


1000000

# Aprendizaje por refuerzo en Python

## Una breve introducción

Antonio Manjavacas
manjavacas@ugr.es
@manjavacas_

*PyDay 2024*