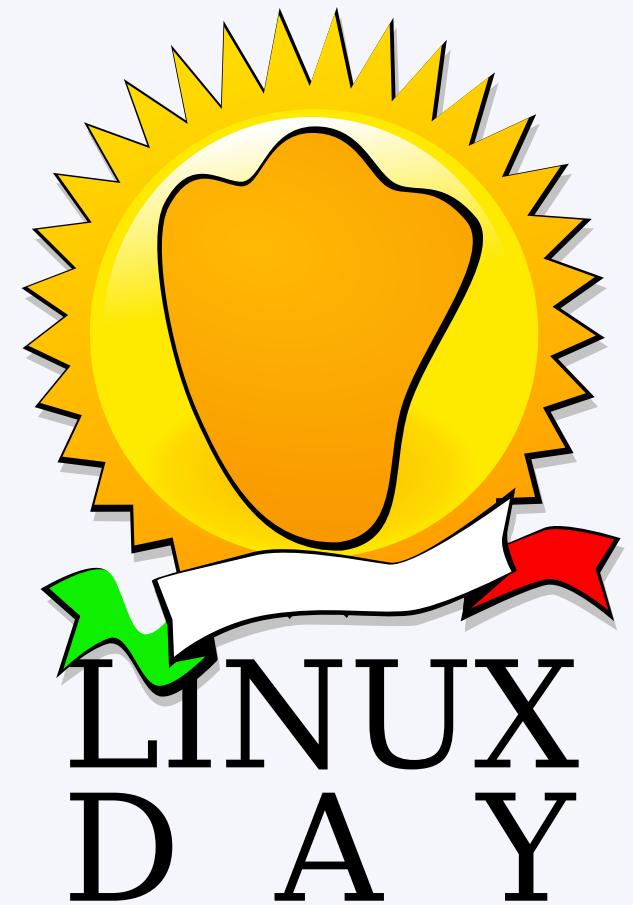
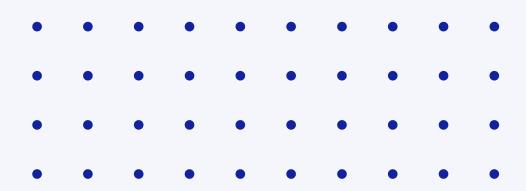


Linux Day 2022

# Il robot e l'algoritmo che gli insegnò a muoversi

Ornella Fanais

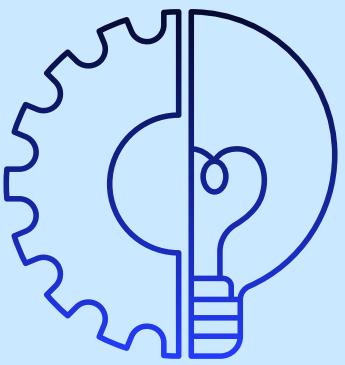
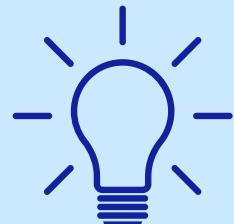
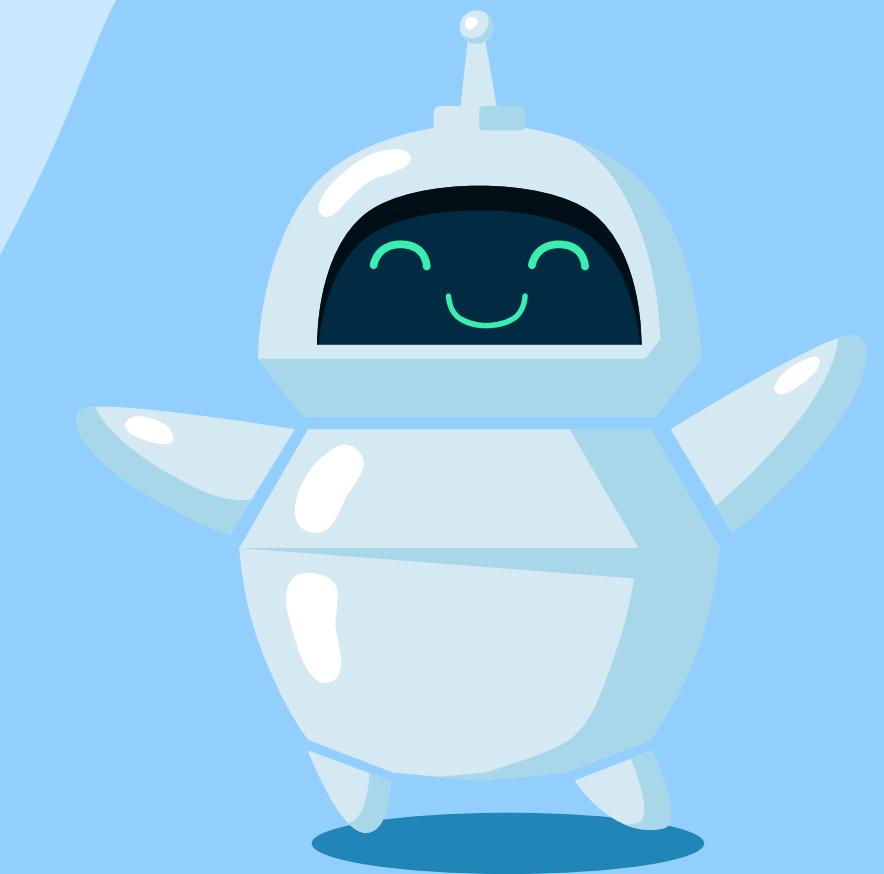


## About me

**Vivo a Cagliari, sono appassionata di robotica e intelligenza artificiale**

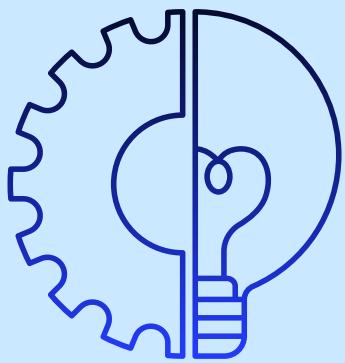
- Laurea Magistrale in Ingegneria Meccanica (università di Cagliari)
- Software engineer presso Avanade

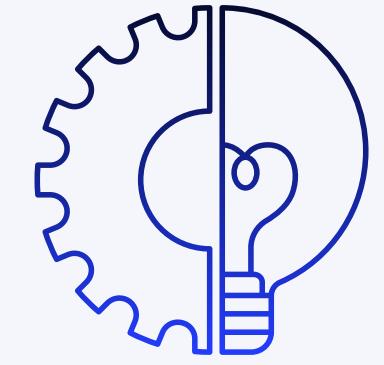
<https://www.linkedin.com/in/ornella-fanais/>



# Sommario

- Machine Learning
- Reinforcement Learning
- Principali elementi RL
- Strumenti per costruire un ambiente robotico
- Studio del problema fisico e del problema RL
- PyBullet 101
- Ambiente OpenAI Gym
- Colab e Gpu





Now, do you love me?  
(Do you love me)  
Now that I can dance!

Do You Love Me?" by The Contours.





# Machine Learning

## Supervised learning

Consiste nel trarre un modello a partire da dati etichettati ed effettuare previsioni relative a dati non disponibili. Le tecniche utilizzate sono la regressione e la classificazione.

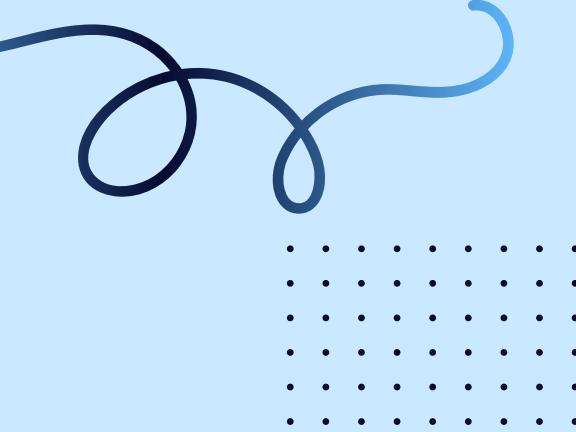
## Unsupervised learning

In questo caso i dati di addestramento non sono etichettati. Le tecniche utilizzate, il clustering consentono di organizzare una serie di informazione all'interno di gruppi significativi.

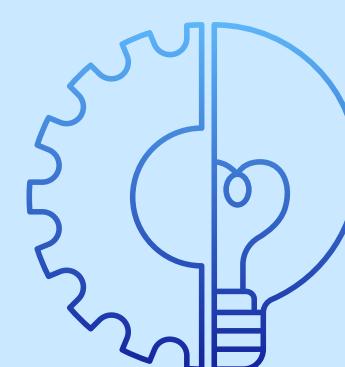
## Reinforcement learning

Il sistema di apprendimento chiamato agente osserva l'ambiente, lo aiuta a selezionare ed eseguire azioni ottenendo in cambio una ricompensa.





# Reinforcement Learning

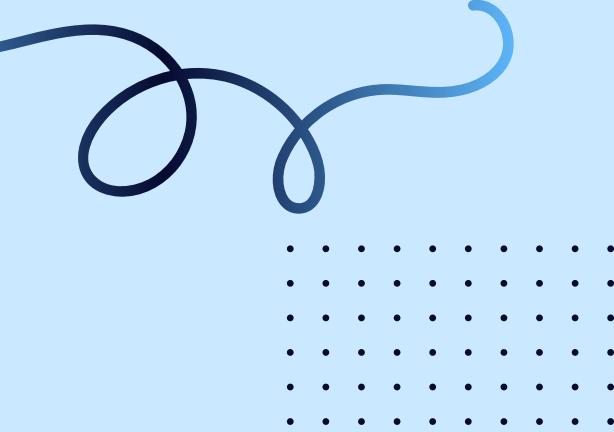


L'apprendimento dell'agente è dato dalla continua interazione con l'ambiente Al fine di raggiungere il corretto addestramento che deve culminare con la massimizzazione della ricompensa.

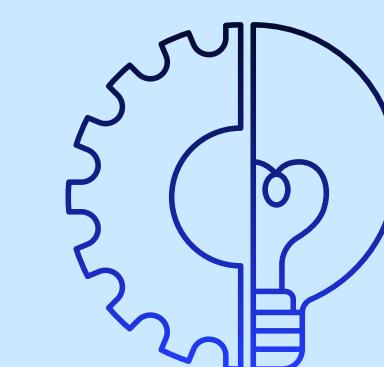


- Esegue l'azione
- Riceve informazioni sullo stato
- Riceve la ricompensa

- Riceve l'azione
- Emette uno stato  
(Modifica il suo stato)
- Emette la ricompensa



# Ambiente RL



Rappresenta l'universo in cui esiste l'agente.  
L'ambiente si trova sempre in uno stato specifico che viene  
modificato dalle azioni che l'agente intraprende.



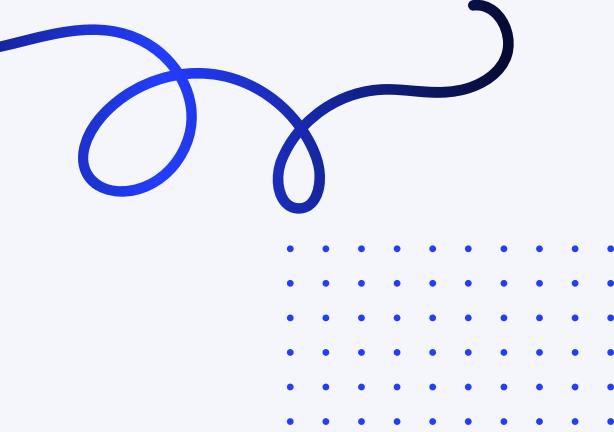
**Azione:**  
eseguita dall'agente  
per modificare lo  
stato dell'ambiente.



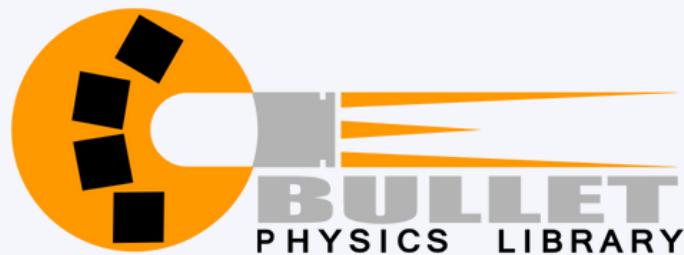
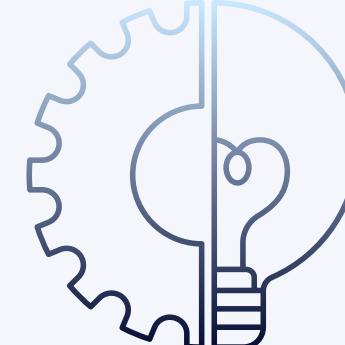
**Stato:**  
Informazione che  
l'ambiente fornisce  
all'agente.



**Ricompensa:**  
Fornisce all'agente un  
feedback relativo all'azione  
intrapresa.



# Strumenti per costruire un ambiente robotico RL



## PyBullet

PyBullet è una libreria che sfrutta un Physics engine, utilizza Python per la simulazione robotica e il Reinforcement Learning. Rappresenta una valida alternativa open source per la physics simulation anche in ambito scientifico.

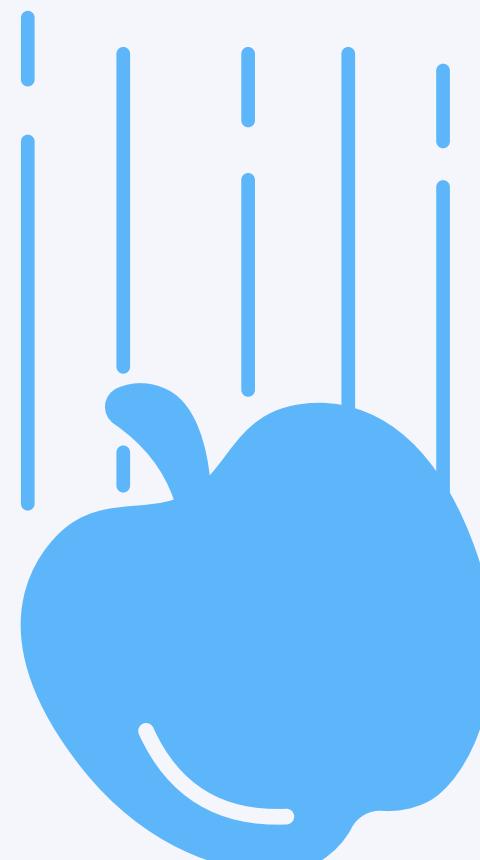
## OpenAI Gym

E' un toolkit open-source che fornisce un'ampia varietà di ambienti simulati rilasciato nel 2016 da Elon Musk e Sam Altman. Lo scopo è quello di fornire un standard di sviluppo per l'intelligenza artificiale facile da configurare e utilizzabile con Python.

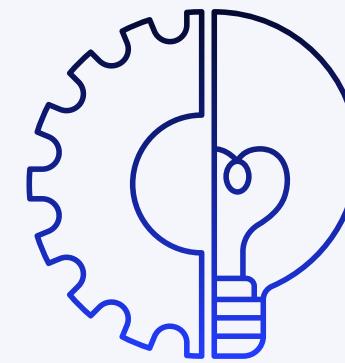
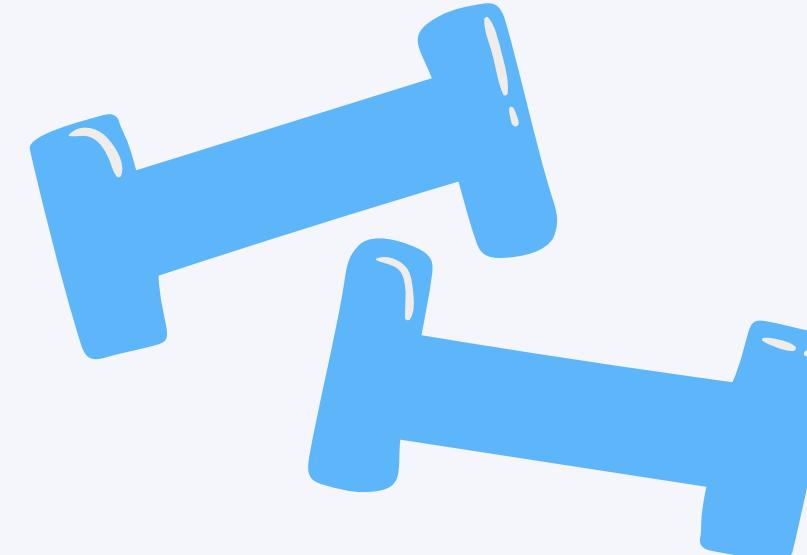
The OpenAI logo, consisting of the word "OpenAI" in a large, light blue, sans-serif font.

# Librerie Python e Analisi

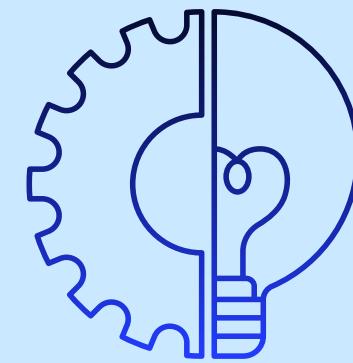
Analisi del problema fisico:  
**PyBullet**  
geometria e cinematica



Analisi del problema RL:  
**Standard OpenAI Gym**  
costruzione dell'ambiente RL  
**PyBullet**  
algoritmi PPO e DQN



# Analisi di un problema fisico con Pybullet



## Caricamento

Occorre conoscere la geometria del sistema oggetto di studio e il suo comportamento cinematico. Utilizzeremo le informazioni all'interno dei file URDF.

## Monitoraggio

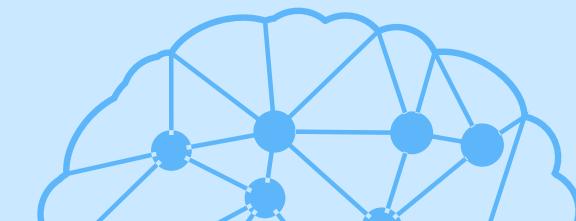
Dobbiamo monitorare i parametri quando vogliamo simulare un determinato evento che prevede il movimento di un corpo.

## Controllo

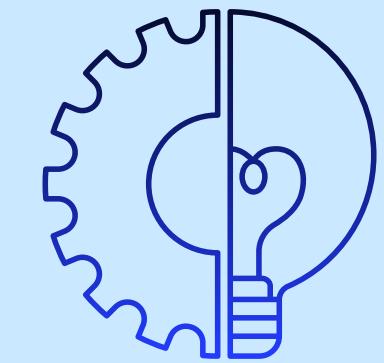
Dobbiamo generare un movimento in modo tale da soddisfare il nostro obiettivo (per esempio la manipolazione di un oggetto attraverso una pinza).

## Cinematica inversa

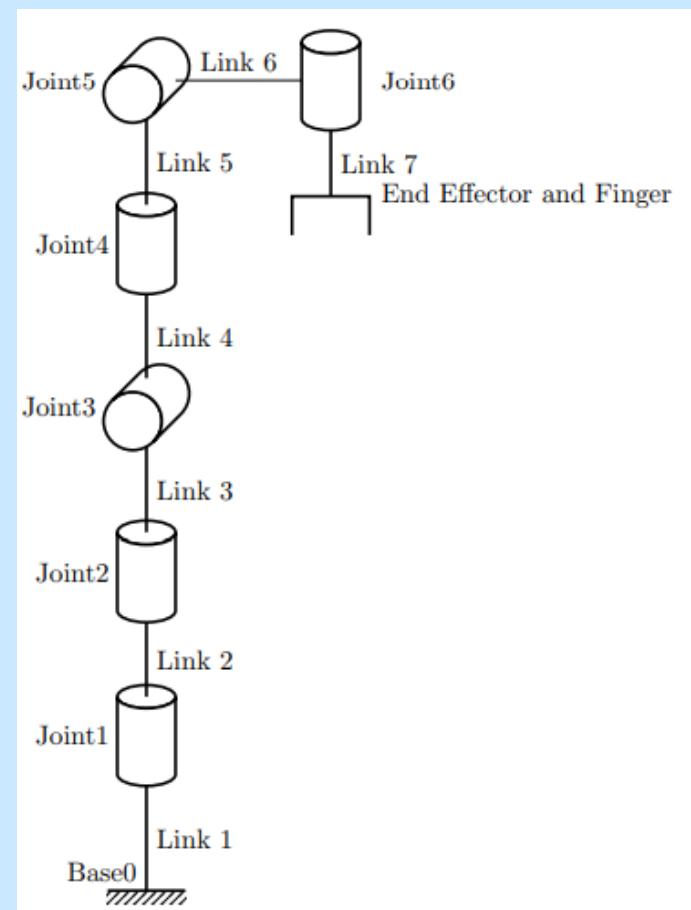
Note le posizioni x,y,z cartesiane dell'oggetto si vuole convertire l'informazione in termini di inclinazione o angoli rispetto a un asse di riferimento differente da quello principale.



# PyBullet 101: Some Basic Stuff



Nell'industria 4.0 è molto importante avvalersi di ambienti di simulazione perché permettono una notevole riduzione dei costi.

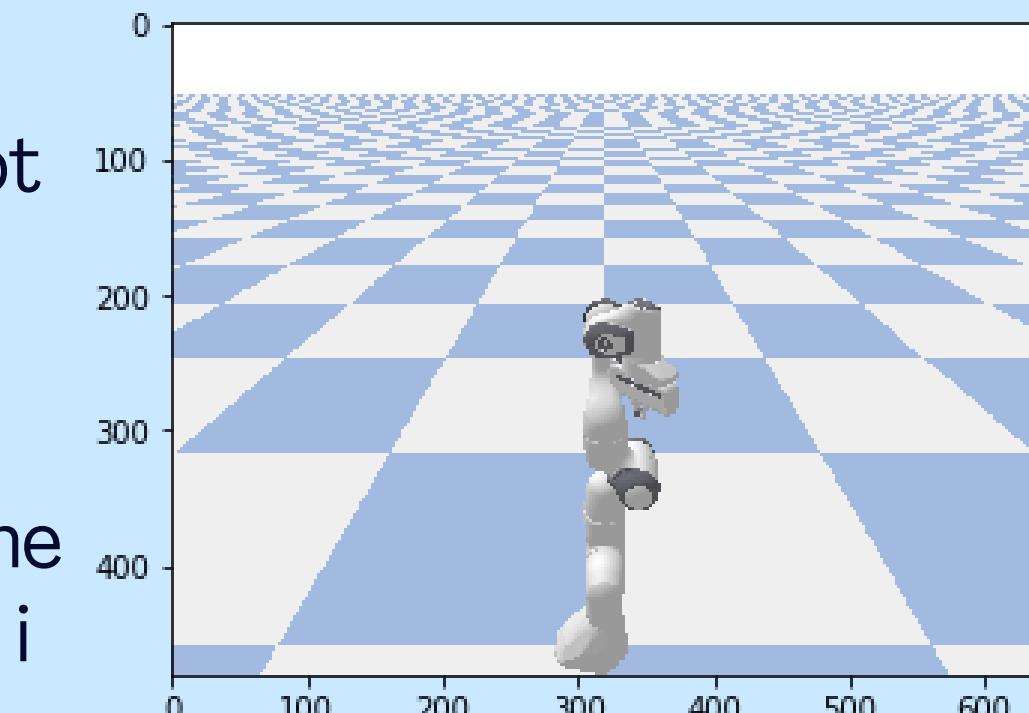


Catena cinematica 7DOF

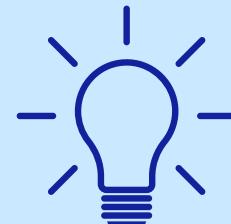
**URDF** (Universal Robot Description File) sono utilizzati nei progetti ROS (Robot Operating System) per descrivere il robot e i suoi elementi.

Il robot è costituito da:

- **links**: i corpi rigidi;
- **joints**: sono i giunti ovvero i motori che consentono al robot di muoversi. Tra i più diffusi si possono annoverare: i giunti prismatici (traslazione) e quelli rotoidali (rotazione).

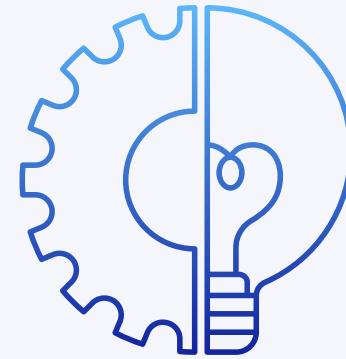


Franka Panda Emilia 7 DOF



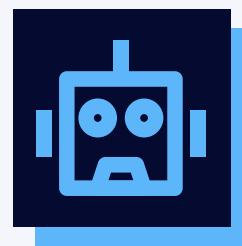


# Caricamento

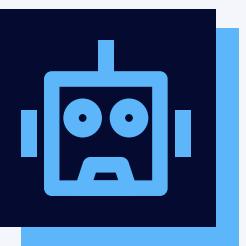


Svolge la funzione di caricamento del file URDF del robot.

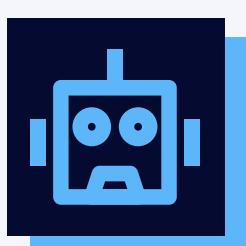
```
loadURDF("franka_panda/panda.urdf", [0, 0, 0],[0, 0, 0, 1], useFixedBase = True)
```



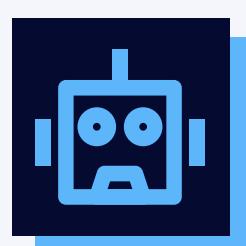
**loadURDF**



**basePosition**  
[0, 0, 0]



**baseOrientation**  
[0, 0, 0,1]



**useFixedBase**  
True

The screenshot shows the GitHub repository page for 'bulletphysics' (https://github.com/bulletphysics/bulletphysics). The page includes the repository's name, description, statistics (64 followers, 2 forks), and links to overview, repositories, projects, packages, and people. Below the main header, there's a section for 'Popular repositories' showing 'bullet3' and 'pybullet\_robots'.

Sono le componenti cartesiane del vettore posizione [x,y,z].

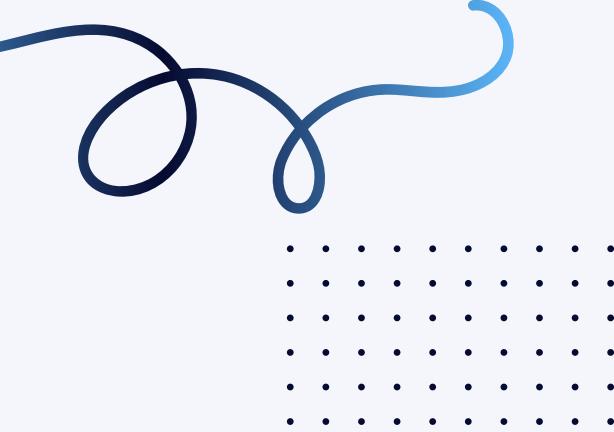
Su ogni giunto sarà posizionata una terna.

Sono le proiezioni dei versori i,j,k sugli assi x,y,z. Si utilizzano gli angoli di Eulero (roll, pitch, yaw) con l'aggiunta di un 4° parametro, da qui i quaternioni.

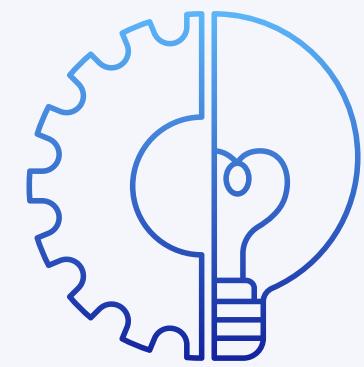
Ogni giunto avrà una terna la cui orientazione farà riferimento ad un sistema cartesiano di riferimento.

Parametro opzionale utile per fissare la base.]





# Monitoraggio



Possiamo ricavare diverse variabili di stato del giunto usando **getJointState**, quali posizione e velocità in riferimento allo spazio di lavoro (**task space**).



Possiamo ottenere informazioni sulla posizione cartesiana e l'orientamento per il centro di massa di ciascuno links utilizzando **getLinkState** riferendoci allo spazio cartesiano (**configuration space**).

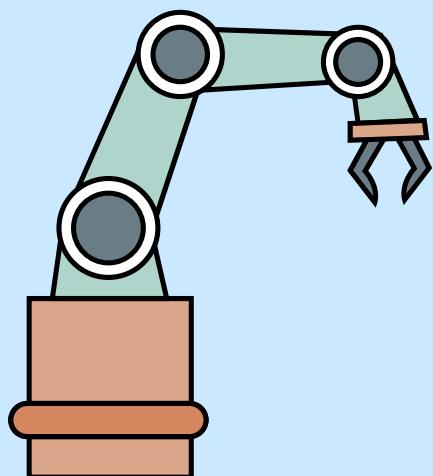


# Configuration Space VS Task Space



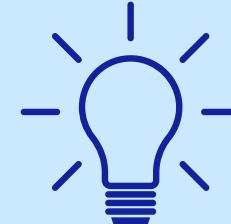
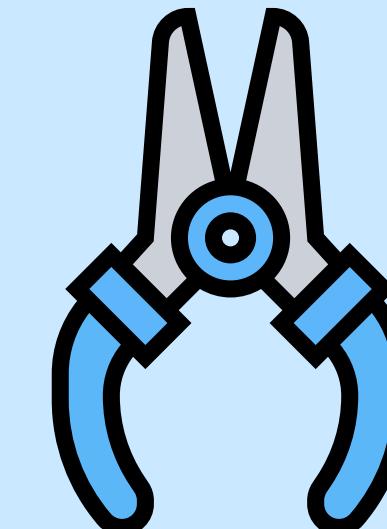
## Configuration Space

è lo spazio che contiene tutte le possibili configurazione dei giunti.



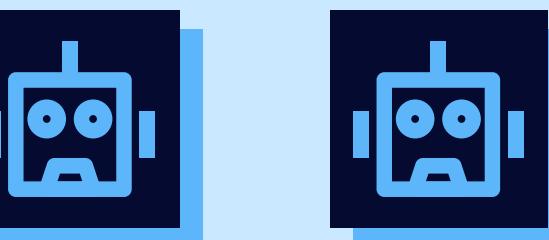
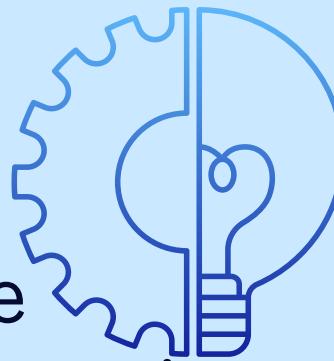
## Task Space

è lo spazio che contiene tutte le possibili posizioni raggiungibili dall'end effector

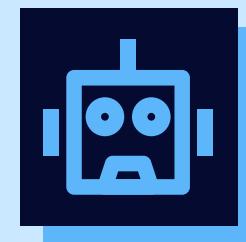


# Azionamento

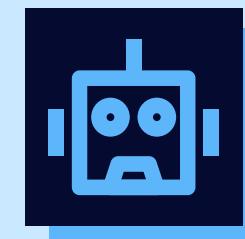
Svolge la funzione di controllo del movimento dei robot quindi dei suoi link e dei giunti in funzione dell'obiettivo da raggiungere. Utilizzato per modificare la velocità, la posizione o applicare una coppia torcente sul giunto.



**loadURDF JointIndex**

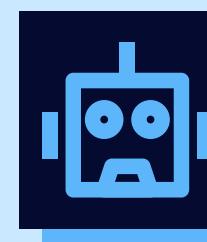


**controlMode**



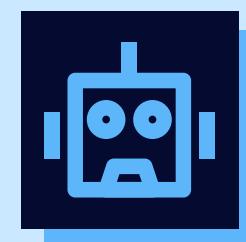
**Position control mode**

Il motore segue la posizione fornita e mantiene la stessa anche se agiscono forze esterne sul sistema. Es.3D printer



**Velocity control mode**

Il giunto dovrà raggiungere la velocità target desiderata data la massima coppia applicata.

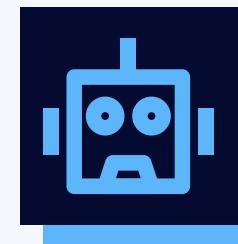
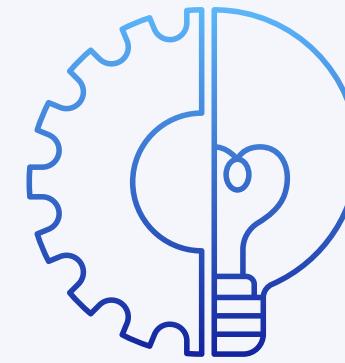


**Toque control mode**

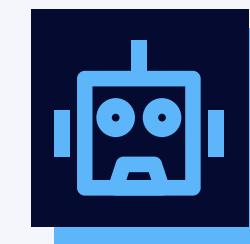
La modalità di controllo della coppia consente di stabilire il valore coppia agente sul motore.



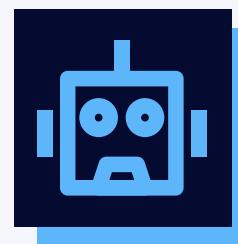
# Azionamento



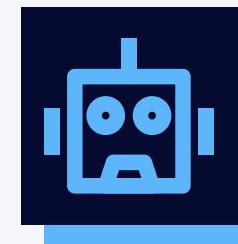
**controlMode**



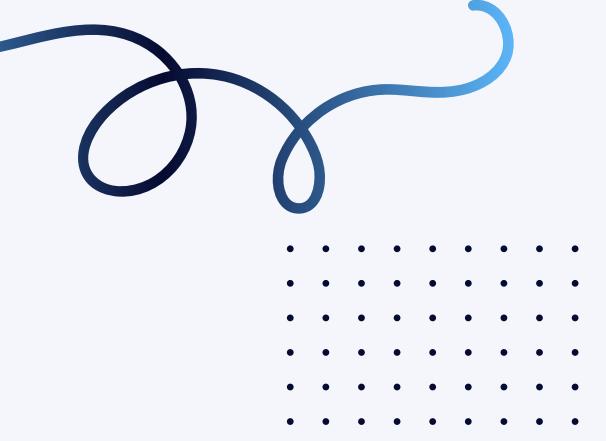
**p.POSITION\_CONTROL**,  
p.setJointMotorControl2( bodyIndex=body,  
jointIndex=joint,  
controlMode=p.POSITION\_  
CONTROL,  
targetPosition=pos,  
force=max\_force)



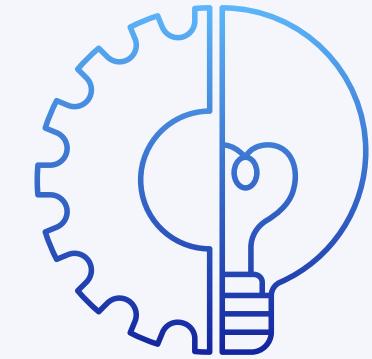
**p.VELOCITY\_CONTROL**  
p.setJointMotorControl2(  
bodyIndex=body,  
jointIndex=joint,  
controlMode=p.VELOCITY\_  
CONTROL,  
targetVelocity=vel,  
force=max\_force)



**p.TORQUE\_CONTROL**  
p.setJointMotorControl  
2(  
bodyIndex=body,  
jointIndex=joint,  
controlMode=p.TORQU  
E\_CONTROL,  
force=torque)



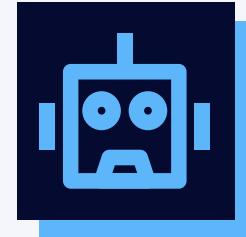
# Cinematica Inversa



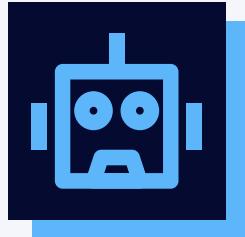
Nota la posizione dell'end effector qual è il valore delle varibili di giunto?

La cinematica inversa ci permette di passare dalle coordinate cartesiane agli angoli dei giunti passando dal world space system al joint space system.

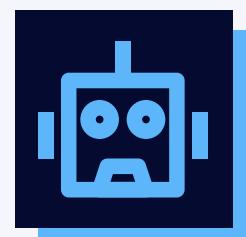
```
calculateInverseKinematics(self.pandaUid,11,newPosition, orientation)
```



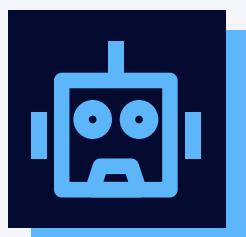
**loadURDF**  
"franka\_panda/panda.urdf"



**endEffectorLinkIndex**



**targetPosition  
(end effector)**



**targetOrientation**



# Esempio di physics simulation



**connect**: connessione al client-server (due server fisici integrati: DIRECT e GUI)



**setGravity**: informazioni sull'accelerazione di gravità



**resetSimulation**,  
**stepSimulation**: per concludere e iniziare la simulazione



```
import pybullet
import pybullet_data

pybullet.connect(pybullet.GUI)
# without GUI: pybullet.connect(pybullet.DIRECT)

pybullet.setGravity(0, 0, -9.81)

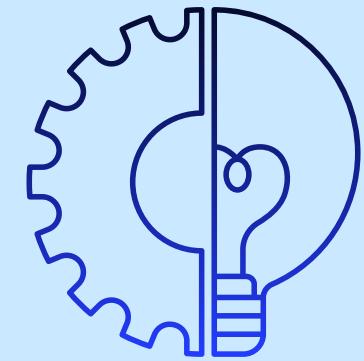
pybullet.setAdditionalSearchPath(pybullet_data.getDataPath())
cobot = pybullet.loadURDF("franka_panda/panda.urdf", [0, 0, 0],
[0, 0, 0, 1], useFixedBase = True)

pybullet.resetSimulation()

for step in range(500):
    joint_two_targ = np.random.uniform(jointLowerLimit,
jointUpperLimit)
    joint_four_targ = np.random.uniform(jointLowerLimit,
jointUpperLimit)
    p.setJointMotorControlArray(cobot, [2, 4],
p.POSITION_CONTROL, [joint_two_targ , joint_four_targ])
    p.stepSimulation()
    print(p.getJointStates(cobot, [2 , 4]))
```



# Analisi di un problema RL



## Inizializzazione

Si devono impostare delle condizioni al contorno, questo significa ad esempio stabilire le direzioni e le posizioni consentite e le forze da applicare.

## Step

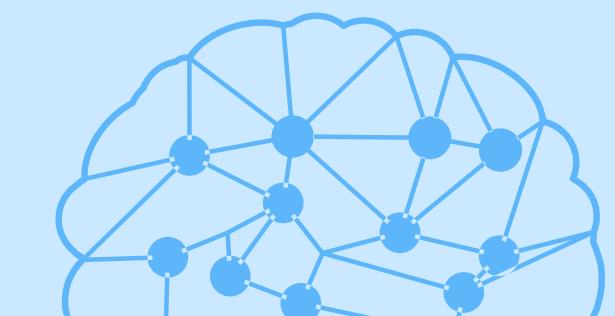
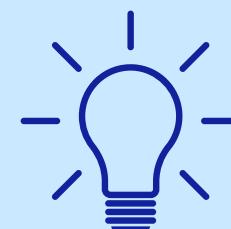
Rappresenta l'obiettivo e racchiude tutte le analisi effettuate nella fase di studio del problema fisico e il sistema di ricompense. Inoltre registra l'andamento dei parametri fornendo l'osservazione dei parametri di stato.

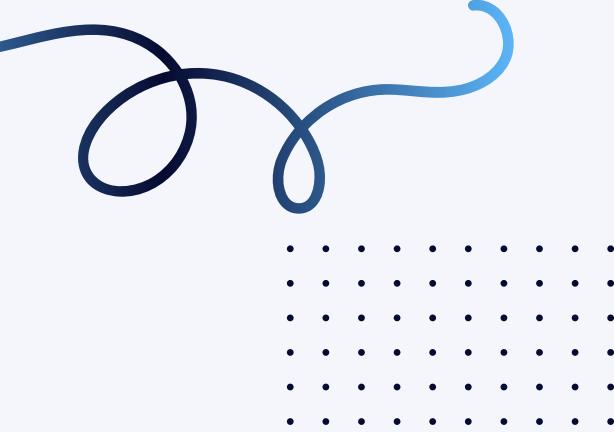
## Reset

Riporta il sistema nella configurazione di partenza.

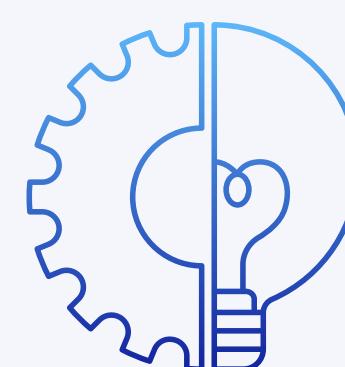
## Render

Fornisce un riscontro visivo della simulazione.

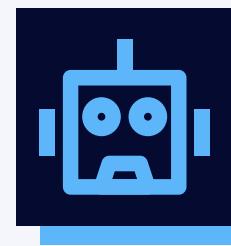




# Ambiente PandaEnv

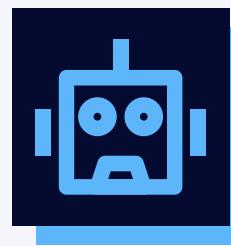


Lo sviluppo di un ambiente RL secondo lo standard OpenAI Gym si basa sui metodi:



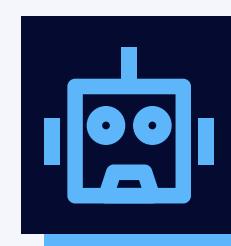
**\_\_init\_\_**

In cui sono presenti le informazioni relative alle azioni da intraprendere e le variabili da monitorare.



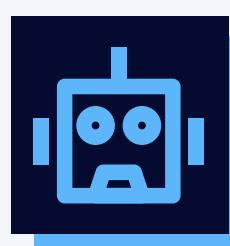
**Step()**

Eseguire un'azione che restituisce la corrente osservazione, ricompensa e l'indicazione sulla fine dell'episodio.



**Reset()**

Riporta l'ambiente al suo stato iniziale ottenendo la prima osservazione di inizializzazione.



**Render()**

Rappresenta in forma grafica l'ambiente.

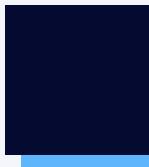
## Inizializzazione `__init__`



**action space:** insieme di azioni che possono essere eseguite, in questo caso sono di tipo continuo:

- **(x,y,z)end-effector che deve raggiungere con un  $d = 0.05$  il target**
- **variabile unica per la pinza**

**al fine di ottenere un array di dimensione 4**



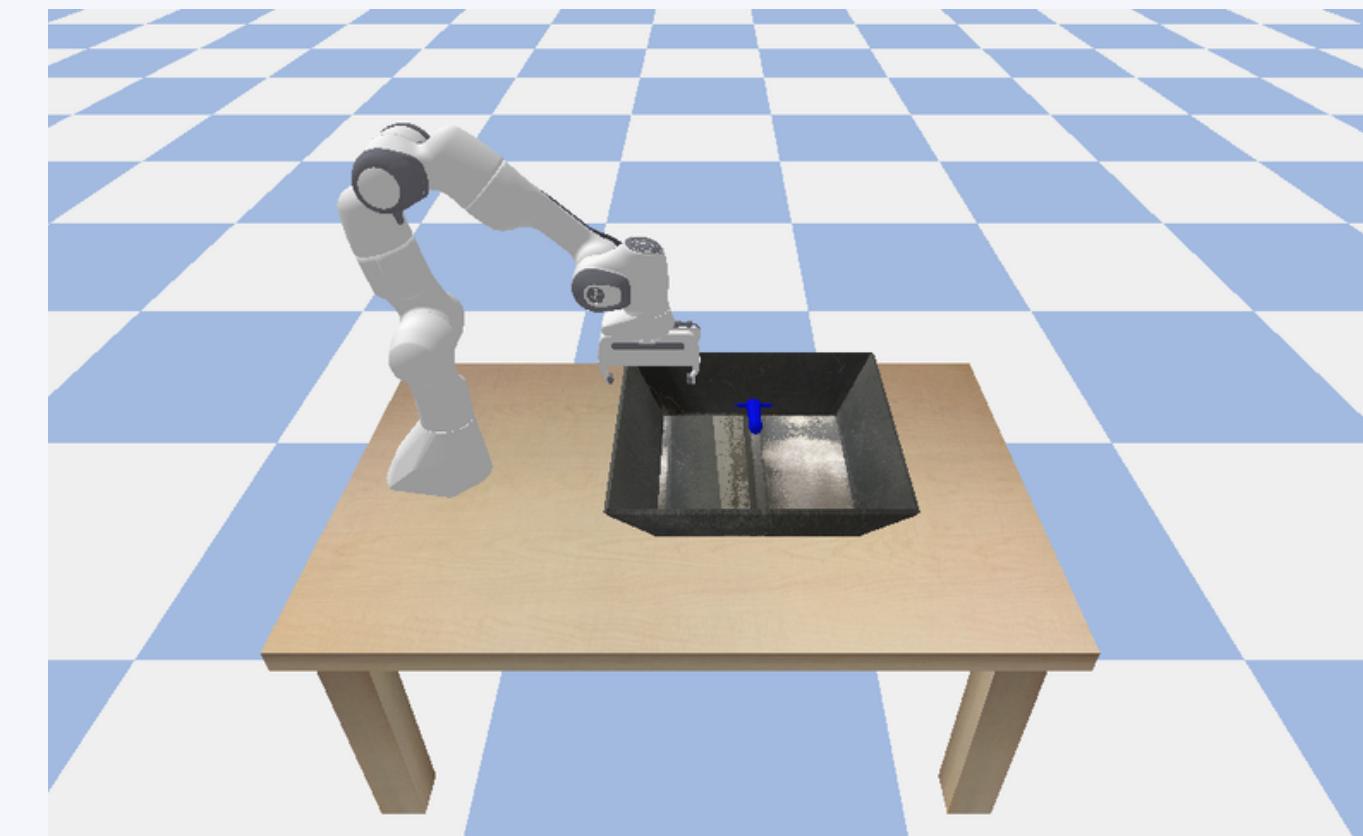
**observation space:** insieme di osservazioni fornite all'agente che identificano lo stato dell'ambiente nelle fasi temporali

- **(x,y,z)end-effector per ogni step raggiunto**
- **le due varibili di giunto**

**al fine di ottenere un array di dimensione 5**



Semplificazione: ipotizziamo che la pinza sia sempre rivolta verso il basso in modo da trascurare dell'orientamento.



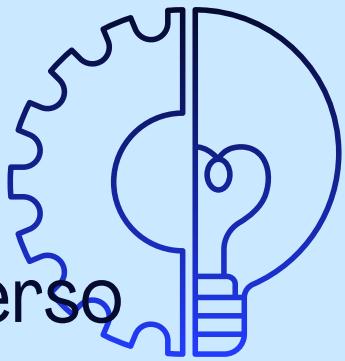
Obiettivo: raggiungere il target posizionato all'interno della scatola.



## Step()



**Determinazione della nuova posizione:** ad ogni step controlliamo il robot attraverso uno spostamento incrementale pari a dv verso la posizione cartesiana desiderata.



**calculateInverseKinematics()** per calcolare le variabili del giunto target per il robot, in input inseriamo la nuova posizione e l'orientamento (0, -pi/2, pi/2) con la pinza diretta verso il basso (per l'orientazione usiamo getQuaternionFromEuler()).

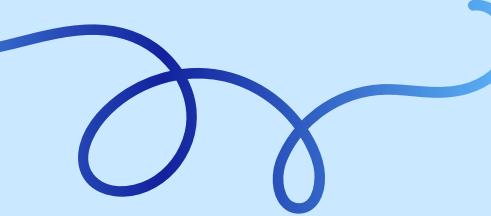


**setJointMotorControlArray()** ad ogni step e quindi per ciascun dv di spostamento registro la posizione dell'end effector utilizzando `getLinkState()` e della pinza utilizzando `getJointState()` ad ogni variazione calcolo la cinematica inversa e opero il controllo per il raggiungimento del target.

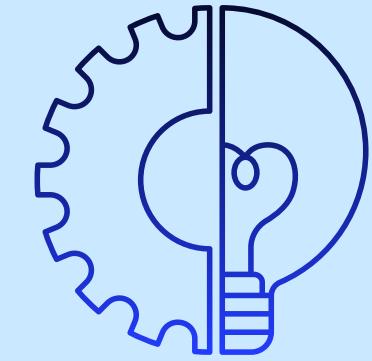


**Sistema di ricompensa:** se il robot afferra l'oggetto e lo raccoglie a una certa altezza pari a 0.3, l'agente ottiene una ricompensa pari a 10. L'episodio termina se si raggiungono il massimo numero di step pari a 20000 oppure se il robot raggiunge l'obiettivo.





## Reset()



**Posizione di restart:** ad ogni episodio il robot deve partire nella stessa posizione del precedente.

**Osservazione()** l'end effector, come per il metodo step, viene monitorato dalla funzione getLinkState(), mentre le variabili congiunte della pinza possono essere ottenute usando getJointState().



# Esempio di simulazione RL

- env.action\_space.sample()**: agente casuale che genera azioni random per testare le funzionalità quindi lo spostamento di end effector e pinze
- env.Reset()**: riporta il robot nella posizione di reset
- Tupla**: racchiude le informazioni per ogni step relative alle varibili osservate, ricompensa, successo e info



```
import gym

images = []
env = PandaEnv()
for i_episode in range(1):
    observation = env.reset()
    for t in range(20):
        #print(observation)

        action = env.action_space.sample()
        observation, reward, done, info = env.step(action)
        print(f" {observation} Reward {reward} Info {info}")
        img = env.render()
        images.append(img)
        if done:
            print("Episode finished after {}"
                  "timesteps".format(t+1))
            break
env.close()
```



# Pybullet e OpenAI Gym = Agente

Pybullet è diventata una libreria di riferimento anche per sviluppo di agenti RL. Attraverso stable\_baselines3 è possibile sviluppare agenti utilizzando gli algoritmi:

- PPO
- DQN
- A2C

Senza dover realizzare una rete neurale con tensorflow.

```
from stable_baselines3 import PPO
import gym

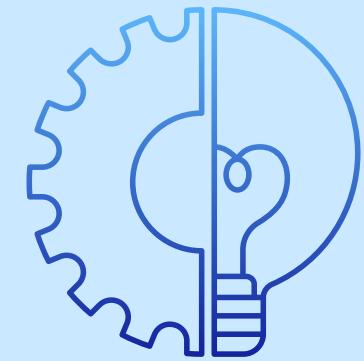
# Parallel environments
env = gym.make("pandaenv-v0")

model = PPO(policy = "MlpPolicy", env = env, verbose=1)
model.learn(total_timesteps=250) #25000

images = []
obs = env.reset()
for i in range(20): #20000
    action, _state = model.predict(obs, deterministic=True)
    obs, reward, done, info = env.step(action)
    print(f" {obs} Reward {reward} info {info}")
    #img = model.env.render(mode='rgb_array')
    #images.append(img)
    if done:
        obs = env.reset()
```



# Colaboratory e GPU



"Colab" permette di scrivere ed eseguire codice Python direttamente dal browser con i seguenti vantaggi:

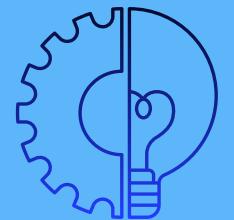
- Nessuna configurazione necessaria
- Accesso gratuito alle GPU

The screenshot shows the Google Drive interface. On the left, there's a sidebar with options like 'Cartella', 'Caricamento di file', 'Caricamento cartella', 'Documenti Google', 'Fogli Google', 'Presentazioni Google', 'Moduli Google', 'Altro', 'Archiviazione', and 'Acquista spazio di archiviazione'. In the main area, there's a search bar 'Cerca in Drive' and a list of files. One file, 'Linux Day 2021 RL Acrobot.ipynb', is open, showing code related to setting up environments. A context menu is open over this file, with 'Google Colaboratory' highlighted.

The screenshot shows the Google Colaboratory interface. At the top, there's a dropdown for 'Acceleratore hardware' with 'GPU' selected. Below it is a checkbox for 'Escludi output delle celle di codice durante il salvataggio del blocco note'. The main area shows a code editor with a command history and a 'Runtime' menu open. The 'Runtime' menu contains various options like 'Esegui tutte', 'Esegui prima', 'Esegui cella evidenziata', etc., along with a list of recent URLs.



## References



- 01** PyBullet Quickstart Guide  
[http://wiki.ros.org/urdf/Tutorials](#)
- 02** <https://github.com/bulletphysics>
- 03** <https://github.com/OrnellaFanais/LinuxDay2022.git>



# Thank You

"Che siano esperienze simulate o sogni, le informazioni sono al tempo stesso realtà e fantasia e in ogni caso, tutti i dati che una persona accumula durante il corso della propria esistenza non sono che una goccia nel mare."

*Batou, Ghost in the shell*

