Empieza a programar o a crear código con IA.

```
# 📌 INSTALAR DEPENDENCIAS EN GOOGLE COLAB
!sudo apt-get update --fix-missing
!sudo apt-get install -y xvfb ffmpeg
!pip install -U gym
!pip install pygame
!pip install keras
!pip install tensorflow
!pip install pyvirtualdisplay

# 🎥 HABILITAR EL RENDERIZADO EN COLAB
from pyvirtualdisplay import Display
display = Display(visible=0, size=(400, 300))
display.start()

print("¡Virtual Display iniciado correctamente!")

# 📌 IMPORTAR LIBRERÍAS NECESARIAS
import numpy as np
import pandas as pd
import gym
import random
import cv2
import base64
from collections import deque
from keras.models import Sequential
from keras.layers import Dense
from keras.optimizers import Adam
from keras.callbacks import TensorBoard
from IPython.display import HTML

# 📌 REGISTRAR MÉTRICAS PARA TENSORBOARD
tensorboard_callback = TensorBoard(log_dir="./logs")

# 🔥 DEFINICIÓN DEL AGENTE DQL
class DQLAgent():
    def __init__(self, env):
        self.state_size = env.observation_space.shape[0]
        self.action_size = env.action_space.n
        self.gamma = 0.95
        self.learning_rate = 0.001
        self.epsilon = 1.0
        self.epsilon_decay = 0.995
        self.epsilon_min = 0.01
        self.memory = deque(maxlen=1000)
        self.model = self.build_model()

    def build_model(self):
        model = Sequential()
        model.add(Dense(48, input_dim=self.state_size, activation='tanh'))
        model.add(Dense(self.action_size, activation='linear'))
        model.compile(loss='mse', optimizer=Adam(learning_rate=self.learning_rate))
        return model
```

```
        def remember(self, state, action, reward, next_state, done):
            self.memory.append((state, action, reward, next_state, done))

        def act(self, state):
            if random.uniform(0,1) <= self.epsilon:
                return env.action_space.sample()
            else:
                act_values = self.model.predict(state, verbose=0)
            return np.argmax(act_values[0])

        def replay(self, batch_size):
            if len(self.memory) < batch_size:
                return
            minibatch = random.sample(self.memory, batch_size)
            for state, action, reward, next_state, done in minibatch:
                target = reward if done else reward + self.gamma * np.amax(self.model.predict(next_state, verbose=0)[0])
                train_target = self.model.predict(state, verbose=0)
                train_target[0][action] = target
                self.model.fit(state, train_target, verbose=0, callbacks=[tensorboard_callback])

        def adaptiveEGreedy(self):
            if self.epsilon > self.epsilon_min:
                self.epsilon *= self.epsilon_decay

    # 📌 INICIALIZAR EL ENTORNO
    env = gym.make('CartPole-v1', render_mode="rgb_array")

    # 📌 ENTRENAMIENTO DEL AGENTE
    if __name__ == "__main__":
        agent = DQLAgent(env)
        batch_size = 16
        episodes = 10

        for e in range(episodes):
            state = env.reset()
            if isinstance(state, tuple):
                state = state[0]
            state = np.reshape(state, [1, 4])
            time = 0

            while True:
                action = agent.act(state)
                next_state, reward, done, _, _ = env.step(action)
                next_state = np.reshape(next_state, [1, 4])
                agent.remember(state, action, reward, next_state, done)
                agent.replay(batch_size)
                agent.adaptiveEGreedy()
                state = next_state

                if done:
                    print(f'Episode: {e}, Time: {time}')
                    break
                time += 1

            if e % 5 == 0:
                agent.model.save('cartpole_dql.keras')
```

```
    print("Entrenamiento finalizado. Guardando modelo...")
    agent.model.save('cartpole_dql_final.keras')

# 🎥 GRABAR VIDEO DEL AGENTE
def record_video(env, agent, video_path="cartpole_video.mp4", frames=500):
    obs = env.reset()
    if isinstance(obs, tuple):
        obs = obs[0]
    obs = np.reshape(obs, [1, 4])

    frame_shape = (600, 400)
    out = cv2.VideoWriter(video_path, cv2.VideoWriter_fourcc(*'mp4v'), 30, frame_shape)

    for _ in range(frames):
        frame = env.render()

        if frame is None:
            print("⚠️ Error: El frame renderizado es None.")
            break

        frame = cv2.cvtColor(frame, cv2.COLOR_RGB2BGR)
        frame = cv2.resize(frame, frame_shape)

        out.write(frame)

        action = np.argmax(agent.model.predict(obs, verbose=0))
        obs, _, done, _, _ = env.step(action)
        obs = np.reshape(obs, [1, 4])

        if done:
            break

    out.release()
    env.close()
    print("🎥 Video guardado correctamente en", video_path)

# 📌 LLAMAR A LA FUNCIÓN PARA GRABAR EL VIDEO
record_video(env, agent, "cartpole_video.mp4")

# 📌 FUNCIÓN PARA MOSTRAR EL VIDEO EN GOOGLE COLAB
def display_video(video_path):
    video_file = open(video_path, "rb").read()
    video_url = f"data:video/mp4;base64,{base64.b64encode(video_file).decode()}"
    return HTML(f'<video width="600" height="400" controls><source src="{video_url}" type="video/mp4"></video>')

# 📌 MOSTRAR EL VIDEO EN COLAB

!ffmpeg -i cartpole_video.mp4 -vcodec libx264 cartpole_video_fixed.mp4
from IPython.display import HTML
import base64

def display_video(video_path):
    video_file = open(video_path, "rb").read()
    video_url = f"data:video/mp4;base64,{base64.b64encode(video_file).decode()}"
    return HTML(f'<video width="600" height="400" controls><source src="{video_url}" type="video/mp4"></video>')
```

```
# 📌 Mostrar el video corregido en Colab
display_video("cartpole_video_fixed.mp4")
```

```
Hit:1 https://cloud.r-project.org/bin/linux/ubuntu jammy-cran40/ InRelease
Hit:2 https://developer.download.nvidia.com/compute/cuda/repos/ubuntu2204/x86_64  InRelease
Hit:3 http://archive.ubuntu.com/ubuntu jammy InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Hit:5 http://archive.ubuntu.com/ubuntu jammy-updates InRelease
Get:6 https://r2u.stat.illinois.edu/ubuntu jammy InRelease [6,555 B]
Hit:7 https://ppa.launchpadcontent.net/deadsnakes/ppa/ubuntu jammy InRelease
Hit:8 http://archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:9 https://ppa.launchpadcontent.net/graphics-drivers/ppa/ubuntu jammy InRelease
Hit:10 https://ppa.launchpadcontent.net/ubuntugis/ppa/ubuntu jammy InRelease
Fetched 6,555 B in 2s (4,079 B/s)
Reading package lists... Done
W: Skipping acquire of configured file 'main/source/Sources' as repository 'https://r2u.stat.illinois.edu/ubuntu jammy InRelease' does not seem to provide it (sources.list entry misspelt?)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
ffmpeg is already the newest version (7:4.4.2-0ubuntu0.22.04.1).
xvfb is already the newest version (2:21.1.4-2ubuntu1.7~22.04.13).
0 upgraded, 0 newly installed, 0 to remove and 31 not upgraded.
Requirement already satisfied: gym in /usr/local/lib/python3.11/dist-packages (0.26.2)
Requirement already satisfied: numpy>=1.18.0 in /usr/local/lib/python3.11/dist-packages (from gym) (1.26.4)
Requirement already satisfied: cloudpickle>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from gym) (3.1.1)
Requirement already satisfied: gym_notices>=0.0.4 in /usr/local/lib/python3.11/dist-packages (from gym) (0.0.8)
Requirement already satisfied: pygame in /usr/local/lib/python3.11/dist-packages (2.6.1)
Requirement already satisfied: keras in /usr/local/lib/python3.11/dist-packages (3.8.0)
Requirement already satisfied: absl-py in /usr/local/lib/python3.11/dist-packages (from keras) (1.4.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.11/dist-packages (from keras) (1.26.4)
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (from keras) (13.9.4)
Requirement already satisfied: namex in /usr/local/lib/python3.11/dist-packages (from keras) (0.0.8)
Requirement already satisfied: h5py in /usr/local/lib/python3.11/dist-packages (from keras) (3.12.1)
Requirement already satisfied: optree in /usr/local/lib/python3.11/dist-packages (from keras) (0.14.0)
Requirement already satisfied: ml-dtypes in /usr/local/lib/python3.11/dist-packages (from keras) (0.4.1)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from keras) (24.2)
Requirement already satisfied: typing-extensions>=4.5.0 in /usr/local/lib/python3.11/dist-packages (from optree->keras) (4.12.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich->keras) (2.18.0)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich->keras) (0.1.2)
Requirement already satisfied: tensorflow in /usr/local/lib/python3.11/dist-packages (2.18.0)
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: flatbuffers>=24.3.25 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (25.2.10)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.6.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (18.1.1)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.11/dist-packages (from tensorflow) (24.2)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<6.0.0dev,>=3.20.3 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (4.25.6)
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.32.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.11/dist-packages (from tensorflow) (75.1.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.0)
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.5.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (4.12.2)
Requirement already satisfied: wrapt>=1.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.17.2)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.70.0)
Requirement already satisfied: tensorboard<2.19,>=2.18 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (2.18.0)
Requirement already satisfied: keras>=3.5.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.8.0)
Requirement already satisfied: numpy<2.1.0,>=1.26.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (1.26.4)
Requirement already satisfied: h5py>=3.11.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (3.12.1)
Requirement already satisfied: ml-dtypes<0.5.0,>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.4.1)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.11/dist-packages (from tensorflow) (0.37.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.11/dist-packages (from astunparse>=1.6.0->tensorflow) (0.45.1)
Requirement already satisfied: rich in /usr/local/lib/python3.11/dist-packages (from keras>=3.5.0->tensorflow) (13.9.4)
```