

Install and import necessary packages

```
In [ ]: !pip install gym
!apt-get install python-opengl -y
!apt install xvfb -y

!pip install gym[atari]

!pip install pyvirtualdisplay
!pip install piglet
```

```

Requirement already satisfied: gym in /usr/local/lib/python3.10/dist-packages (0.25.2)
Requirement already satisfied: numpy>=1.18.0 in /usr/local/lib/python3.10/dist-packages (from gym) (1.23.5)
Requirement already satisfied: cloudpickle>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from gym) (2.2.1)
Requirement already satisfied: gym-notices>=0.0.4 in /usr/local/lib/python3.10/dist-packages (from gym) (0.0.8)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package python-opengl
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libfontenc1 libxfont2 libxkbfile1 x11-xkb-utils xfonts-base xfonts-encodings xfonts-utils
  xserver-common
The following NEW packages will be installed:
  libfontenc1 libxfont2 libxkbfile1 x11-xkb-utils xfonts-base xfonts-encodings xfonts-utils
  xserver-common xvfb
0 upgraded, 9 newly installed, 0 to remove and 32 not upgraded.
Need to get 7,814 kB of archives.
After this operation, 11.9 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/main amd64 libfontenc1 amd64 1:1.1.4-1build3 [14.7 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 libxfont2 amd64 1:2.0.5-1build1 [94.5 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy/main amd64 libxkbfile1 amd64 1:1.1.0-1build3 [71.8 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy/main amd64 x11-xkb-utils amd64 7.7+5build4 [172 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy/main amd64 xfonts-encodings all 1:1.0.5-0ubuntu2 [578 kB]
Get:6 http://archive.ubuntu.com/ubuntu jammy/main amd64 xfonts-utils amd64 1:7.7+6build2 [94.6 kB]
Get:7 http://archive.ubuntu.com/ubuntu jammy/main amd64 xfonts-base all 1:1.0.5 [5,896 kB]
Get:8 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 xserver-common all 2:21.1.4-2ubuntu1.7~22.04.8 [28.6 kB]
Get:9 http://archive.ubuntu.com/ubuntu jammy-updates/universe amd64 xvfb amd64 2:21.1.4-2ubuntu1.7~22.04.8 [863 kB]
Fetched 7,814 kB in 1s (6,305 kB/s)
Selecting previously unselected package libfontenc1:amd64.
(Reading database ... 121730 files and directories currently installed.)
Preparing to unpack .../0-libfontenc1_1%3a1.1.4-1build3_amd64.deb ...
Unpacking libfontenc1:amd64 (1:1.1.4-1build3) ...
Selecting previously unselected package libxfont2:amd64.
Preparing to unpack .../1-libxfont2_1%3a2.0.5-1build1_amd64.deb ...
Unpacking libxfont2:amd64 (1:2.0.5-1build1) ...
Selecting previously unselected package libxkbfile1:amd64.
Preparing to unpack .../2-libxkbfile1_1%3a1.1.0-1build3_amd64.deb ...
Unpacking libxkbfile1:amd64 (1:1.1.0-1build3) ...
Selecting previously unselected package x11-xkb-utils.
Preparing to unpack .../3-x11-xkb-utils_7.7+5build4_amd64.deb ...
Unpacking x11-xkb-utils (7.7+5build4) ...
Selecting previously unselected package xfonts-encodings.
Preparing to unpack .../4-xfonts-encodings_1%3a1.0.5-0ubuntu2_all.deb ...

```

```

Unpacking xfonts-encodings (1:1.0.5-0ubuntu2) ...
Selecting previously unselected package xfonts-utils.
Preparing to unpack .../5-xfonts-utils_1%3a7.7+6build2_amd64.deb ...
Unpacking xfonts-utils (1:7.7+6build2) ...
Selecting previously unselected package xfonts-base.
Preparing to unpack .../6-xfonts-base_1%3a1.0.5_all.deb ...
Unpacking xfonts-base (1:1.0.5) ...
Selecting previously unselected package xserver-common.
Preparing to unpack .../7-xserver-common_2%3a21.1.4-2ubuntu1.7~22.04.8_all.deb
...
Unpacking xserver-common (2:21.1.4-2ubuntu1.7~22.04.8) ...
Selecting previously unselected package xvfb.
Preparing to unpack .../8-xvfb_2%3a21.1.4-2ubuntu1.7~22.04.8_amd64.deb ...
Unpacking xvfb (2:21.1.4-2ubuntu1.7~22.04.8) ...
Setting up libfontenc1:amd64 (1:1.1.4-1build3) ...
Setting up xfonts-encodings (1:1.0.5-0ubuntu2) ...
Setting up libxkbfile1:amd64 (1:1.1.0-1build3) ...
Setting up libxfont2:amd64 (1:2.0.5-1build1) ...
Setting up x11-xkb-utils (7.7+5build4) ...
Setting up xfonts-utils (1:7.7+6build2) ...
Setting up xfonts-base (1:1.0.5) ...
Setting up xserver-common (2:21.1.4-2ubuntu1.7~22.04.8) ...
Setting up xvfb (2:21.1.4-2ubuntu1.7~22.04.8) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for fontconfig (2.13.1-4.2ubuntu5) ...
Processing triggers for libc-bin (2.35-0ubuntu3.4) ...
/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc_proxy.so.2 is not a symbolic lin
k

/sbin/ldconfig.real: /usr/local/lib/libtbbbind.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbb.so.12 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbmalloc.so.2 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_0.so.3 is not a symbolic link

/sbin/ldconfig.real: /usr/local/lib/libtbbbind_2_5.so.3 is not a symbolic link

Requirement already satisfied: gym[atari] in /usr/local/lib/python3.10/dist-packa
ges (0.25.2)
Requirement already satisfied: numpy>=1.18.0 in /usr/local/lib/python3.10/dist-pa
ckages (from gym[atari]) (1.23.5)
Requirement already satisfied: cloudpickle>=1.2.0 in /usr/local/lib/python3.10/di
st-packages (from gym[atari]) (2.2.1)
Requirement already satisfied: gym-notices>=0.0.4 in /usr/local/lib/python3.10/di
st-packages (from gym[atari]) (0.0.8)
Collecting ale-py~0.7.5 (from gym[atari])
  Downloading ale_py-0.7.5-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_6
4.whl (1.6 MB)
    _____ 1.6/1.6 MB 10.7 MB/s eta 0:00:00
Requirement already satisfied: importlib-resources in /usr/local/lib/python3.10/d
ist-packages (from ale-py~0.7.5->gym[atari]) (6.1.1)
Installing collected packages: ale-py
Successfully installed ale-py-0.7.5
Collecting pyvirtualdisplay
  Downloading PyVirtualDisplay-3.0-py3-none-any.whl (15 kB)
Installing collected packages: pyvirtualdisplay
Successfully installed pyvirtualdisplay-3.0
Collecting piglet

```

```

Downloading piglet-1.0.0-py2.py3-none-any.whl (2.2 kB)
Collecting piglet-templates (from piglet)
  Downloading piglet_templates-1.3.0-py3-none-any.whl (67 kB)
    67.5/67.5 kB 2.4 MB/s eta 0:00:00
Requirement already satisfied: pyparsing in /usr/local/lib/python3.10/dist-packag
es (from piglet-templates->piglet) (3.1.1)
Requirement already satisfied: attrs in /usr/local/lib/python3.10/dist-packages
(from piglet-templates->piglet) (23.2.0)
Requirement already satisfied: astunparse in /usr/local/lib/python3.10/dist-packa
ges (from piglet-templates->piglet) (1.6.3)
Requirement already satisfied: markupsafe in /usr/local/lib/python3.10/dist-packa
ges (from piglet-templates->piglet) (2.1.4)
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/di
st-packages (from astunparse->piglet-templates->piglet) (0.42.0)
Requirement already satisfied: six<2.0,>=1.6.1 in /usr/local/lib/python3.10/dist-
packages (from astunparse->piglet-templates->piglet) (1.16.0)
Installing collected packages: piglet-templates, piglet
Successfully installed piglet-1.0.0 piglet-templates-1.3.0

```

To activate virtual display we need to run a script once for training an agent, as follows:

```

In [ ]: from pyvirtualdisplay import Display
display = Display(visible=0, size=(1400, 900))
display.start()

```

```

Out[ ]: <pyvirtualdisplay.display.Display at 0x7fa7f6f5d780>

```

```

In [ ]: # This code creates a virtual display to draw game images on.
# If you are running locally, just ignore it
import os
if type(os.environ.get("DISPLAY")) is not str or len(os.environ.get("DISPLAY"))=
!bash ../xvfb start
%env DISPLAY=:1

```

```

In [ ]: import gym
from gym import logger as gymlogger
from gym.wrappers.record_video import RecordVideo
gymlogger.set_level(40) # error only
import tensorflow as tf
import numpy as np
import random
import matplotlib
import matplotlib.pyplot as plt
%matplotlib inline
import math
import glob
import io
import base64
from IPython.display import HTML

from IPython import display as ipythondisplay

"""
Utility functions to enable video recording of gym environment and displaying it
To enable video, just do "env = wrap_env(env)"
"""

def show_video():
    mp4list = glob.glob('video/*.mp4')

```

```

if len(mp4list) > 0:
    mp4 = mp4list[0]
    video = io.open(mp4, 'r+b').read()
    encoded = base64.b64encode(video)
    ipythondisplay.display(HTML(data='''<video alt="test" autoplay
                                loop controls style="height: 400px;">
                                <source src="data:video/mp4;base64,{0}" type="video/mp4" />
                                </video>'''.format(encoded.decode('ascii'))))
else:
    print("Could not find video")

def wrap_env(env):
    env = RecordVideo(env, './video')
    return env

```

OpenAI Gym

OpenAI gym is a python library that wraps many classical decision problems including robot control, videogames and board games. We will use the environments it provides to test our algorithms on interesting decision problems.

Gym documentation: https://www.gymlibrary.dev/content/basic_usage/#

Refer to the docstrings of github code to understand the attributes of the environment.

<https://github.com/openai/gym/blob/dcd185843a62953e27c2d54dc8c2d647d604b635/gym/L18C20>



Environment object

- `env.observation_space` : state space, all possible states.
- `env.action_space` : all possible actions the agent can take.
- `env.state` : Current state the agent is in.
- `env.reset()` : reset environment to initial state, return first observation
- `env.render()`: show current state.
- `env.step(action)` : commit action a and return (new observation, reward, is done, info)

MountainCar

```

In [ ]: # env = gym.make('CartPole-v0')
env = gym.make('MountainCar-v0')

# env = wrap_env(env)

print('observation space:', env.observation_space)

```

```

print('action space:', env.action_space)

obs = env.reset()

print('initial observation:', obs)

action = env.action_space.sample() # take a random action
print("action: ", action)

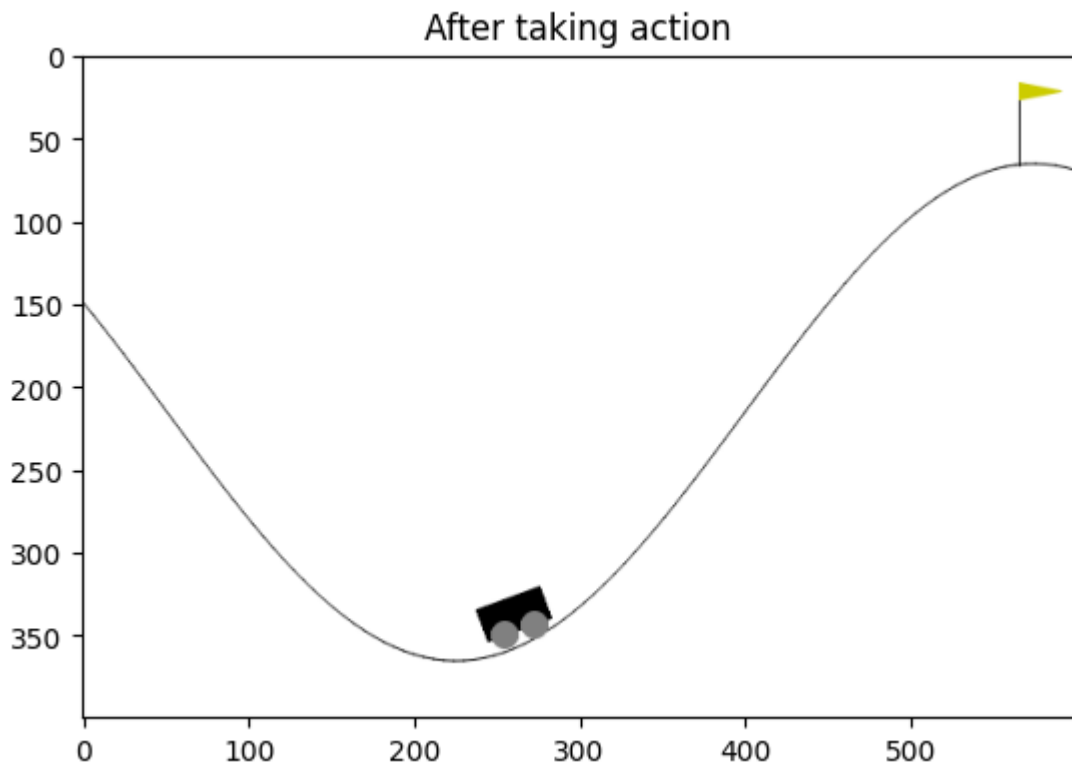
obs, r, done, info = env.step(action)
print('\nnext observation:', obs)
print('reward:', r)
print('done:', done)
print('info:', info)
plt.title("After taking action")
plt.imshow(env.render('rgb_array'))

```

observation space: Box([-1.2 -0.07], [0.6 0.07], (2,), float32)
 action space: Discrete(3)
 initial observation: [-0.4057507 0.]
 action: 1

next observation: [-0.40661624 -0.00086556]
 reward: -1.0
 done: False
 info: {}

Out[]: <matplotlib.image.AxesImage at 0x7fa766ca2f20>



Random action

```

In [ ]: '''CartPole problem use random action'''
# env = gym.make('CartPole-v0')
env = gym.make('MountainCar-v0')
env = wrap_env(env) # defined before for rendering online

```

```
observation = env.reset()

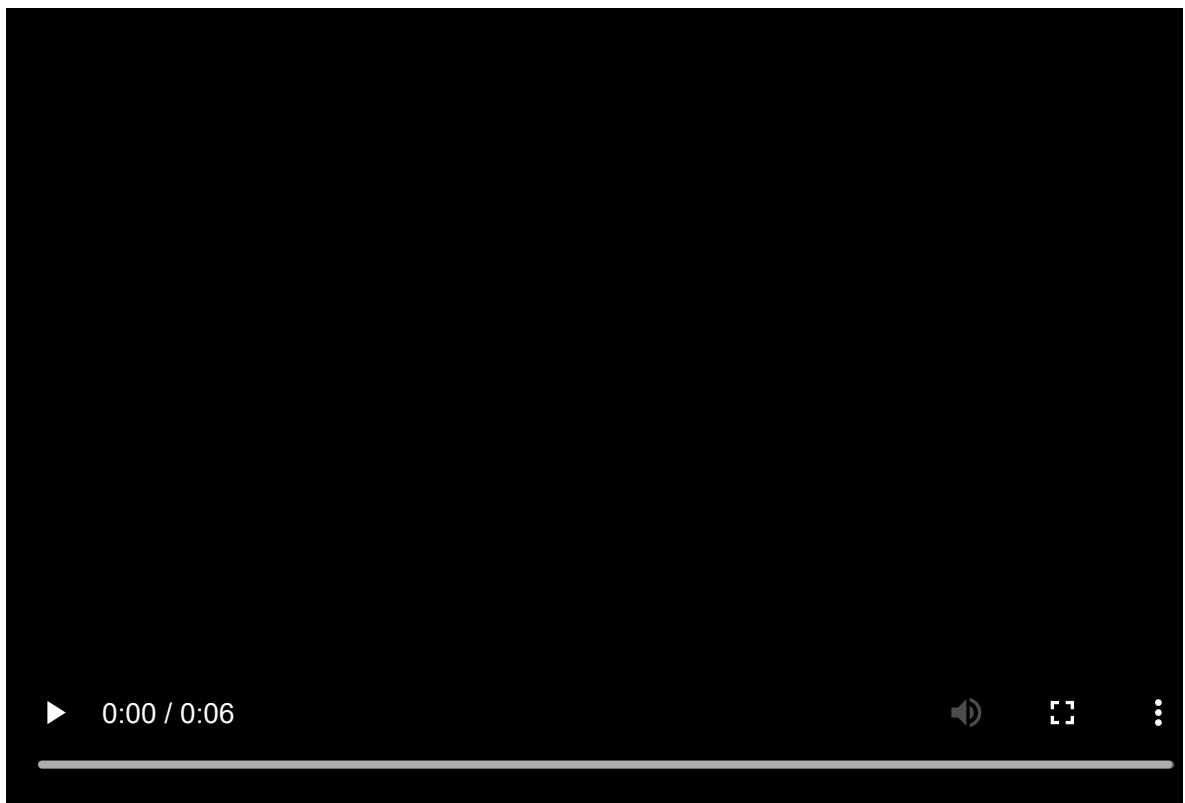
total_reward = 0

while True:
    env.render()

    # your agent goes here
    action = env.action_space.sample() # take a random action
    observation, reward, done, info = env.step(action)
    # print(reward)
    total_reward += reward

    if done:
        break;

env.close()
show_video()
print(total_reward)
```



-200.0

Intutive action

- Accelerate to the left when car is on the left.
- Accelerate to the right when the car is on the right.

```
In [ ]: env = gym.make('MountainCar-v0')
```

```
In [ ]: def policy(env):
        if env.state[1]>0:
            action = 2
        else:
```

```
    action = 1
    return action
```

```
In [ ]: env = gym.make('MountainCar-v0')
env = wrap_env(env) # defined before for rendering online

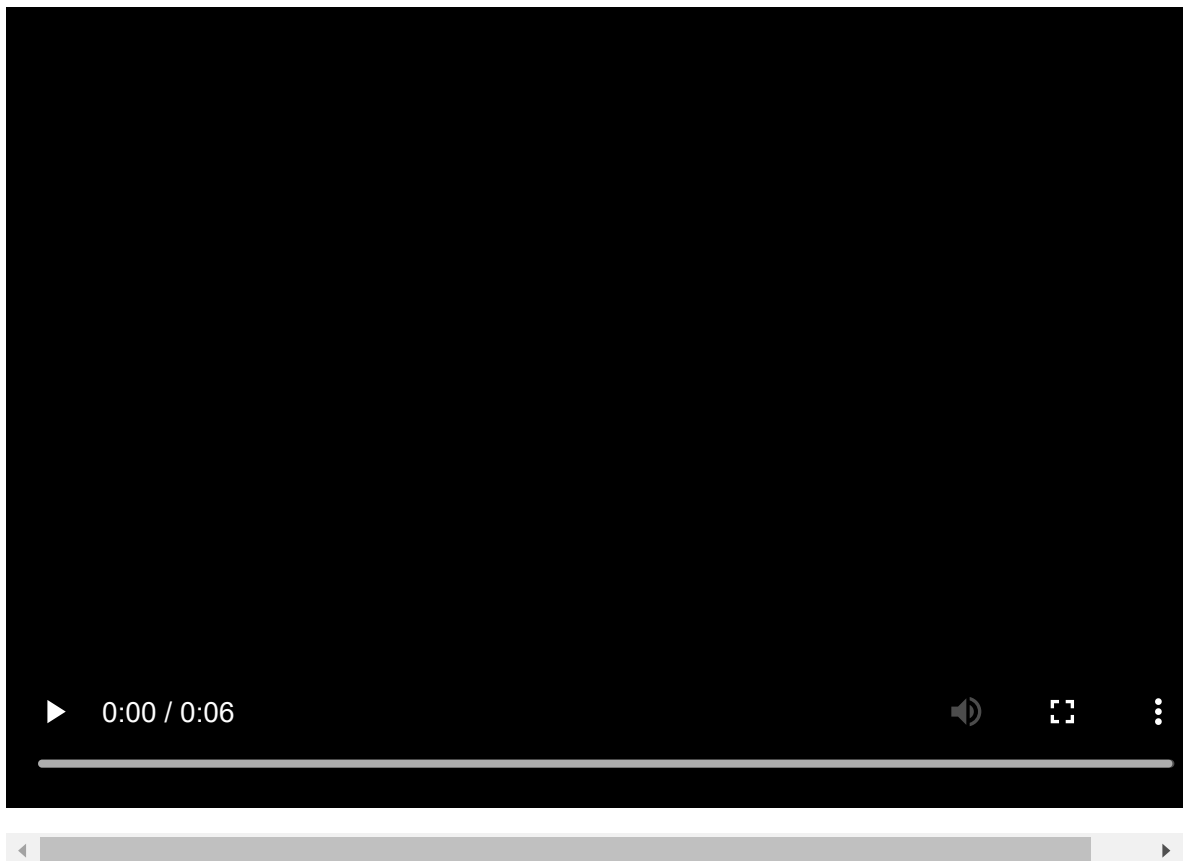
observation = env.reset()

while True:
    # env.render()

    # your agent goes here
    action = policy(env) # take a random action
    observation, reward, done, info = env.step(action)
    # print(reward)

    if done:
        break;

env.close()
show_video()
```



Frozen Lake

```
In [ ]: env = gym.make('FrozenLake-v1', desc=None, map_name="4x4", is_slippery=True)
print("State space: ", env.observation_space)
print("Action space: ", env.action_space)
env.reset()
print("Current state: ", env.s)
```


State space: Discrete(16)
Action space: Discrete(4)
Current state: 0

Random action

```
In [ ]: env = gym.make('FrozenLake-v1', desc=None, map_name="4x4", is_slippery=True)
env = wrap_env(env) # defined before for rendering online

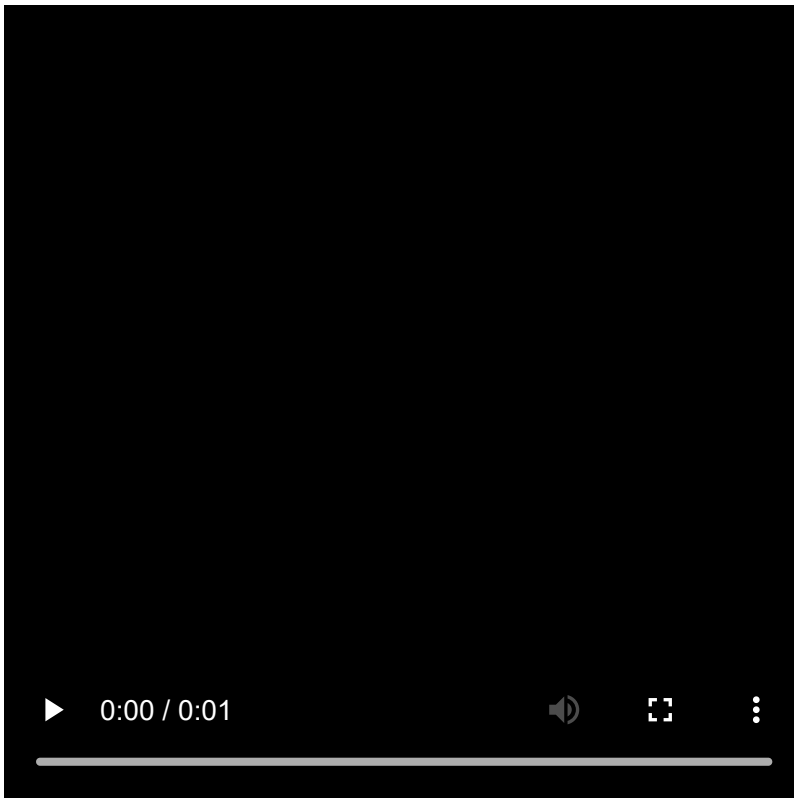
observation = env.reset()


while True:
    # your agent goes here
    action = env.action_space.sample() # take a random action
    print(action)
    observation, reward, done, info = env.step(action)
    # print(reward)

    if done:
        break;

env.close()
show_video()
```

2
0
2
1
3
2



 Select a [gym environment](#) and understand the environment's states, actions and rewards by going through the Openai Gym's [documentation](#) and [github codes](#).

- Explain in your own words what the env is about and what needs to be achieved. Give a one liner explaining what the observation space and action space is for the selected environment.
- Import the environment and make the agent take "random actions". Print the total reward the agent received at the end of an episode.
- Calculate the average total reward by simulating multiple episodes. (atleast 1000)
- Try modifying the actions to maximise the total reward. If the chosen environment is complex to hardcode the actions, you can just explain in your own words.

```
In [ ]: env = gym.make("Taxi-v3")
print("State Space: ",env.observation_space)
print("Action Space: ",env.action_space)
env.reset()

print("Current State: ",env.s)
```

```
State Space: Discrete(500)
Action Space: Discrete(6)
Current State: 27
```

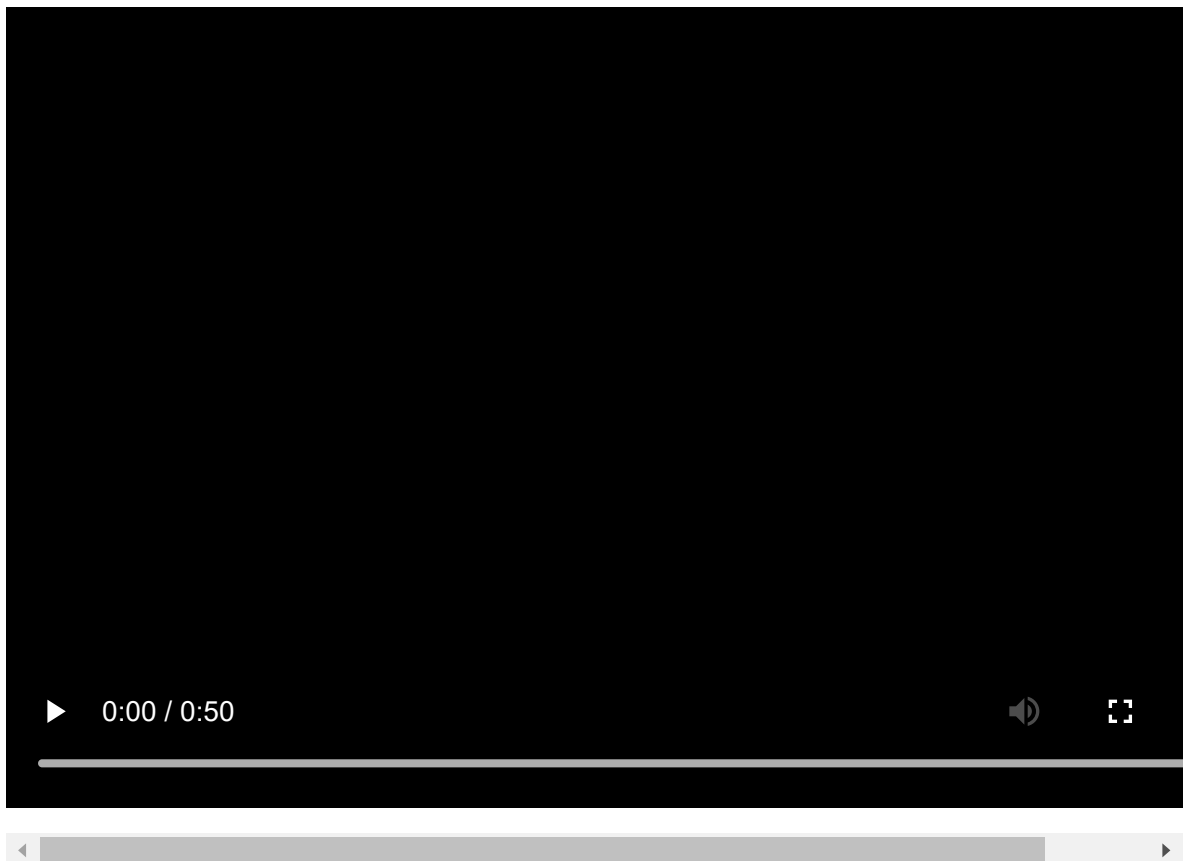
```
In [ ]: env = gym.make("Taxi-v3")
env = wrap_env(env)

observation = env.reset()

while True:
    action = env.action_space.sample()
    observation, reward, done, info = env.step(action)

    if done:
        break

env.close()
show_video()
```



Explain in your own words what the env is about and what needs to be achieved. Give a one liner explaining what the observation space and action space is for the selected environment.

The Taxi-v3 environment is a classic gridworld problem with a taxi navigating through a 5x5 grid. The taxi has to pick up a passenger at one location and drop them off at another location. The grid also contains walls, and there are four special locations - the passenger's starting position, the passenger's destination, the taxi's current position, and the destination for the taxi drop-off.

The observation space is a discrete space of size 500, representing the 25x5 grid cells of the environment. Each state corresponds to the possible combinations of the taxi's location, the passenger's location, and the destination location. The state is represented as a single integer between 0 and 499.

The action space is a discrete space of size 6, representing the six possible actions the agent can take at each state: move south, north, east, west, pick up the passenger, or drop off the passenger. The actions are represented as integers: 0 (south), 1 (north), 2 (east), 3 (west), 4 (pickup), and 5 (drop-off).

In this environment, the agent receives a reward of +20 for a successful drop-off, -1 for each time step, -10 for illegal pick-up or drop-off actions, and -1 for attempting to move through a wall. The goal of the agent is to maximize its cumulative reward while successfully picking up and dropping off the passenger.

Calculate the average total reward by simulating multiple episodes. (atleast 1000)

```
In [ ]: env = gym.make("Taxi-v3")
env = wrap_env(env) # defined before for rendering online
# Number of episodes to simulate
num_episodes = 1000

# List to store total rewards for each episode
total_rewards = []

# Simulate episodes
for episode in range(num_episodes):
    state = env.reset()
    episode_reward = 0

    # Run the episode until it's done
    while True:
        # Replace this line with your agent's policy to choose an action
        action = env.action_space.sample()

        # Take a step in the environment
        next_state, reward, done, _ = env.step(action)

        # Accumulate the reward for this step
        episode_reward += reward

        # Check if the episode is done
        if done:
            break

        # Update the current state for the next step
        state = next_state

    # Store the total reward for this episode
    total_rewards.append(episode_reward)

# Calculate the average total reward
average_reward = np.mean(total_rewards)
print(average_reward)
```

/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass the result to `transformed_cell` argument and any exception that happen during the transform in `preprocessing_exc_tuple` in IPython 7.17 and above.

and should_run_async(code)

-773.384

Try modifying the actions to maximise the total reward. If the chosen environment is complex to hardcode the actions, you can just explain in your own words.

```
In [ ]: import gym

def taxi_policy(observation):
    # Extract relevant information from the observation
    row = observation // 5
    col = observation % 5
```

```
    if row == 3 and col < 4: # If the agent is on the bottom row and not at the
        action = 0 # Move up
    elif col < 4: # If the agent is not at the goal
        action = 2 # Move down
    elif col == 4: # If the agent is at the goal
        action = 5 # Drop off
    else:
        action = 1 # Move right

    return action

# Create the Taxi-v3 environment
env = gym.make('Taxi-v3')

# Wrap the environment for video recording
env = wrap_env(env)

observation = env.reset()
total_reward = 0

while True:
    env.render() # Uncomment this line if you want to visualize the environment

    # Your agent's policy
    action = taxi_policy(observation)

    # Take a step in the environment
    observation, reward, done, info = env.step(action)

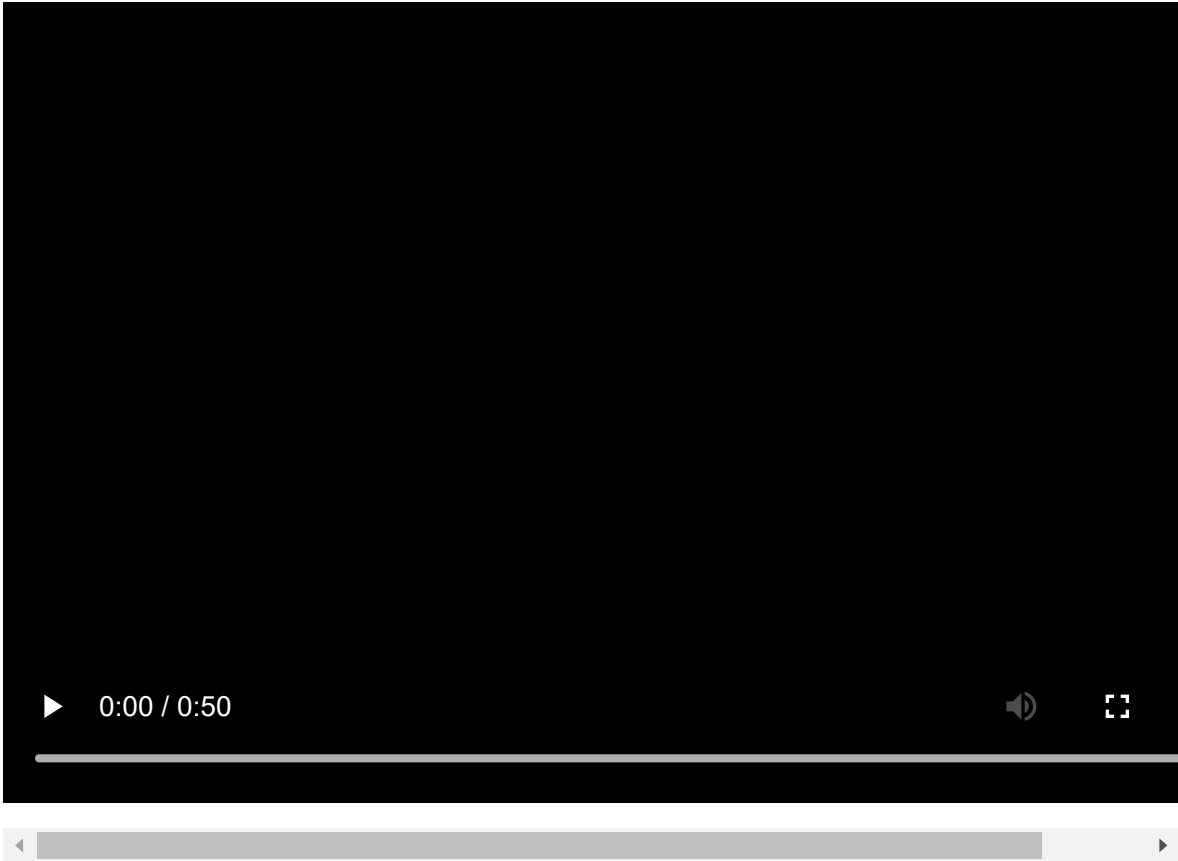
    total_reward += reward

    if done:
        break

env.close()

# Show the recorded video

# Display the recorded video
show_video()
print("Total Reward:", total_reward)
```



Total Reward: -2000