Install and import necessary packages

```
!pip install gym
!apt-get install python-opengl -y
!apt install xvfb -y

!pip install gym[atari]

!pip install pyvirtualdisplay
!pip install piglet
```

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

```
from pyvirtualdisplay import Display
display = Display(visible=0, size=(1400, 900))
display.start()


# 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"))==0:
    !bash ../xvfb start
    %env DISPLAY=:1
```

```
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/envs/classic_control/mountain_car.py#L18C1-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

```
# 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'))
```

Random action

```
'''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)
```

Intutive action

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

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


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

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

observation = env.reset()

while True:
  # env.render()
```

Frozen Lake

```
  observation, reward, done, info = env.step(action)

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)

show_video()
```

Random action

```
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()
```

📔 Select a gym environment and understand the evironment'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