

Practice OpenAI Gym

1. Implementation in OpenAI Gym

Install python in your system after that follow the given steps:

Steps:

1. Install virtualenv:

```
$pip install virtualenv
```

If pip is not installed on your system, you can install it by typing

```
sudo easy_install pip
```

2. Create a virtual environment named openai-gym using the virtualenv tool:

```
$virtualenv openai-gym
```

3. Activate the openai-gym virtual environment:

```
$source openai-gym/bin/activate
```

4. Install all the packages for the Gym toolkit from upstream:

```
$pip install -U gym
```

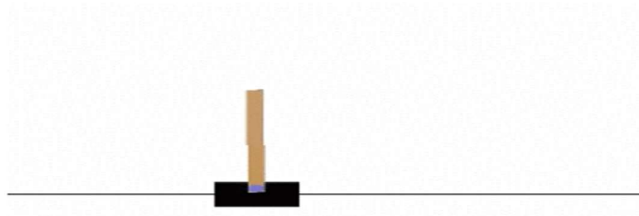
5. Test to make sure the installation is successful:

```
$python -c 'import gym; gym.make("CartPole-v0");'
```

2. Classic control Example: These are different classic controls for small-scale reinforcement learning tasks, mainly from reinforcement learning literature.

Cartpole

The CartPole problem is also known as the “Inverted Pendulum” problem. It has a pole attached to a cart. Since the pole’s center of mass is above its pivot point, it’s an unstable system. The pole starts in an upright position with a small perturbation. The goal is to move the cart left and right to keep the pole from falling.



```
import gym
env = gym.make('CartPole-v0') #CartPole is just an example.
env.reset()
env.render()
```

```
Anaconda Prompt (anaconda3)

(base) C:\Users\91751>pip install virtualenv
'$pip' is not recognized as an internal or external command,
operable program or batch file.

(base) C:\Users\91751>pip install virtualenv
Collecting virtualenv
  Downloading virtualenv-20.15.0-py2.py3-none-any.whl (10.1 MB)
    | 10.1 MB 2.2 MB/s
Requirement already satisfied: six<2,>=1.9.0 in c:\users\91751\anaconda3\lib\site-packages (from virtualenv) (1.15.0)
Collecting filelock<4,>=3.2
  Downloading filelock-3.7.1-py3-none-any.whl (10 kB)
Collecting distlib<1,>=0.3.1
  Downloading distlib-0.3.4-py2.py3-none-any.whl (461 kB)
    | 461 kB 3.3 MB/s
Collecting platformdirs<3,>=2
  Downloading platformdirs-2.5.2-py3-none-any.whl (14 kB)
Installing collected packages: platformdirs, filelock, distlib, virtualenv
  Attempting uninstall: filelock
    Found existing installation: filelock 3.0.12
    Uninstalling filelock-3.0.12:
      Successfully uninstalled filelock-3.0.12
  Successfully installed distlib-0.3.4 filelock-3.7.1 platformdirs-2.5.2 virtualenv-20.15.0

(base) C:\Users\91751>virtualenv openai-gym
created virtual environment CPython3.8.8.final.0-64 in 12067ms
  creator CPythonWindows(dest=C:\Users\91751\openai-gym, clear=False, no_vcs_ignore=False, global=False)
  seeder FromAppData(download=False, pip=bundle, setuptools=bundle, wheel=bundle, via=copy, app_data_dir=C:\Users\91751\AppData\Local\pypa\virtualenv)
    added seed packages: pip==22.1.2, setuptools==62.6.0, wheel==0.37.1
  activators BashActivator,BatchActivator,FishActivator,NushellActivator,PowerShellActivator,PythonActivator

(base) C:\Users\91751>pip install -U gym
Requirement already satisfied: gym in c:\users\91751\anaconda3\lib\site-packages (0.24.1)
Requirement already satisfied: gym-notices>=0.0.4 in c:\users\91751\anaconda3\lib\site-packages (from gym) (0.0.7)
Requirement already satisfied: importlib-metadata>=4.8.0 in c:\users\91751\anaconda3\lib\site-packages (from gym) (4.12.0)
Requirement already satisfied: numpy>=1.18.0 in c:\users\91751\anaconda3\lib\site-packages (from gym) (1.20.1)
Requirement already satisfied: cloudpickle>=1.2.0 in c:\users\91751\anaconda3\lib\site-packages (from gym) (1.6.0)
Requirement already satisfied: zipp>=0.5 in c:\users\91751\anaconda3\lib\site-packages (from importlib-metadata>=4.8.0->gym) (3.4.1)

Jupyter Untitled4 Last Checkpoint: 12/30/2020 (unsaved changes)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3
In [1]: import gym
env = gym.make('CartPole-v0')
env.reset()
env.render()

C:\Users\91751\anaconda3\lib\site-packages\gym\envs\register.py:102: DeprecationWarning: Call to deprecated method register of class GymRegistry. You should consider upgrading to version 'v1'.
  logger.warn(
C:\Users\91751\anaconda3\lib\site-packages\gym\utils\passive_env_checker.py:102: DeprecationWarning: Call to deprecated method register of class GymRegistry. You should consider upgrading to version 'v1'.
  logger.warn(

Out[1]: True

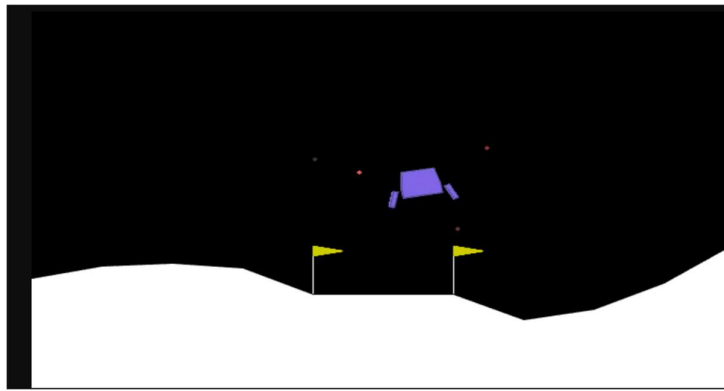
In [ ]:
```

3. **Box2d:** This is an open-source physics engine used for simulating rigid bodies in 2D. The Gym toolkit has a few continuous control tasks that are developed using the Box2D simulator. Figure 14 shows the example of Box2D environments. Box2d is a 2D physics engine.

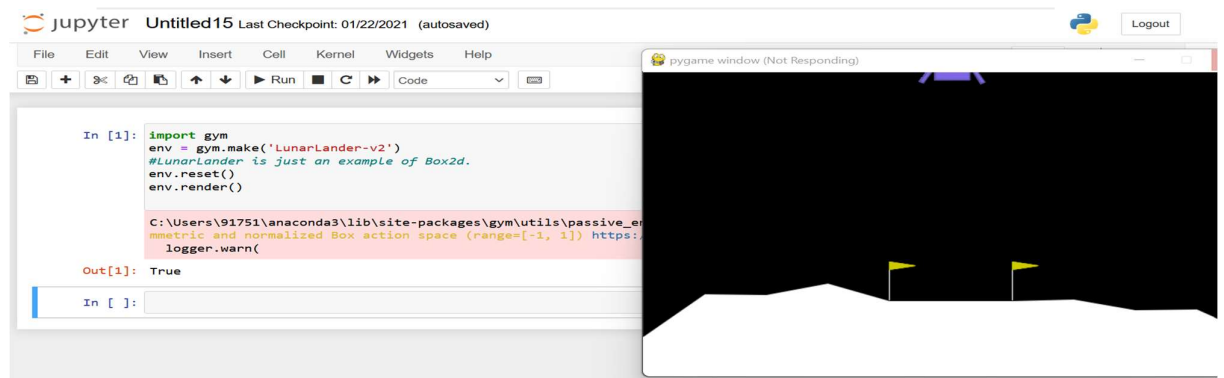
```
pip install -e '[box2d]' or pip install Box2D
```

LunarLander

This environment is a classic rocket trajectory optimization problem. According to Pontryagin’s maximum principle, it is optimal to fire the engine at full throttle or turn it off. This is the reason why this environment has discrete actions: engine on or off. There are two environment versions: discrete or continuous. The landing pad is always at coordinates (0,0). The coordinates are the first two numbers in the state vector. Landing outside of the landing pad is possible. Fuel is infinite, so an agent can learn to fly and then land on its first attempt.



```
import gym
env = gym.make('LunarLander-v2') #LunarLander is just an example of Box2d.
env.reset()
env.render()
```



```
(base) C:\Users\91751>pip install Box2D
Collecting Box2D
  Downloading Box2D-2.3.10-cp38-cp38-win_amd64.whl (1.3 MB)
    |#####| 1.3 MB 595 kB/s
Installing collected packages: Box2D
Successfully installed Box2D-2.3.10

(base) C:\Users\91751>
```

4. Algorithmic

They provide tasks that require an agent to perform computations, such as the addition of multi-digit numbers, copying data from an input sequence, reversing sequences, and so on. These are a variety of algorithmic tasks, such as learning to copy a sequence. There's no extra dependency to install, so to get started, you can just do:

```
import gym
env = gym.make('Copy-v0')
```

```
env.reset()
env.render()
```

```
In [2]: import gym
env = gym.make('Copy-v0')
env.reset()
env.render()

[2022-06-29 11:43:12,744] Making new env: Copy-v0
Total length of input instance: 2, step: 0
=====
Observation Tape : 0
Output Tape      : 0
Targets          : 0

C:\Users\91751\anaconda3\lib\site-packages\gym\envs\registration.py:17: PkgResourcesDeprecationWarning: Parameters to load are deprecated. Call .resolve and .require separately.
  result = entry_point.load(False)

Out[2]: <ipykernel.ostream.OutStream at 0x2a0a0cbe760>
```

5. Toy text

Open AI Gym also has some simple text-based environments under this category. These include some classic problems such as Frozen Lake, where the goal is to find a safe path to cross a grid of ice and water tiles. It is categorized under toy text because it uses a simpler environment representation—mostly through text. Toy environments which are text-based. There's no extra dependency to install, so to get started, you can just do:

'FrozenLake-v0'

Frozen lake involves crossing a frozen lake from Start(S) to Goal (G) without falling into any Holes(H) by walking over the Frozen(F) lake. The agent may not always move in the intended direction due to the slippery nature of the frozen lake.



Output of the the method env.render()

So, the same grid we saw in the [previous post](#) now is represented by a matrix of characters. Their meaning is as follows:

- **S**: initial state
- **F**: frozen lake
- **H**: hole
- **G**: the goal
- **Red square**: indicates the current position of the player

```
import gym
env = gym.make('FrozenLake-v0')
env.reset()
env.render()
```

```
In [1]: import gym
env = gym.make('FrozenLake-v0')
env.reset()
env.render()
```

[2022-06-29 11:44:14,809] Making new env: FrozenLake-v0
C:\Users\91751\anaconda3\lib\site-packages\gym\envs\registration.py:17: PkgResourcesDeprecationWarning: Parameters to load are deprecated. Call .resolve and .require separately.
result = entry_point.load(False)

```

$FFF
FHHH
FFHH
HFFG

```

6. Atari Game:

These offer interfaces to several classic Atari console games. These environment interfaces are wrappers on top of the Arcade Learning Environment (ALE). They provide the game's screen images or RAM as input to train your agents. The Atari environment consists of a wide range of classic Atari videogames. Install the dependencies

`pip install -e '[atari]'` (you'll need CMake installed) or `pip install atari-py` after that write the code `python -m atari_py.import_roms source of ROMS`

eg: `python -m atari_py.import_roms "C:\Users\91751\Downloads\ROMS"`

download the ROMS here:

http://www.atarimania.com/rom_collection_archive_atari_2600_roms.html

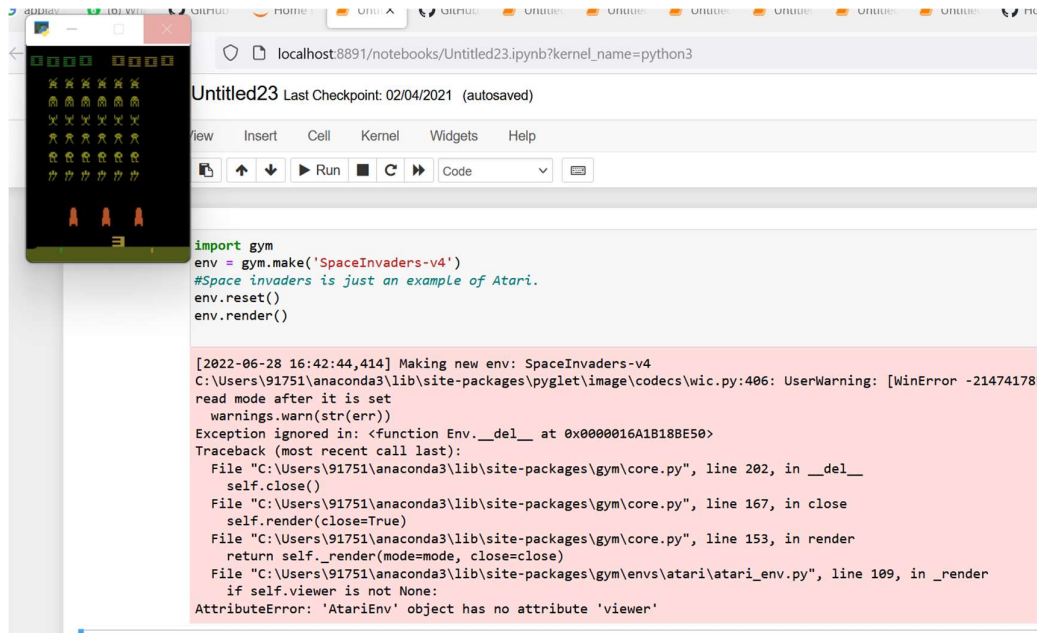
and then follow the commands below:

'SpaceInvaders-v0'

Your objective is to destroy the space invaders by shooting your laser cannon at them before they reach the Earth. The game ends when all your lives are lost after taking enemy fire, or when they reach the earth.



```
import gym
env = gym.make('SpaceInvaders-v0')
#Space invaders is just an example of Atari.
env.reset()
env.render()
```



```
(base) C:\Users\91751>pip install atari-py
Requirement already satisfied: atari-py in c:\users\91751\anaconda3\lib\site-packages (0.2.9)
Requirement already satisfied: numpy in c:\users\91751\anaconda3\lib\site-packages (from atari-py) (1.20.1)
Requirement already satisfied: six in c:\users\91751\anaconda3\lib\site-packages (from atari-py) (1.15.0)

(base) C:\Users\91751>python -m atari_py.import_roms "C:\Users\91751\Downloads\ROMS"
copying adventure.bin from C:\Users\91751\Downloads\ROMS\Adventure (1980) (Atari, Warren Robinett) (CX2613, CX2613P)
~.bin to C:\Users\91751\anaconda3\lib\site-packages\atari_py\atari_roms\adventure.bin
copying air_raids.bin from C:\Users\91751\Downloads\ROMS\Air Raid (Men-A-Vision) (PAL) ~.bin to C:\Users\91751\anaconda3\lib\site-packages\atari_py\atari_roms\air_raids.bin
copying alien.bin from C:\Users\91751\Downloads\ROMS\Alien (1982) (20th Century Fox Video Games, Douglas 'Dallas' Neubauer) (11006) ~.bin to C:\Users\91751\anaconda3\lib\site-packages\atari_py\atari_roms\alien.bin
copying amidar.bin from C:\Users\91751\Downloads\ROMS\Amidar (1982) (Parker Brothers, Ed Temple) (PB5310) ~.bin to C:\Users\91751\anaconda3\lib\site-packages\atari_py\atari_roms\amidar.bin
copying assault.bin from C:\Users\91751\Downloads\ROMS\Assault (AKA Sky Alien) (1983) (Bomb - Onbase) (CA281) ~.bin to C:\Users\91751\anaconda3\lib\site-packages\atari_py\atari_roms\assault.bin
copying asterix.bin from C:\Users\91751\Downloads\ROMS\Asterix (AKA Taz) (1984) (Atari, Jerome Domurat, Steve Wita) (2696) ~.bin to C:\Users\91751\anaconda3\lib\site-packages\atari_py\atari_roms\asterix.bin
copying asteroids.bin from C:\Users\91751\Downloads\ROMS\Asteroids (1981) (Atari, Brad Stewart - Sears) (CX2649 - 49) ~.bin to C:\Users\91751\anaconda3\lib\site-packages\atari_py\atari_roms\asteroids.bin
```

OpenAI Gym Practice 2

1. Cartpole Problem

CartPole is a game in the Open-AI Gym reinforced learning environment. It has a pole attached to a cart. This is an unstable system. The goal is to move the cart left and right to keep the pole from falling. In the Open AI CartPole environment, the status of the system is specified by an “observation” of four parameters (x , v , θ , ω), where x : the horizontal position of the cart (positive means to the right), v : the horizontal velocity of the cart (positive means moving to the right), θ : the angle between the pole and the vertical position (positive means clock-wise), ω : angular velocity of the pole (positive means rotating clock-wise). Given an observation, a player can perform either one of the following two possible “actions”: 0: pushing the cart to the left, 1: pushing the cart to the right

Initialization

```
import gym

def query_environment(name):
    env = gym.make(name)
    spec = gym.spec(name)
    print(f"Action Space: {env.action_space}")
    print(f"Observation Space: {env.observation_space}")
    print(f"Max Episode Steps: {spec.max_episode_steps}")
    print(f"Nondeterministic: {spec.nondeterministic}")
    print(f"Reward Range: {env.reward_range}")
    print(f"Reward Threshold: {spec.reward_threshold}")
```


Environment

```
[ ] query_environment("CartPole-v1")
```

If rendering does not work in colab, try the following.

```
!pip install gym pyvirtualdisplay > /dev/null 2>&1
!apt-get install -y xvfb python-opengl ffmpeg > /dev/null 2>&1
!apt-get update > /dev/null 2>&1
!apt-get install cmake > /dev/null 2>&1
!pip install --upgrade setuptools 2>&1
!pip install ez_setup > /dev/null 2>&1
!pip install gym[atari] > /dev/null 2>&1
```

Utility Functions

```
import gym
from gym.wrappers import Monitor
import glob
import io
import base64
def wrap_env(env):
    env = Monitor(env, './video', force=True)
    return env
CartPole Simulation
```

```
env = wrap_env(gym.make("CartPole-v1"))
#env = wrap_env(gym.make("Atlantis-v0"))

observation = env.reset()

while True:
    env.render()
    #your agent goes here
    action = env.action_space.sample()

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

    if done:
        break;

env.close()
show_video()
```

Try simulations of different environments using gym library page. <https://www.gymnasium.ml/>

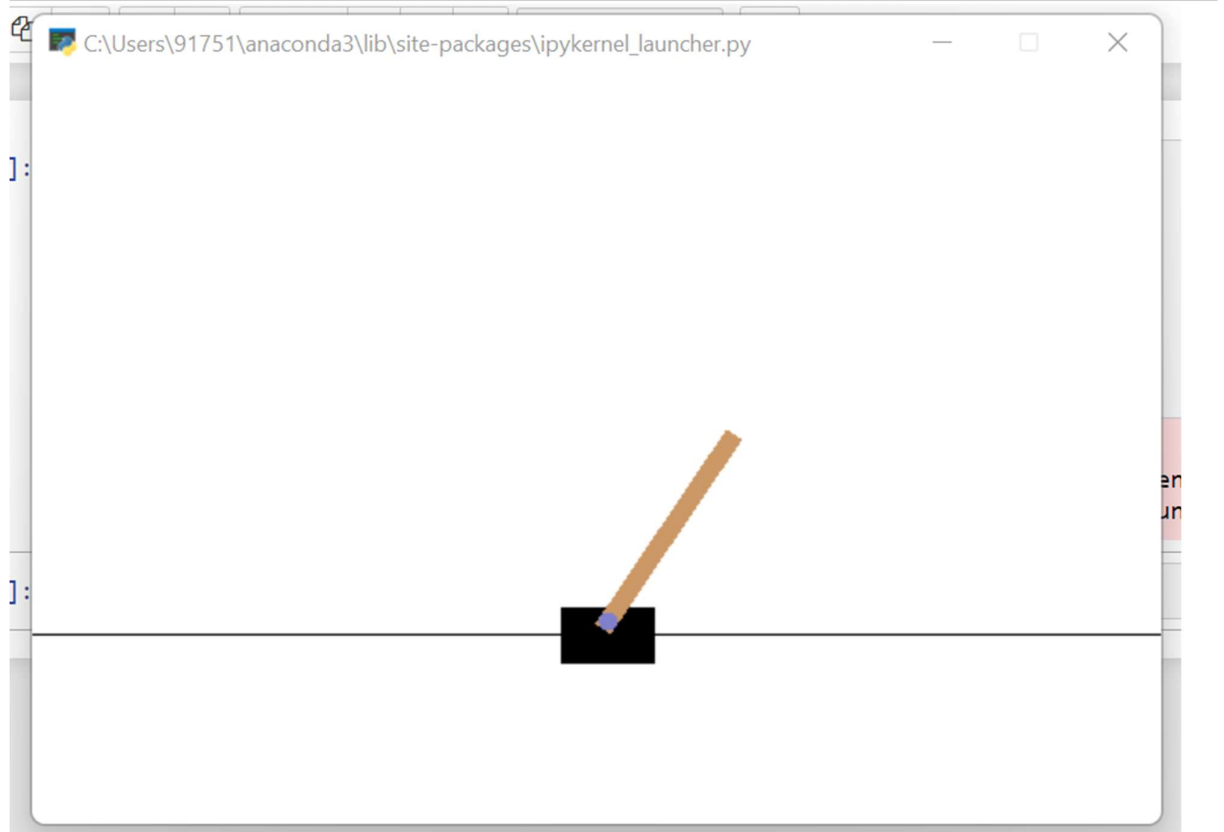
Exercise:

1. Create the list of environments
 - a) MountainCar-v0
 - b) LunarLander-v2
 - c) BipedalWalker-v2
 - d) Acrobot-v1
 - e) Pendulum-v1
 - f) CarRacing-v1
 - g) ALE/AirRaid-v5
2. Write a Openai Gym CartPole random action program the goal of the program is move the cart left and right to keep the pole from falling.

```
: import gym
env = gym.make('CartPole-v0')

env.reset()
for _ in range(30000):
    env.render()
    env.step(env.action_space.sample()) # take a random action
env.close()
```

[2022-07-01 15:00:55,169] Making new env: CartPole-v0
[2022-07-01 15:00:55,926] You are calling 'step()' even though this environment has already returned done = True. You should always call 'reset()' once you receive 'done = True' -- any further steps are undefined behavior.



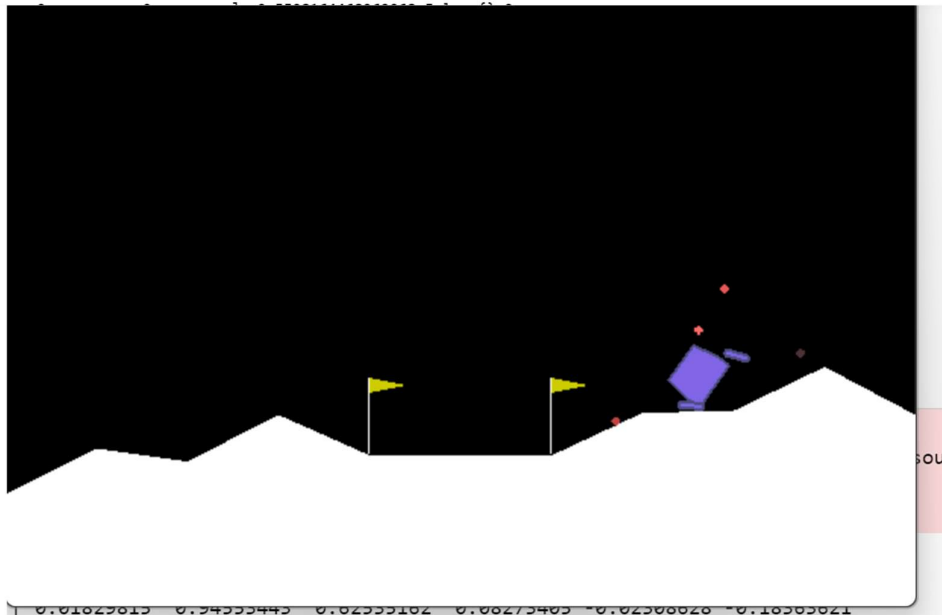
3. Create LunarLander-v2 environment and implement this for a random-action interacting environment (Landing pad is always at coordinates (0,0). Coordinates are the first two numbers in state vector. Reward for moving from the top of the screen to landing pad and zero speed is about 100 .. 140 points. If lander moves away from landing pad it loses

reward back. Episode finishes if the lander crashes or comes to rest, receiving additional -100 or +100 points. Each leg ground contact is +10. Firing main engine is -0.3 points each frame. Solved is 200 points. Landing outside landing pad is possible. Fuel is infinite, so an agent can learn to fly and then land on its first attempt. Four discrete actions available: do nothing, fire left orientation engine, fire main engine, fire right orientation engine.)

```
In [1]: import gym
import random
env = gym.make("LunarLander-v2")
def Random_games():
    for episode in range(10):
        env.reset()
        while True:
            env.render()
            action = env.action_space.sample()
            next_state, reward, done, info = env.step(action)
            print(next_state, reward, done, info, action)
            if done:
                break
Random_games()
```

```
[2022-07-01 16:15:32,728] Making new env: LunarLander-v2
C:\Users\91751\anaconda3\lib\site-packages\gym\envs\registration.py:17: PkgResourcesDeprecationWarning: Parameters to load are deprecated. Call .resolve and .require separately.
result = entry_point.load(False)
```

```
[ 0.01213741  0.94429169  0.61384449  0.10973214 -0.01390531 -0.13761287
 0.          0.          ] -0.2744089400192422 False {} 0
[ 0.01829815  0.94553443  0.62535162  0.08273405 -0.02308628 -0.18363621
 0.          0.          ] -1.8048143469559637 False {} 3
[ 0.02436543  0.94638034  0.61361904  0.05627616 -0.02990531 -0.13639287
 0.          0.          ] 0.6505005533665849 False {} 1
[ 0.03043289  0.94682666  0.61363754  0.0296045  -0.03672478 -0.13640186
 0.          0.          ] 0.6505005533665849 False {} 1
```



4. Create a BipedalWalker-v2 environment and implement this for a random-action interacting environment (Bipedal Walker has two legs. Each leg has two joints. You have to teach the Bipedal-walker to walk by applying the torque on these joints. Therefore the size of our action space is four which is the torque applied on four joints. You can use the torque in the range of $(-1, 1)$)

