Install Libraries that would be needed in this notebook

```
!apt install swig cmake

!pip install stable-baselines3==2.0.0a5
!pip install swig
!pip install gymnasium[box2d]
!pip install huggingface_sb3

```

Show hidden output

The flow of creating and using an environment

1. Create an environment.
2. Reset the environment to the initial state.
3. Sample an action.
4. Take that action.
5. Check if the action leads to termination (end of episode) or truncation (exceed time or physical space limit)
6. Repeat step 3 to 5 for a number of times.
7. Close the environment.

Some important reference / links to look up for

- Learn the environment parameter from the documentations.
  - For instance, https://gymnasium.farama.org/environments/box2d/lunar_lander/ provides details on the state space, action space and reward.
- Learn how the environment works as an object.
  - https://gymnasium.farama.org/api/env/#gymnasium.Env provides details on the functions / methods generally used by an env object.

```
import gymnasium as gym

# 1. First create an environment (This example focuses on the Lunar Lander Environment)
# https://gymnasium.farama.org/environments/box2d/lunar_lander/
env = gym.make("LunarLander-v2")

# 2. Reset the environment to the initial state
observation, info = env.reset()

# 3. Randomly sample an action
# 4. And take the action
# 5. Check if the action leads to termination (end of episode) or truncation (exceed timelimit or physical space limit)
# 6. Repeat step 3 to 5 for a number of times

for _ in range(20):

  # 3. Randomly sample an action
  action = env.action_space.sample()
  print(f"Action taken: {action}")

  # 4. And take the action (https://gymnasium.farama.org/api/env/#gymnasium.Env.step)
  observation, reward, terminated, truncated, info = env.step(action)

  # 5. Check if the action leads to termination or truncation
  if terminated or truncated:
    print("Environment is reset!")
    observation, info = env.reset()

# 7. Close the environment
env.close()
```

```
Action taken: 3
Action taken: 2
Action taken: 3
Action taken: 0
Action taken: 1
Action taken: 0
Action taken: 2
Action taken: 2
Action taken: 1
Action taken: 1
Action taken: 2
Action taken: 3
Action taken: 0
Action taken: 1
Action taken: 1
Action taken: 0
Action taken: 3
Action taken: 2
Action taken: 1
Action taken: 2
```

Visualize Observation Space

- To study the parameter represented by each value in the observation space, kindly refer to: https://gymnasium.farama.org/environments/box2d/lunar_lander/#

```python
import gymnasium as gym

env = gym.make('LunarLander-v2')

print(f"The shape of the observation space: {env.observation_space.shape}")
print(f"A sample of the observation space: {env.observation_space.sample()}")
```

```
The shape of the observation space: (8,)
A sample of the observation space: [-75.64905     78.99128     -2.5721257    4.0058837   -2.0155616
   4.15837      0.38517615   0.49400613]
```

Visualize Action Space

```python
import gymnasium as gym

env = gym.make("LunarLander-v2")

print(f"The available action in the action space: {env.action_space} / {env.action_space.n}")
print(f"A sample of the action space: {env.action_space.sample()}")
```

```
The available action in the action space: Discrete(4) / 4
A sample of the action space: 2
```

Train an agent using a Vectorised Environment

```python
import gymnasium as gym
from stable_baselines3 import PPO
from stable_baselines3.common.env_util import make_vec_env


#1. Make a vectorised environment - allowing the model to train with diversity across environments
# https://stable-baselines3.readthedocs.io/en/master/common/env_util.html#stable_baselines3.common.env_util.make_vec_env
env = make_vec_env(env_id = "LunarLander-v2",
                   n_envs = 16)

# 2. Create a PPO model
# https://stable-baselines3.readthedocs.io/en/master/modules/ppo.html
model = PPO(policy = 'MlpPolicy',
            env = env,
            n_steps = 1024,
            batch_size = 64,
            n_epochs = 4,
            gamma = 0.999,
            gae_lambda = 0.98,
            ent_coef = 0.01,
            verbose = 1)

# 3. Train the PPO model
model.learn(total_timesteps = 1e6)

# 4. Save the trained PPO model
model_name = "ppo-LunarLander-v2"
model.save(model_name)
```

Show hidden output

Evaluate the trained agent

```
import gymnasium as gym
from stable_baselines3 import PPO
from stable_baselines3.common.env_util import make_vec_env
from stable_baselines3.common.evaluation import evaluate_policy
from stable_baselines3.common.monitor import Monitor

# 1. Create a new environment for evaluation
eval_env = Monitor(gym.make("LunarLander-v2", render_mode = 'rgb_array'))

# 2. Evaluate the model (P/S: Taking models from the previous cell block)
# https://stable-baselines3.readthedocs.io/en/master/common/evaluation.html#stable_baselines3.common.evaluation.evaluate_policy
mean_reward, std_reward = evaluate_policy(model = model,
                                          env = eval_env,
                                          n_eval_episodes = 10,
                                          deterministic = True) # Take deterministic actions (instead of sampling / stochastic)

print(f"The earned reward is : {mean_reward:.2f} +/- {std_reward:.2f}")
```

```
The earned reward is : 240.90 +/- 19.26
```

Publish the trained model on the Hub

- Create a new token with with **write role** here: https://huggingface.co/settings/tokens
- Once the model is published on Hub, the result may be accessed via the **repo_url** in the output.

```
from huggingface_hub import notebook_login

notebook_login()
!git config --global credential.helper store
```

```
from stable_baselines3.common.vec_env import DummyVecEnv
from huggingface_sb3 import package_to_hub

# https://huggingface.co/docs/hub/en/stable-baselines3
package_to_hub(model = model, # Taking model from the previous cell block
               model_name = model_name, # Taking model name from the previous cell block
               model_architecture = "PPO",
               env_id = "LunarLander-v2",
               eval_env = DummyVecEnv([lambda: Monitor(gym.make("LunarLander-v2", render_mode = "rgb_array"))]), # Expecting a list
               repo_id = "wengti0608/ppo-LunarLander-v2",
               commit_message = "First Commit")
```

Show hidden output

Load a saved LunarLander model from the Hub

- I am loading a model that was trained with gym (old version of gymnasium). Therefore, I am using shimmy to ensure version compatibility.
- But this will lead to reinstallation of gym which will crash with gymnasium, subsequently leading to crash. Therefore, I try to reinstall all the libraries that are needed again.

```
!pip install shimmy
!apt install swig cmake

!pip install stable-baselines3==2.0.0a5
!pip install swig
!pip install gymnasium[box2d] shimmy
!pip install huggingface_sb3
```

Show hidden output

- You may find a full list of trained models of LunarLander-v2 here: https://huggingface.co/models?search=LunarLander-v2
- For this tutorial, we are using the following repository: https://huggingface.co/Classroom-workshop/assignment2-omar

```
from huggingface_sb3 import load_from_hub
from stable_baselines3 import PPO
```

```

# The loaded model was trained on Python 3.8 that uses pickle protocol of 5
# But Python 3.6 and 3.7 use pickle protocol 4
# Therefore, a placeholder custom_objects need to be loaded
custom_objects = {
    "learning_rate" : 0.0,
    "lr_schedule": lambda _: 0.0,
    "clip_range": lambda _: 0.0,
}

# https://huggingface.co/blog/sb3
checkpoint = load_from_hub(repo_id = "Classroom-workshop/assignment2-omar",
                           filename = "ppo-LunarLander-v2.zip")

# https://stable-baselines3.readthedocs.io/en/master/modules/ppo.html
model = PPO.load(path = checkpoint,
                 custom_objects = custom_objects,
                 print_system_info = True)
```

Show hidden output

Evaluate the loaded model

```
import gymnasium as gym
from stable_baselines3.common.evaluation import evaluate_policy
from stable_baselines3.common.monitor import Monitor

eval_env = Monitor(gym.make("LunarLander-v2", render_mode = "rgb_array"))

mean_reward, std_reward = evaluate_policy(model = model,
                                          env = eval_env,
                                          n_eval_episodes = 10,
                                          deterministic = True) # Taking the loaded model from the previous cell block

print(f"The reward earned by the loaded model: {mean_reward:.2f} +/- {std_reward:.2f}")
```

```
The reward earned by the loaded model: 294.90 +/- 14.55
```

```
Start coding or generate with AI.
```