In [1]:
```
#Implementation of Reinforcement Algorithm.
```

In [2]:
```
import gym
import numpy as np
```

In [3]:
```
from stable_baselines3 import PPO
```

In [4]:
```
from stable_baselines3.ppo.policies import MlpPolicy
```

In [5]:
```
env = gym.make('CartPole-v1')

model = PPO(MlpPolicy, env, verbose=0)
```

In [6]:
```
def evaluate(model, num_episodes=100):
    """
    Evaluate a RL agent
    :param model: (BaseRLModel object) the RL Agent
    :param num_episodes: (int) number of episodes to evaluate it
    :return: (float) Mean reward for the last num_episodes
    """
    # This function will only work for a single Environment
    env = model.get_env()
    all_episode_rewards = []
    for i in range(num_episodes):
        episode_rewards = []
        done = False
        obs = env.reset()
        while not done:
            # _states are only useful when using LSTM policies
            action, _states = model.predict(obs)
            # here, action, rewards and dones are arrays
            # because we are using vectorized env
            obs, reward, done, info = env.step(action)
            episode_rewards.append(reward)

        all_episode_rewards.append(sum(episode_rewards))

    mean_episode_reward = np.mean(all_episode_rewards)
    print("Mean reward:", mean_episode_reward, "Num episodes:", num

    return mean_episode_reward
```

In [7]:
```
# Random Agent, before training
mean_reward_before_train = evaluate(model, num_episodes=100)
```

```
Mean reward: 23.52 Num episodes: 100
```

In [8]:
```
from stable_baselines3.common.evaluation import evaluate_policy
```

In [9]:
```
mean_reward, std_reward = evaluate_policy(model, env, n_eval_episod

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

```
mean_reward:9.55 +/- 0.75

/opt/anaconda3/lib/python3.9/site-packages/stable_baselines3/commo
n/evaluation.py:65: UserWarning: Evaluation environment is not wra
pped with a ``Monitor`` wrapper. This may result in reporting modi
fied episode lengths and rewards, if other wrappers happen to modi
fy these. Consider wrapping environment first with ``Monitor`` wra
pper.
  warnings.warn(
```

In [10]:
```
# Train the agent for 10000 steps
model.learn(total_timesteps=10000)
```

Out[10]:
```
<stable_baselines3.ppo.ppo.PPO at 0x1542baeb0>
```

In [11]:
```
# Evaluate the trained agent
mean_reward, std_reward = evaluate_policy(model, env, n_eval_episod

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

```
mean_reward:441.12 +/- 92.79
```

In [12]:
```
# Set up fake display; otherwise rendering will fail
import os
os.system("Xvfb :1 -screen 0 1024x768x24 &")
os.environ['DISPLAY'] = ':1'
```

```
sh: Xvfb: command not found
```

In [13]:
```
import base64
from pathlib import Path

from IPython import display as ipythondisplay

def show_videos(video_path='', prefix=''):

  html = []
  for mp4 in Path(video_path).glob("{}*.mp4".format(prefix)):
      video_b64 = base64.b64encode(mp4.read_bytes())
      html.append('''<video alt="{}" autoplay
                    loop controls style="height: 400px;">
                    <source src="data:video/mp4;base64,{}" type="vi
                </video>'''.format(mp4, video_b64.decode('ascii')))
  ipythondisplay.display(ipythondisplay.HTML(data="<br>".join(html)
```

In [14]:
```
from stable_baselines3.common.vec_env import VecVideoRecorder, Dumm

def record_video(env_id, model, video_length=500, prefix='', video_
  """
  :param env_id: (str)
  :param model: (RL model)
  :param video_length: (int)
  :param prefix: (str)
  :param video_folder: (str)
  """
  eval_env = DummyVecEnv([lambda: gym.make(env_id)])
  # Start the video at step=0 and record 500 steps
  eval_env = VecVideoRecorder(eval_env, video_folder=video_folder,
                              record_video_trigger=lambda step: ste
                              name_prefix=prefix)

  obs = eval_env.reset()
  for _ in range(video_length):
    action, _ = model.predict(obs)
    obs, _, _, _ = eval_env.step(action)

  # Close the video recorder
  eval_env.close()
```

In [15]:
```
record_video('CartPole-v1', model, video_length=500, prefix='ppo-ca
```

```
---------------------------------------------------------------------
---------
ModuleNotFoundError                       Traceback (most recent c
all last)
File /opt/anaconda3/lib/python3.9/site-packages/gym/envs/classic_c
ontrol/rendering.py:15, in <module>
     14 try:
---> 15     import pyglet
     16 except ImportError as e:

ModuleNotFoundError: No module named 'pyglet'

During handling of the above exception, another exception occurred
:

ImportError                               Traceback (most recent c
all last)
Input In [15], in <cell line: 1>()
----> 1 record_video('CartPole-v1', model, video_length=500, prefi
x='ppo-cartpole')

Input In [14], in record_video(env_id, model, video_length, prefix
, video_folder)
     12 # Start the video at step=0 and record 500 steps
     13 eval_env = VecVideoRecorder(eval_env, video_folder=video_f
older,
```

```
     14                                record_video_trigger=lambda
step: step == 0, video_length=video_length,
     15                                name_prefix=prefix)
---> 17 obs = eval_env.reset()
     18 for _ in range(video_length):
     19   action, _ = model.predict(obs)

File /opt/anaconda3/lib/python3.9/site-packages/stable_baselines3/
common/vec_env/vec_video_recorder.py:68, in VecVideoRecorder.reset
(self)
     66 def reset(self) -> VecEnvObs:
     67     obs = self.venv.reset()
---> 68     self.start_video_recorder()
     69     return obs

File /opt/anaconda3/lib/python3.9/site-packages/stable_baselines3/
common/vec_env/vec_video_recorder.py:80, in VecVideoRecorder.start
_video_recorder(self)
     75 base_path = os.path.join(self.video_folder, video_name)
     76 self.video_recorder = video_recorder.VideoRecorder(
     77     env=self.env, base_path=base_path, metadata={"step_id"
: self.step_id}
     78 )
---> 80 self.video_recorder.capture_frame()
     81 self.recorded_frames = 1
     82 self.recording = True

File /opt/anaconda3/lib/python3.9/site-packages/gym/wrappers/monit
oring/video_recorder.py:132, in VideoRecorder.capture_frame(self)
    129 logger.debug("Capturing video frame: path=%s", self.path)
    131 render_mode = "ansi" if self.ansi_mode else "rgb_array"
--> 132 frame = self.env.render(mode=render_mode)
    134 if frame is None:
    135     if self._async:

File /opt/anaconda3/lib/python3.9/site-packages/stable_baselines3/
common/vec_env/dummy_vec_env.py:87, in DummyVecEnv.render(self, mo
de)
     75 """
     76 Gym environment rendering. If there are multiple environme
nts then
     77 they are tiled together in one image via ``BaseVecEnv.rend
er()``.
   (...)
     84 :param mode: The rendering type.
     85 """
     86 if self.num_envs == 1:
---> 87     return self.envs[0].render(mode=mode)
     88 else:
     89     return super().render(mode=mode)

File /opt/anaconda3/lib/python3.9/site-packages/gym/core.py:295, i
n Wrapper.render(self, mode, **kwargs)
```

```
    294 def render(self, mode="human", **kwargs):
--> 295     return self.env.render(mode, **kwargs)

File /opt/anaconda3/lib/python3.9/site-packages/gym/envs/classic_c
ontrol/cartpole.py:179, in CartPoleEnv.render(self, mode)
    176 cartheight = 30.0
    178 if self.viewer is None:
--> 179     from gym.envs.classic_control import rendering
    181     self.viewer = rendering.Viewer(screen_width, screen_he
ight)
    182     l, r, t, b = -cartwidth / 2, cartwidth / 2, cartheight
/ 2, -cartheight / 2

File /opt/anaconda3/lib/python3.9/site-packages/gym/envs/classic_c
ontrol/rendering.py:17, in <module>
     15     import pyglet
     16 except ImportError as e:
---> 17     raise ImportError(
     18         """
     19     Cannot import pyglet.
     20     HINT: you can install pyglet directly via 'pip install
pyglet'.
     21     But if you really just want to install all Gym depende
ncies and not have to think about it,
     22     'pip install -e .[all]' or 'pip install gym[all]' will
do it.
     23     """
     24     )
     26 try:
     27     from pyglet.gl import *

ImportError:
    Cannot import pyglet.
    HINT: you can install pyglet directly via 'pip install pyglet'
.
    But if you really just want to install all Gym dependencies an
d not have to think about it,
    'pip install -e .[all]' or 'pip install gym[all]' will do it.
```

In [ ]:
```
show_videos('videos', prefix='ppo')
```

In [ ]:
```

```