Names: Priel Kaizman (315627448), Adir Serruya (316472455)

DRL – Assignment 2

Section 1 – REINFORCE

Answers:

1. The advantage estimate reflects the difference between the actual return compared to a baseline (usually the value function). This approach provides a relative performance metric, identifying actions that perform better than average by comparing them to the general level of rewards in the environment, therefor it helps isolate the actions that are truly better in a given policy.
It's better to follow the gradient computed with the advantage estimate instead of just the return itself, since it reduces the variance of the learning process and accelerates convergence, contributing to a more stable and efficient policy optimization. In addition, since this approach values the relative quality of actions it improves learning effectiveness.

2. The prerequisite condition for the equation to be true is that the baseline $b(s_t)$ is independent from the action a_t.
Proving the equation $\mathrm{E}_{\pi_\theta}[\nabla log\pi_\theta(a_t|s_t)b(s_t)] = 0$:

$$\mathrm{E}_{\pi_\theta}[\nabla log\pi_\theta(a_t|s_t)b(s_t)] = \sum_{a_t} \pi_\theta(a_t|s_t)\, \nabla log\pi_\theta(a_t|s_t)b(s_t)$$

Since $\nabla log\pi_\theta(a_t|s_t) = \frac{\nabla\pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)}$ we get:

$$\sum_{a_t} \pi_\theta(a_t|s_t)\, \nabla log\pi_\theta(a_t|s_t)b(s_t) = \sum_{a_t} \pi_\theta(a_t|s_t) \frac{\nabla\pi_\theta(a_t|s_t)}{\pi_\theta(a_t|s_t)} b(s_t)$$

Since $b(s_t)$ is not dependent on a_t (the prerequisite for the equation to be true), we can factor it out, also we can cancel out $\pi_\theta(a_t|s_t)$:

$$\sum_{a_t} \cancel{\pi_\theta(a_t|s_t)} \frac{\nabla\pi_\theta(a_t|s_t)}{\cancel{\pi_\theta(a_t|s_t)}} b(s_t) = b(s_t) \sum_{a_t} \nabla\pi_\theta(a_t|s_t) = b(s_t)\nabla \sum_{a_t} \pi_\theta(a_t|s_t)$$

The sum of the probabilities is always 1, the derivative or a constant is 0, therefor:

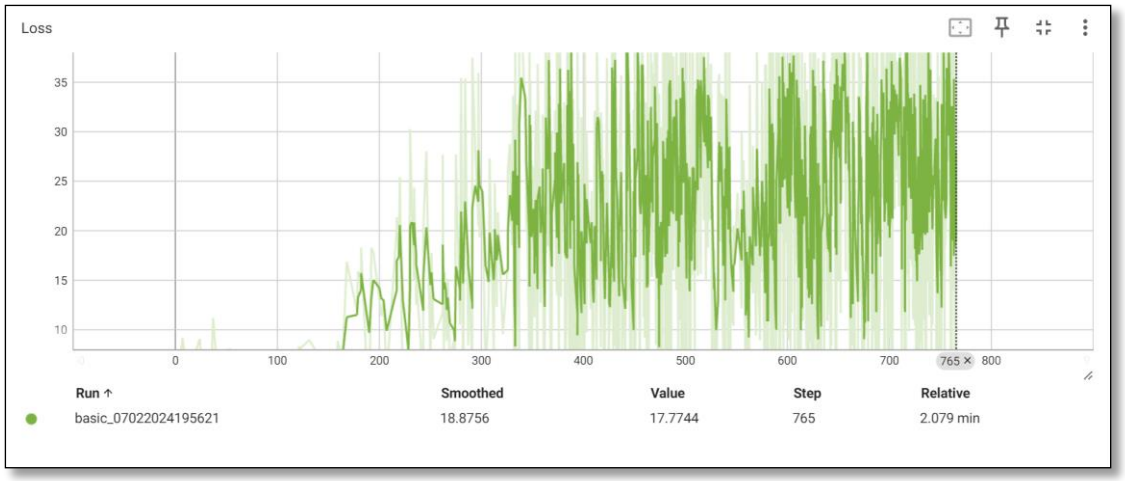
$$b(s_t)\nabla \textstyle\sum_{a_t} \pi_\theta(a_t|s_t) = b(s_t)\nabla 1 = b(s_t)\cdot 0 = 0 \;\rightarrow\; \mathrm{E}_{\pi_\theta}[\nabla log\pi_\theta(a_t|s_t)b(s_t)] = 0$$

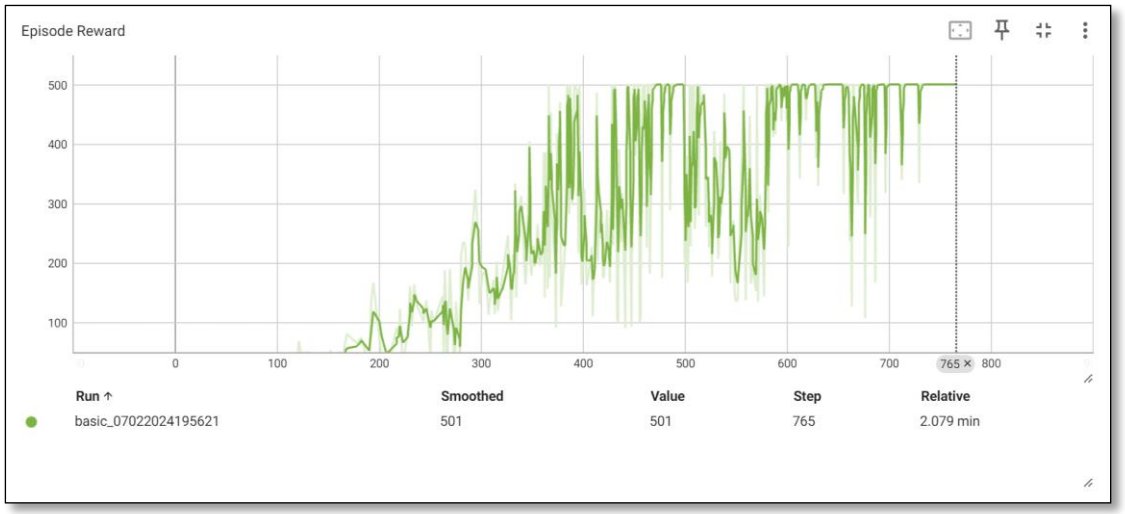
Results:

We kept the same policy network structure with the same hyperparameters and added a value network with a single layer that has the same number of units as the policy network layer (12).
We found that after the change the convergence time was much shorter and generally we saw an overall improvement in performance.

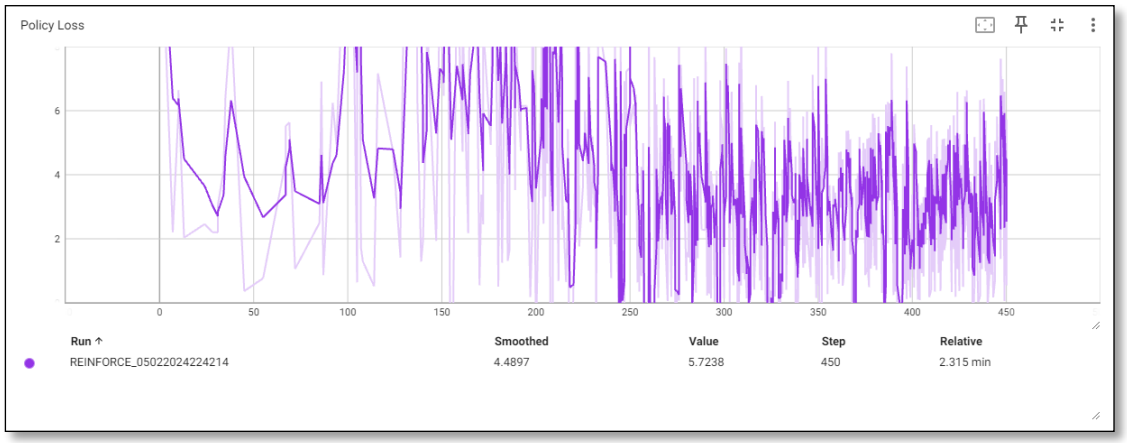
Before the change it took **766** episodes to reach the MA over 100 episodes of 475 while after the change it only took **451** episodes.
Before the change the run took 2.079 minutes and after the change it took 2.315 minutes.
In addition we found unsurprisingly that the loss declined more consistently after the change and that the rewards per episode were more consistently higher after the change.

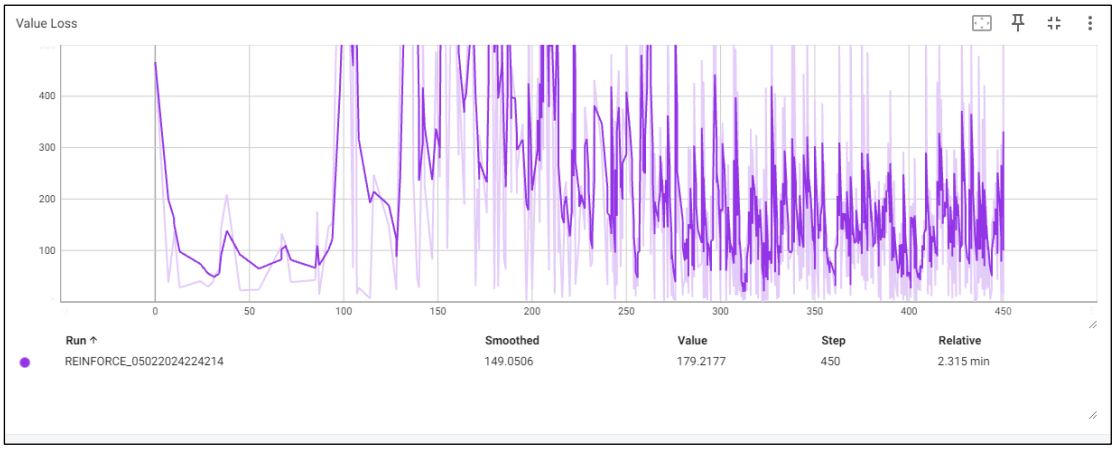
Plots - before the change:

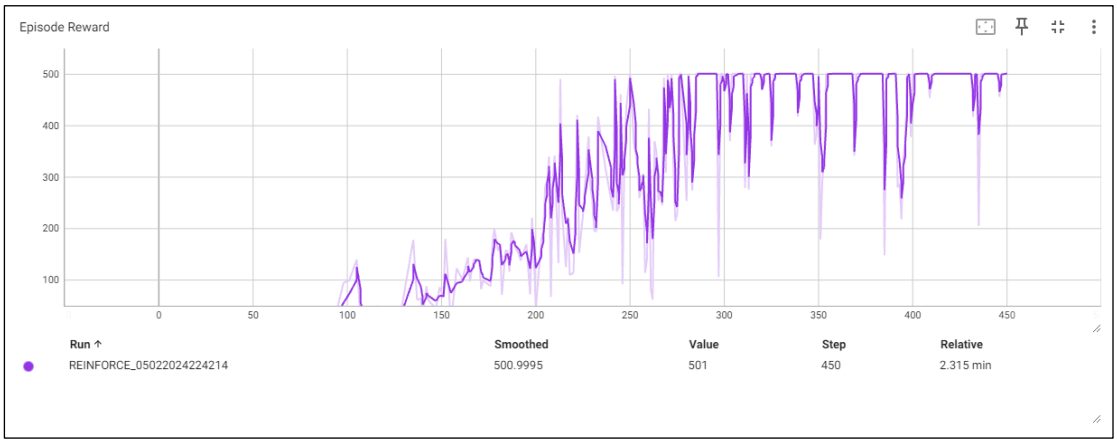




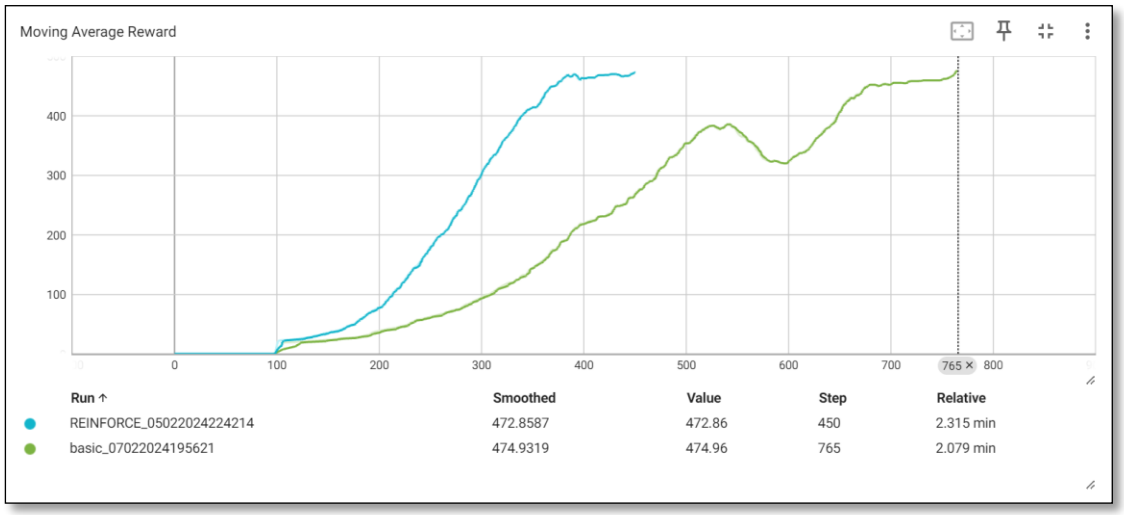
Plots - after the change:







Moving average comparison:



Section 2 – Advanced actor-critic

Answers:

1. The advantage estimate as $\hat{A}_t = \hat{Q}_w(s_t, a_t) - \hat{V}_v(s_t)$ measures how much better an action a_t is compared to the policy's average action at state s_t.
The TD-Error at time t is given by:

$$\delta_t = R_{t+1} + \gamma \hat{V}_v(S_{t+1}) - \hat{V}_v(S_t)$$

If we assume that $\hat{Q}_w(s_t, a_t)$ can be decomposed into the immediate reward R_{t+1} and the discounted value of the next state $\gamma \hat{V}_v(S_{t+1})$ (which is part of the bellman equation for Q) then we have:

$$\hat{Q}_w(s_t, a_t) = R_{t+1} + \gamma \hat{V}_v(S_{t+1})$$

Substituting this into the advantage equation:

$$\hat{A}_t = \left(R_{t+1} + \gamma \hat{V}_v(S_{t+1})\right) - \hat{V}_v(S_t) = \delta_t$$

This shows that the TD error δ_t is the same as the advantage estimate $\hat{A}_t$ when the $\hat{Q}$ function is decomposed in this manner.

2. Actor: This is the policy function, typically denoted as $\pi(a|s,\theta)$, where θ represents the parameters of the model.
The actor decides which action to take given the current state.
Its role is to learn a policy that maps states to the optimal actions that maximize the expected reward over time.

Critic: This function evaluates the chosen actions by estimating the value function, denoted as $V(s,w)$ or $Q(s,a,w)$, with w representing the value function's parameters.
The critic's role is to assess the quality of the actions taken by the actor by estimating the expected returns. The critic helps to reduce the variance of the policy gradient and accelerates the learning process by providing a baseline (value estimate) to compare the policy's actions against.

Results:

We implemented the actor-critic model with increased complexity since we found it produced better results than an actor-critic model with the same single layer networks. Our actor network has 2 layers with 32 units and the critic network has 3 layers with 128 units, we kept the rest of the hyperparameters the same as the 2 previous models.

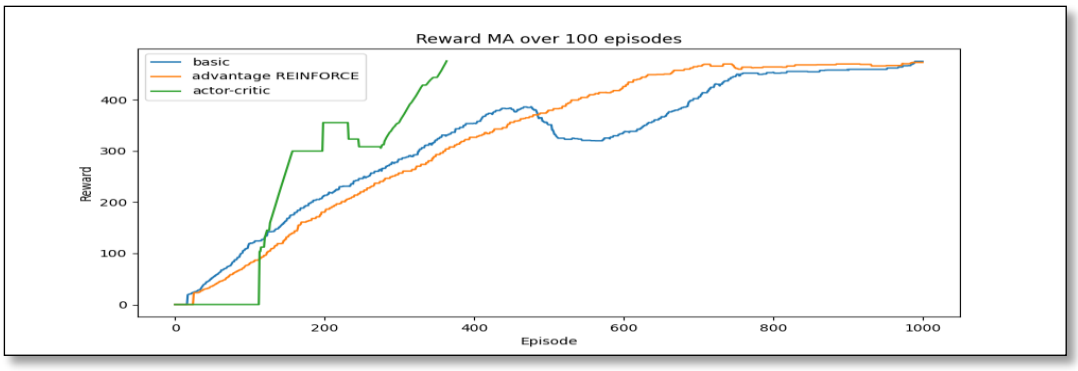
The actor-critic model reached the MA over 100 episodes of 475 within **364** episodes, significantly faster than the previous 2 models (**451** and **766)**.
In addition, the reward per episode was much more stable and quickly reached the maximum of 501, at around episode 100 (compared to episode ~275 for the Advantage REINFORCE model and ~450 for the basic model).

Comparison of the time to solve the problem:

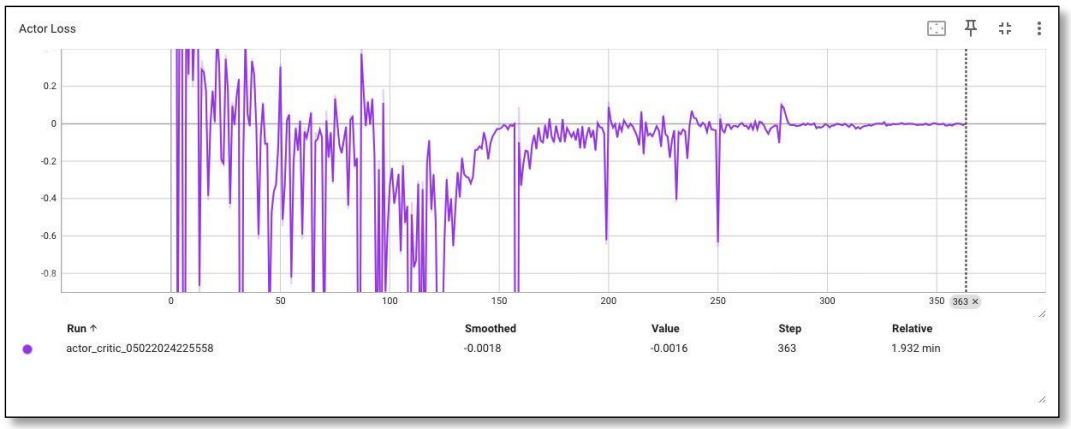
Basic – 2.079 minutes

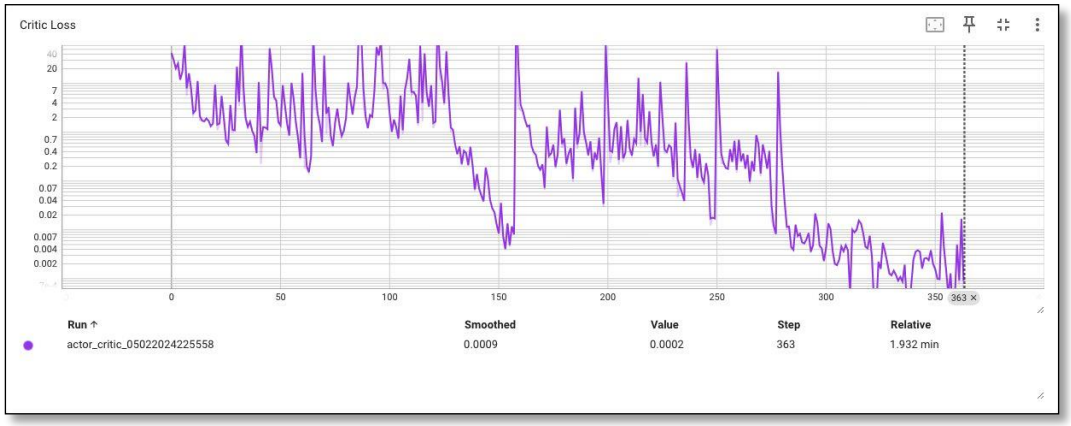
Advantage REINFORCE – 2.315 minutes

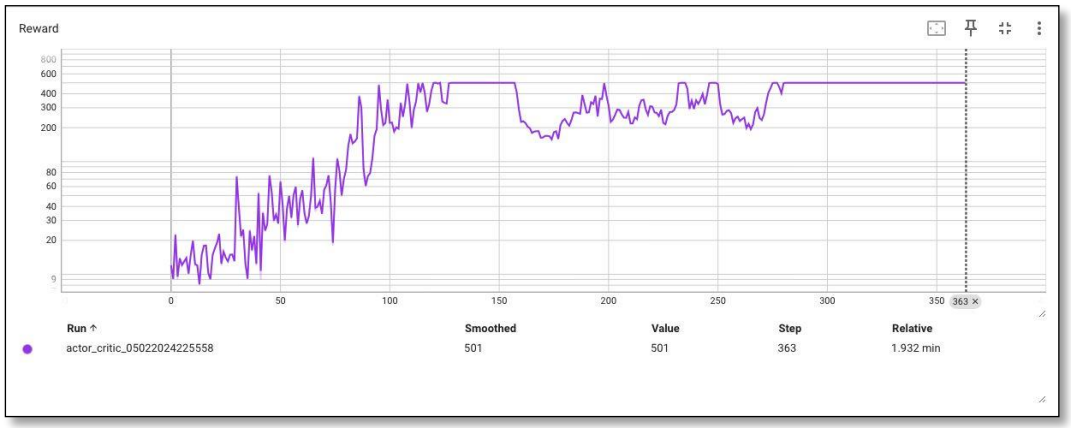
Actor-Critic – 1.932 minutes

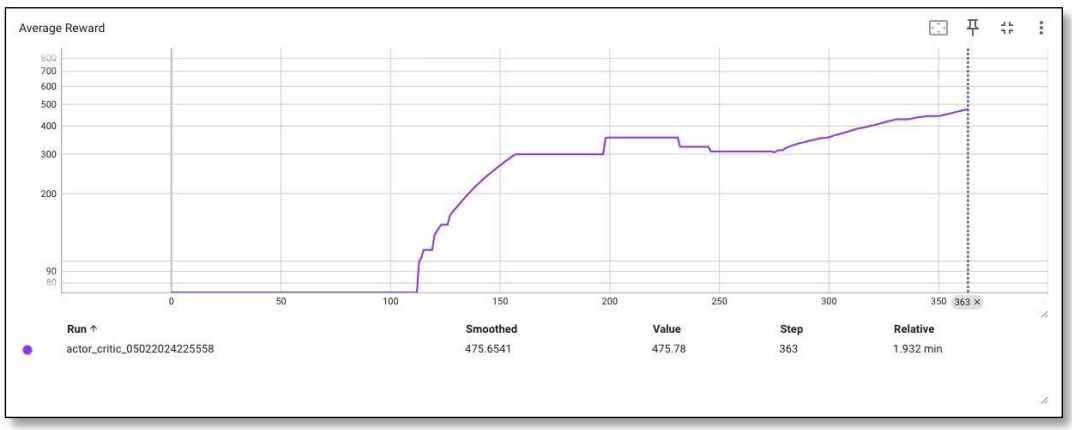


Plots:


Actor Loss
0.2
0
-0.2
-0.4
-0.6
-0.8
0
50
100
150
200
250
300
350
363
Run
Smoothed
Value
Step
Relative
actor_critic_05022024225558
-0.0018
-0.0016
363
1.932 min


Critic Loss
40
20
7
4
2
0.7
0.4
0.2
0.07
0.04
0.02
0.007
0.004
0.002
0
50
100
150
200
250
300
350
363
Run
Smoothed
Value
Step
Relative
actor_critic_05022024225558
0.0009
0.0002
363
1.932 min


Reward
800
600
400
300
200
80
60
40
30
20
9
0
50
100
150
200
250
300
350
363
Run
Smoothed
Value
Step
Relative
actor_critic_05022024225558
501
501
363
1.932 min


Average Reward
800
700
600
500
400
300
200
90
80
0
50
100
150
200
250
300
350
363
Run
Smoothed
Value
Step
Relative
actor_critic_05022024225558
475.6541
475.78
363
1.932 min

Instructions for running the scripts

1. Preliminary Setup:

Ensure that all required packages are installed in your environment, these dependencies are listed in the requirements.txt file.
You can install them using the command: pip install -r requirements.txt

2. Script Descriptions:

- policy_gradients.py: This is the original script you provided with minor fixes (importing gymnasium, taking to account max step reached).

- REINFORCE.py: This script is for section 1 and contains the implementation of REINFONCE as described in the task.

- actor_critic.py: This script is for section 2 and contains the implementation of actor-critic as described in the task.

3. Running the Scripts:

- To execute a script, simply run it in your Python environment. For example by using the command: python policy_gradients.py

- Repeat the process for REINFORCE.py and actor_critic.py as needed.