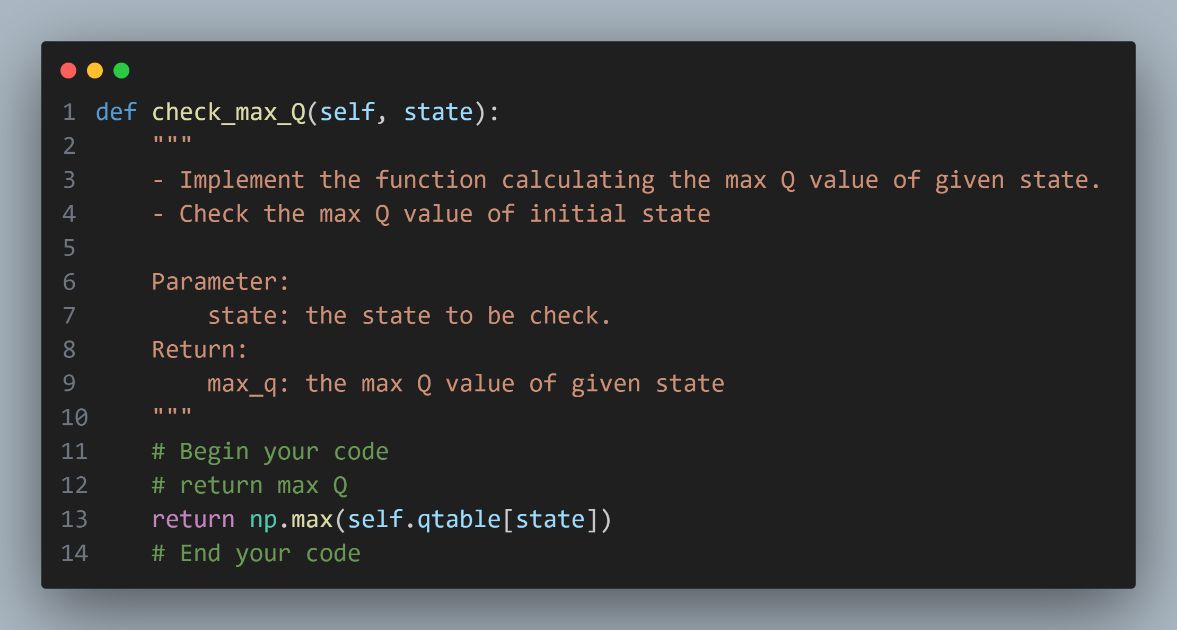
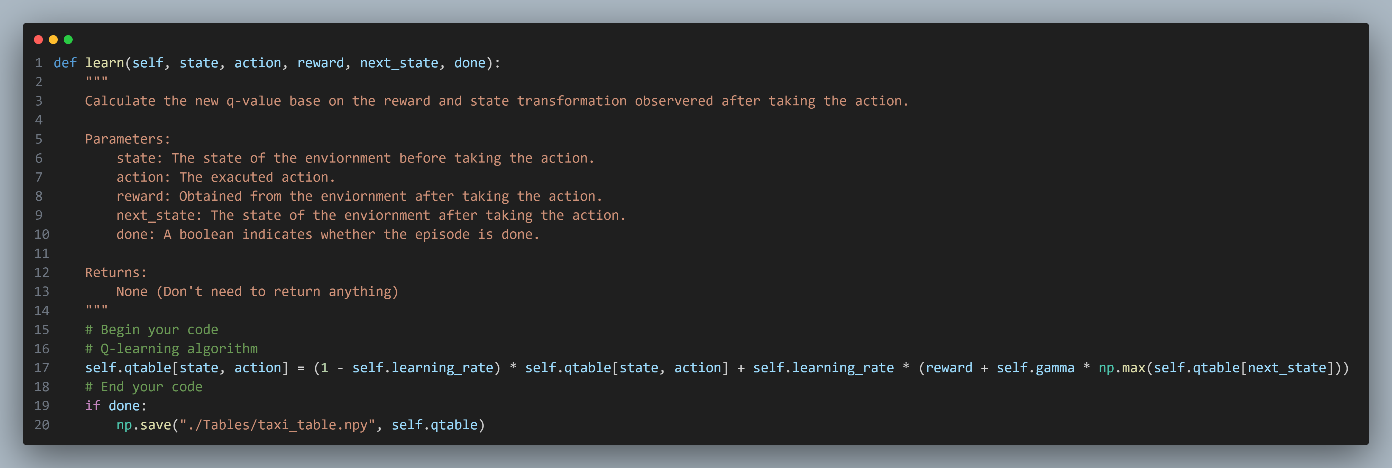
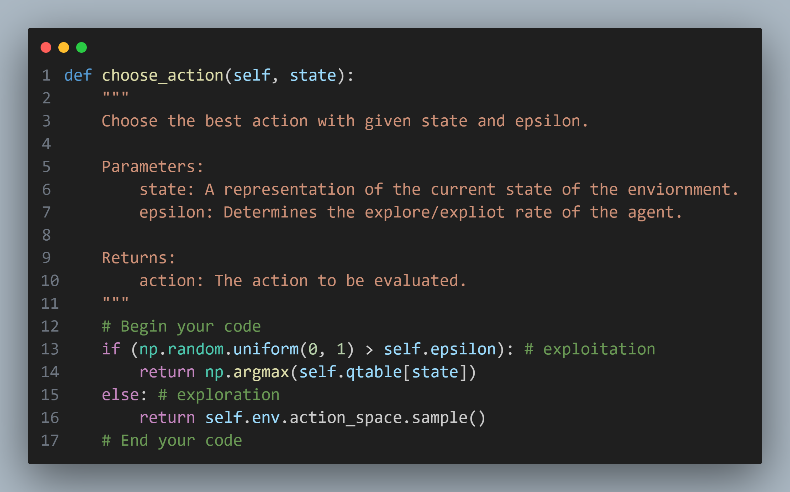
**Homework 4:**

**Reinforcement Learning**

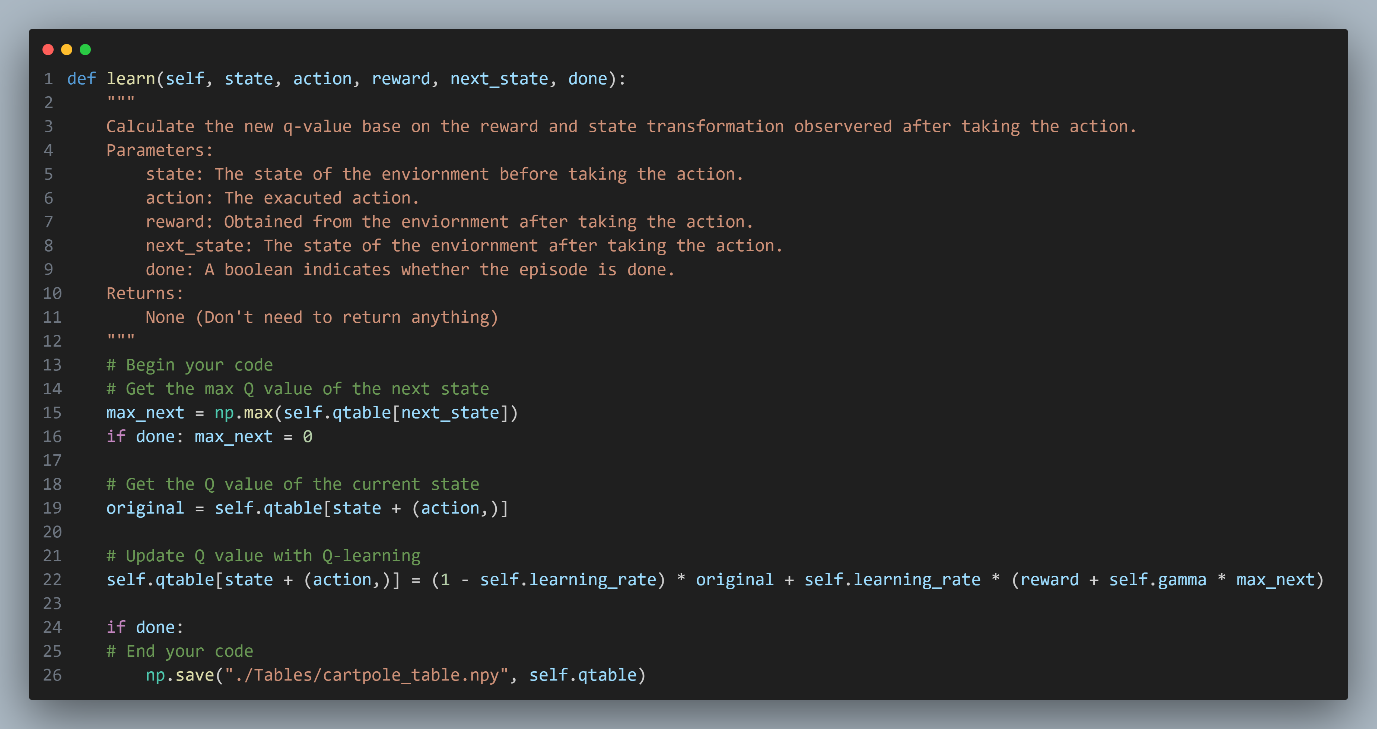
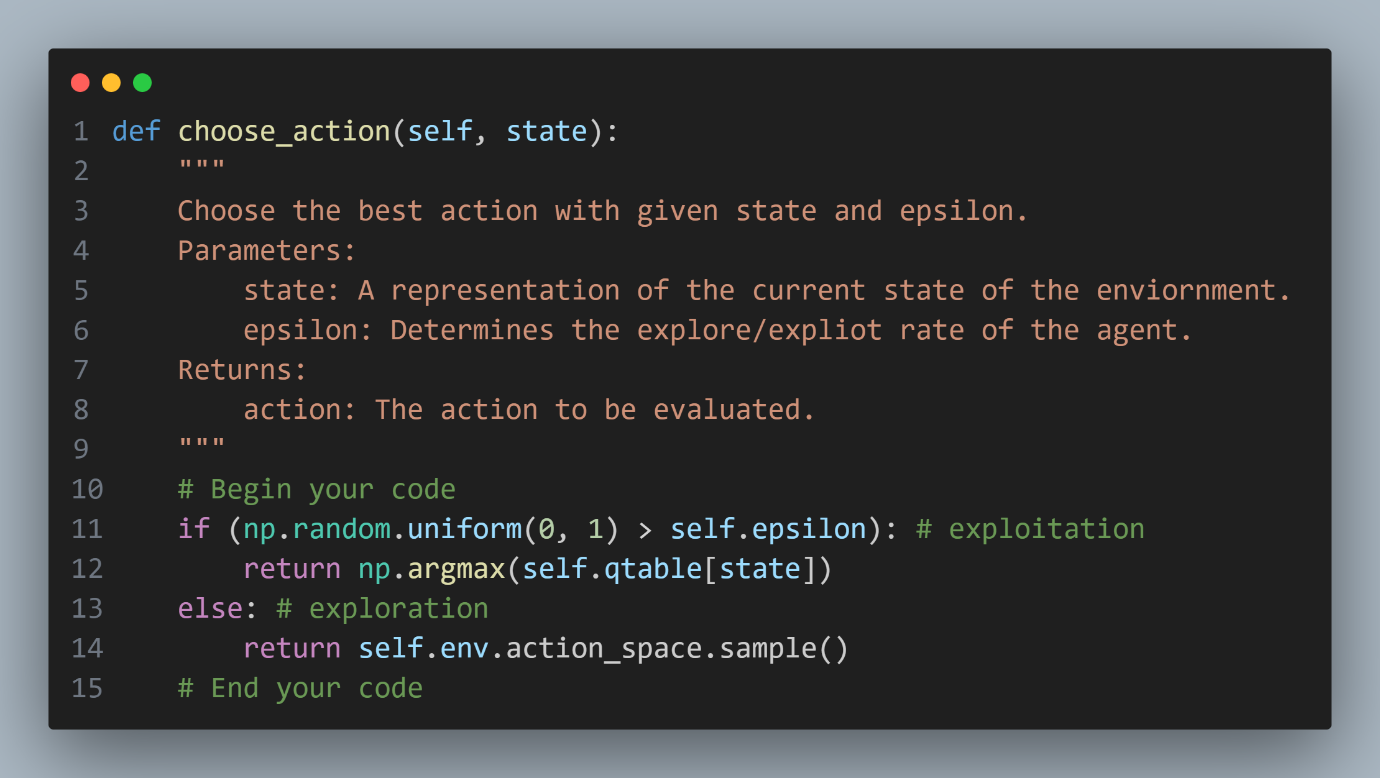
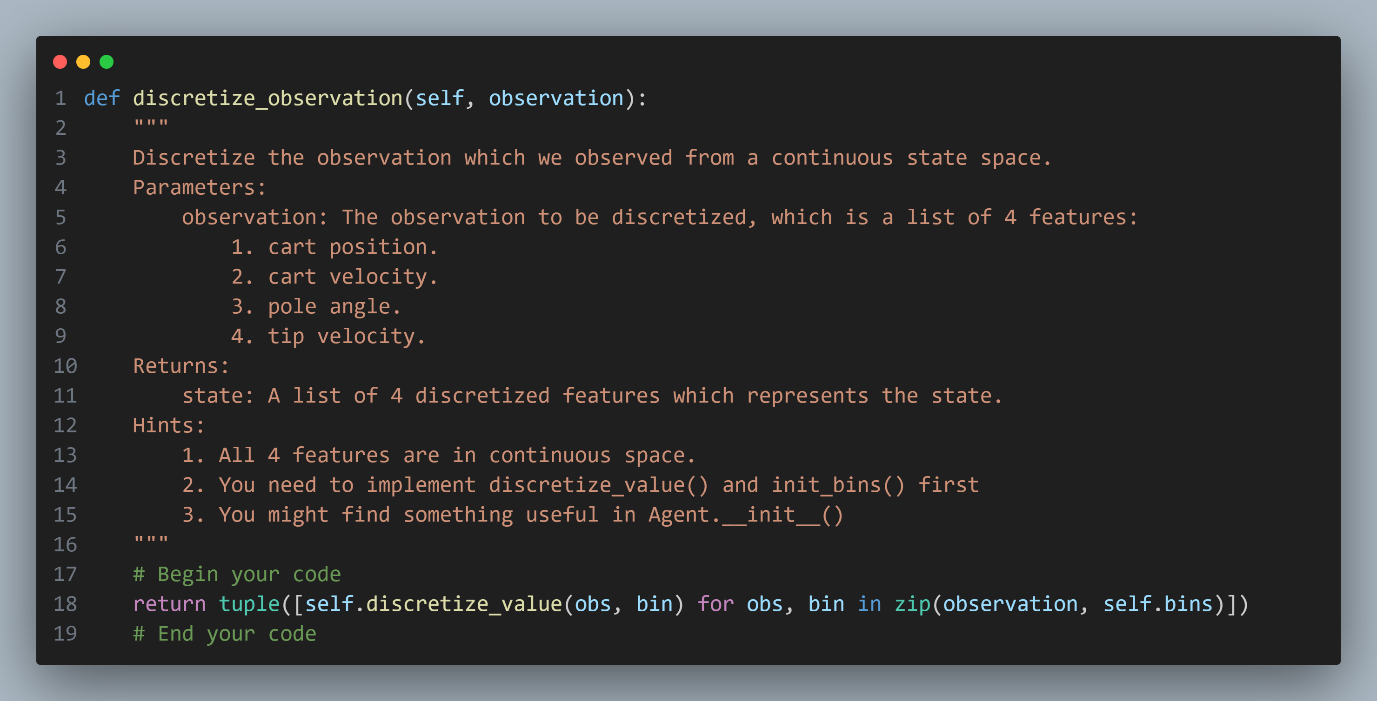
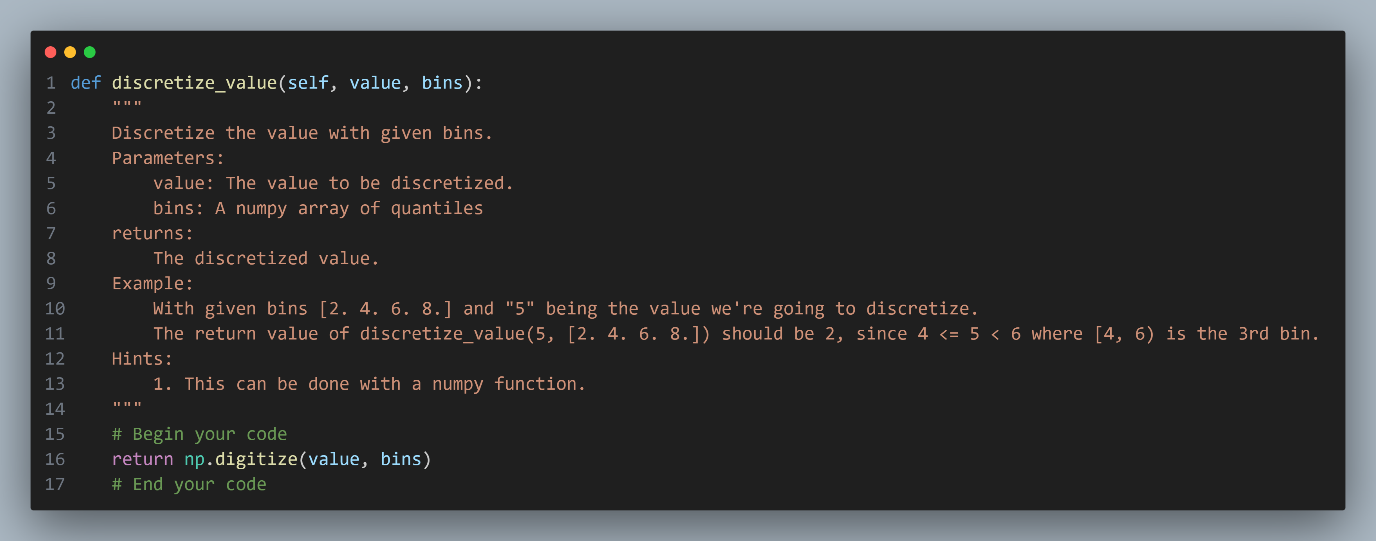
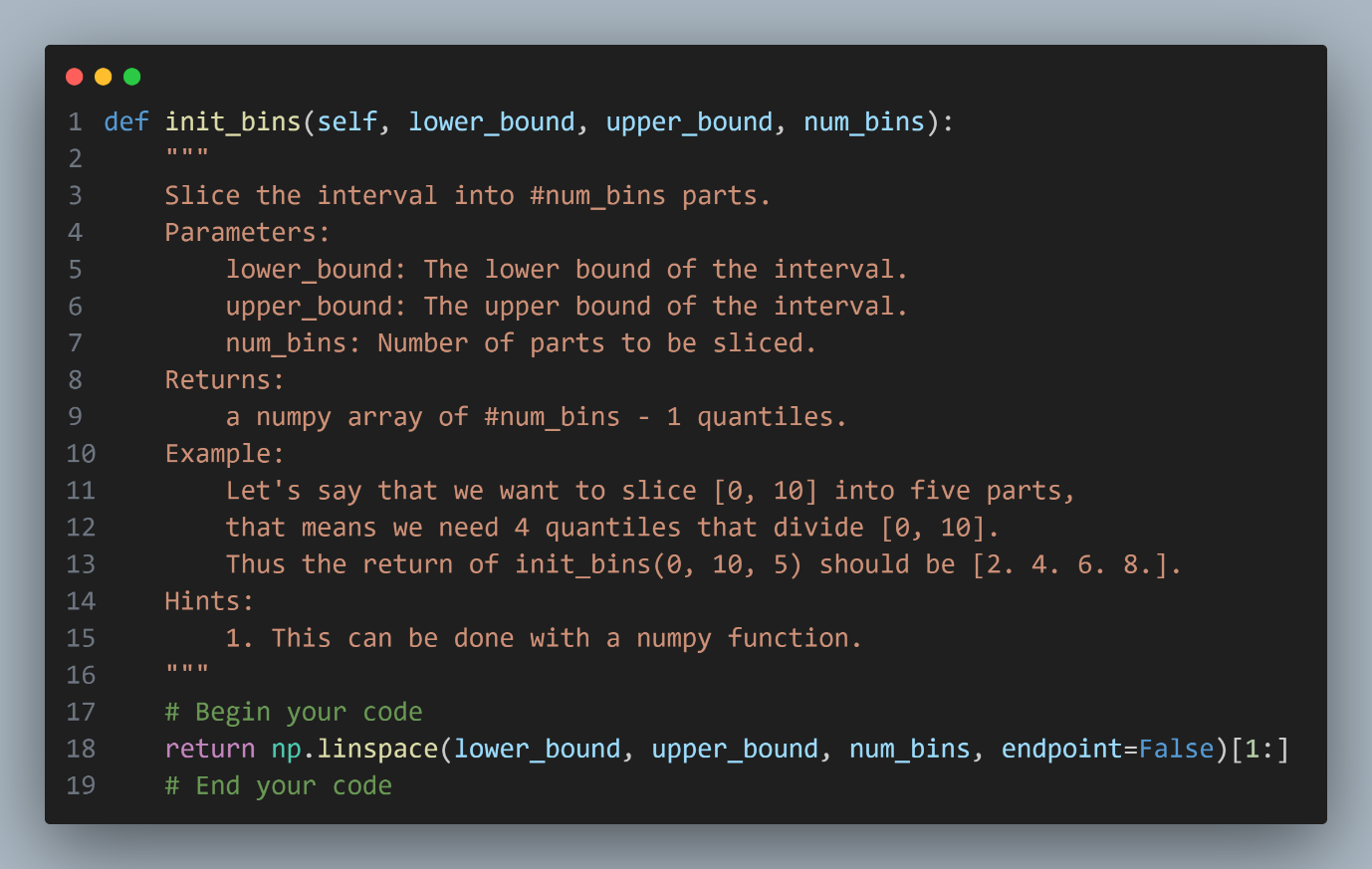
**Report Template**

**Part I. Implementation (-5 if not explain in detail):**

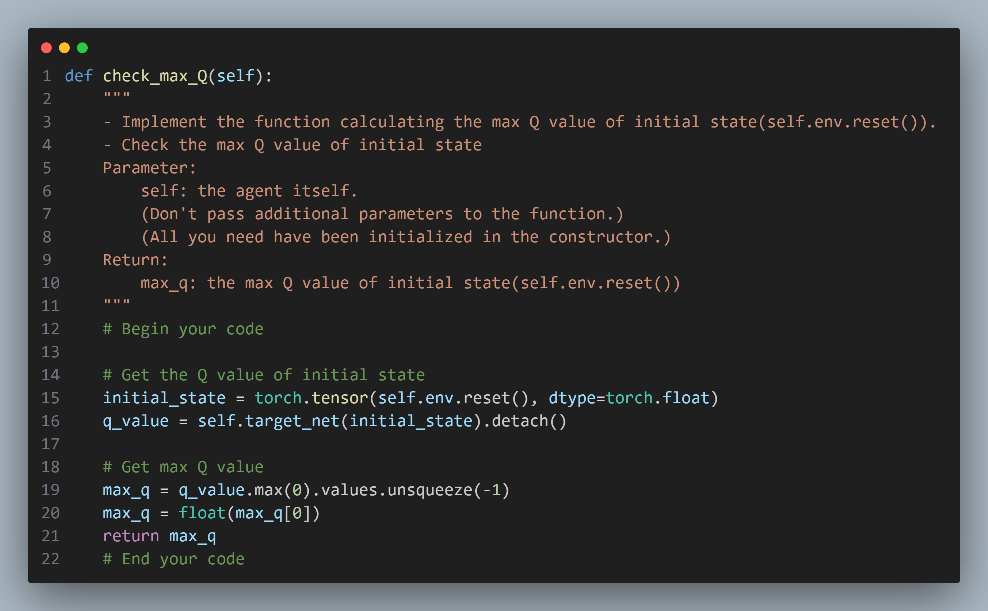
* **taxi.py**

****

* **cartpole.py**

****

* **DQN.py**

****

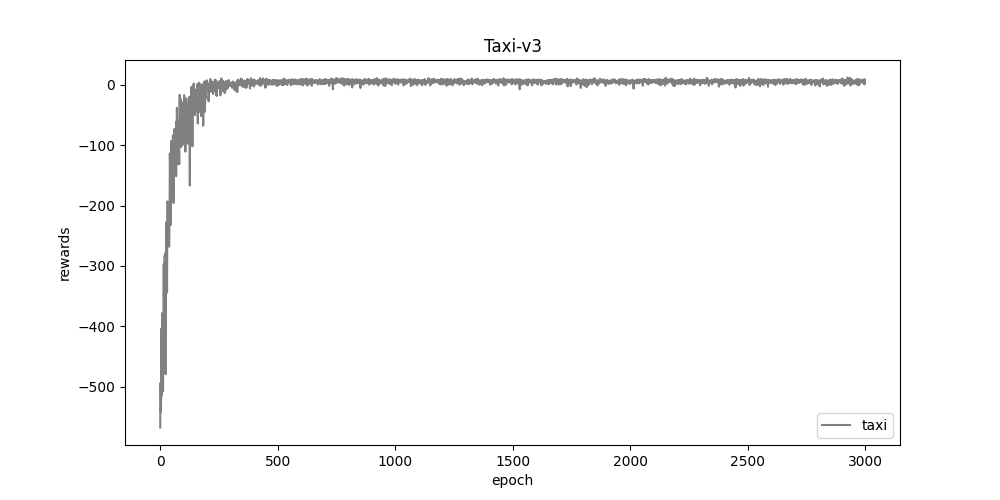
**一張含有 文字 的圖片

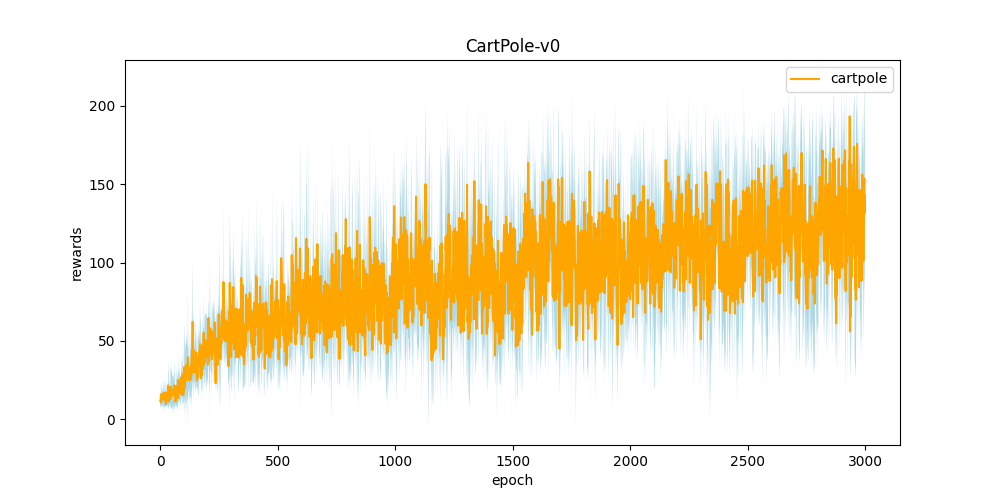
自動產生的描述**

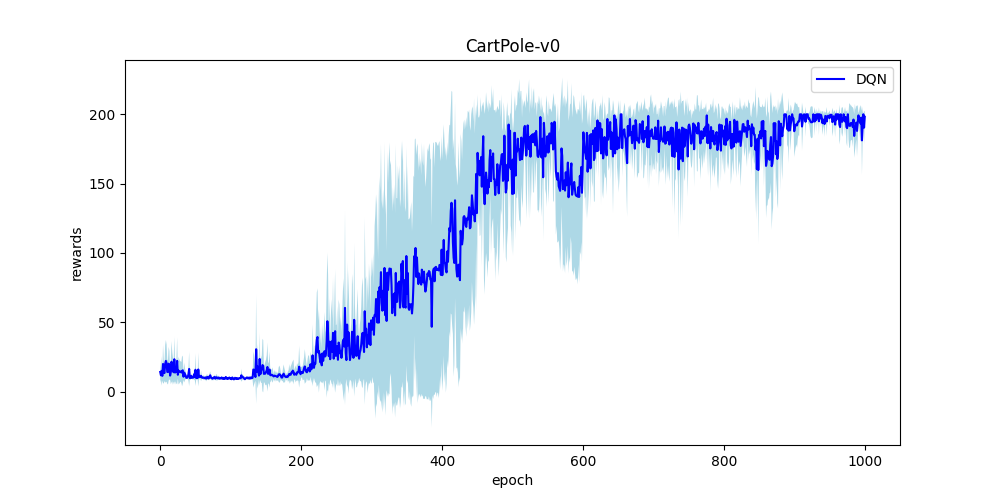
**Part II. Experiment Results:**

**Please paste taxi.png, cartpole.png, DQN.png and compare.png here.**

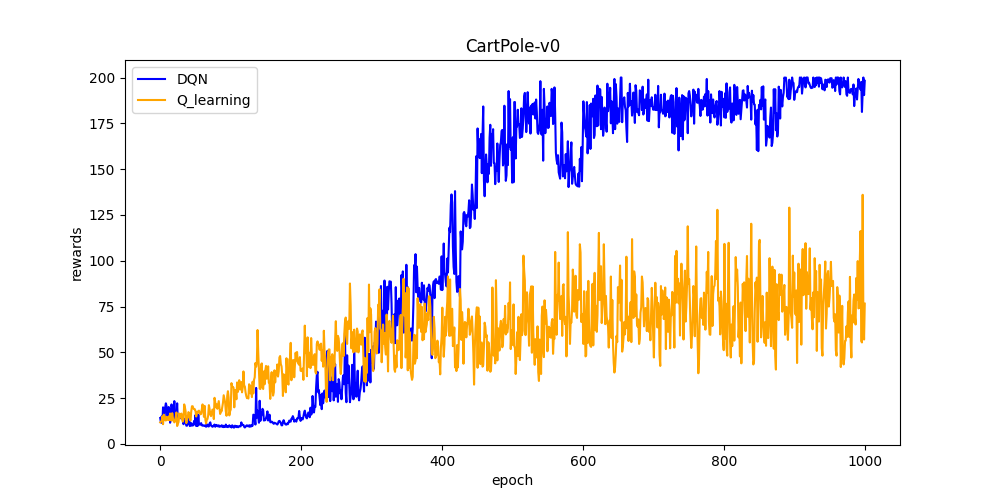
**1. taxi.png:**

****

**2. cartpole.png**

**3. DQN.png**

**4. compare.png**

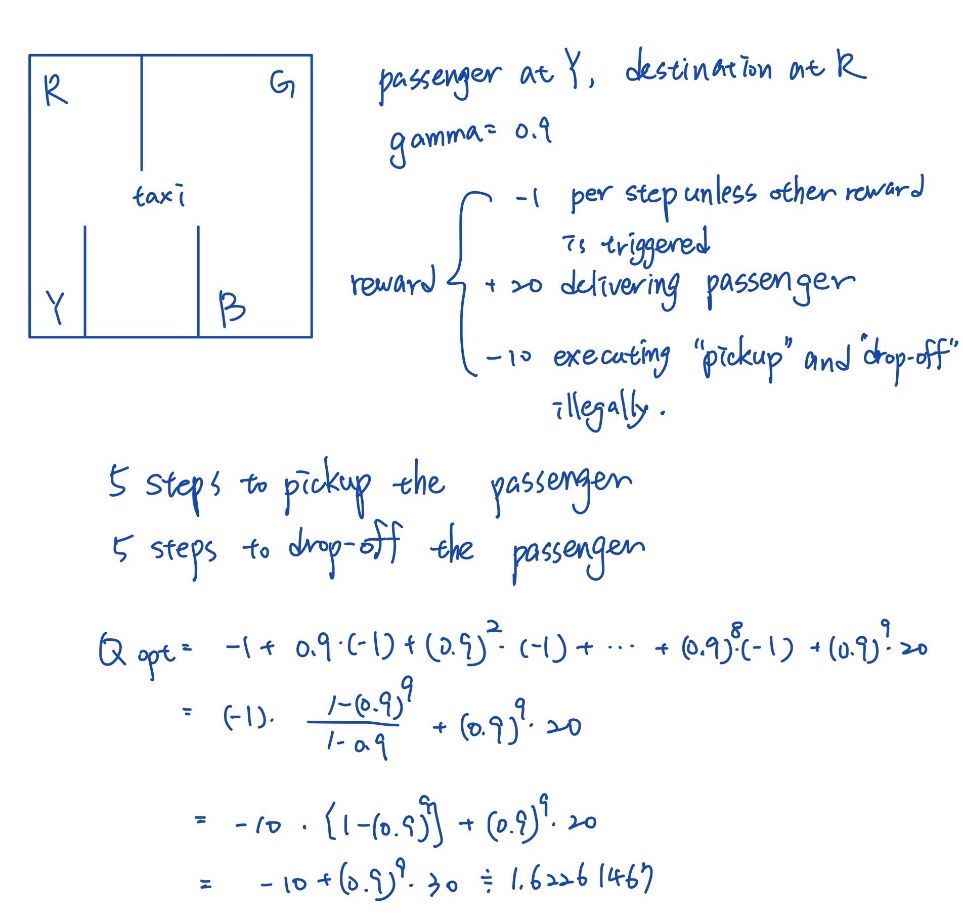
****

**Part III. Question Answering (50%):**

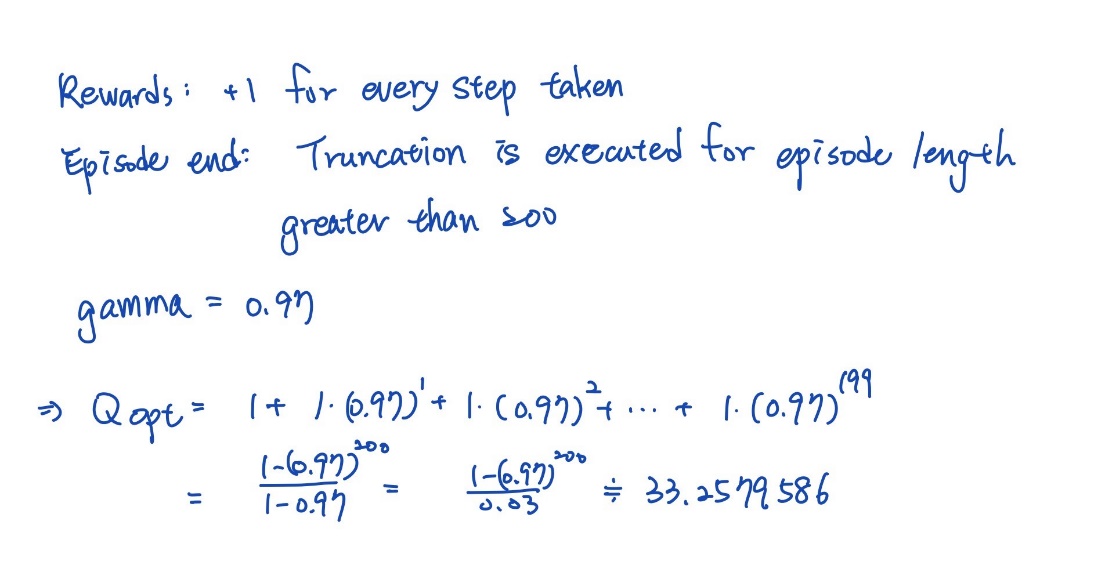
1. Calculate the optimal Q-value of a given state in Taxi-v3, and compare with the Q-value you learned (Please screenshot the result of the “check\_max\_Q” function to show the Q-value you learned). **(10%)**

一張含有 文字 的圖片

自動產生的描述



1. Calculate the max Q-value of the initial state in CartPole-v0, and compare with the Q-value you learned. (Please screenshot the result of the “check\_max\_Q” function to show the Q-value you learned) **(10%)**



1. Why do we need to discretize the observation in Part 2? **(3%)**

Because the interval is continuous, which is not easy to determine the state of the cartpole. As a result, we need to discretize it to get the state.

1. How do you expect the performance will be if we increase “num\_bins”? **(3%)**

In my opinion, the performance will become better if we increase the num\_bins, which represents the number of the states in the bounded interval. Because when we increase the number of the states, it implies that we have more states to approximate the continuous interval, which leads to the better performance.

1. Is there any concern if we increase “num\_bins”? **(3%)**

Increasing "num\_bins" can result in concerns such as longer time required to update and save the Q table due to the increased number of states, as well as the increased memory required to save the larger Q table.

1. Which model (DQN, discretized Q learning) performs better in Cartpole-v0, and what are the reasons? **(5%)**

DQN outperforms discretized Q learning in the Cartpole-v0 environment. The reason for this is that Q learning discretizes the continuous data into states, which can result in data loss. In contrast, DQN can use the continuous data and preserve more details, leading to better performance.

1. What is the purpose of using the epsilon greedy algorithm while choosing an action? **(3%)**

The epsilon-greedy algorithm serves the purpose of balancing exploration and exploitation in action selection. It allows for the selection of the best-known action while also exploring new options, thus taking advantage of prior knowledge and discovering new possibilities.

1. What will happen, if we don’t use the epsilon greedy algorithm in the CartPole-v0 environment? **(3%)**

If the epsilon-greedy algorithm is not used in the CartPole-v0 environment, there are two potential scenarios. If only exploration is used, there will be no way to choose the best-known action, as all actions will be random. If only exploitation is used, the algorithm can only rely on the known information and may miss unknown high-performance conditions without any randomness or exploration.

1. Is it possible to achieve the same performance without the epsilon greedy algorithm in the CartPole-v0 environment? Why or Why not? **(3%)**

It may be possible to achieve the same performance without the epsilon-greedy algorithm in the CartPole-v0 environment if we can find another method to replace the learning rate in the algorithm while maintaining the same proportion of exploration/exploitation. With the same distribution, there is a possibility to achieve the same performance.

1. Why don’t we need the epsilon greedy algorithm during the testing section? **(3%)**

The epsilon-greedy algorithm is not needed during the testing section because it is used for exploration and exploitation during training. In the testing section, the goal is simply to find the best path and obtain the best reward, and therefore there is no need to use the algorithm to train the data.

1. Why does “with torch.no\_grad():“ do inside the “choose\_action” function in DQN? **(4%)**

The use of "with torch.no\_grad():" inside the "choose\_action" function in DQN disables gradient calculation for every tensor, meaning that requires\_grad is set to False. This is because the function is used to choose an action and update the Q-values, but the gradients are not needed for these operations and can be disabled to improve computational efficiency.