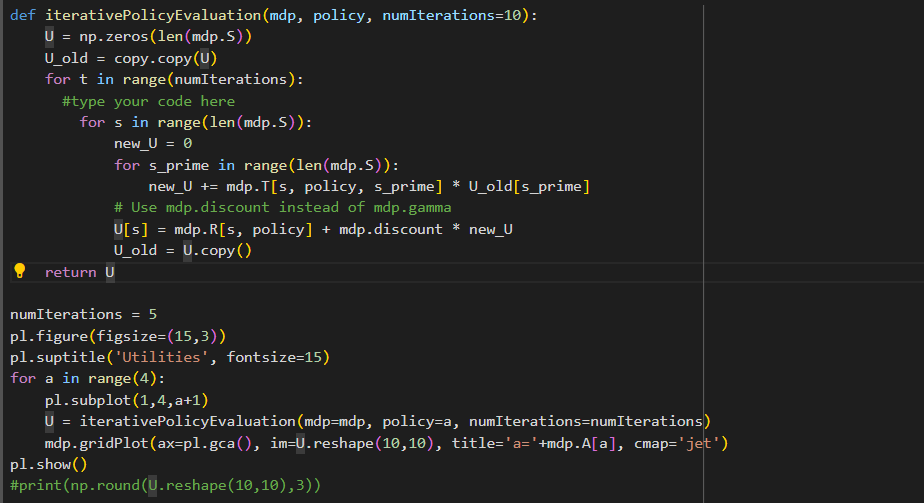
# **Lab 08**

IT21184444

GitHub Link- <https://github.com/hansika99u/DL-Lab08.git>

## Question 01

3.



A screen shot of a computer program

Description automatically generated

A screen shot of a computer

Description automatically generated

**Increased the Grid Size  
  
A computer screen shot of a program

Description automatically generated**

**A computer screen shot of a number

Description automatically generated**

A graph of blue lines

Description automatically generated

## Question 02

**Model-Based Algorithms:**

Model-based algorithms rely on learning or utilizing a model of the environment, which includes both the transition dynamics (how actions alter the environment) and the reward function. In these algorithms, the agent either learns or is given a model that predicts the next state and reward based on the current state and action. This model is then used to plan actions by simulating potential future states and rewards.

Examples of model-based algorithms include Dynamic Programming methods like Value Iteration and Policy Iteration, as well as Model Predictive Control.

**Advantages:** These methods are generally more efficient since the agent can simulate various outcomes before taking any action.

**Disadvantages:** They require an accurate model of the environment, which can be challenging or even impossible to obtain in complex or unfamiliar settings.

**Model-Free Algorithms:**

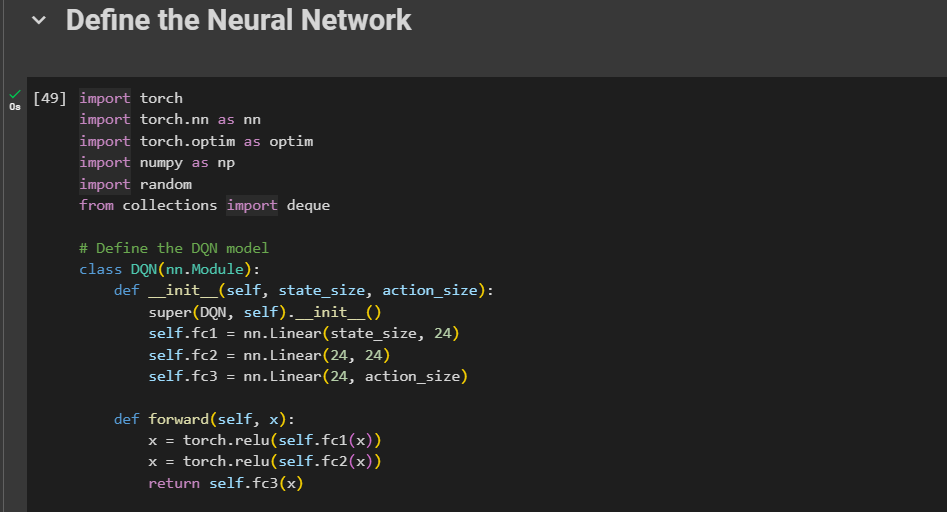
Model-free algorithms do not require learning the dynamics of the environment. Instead, they focus on directly learning a policy (which guides the agent's actions) or a value function (which estimates the value of states or state-action pairs). The agent gathers experience by interacting with the environment and improves its behavior based on these experiences, without attempting to understand the transition model.

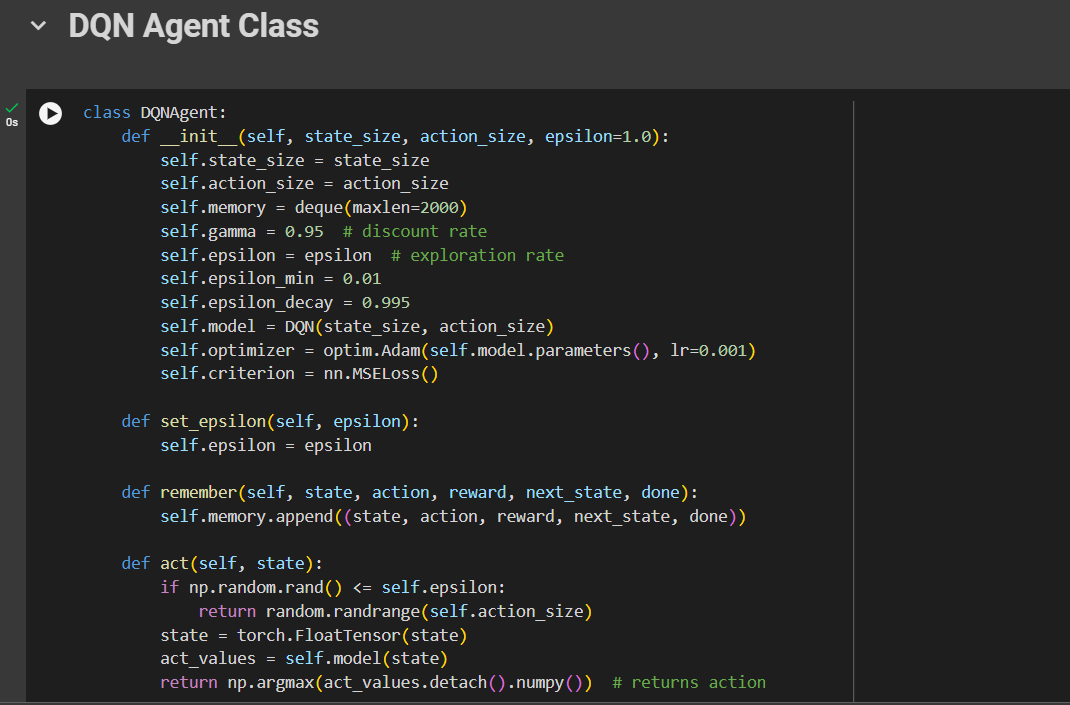
Examples of model-free algorithms include Q-Learning, SARSA, Policy Gradient methods, and Deep Q-Networks (DQN).

**Advantages:** These algorithms are simpler as they don't need a model of the environment, making them suitable for complex or unknown environments.

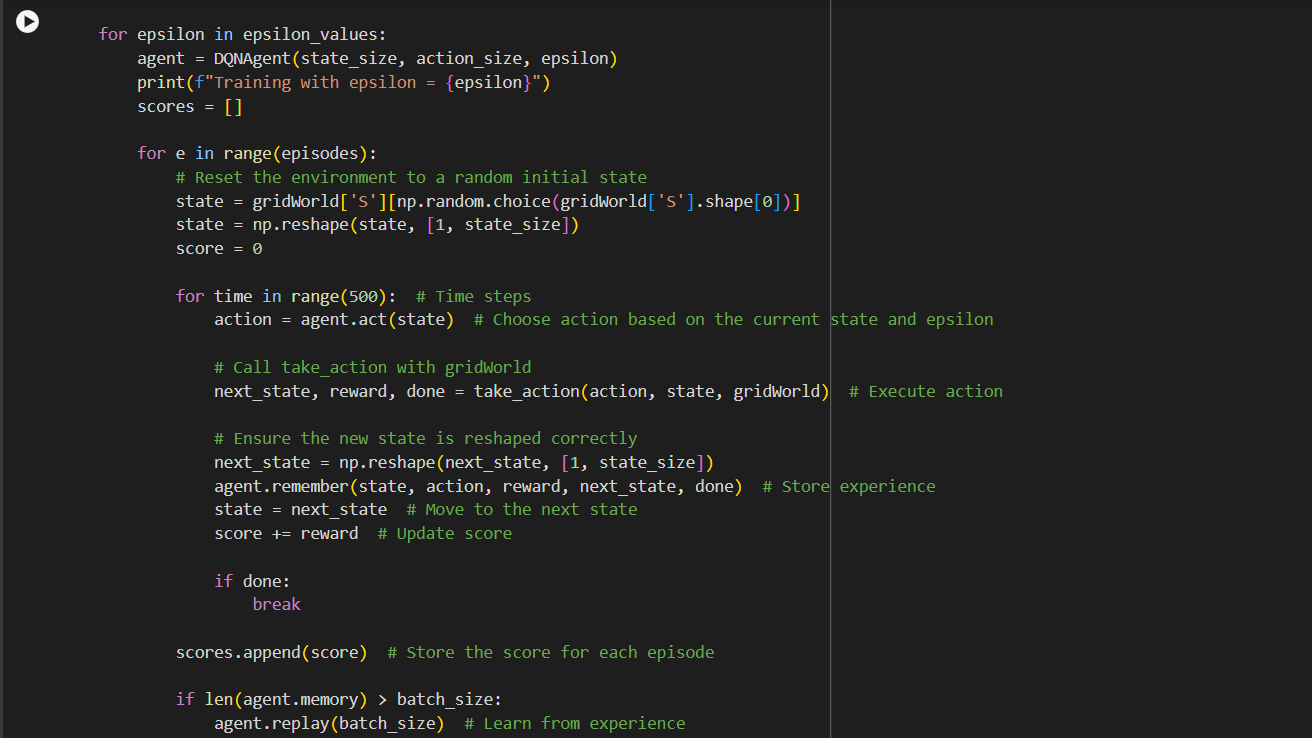
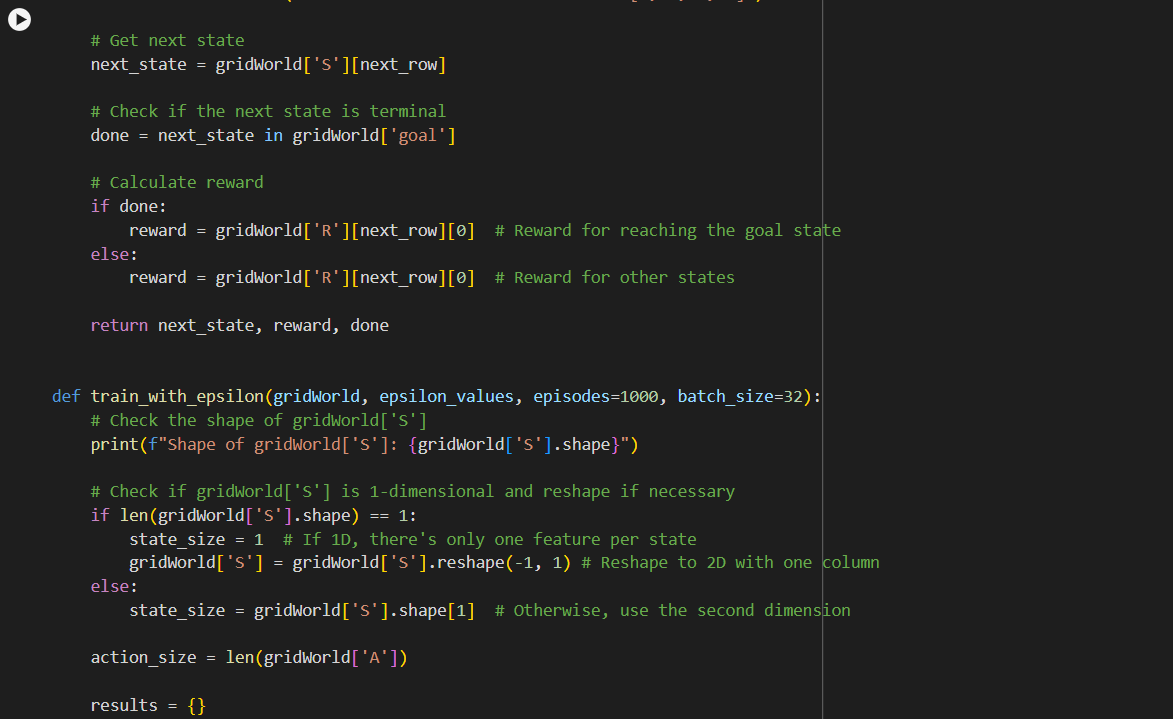
**Disadvantages:** They tend to be less sample-efficient, requiring a large number of interactions with the environment to learn optimal behavior.

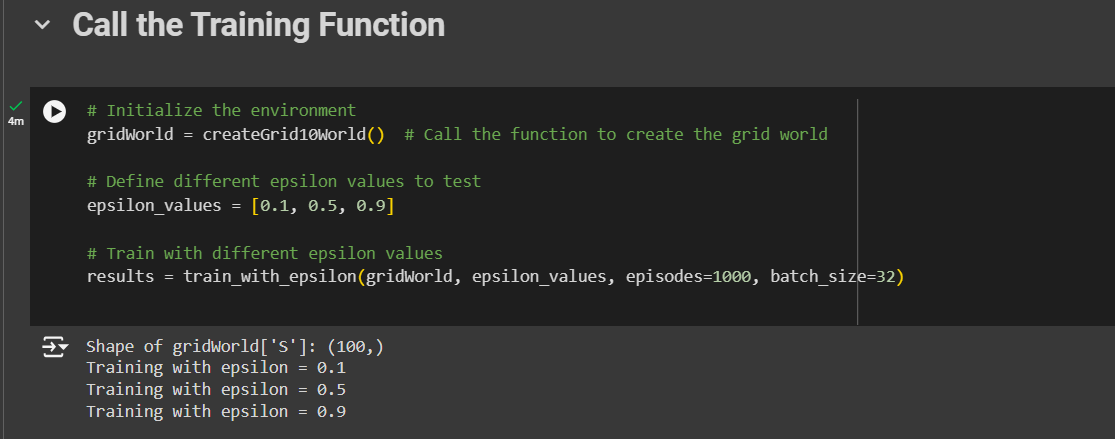
## Question 03

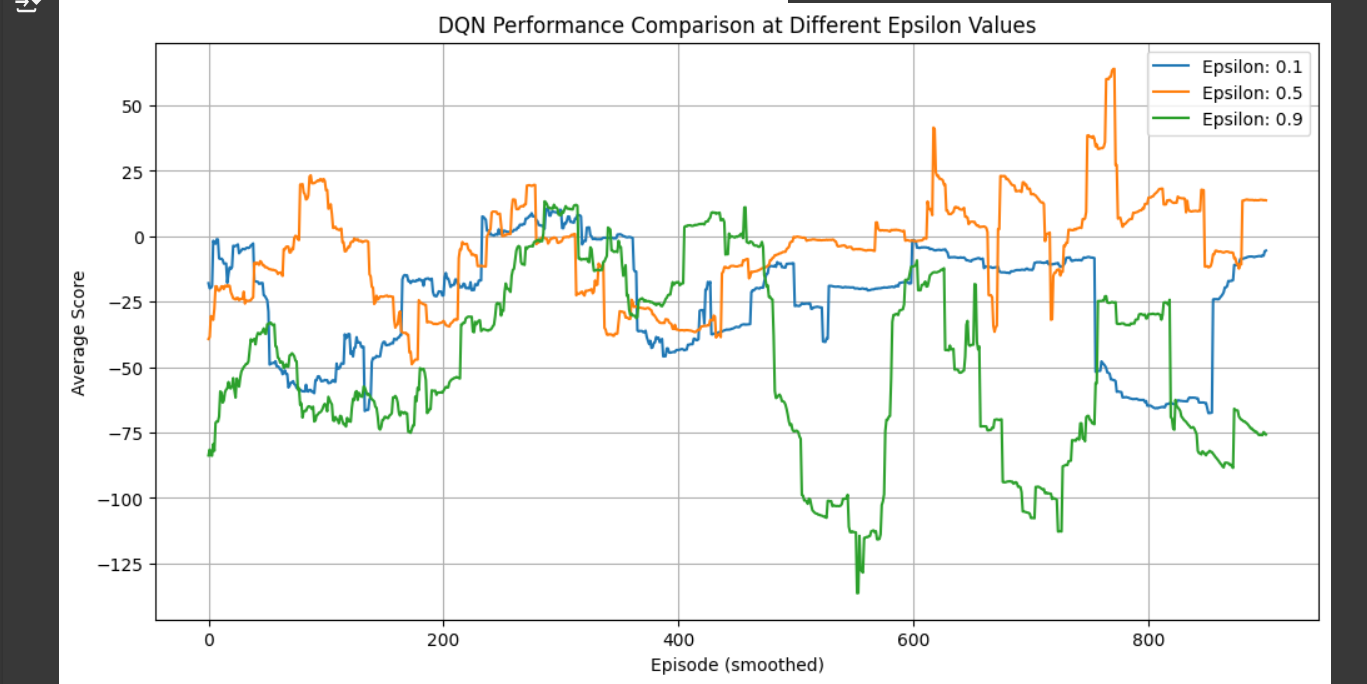


A screen shot of a computer program

Description automatically generatedA screenshot of a computer program

Description automatically generatedA screen shot of a computer code

Description automatically generated



**Epsilon = 0.1 (Blue Curve)**  
At an epsilon value of 0.1, the DQN agent leans more towards exploiting its learned knowledge rather than exploring new actions. This leads to behavior where the agent sticks to familiar strategies, minimizing experimentation. The performance remains stable, with fewer dramatic spikes or drops, indicating that the agent has converged to a reasonably good policy early on. However, due to the limited exploration, the agent’s performance, while consistent, may not reach the highest possible peaks. Convergence is fast, and the agent stabilizes quickly around a set policy, though the performance may be suboptimal.

**Epsilon = 0.5 (Orange Curve)**  
With an epsilon of 0.5, the DQN agent maintains a balance between exploration and exploitation. In this setting, the agent tries new actions while also leveraging its learned policy. This results in more performance fluctuations compared to an epsilon of 0.1. The agent occasionally achieves higher scores but also experiences setbacks as it explores suboptimal strategies. This variance reflects ongoing policy adjustments as the agent continues to explore. Convergence is slower due to continued exploration, but the agent may discover better strategies over time.

**Epsilon = 0.9 (Green Curve)**  
At an epsilon of 0.9, the agent heavily favors exploration over exploitation, frequently trying new actions instead of relying on its learned strategies. As a result, performance is erratic, with sharp declines and recoveries. The agent is in constant experimentation mode, which leads to frequent discoveries of new strategies, though many are ineffective, resulting in inconsistent scores. Convergence is much slower compared to lower epsilon values as the agent struggles to settle on a stable policy. Despite the inconsistency, the extensive exploration might eventually uncover better strategies over time.

In summary, different epsilon values create a trade-off between exploration and exploitation in DQN models. A low epsilon (0.1) promotes stability and fast convergence but may limit the potential for discovering optimal strategies due to reduced exploration. A medium epsilon (0.5) balances exploration and exploitation, leading to moderate performance gains with more variability. A high epsilon (0.9) encourages extensive exploration, resulting in slower convergence and unstable performance but with the potential for long-term strategy improvement. The choice of epsilon depends on the desired balance between quick convergence and discovering better strategies.