Implementation of Qtable:

The data structure I chose for the initialization of Qtable is a 2D array in which I took all the possible states in account. At first, I chose a dictionary as my initial Qtable in that I tried using various inbuilt functions to get the action from the table but it was a little complicated but with the array I was able to play with the Qtable as I wished it made my work easier. The 2D array makes it easy to index and update the Q-values for a specific state-action pair, which is essential for implementing the Q-learning algorithm. It allowed me for a compact and convenient representation of the Q-values for all state-action pairs in the environment.

Q-values update:

In the snake environment we update the Q-values using the Q-learning algorithm.

1. First, we initialize the Qtable as stated above.
2. At each of the step the agent selects an action using the select greedy and select action function. Let's say at time t the agent is at state (1,1) and selects action "right".
3. The environment updates the state based on the selected action and the agent receives a reward. In this case the next state would be (1,2) and the reward might be 1 for eating an apple.
4. The agent then updates the Q-value for the state-action pair (1,1, right) using the Q-learning update rule:
   Q [(1,1)] [2] = Q [(1,1)] [2] + alpha * (reward + gamma * max (Q [(1,2)]) - Q [(1,1)] [2]), where alpha is the learning rate and gamma is the discount factor
5. Repeat the steps 2 to 4 until the agent reaches a terminal state or a maximum number of steps is reached.

This process continues until the agent has explored the environment enough times and the Q-values converge to optimal values that give the maximum expected reward.

Comments on the snake experiment:

At first without implementing the Q-learning I ran the code and saw how the snake tries to get the apple as we know those were random actions it did not perform well. But, after the implementation of the Q-learning in the agent the snake got better at catching the apples although when I saw the snake movement in the terminator the snake gets hard choosing the correct action if the apple is near the wall but if it is in the center of the grid the snake had no issues at most of the time. This is what I noted from the terminator.

Experiment 1 (epsilon=0.1):

Expectation: The epsilon is the exploration-exploitation trade-off which means that it is the balance between choosing the actions based on the current best knowledge (exploitation) or trying out new actions to gather more information (exploration). Since the epsilon is low the action will be mostly selected as greedy so I expected it to give a moderately good performance.

Result: The result was surprising it performed more than I expected this is because I think the action was mostly greedy rather than random selection of action and this had the following for the statistics:

```
** Mean: Return= -31.103, #Apples=44.3, #Stops=18.0, #GoodSteps=833.6, #StatesVisited=75.3
```

Experiment 2 (epsilon=0.3):

Expectation: After the first experiment I thought increasing the epsilon would lead to less selection of greedy action than the previous experiment so I expected this experiment to perform not that good than the previous experiment.

Result: The result was just as I expected the actions might be random more in this experiment than the previous experiment but also surprisingly it was not a very big difference rather it was just a slight variation not a significant drop in the statistics as shown below:

```
** Mean: Return= -43.343, #Apples=40.4, #Stops=22.8, #GoodSteps=826.8, #StatesVisited=80.9
```

But in terms of unique states visited this overtook the previous experiment.

Experiment 3 (epsilon=0.5):

Expectation: After two experiments I expect this experiment to perform less than the previous two because we are increasing the epsilon value again and this time it maybe more random action than the previous two experiments.

Result: But the result was not what I expected rather it was better than the previous one (Experiment 2) but not better than the first (Experiment 1). At this moment I realized that this is Q-learning and we might not know what is going to happen next :)

```
** Mean: Return=  -5.570, #Apples=41.5, #Stops=17.2, #GoodSteps=804.2, #StatesVisited=74.6
```

Experiment 4 (epsilon=0.7):

Expectation: Now with the more increased epsilon I was expecting this to perform not good than the others because the select action function will choose the greedy or random from this value so it would in case produce more random action

Result: The result was just as I expected because of the select action function depends on this parameter to choose the best action the more the value higher the chance of bad results in my point of view. The statistics for this are shown below:

```
** Mean: Return= -15.481, #Apples=34.0, #Stops=20.8, #GoodSteps=791.6, #StatesVisited=74.1
```

Experiment 5 (epsilon=0.9):

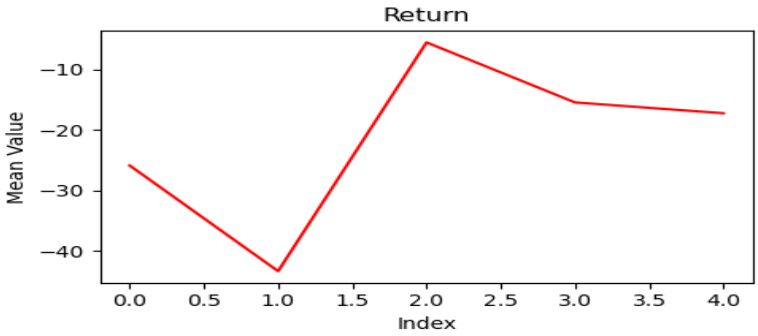
Expectation: Since again the epsilon is higher than the previous experiments I expect this to perform not good than any of the other experiments including the experiment 4 with epsilon=0.7.

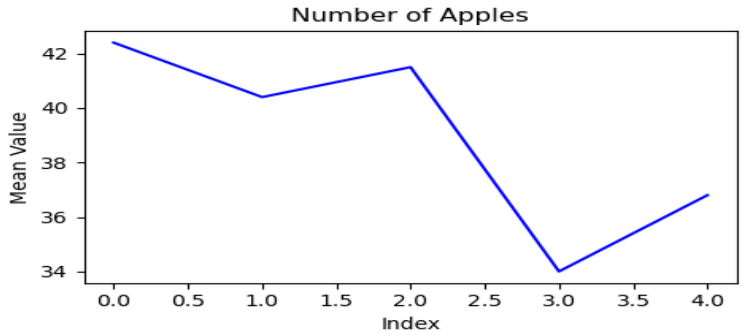
Result: Again, the result was not I was expecting because the qtable updates as I stated in the above maybe this one has a better state and for that better action was chosen. The statistics of this are show:

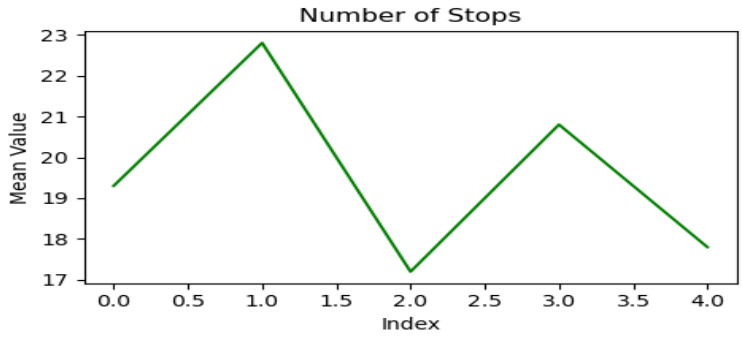
```
** Mean: Return= -17.266, #Apples=36.8, #Stops=17.8, #GoodSteps=767.9, #StatesVisited=73.6
```

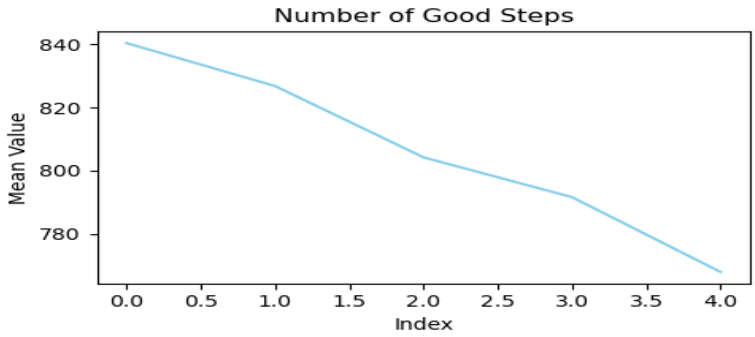
General Reflections:

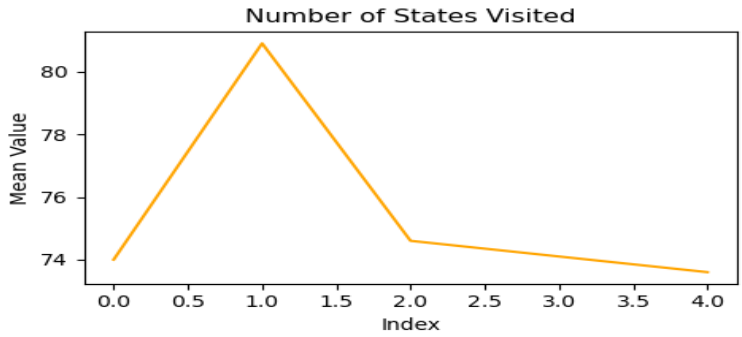
The difficulty of this assignment was 10/10 for me because I chose a data structure at first which was not that convenient than the 2D array data structure. The main reason it was hard because while restarting and running the kernel it takes time to get updated in the q-learning file which I didn't notice earlier and changed the code every time it showed me an error but eventually I figured it out by putting a dummy print statement to figure out when the code is updating in the q-learning. From this exercise I learned that Qtable data structure is the important thing to choose and I also got to know how the snake performs after the Q-learning. Overall this assignment took me a while to figure out.


Return
Mean Value
−10
−20
−30
−40
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
Index


Number of Apples
Mean Value
42
40
38
36
34
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
Index


Number of Stops
Mean Value
23
22
21
20
19
18
17
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
Index


Number of Good Steps
Mean Value
840
820
800
780
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
Index


Number of States Visited
Mean Value
80
78
76
74
0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0
Index