Assignment - 4

Academic Integrity:

I (We) certify that the code and data in this assignment were generated independently, using only the tools and resources defined in the course and that I (we) did not receive any external help, coaching or contributions during the production of this work.

PART 1:

1. Describe the environment that you defined. Provide a set of actions, states, rewards, main objective, etc.

Answer) **Theme**: Tom and Jerry Grid World with cheese as positive rewards and a trap or tom as the negative rewards.

Actions: Up, Down Left, Right

States: {S1 = (0,0), S2 = (0,1), S3 = (0,2), S4 = (0,3), S5 = (0,4), S6 = (1,0), S7 = (1,1), S8 = (1,2), S9 = (1,3), S10 = (1,4), S11 = (2,0), S12 = (2,1), S13 = (2,2), S14 = (2,3), S15 = (2,4), S16 = (3,0)}

Rewards: +5,+8,-2,-3,+10

Objective: To reach the goal without being caught by tom or in the trap.

Total number of states = 16

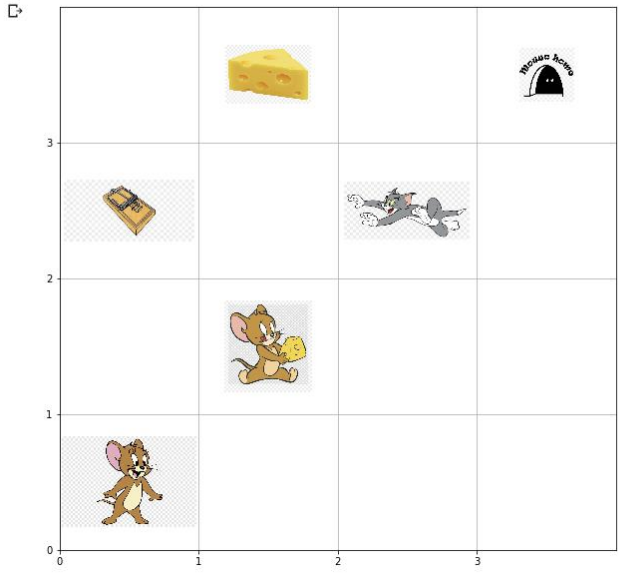
Total number of rewards = 5

Total number of actions = 4

2. Provide visualization of your environment.

Answer)

```
env = GridEnvironment()
obs = env.reset()
env.render()
```



3. Safety in AI: Write a brief review explaining how you ensure the safety of your environment.
Answer) The clip function in the code helps the environment to not cross the threshold or the grid value and move out. Also, I have implemented some of the conditions where in, if the value of the action function forces the agent to move out of the grid the conditions placed will not allow it to do so. These are the safety measures I took for my environment to be safe.

PART 2:

1. Show and discuss the results after applying SARSA to solve the environment defined in Part I.
Answer) After applying the SARSA algorithm to the environment, the Q table is as follows:

```
print(Q)
```

```
[[2.45086717 6.90325125 1.76986111 9.62685323]
 [0.         5.175083   0.57       0.3       ]
 [0.3        0.         0.3        0.3       ]
 [0.3        0.3        0.         0.3       ]
 [2.15723292 0.4766374  0.         0.        ]
 [0.         0.         0.         0.        ]
 [0.9434457  0.         0.         0.        ]
 [0.62838    0.3        0.591      0.6141189 ]
 [0.         0.         2.13446896 0.        ]
 [0.3        0.         1.67326874 0.        ]
 [0.         0.32247    1.61087476 0.        ]
 [0.3        1.52157054 0.         0.3       ]
 [0.         0.         0.         0.        ]
 [0.3        0.321      0.         0.        ]
 [0.         0.         0.         0.        ]
 [0.         0.         0.         0.        ]]
```

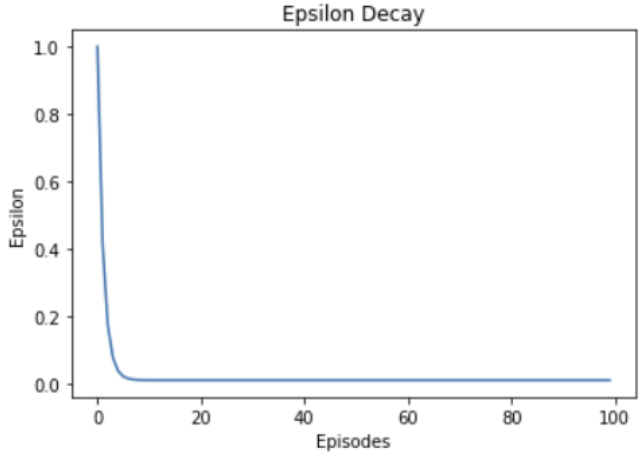
The Q table is updated randomly by choosing the value of greedy by the method of exploration there by making the environment to explore all possible way to reach its goal.

For this result is used an epsilon of exponential (np.exp()), with which I was able to choose the greedy approach.

2. Provide a plot for epsilon decay.
Answer)

```
# print(epsilon_hist)
plt.plot(epsilon_hist)
plt.title('Epsilon Decay')
plt.xlabel('Episodes')
plt.ylabel('Epsilon')
```

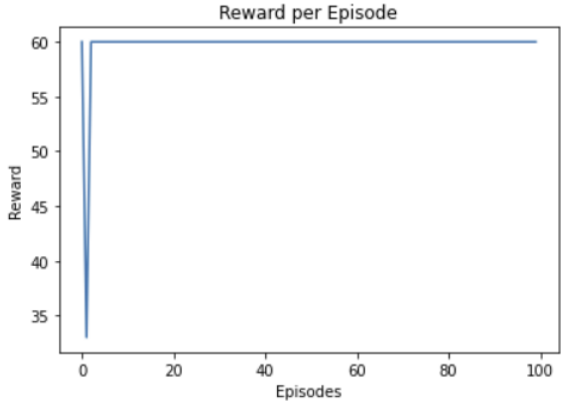
```
Text(0, 0.5, 'Epsilon')
```



3. Provide a plot for the total reward per episode.
Answer)

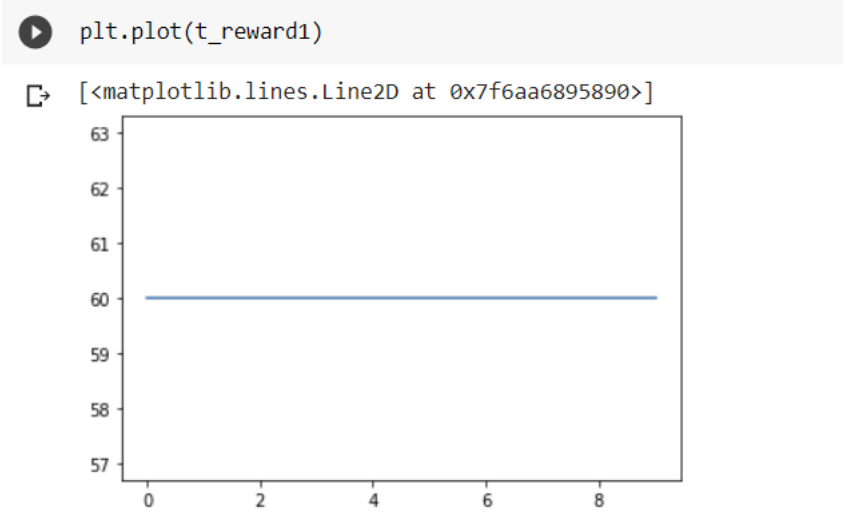
```
# print(t_reward)
plt.plot(t_reward)
plt.title('Reward per Episode')
plt.xlabel('Episodes')
plt.ylabel('Reward')
```

```
Text(0, 0.5, 'Reward')
```



4. Provide the evaluation results. Run your environment for at least 10 episodes, where the agent chooses only greedy actions from the learnt policy. Plot should include the total reward per episode.

Answer) The below graph shows the total reward graph when the agent chooses only greedy action from the learnt policy.



5. Give your interpretation of the results.
Answer) From the results, it is clear that the greedy policy approach will result in a constant reward for the environment there by making it the exploitation approach. This proves that a exploitation method is the easiest way to get a SARSA greedy policy approach to make it easy to get the rewards.

6. Briefly explain these tabular methods: SARSA and Q-learning. Provide their update functions and key features.
Answer) SARSA and Q-learning are two reinforcement learning methods that do not require model knowledge, only observed rewards from many experiments. SARSA and Q-learning make the update after each step.

Update Functions:

SARSA:

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma Q(s',a') - Q(s,a))]$$

Q-Learning:

$$Q(s,a) = Q(s,a) + \alpha[r + \gamma \max_{u} Q(s',u) - Q(s,a))]$$

7. Provide the analysis after tuning at least two hyperparameters from the list above.

Answer) The Hyperparameters that I tuned were maximum timesteps, total number of episodes and also learning rate.

At first, I tuned the maximum timesteps and total number of episodes this resulted in a much clear reward. This helped me to understand how the reward was varying when the timesteps were varying in each episode. Lesser the episodes easier the rewards to be detected.

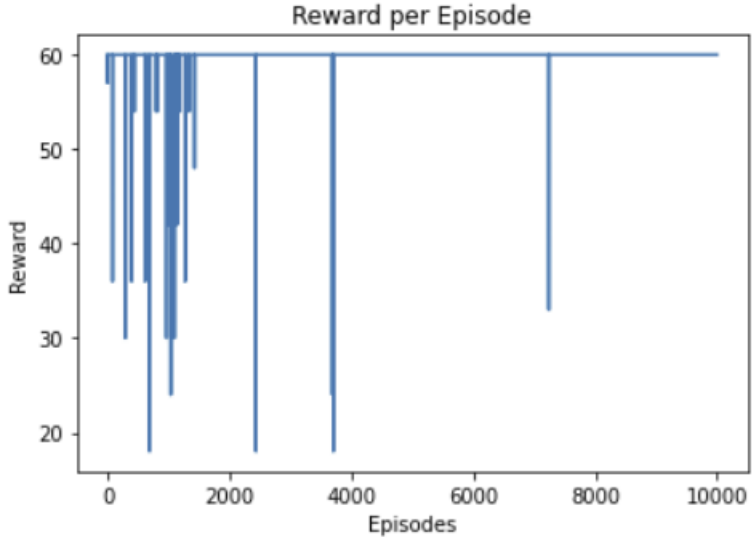
In the second method, I change the learning rate to various decimal places, this resulted in drastic change in the Q table there by making learning rate a key factor that influences the Q table.

8. Try at least 3 different values for each of the parameters that you choose. Provide the reward graphs and your explanation for each of the results. In total you should have at least 6 graphs and your explanations.

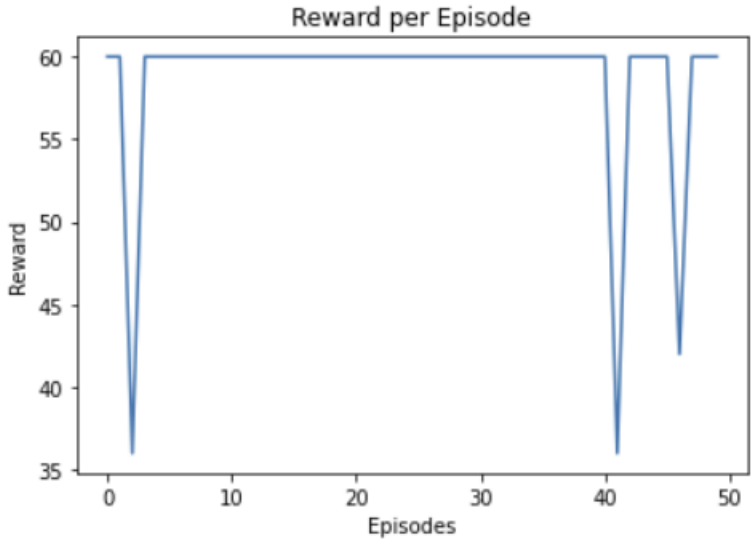
Answer)

Graph 1: # Number of episodes = 10000 & max timesteps = 100

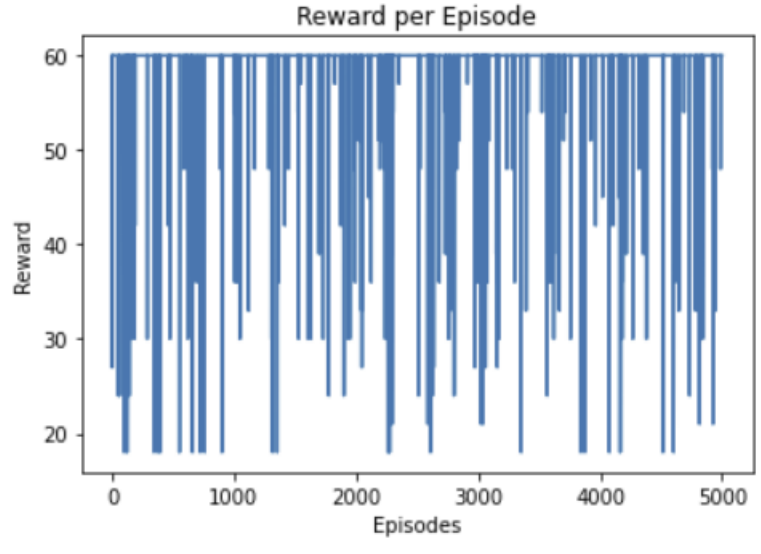
```
Text(0, 0.5, 'Reward')
```



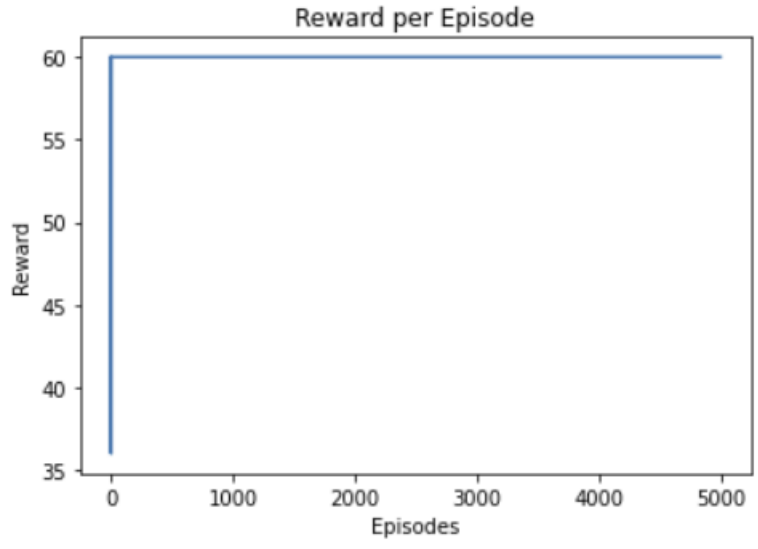
Graph 2: # Number of episodes = 50 & max timesteps = 5



Graph 3: # Number of episodes = 5000 & max timesteps = 5

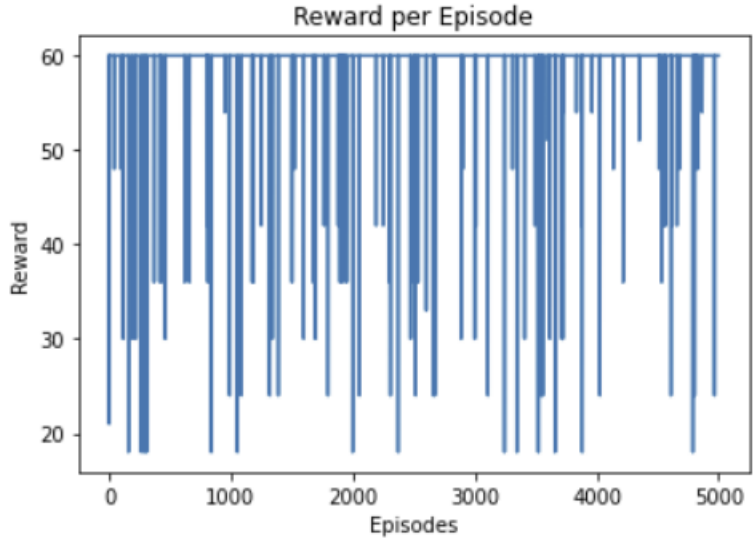


Graph 4: #learning rate = 0.01



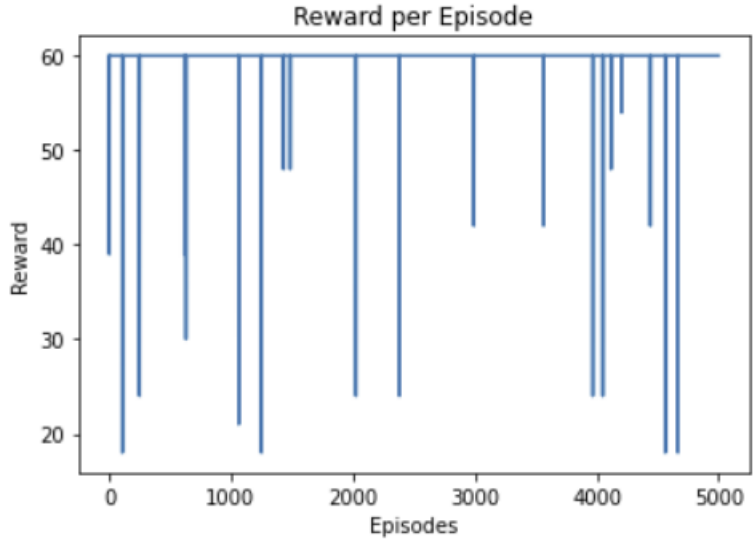
Graph 5:

#learning rate = 0.0001



Graph 6:

#learning rate = 0.5



References:

https://medium.com/swlh/introduction-to-reinforcement-learning-coding-sarsa-part-4-2d64d6e37617
https://www.w3schools.com/python/numpy/default.asp
https://www.w3schools.com/python/matplotlib_intro.asp
https://tcnguyen.github.io/reinforcement_learning/sarsa_vs_q_learning.html