CSE 546: Reinforcement Learning
Assignment 1

Sannihith Kilaru

UB Person #: 50428194

Abstract

Created a grid world(deterministic/stochastic) in which an agent can move up, right, left, down. The actions done in the environment can be stochastic or deterministic. Apply two tabular methods to solve both deterministic and stochastic environments.

```
-0.5

 0.0

 05

 ]0
```

Part 1 [Total: 50 points] - Defining RL environments:

- For defining the RL environments which are for this case Deterministic and Stochastic environment that are based on Markov Decision process. I have used OpenAI gym environment in google colab

Report Q/A for part 1:

1. Describe the deterministic and stochastic environments, which were defined (set of actions/states/rewards, main objective, etc).

   Ans:
   - Action Set: { RIGHT, DOWN, LEFT, UP}
   - State set: {S1, S2, S3,…, S16}
   - Reward Set: {10,5,3,1}
   - Agent position: (0,0)
   - Goal position: (3,3)
   - Objective: Main objective of the agent is to reach the goal position which gives the highest reward to the agent of +10.
   -

   Deterministic environment: In this environment the transition probability is defined as P(s ′ , r|s, a) = {0, 1}, which means that the agent takes the action that it wanted to with a 100% probability

   Stochastic environment: In this environment the transition probability is defined as $\sum_{s',r} P(s', r|s, a) = 1$, which means that the agent takes the action that it wanted to with some uncertainty. Probability of that agent doing that action <100%

   For example, if there is play in a robot's gear and it doesn't complete one full rotation, then it will not have moved as much as it wanted to. This creates uncertainty.

2. Provide visualizations of your environments.

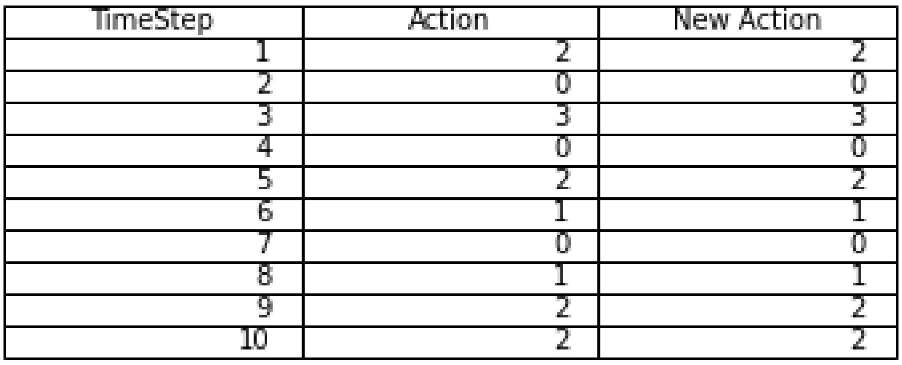
   Ans:
   For visualizing the movement of the agent in the environment I have shown 10 timesteps of the agent's movement in both environments. You can see stochasticity being introduced from the table.

-0.5

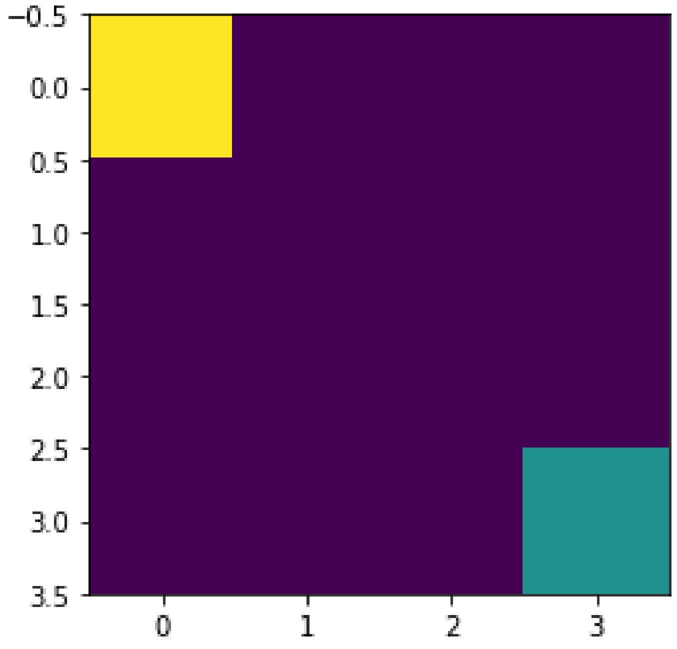
0.0 Visualisation (Deterministic environment):

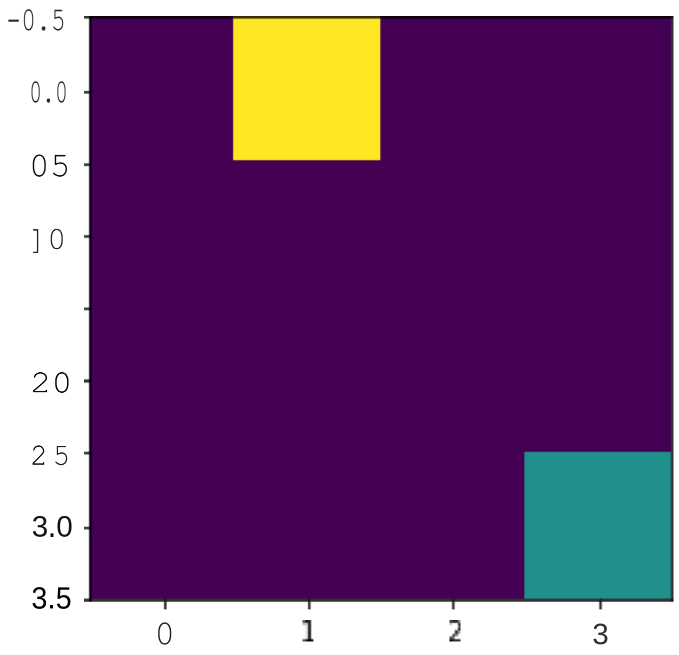
05

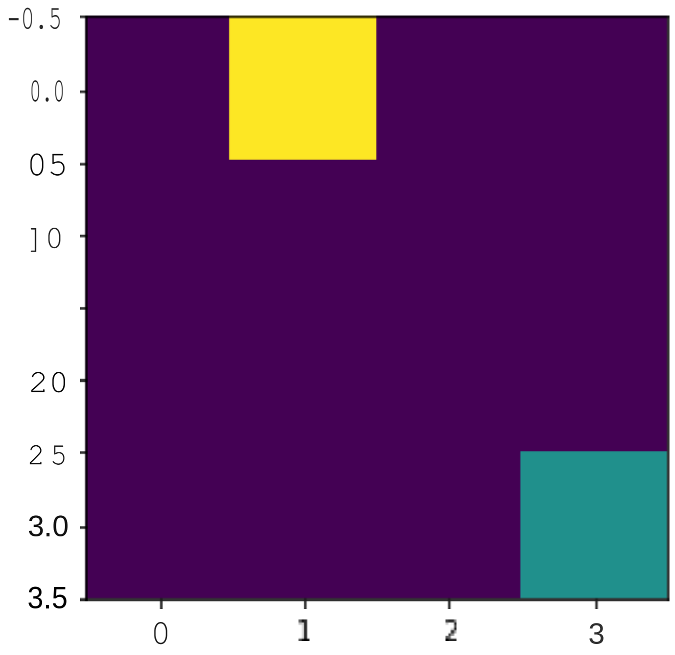
]0

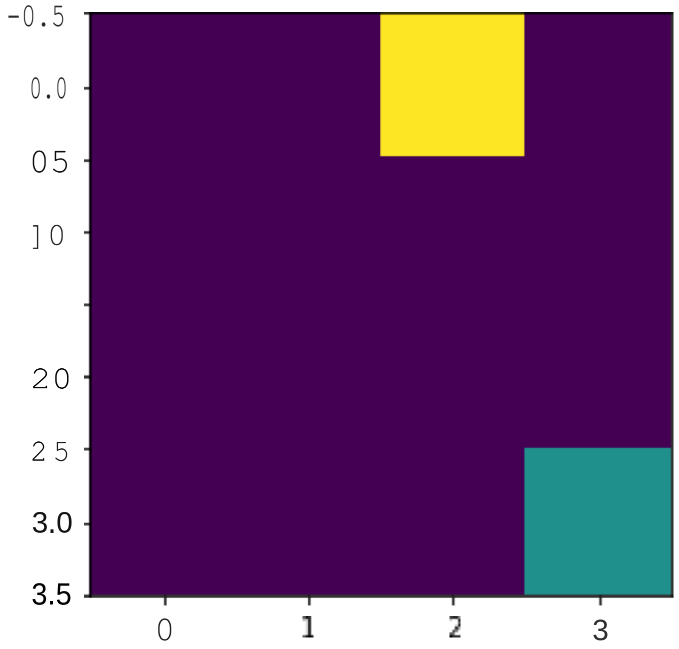


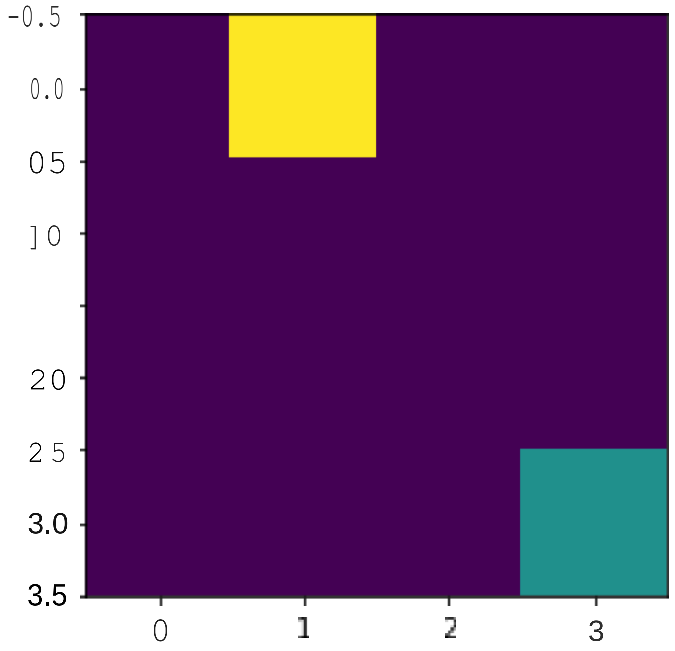
TimeStep	Action	New Action
1	2	2
2	0	0
3	3	3
4	0	0
5	2	2
6	1	1
7	0	0
8	1	1
9	2	2
10	2	2

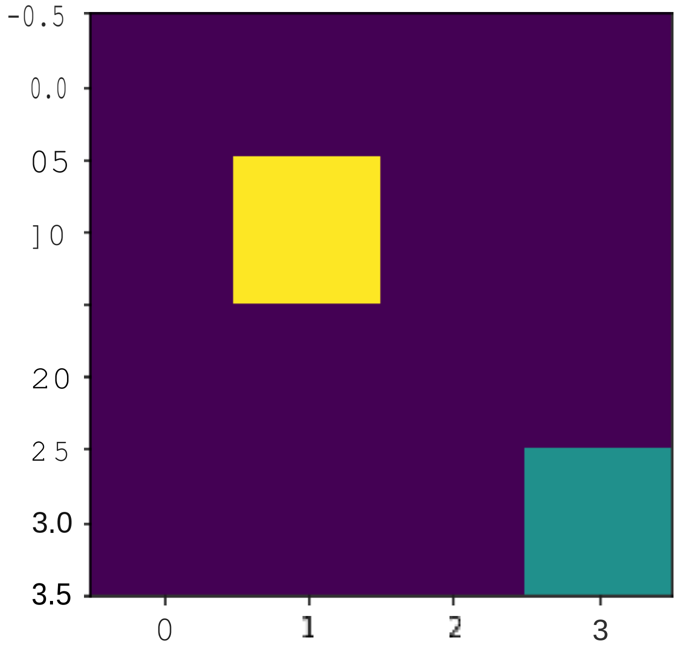


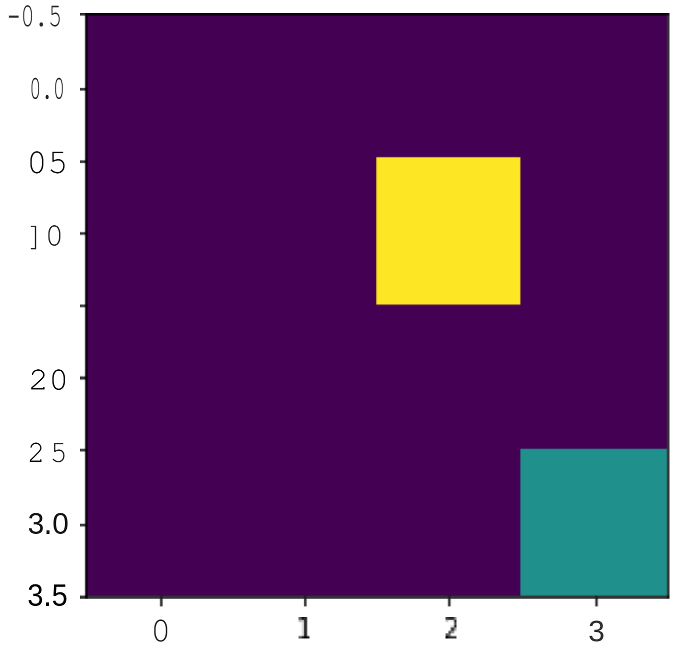

-0.5
0.0
05
]0
20
25
3.0
3.5
0
1
2
3

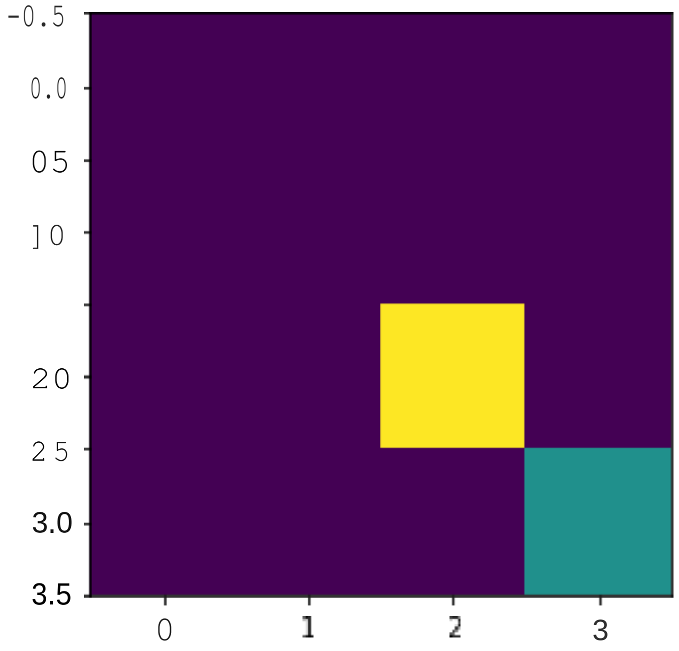

-0.5
0.0
05
]0
20
25
3.0
3.5
0
1
2
3

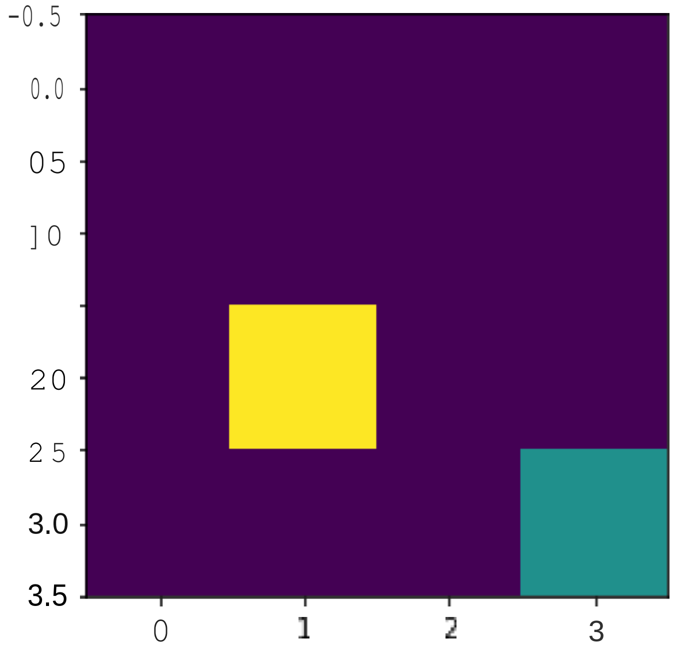

-0.5
0.0
05
10
20
25
3.0
3.5
0
1
2
3

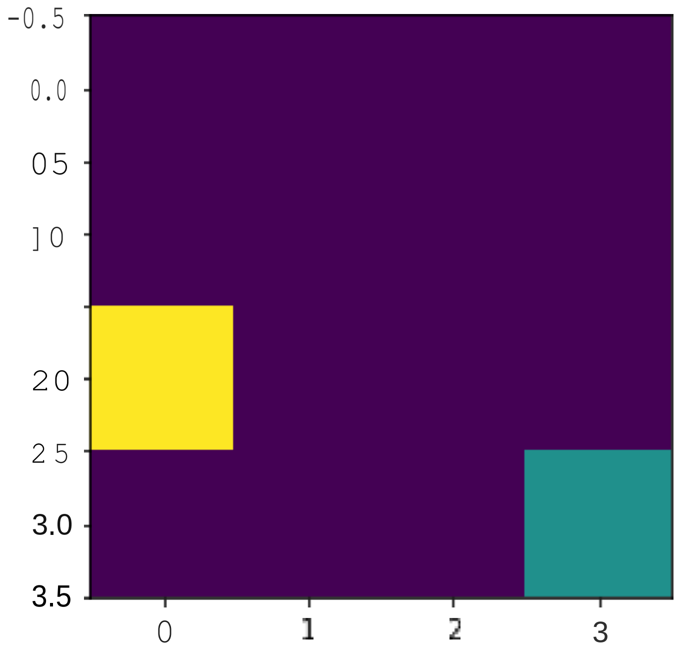

-0.5
0.0
05
10
20
25
3.0
3.5
0
1
2
3


-0.5
0.0
05
]0
20
25
3.0
3.5
0
1
2
3


-0.5
0.0
05
]0
20
25
3.0
3.5
0
1
2
3

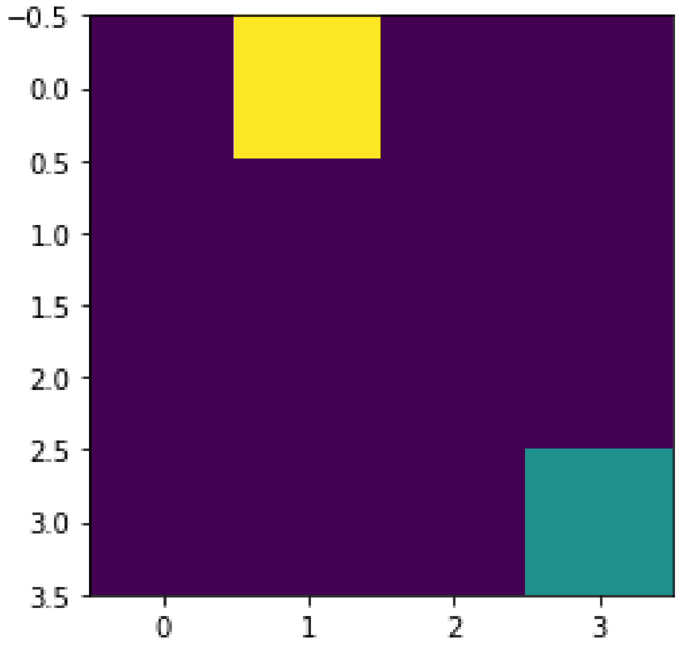

-0.5
0.0
05
10
20
25
3.0
3.5
0
1
2
3

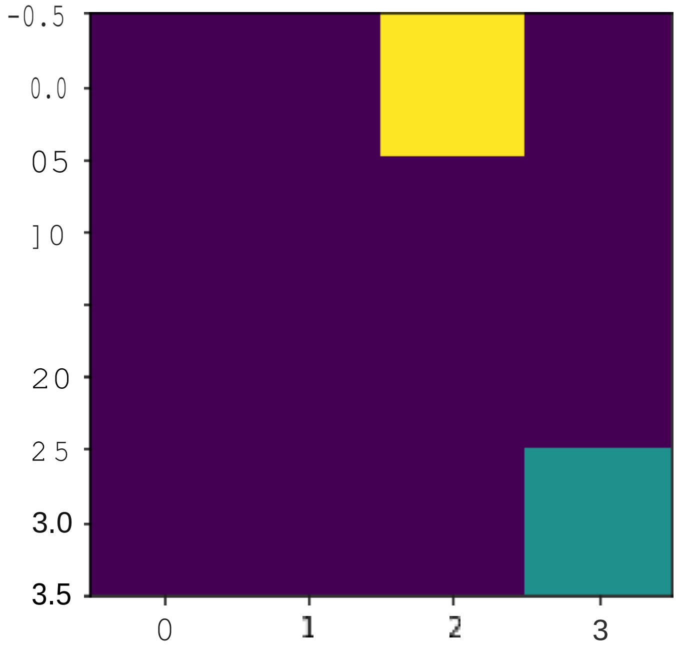

-0.5
0.0
05
10
20
25
3.0
3.5
0
1
2
3

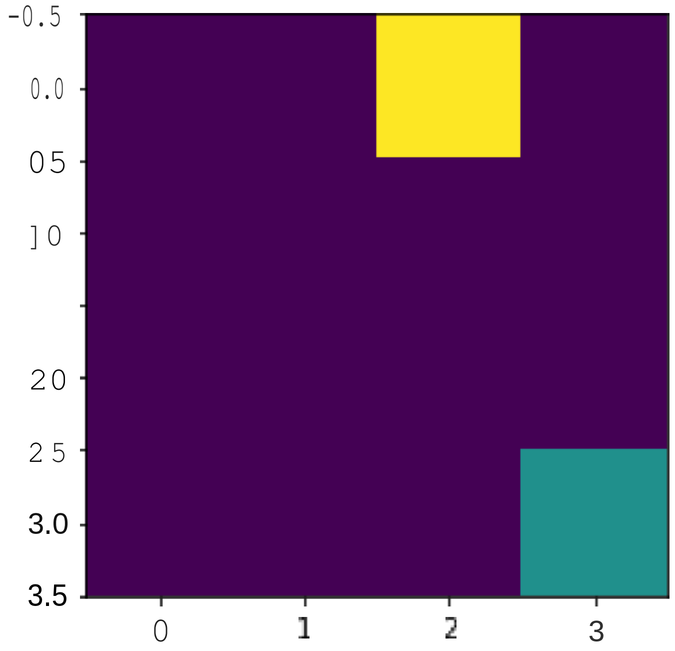

-0.5
0.0
05
10
20
25
3.0
3.5
0
1
2
3

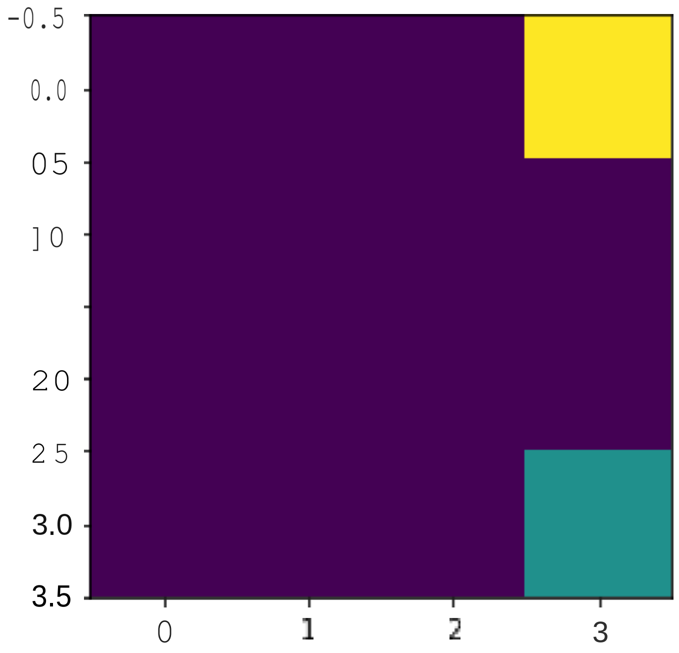
Visualisation (Stochastic Environment):

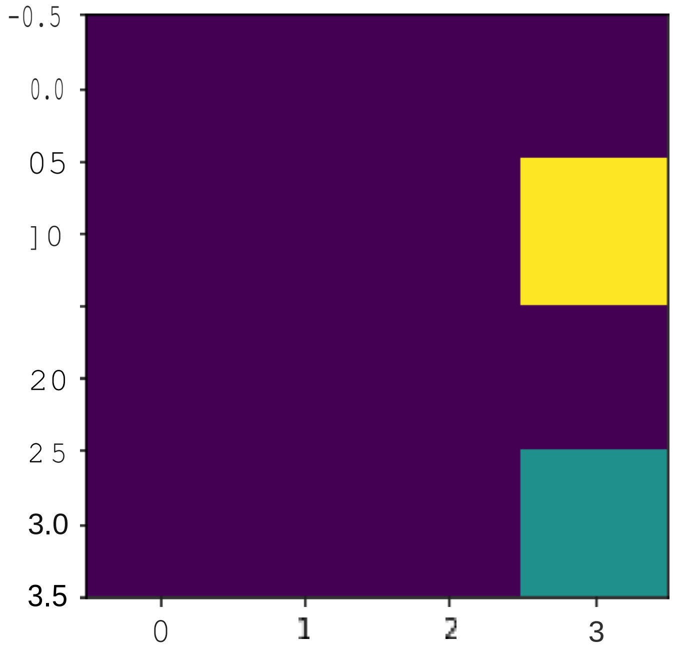
TimeStep	Action	New Action
1	0	0
2	0	0
3	3	3
4	0	0
5	1	1
6	2	0
7	3	0
8	3	0
9	1	1
10	3	3

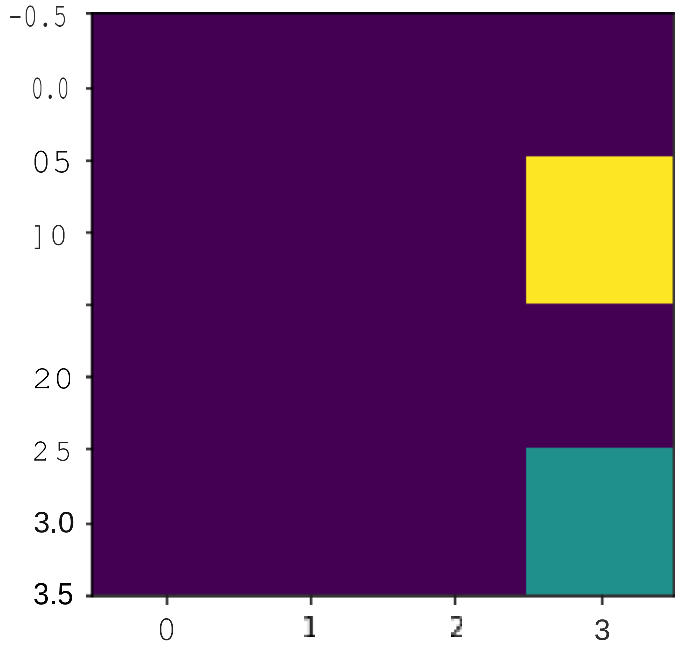


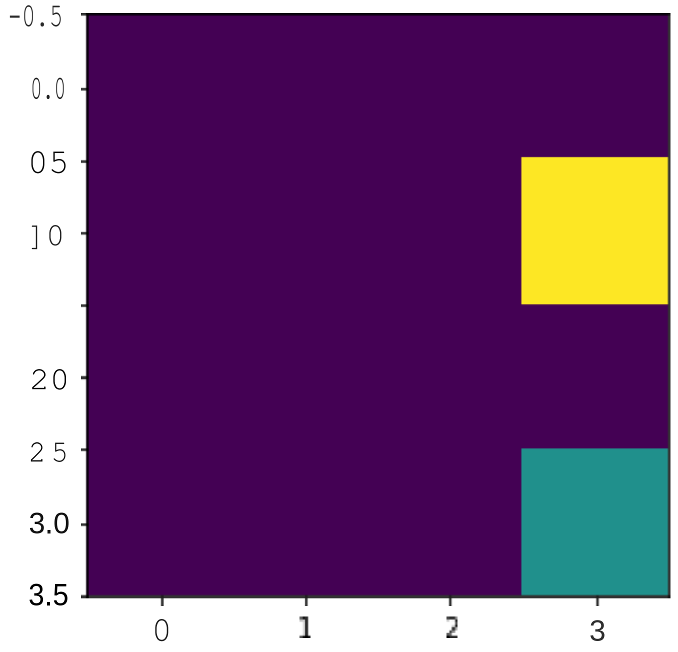

-0.5
0.0
05
]0
20
25
3.0
3.5
0
1
2
3

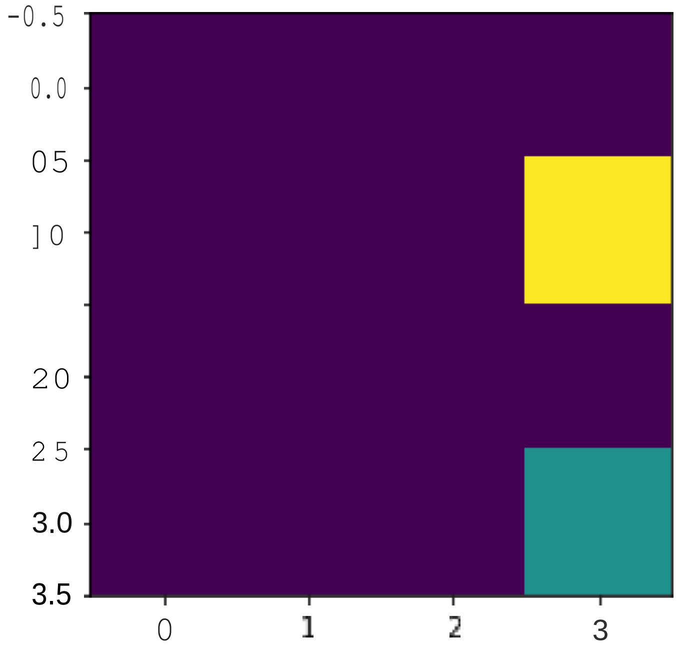

-0.5
0.0
05
]0
20
25
3.0
3.5
0
1
2
3

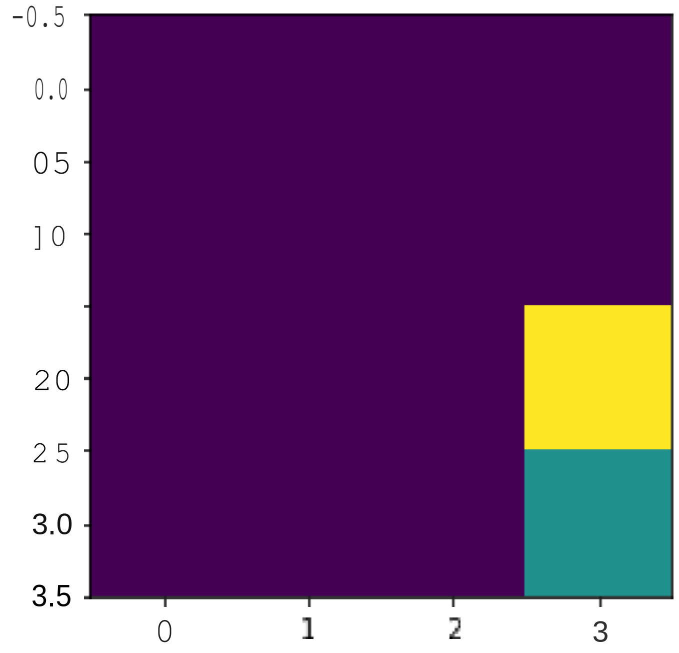

-0.5
0.0
05
10
20
25
3.0
3.5
0
1
2
3

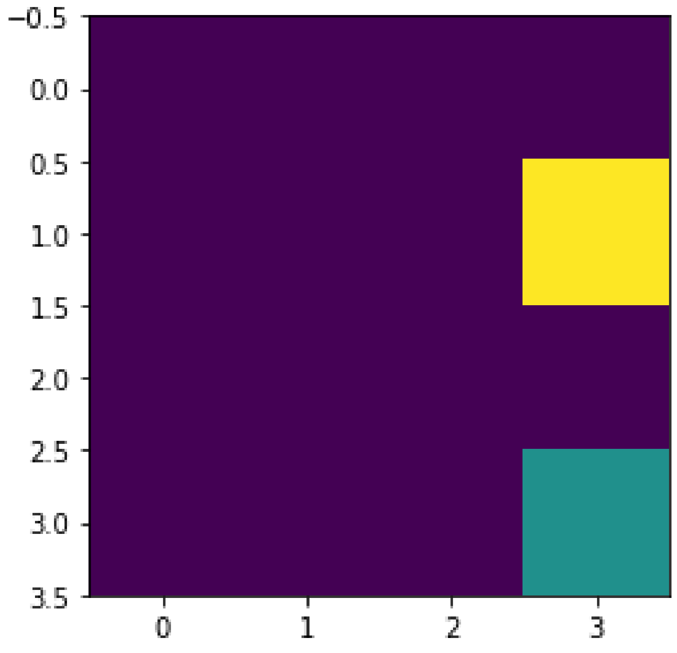

-0.5
0.0
05
10
20
25
3.0
3.5
0
1
2
3


-0.5
0.0
05
]0
20
25
3.0
3.5
0
1
2
3

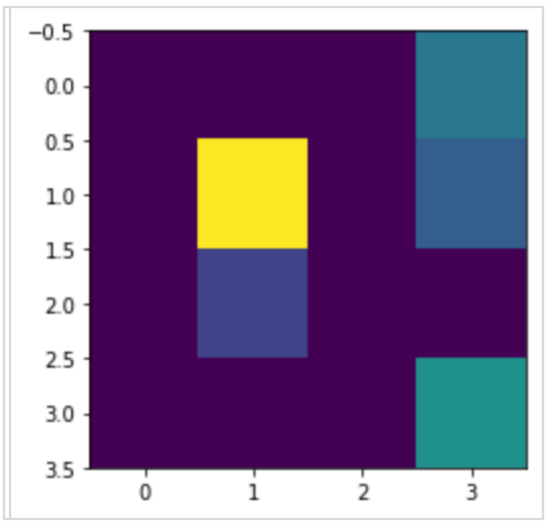

-0.5
0.0
05
]0
20
25
3.0
3.5
0
1
2
3


-0.5
0.0
05
10
20
25
3.0
3.5
0
1
2
3


-0.5
0.0
05
10
20
25
3.0
3.5
0
1
2
3



For visualizing the environment with rewards and agent positions.



Agent is yellow.
As you can see, there are 4 rewards here.
At (0,3) I have put a candy with reward of +3,
At (1,3) I have put a diamond with reward of +5,
At (2,1) I have put a coin with reward of +1,
At (3,3) I have put a gold bar with reward of +10,

3. How did you define the stochastic environment?

   Ans:
   - action 'Up': There's a 90% chance that the agent takes the action 'Up' and a 10% chance that the agent takes any random action.
   - action 'Down': There's a 90% chance that the agent takes the action 'Down' and a 10% chance that the agent takes any random action.
   - action 'Right': There's a 90% chance that the agent takes the action 'Right' and a 10% chance that the agent takes any random action.
   - action 'Left': There's a 90% chance that the agent takes the action 'Left' and a 10% chance that the agent takes any random action.

4. What is the difference between the deterministic and stochastic environments?

   Ans: In a deterministic environment if your agent selects an action it will follow that action with 100% probability whereas in a stochastic environment if the agent selects an action it can go to more than one possible next state, that is, the probability of the agent taking the action is <100%.

5. Write a brief review explaining how you ensure the safety of your environments.

   Ans: We can take some of the few measures as follow:
   - Agent shouldn't go out of bounds(environment). Achieved it using np.clip() .
   - If the reward for each step is 0 then the agent might not learn the shortest path. That is why I am penalizing the agent with a -1 reward for every step it takes.
   - To ensure the agent doesn't move without any goal/restrictions I set the maximum time steps for which the agent will run and after reaching the maximum time steps or goal the agent should stop.
   - The environment has a start position and end goal position, so that every time episode is over resets or reaches the goal, it should come back to the start position again.
   - For the stochastic environment, I introduced 10% stochasticity. This is ensured by using a random value between [0,1] and checking if it is greater than 0.9. The amount of stochasticity is important in replicating stochasticity from real world problems.

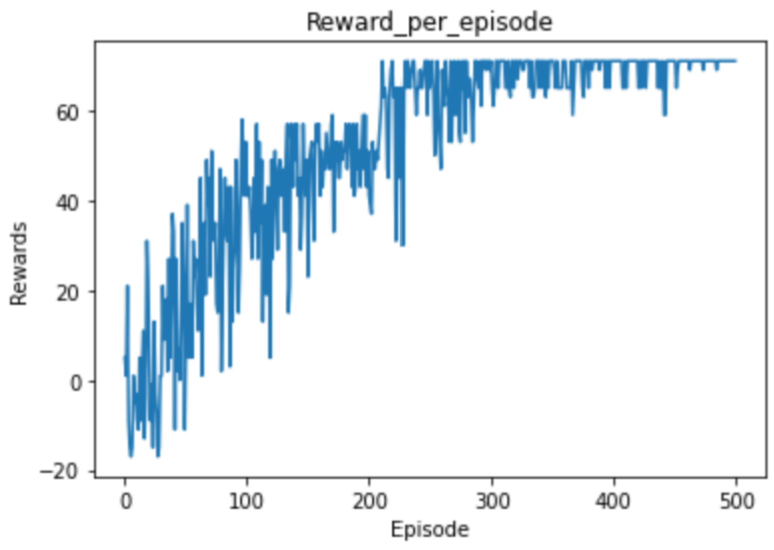
Part 2 [Total: 50 points] - Applying tabular methods:

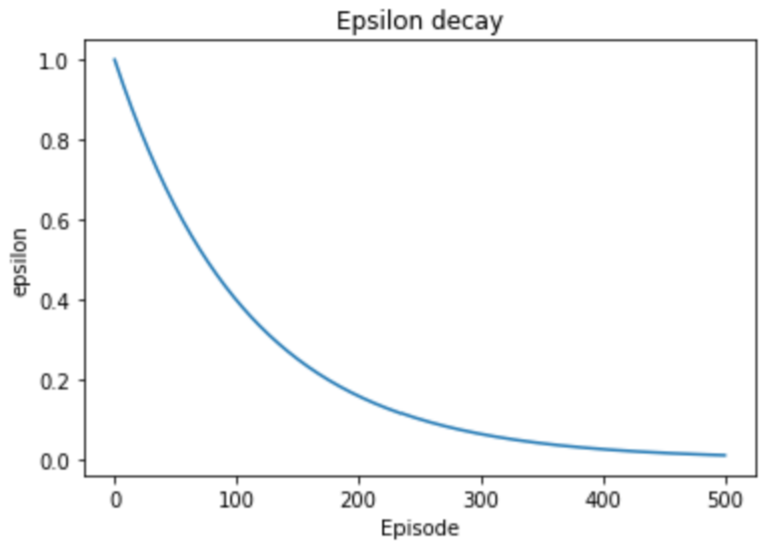
- Applying two tabular methods to solve both the deterministic and stochastic environments that were defined in Part 1.
- I've chosen Q learning and SARSA for my implementation.

Results:

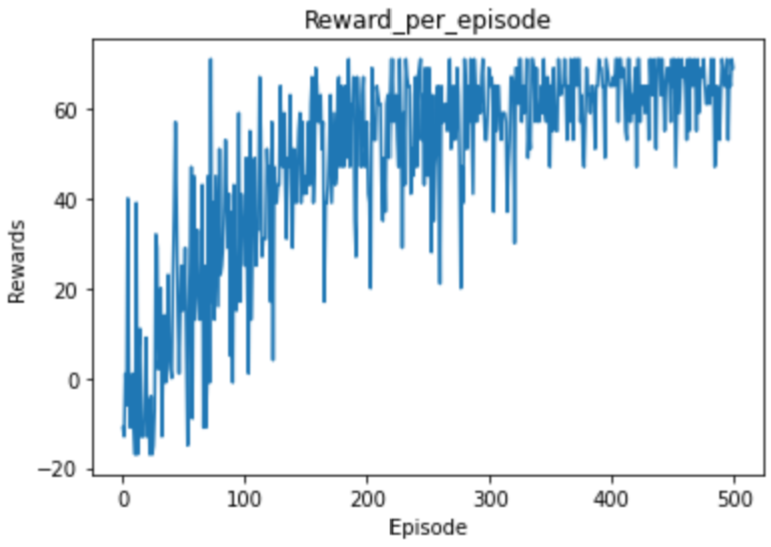
Q1:

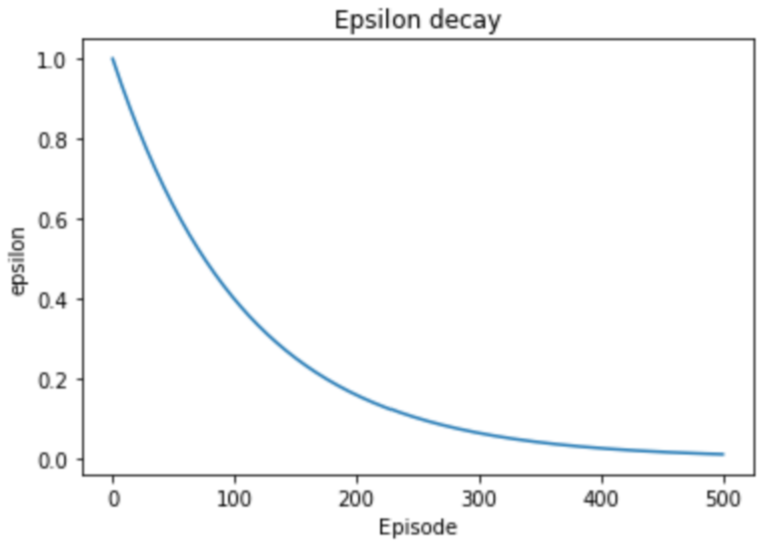
- Applying Q-learning to solve the deterministic environment defined in Part 1.:



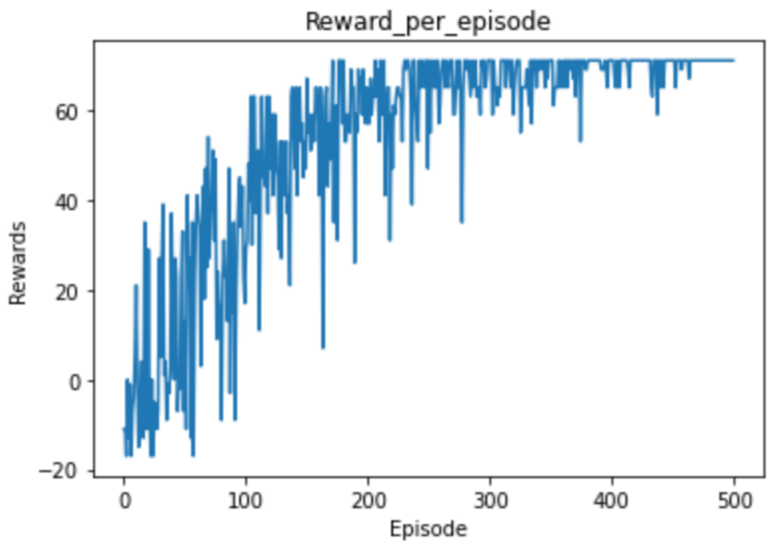


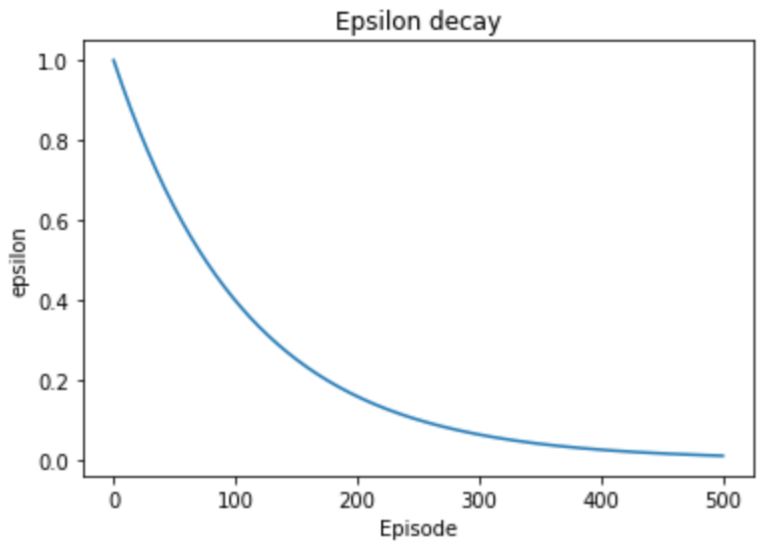
- Applying Q-learning to solve the stochastic environment defined in Part 1.:



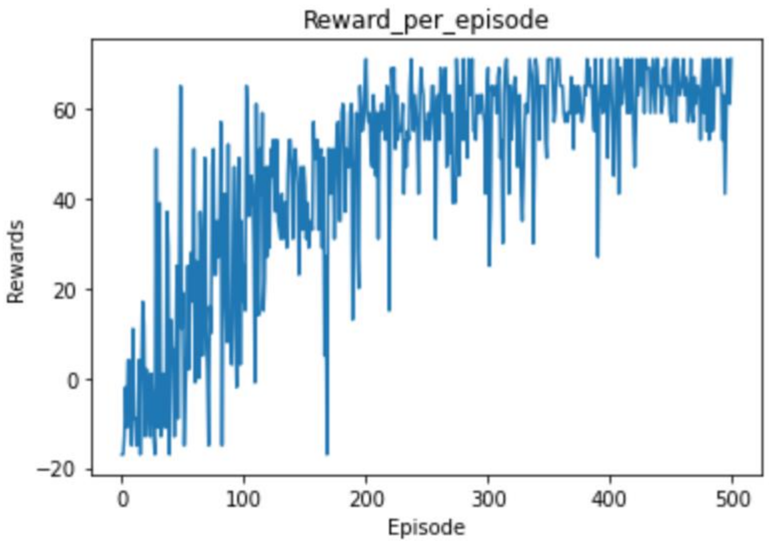


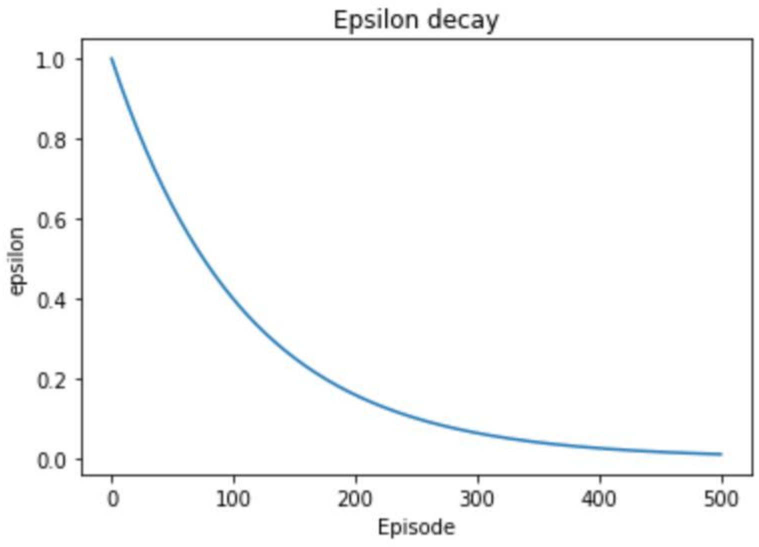
- Applying SARSA to solve the deterministic environment defined in Part 1:



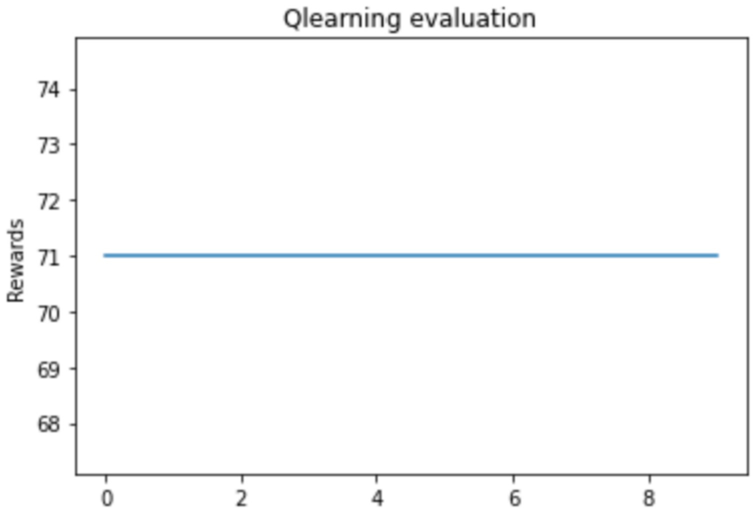


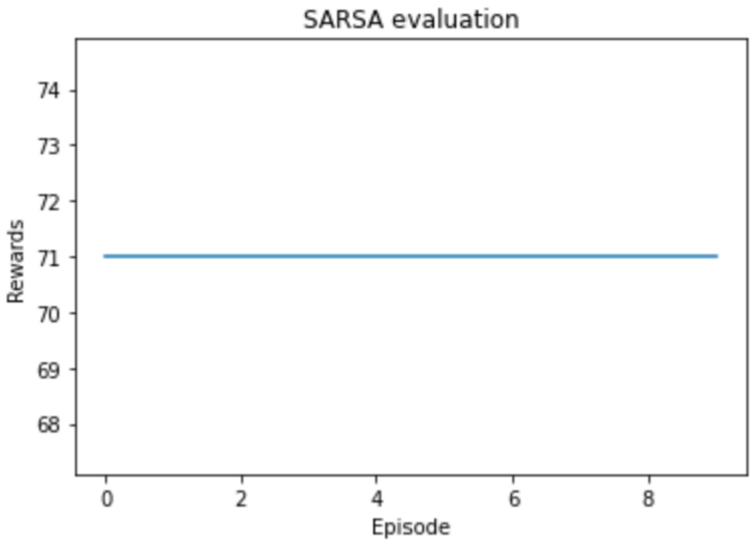
- Applying SARSA to solve the stochastic environment defined in Part 1.





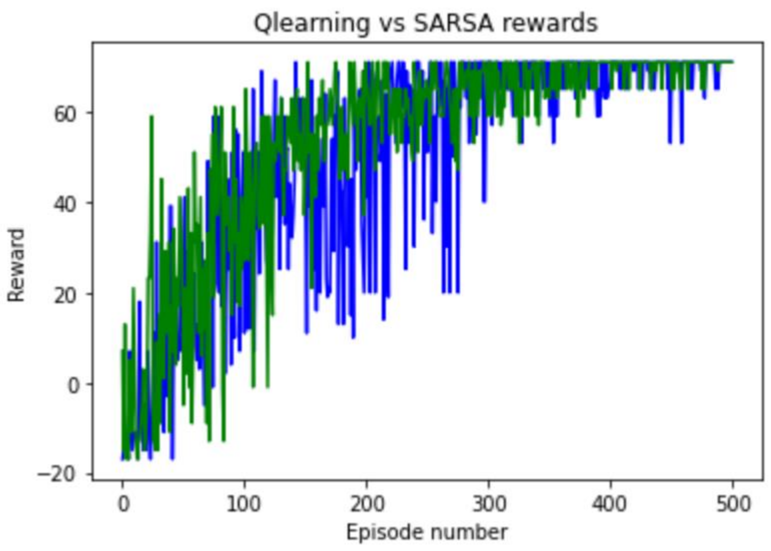
- The evaluation results. Running environment for 10 episodes, where the agent chooses only greedy actions from the learnt policy.





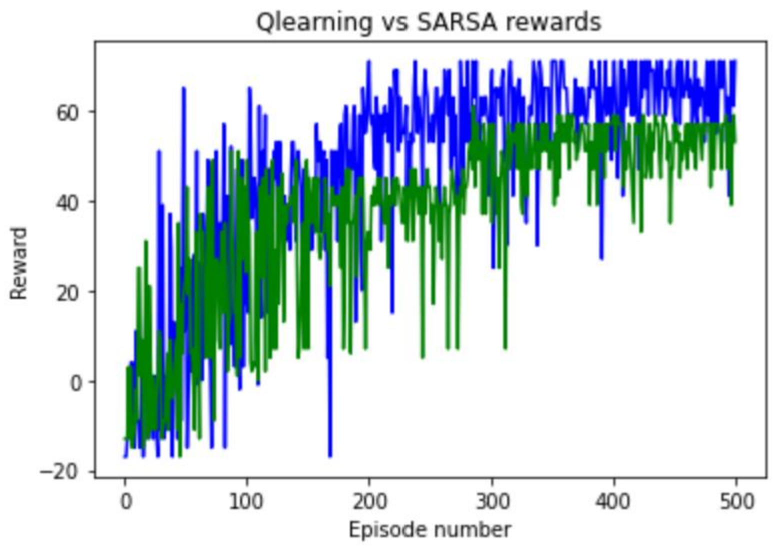
Q2:

Comparing the performance of both algorithms on the same deterministic environment (by plotting rewards per episode):



- We can see both methods achieve the goal and maximise the rewards per episode. Due to the environment being deterministic we don't see many fluctuations in the rewards per episode.
- After training we see that the reward per episode is almost the same. Therefore, both the algorithms are working well and are converging almost at the same point.

Q3:
Comparing the performance of both algorithms on the same stochastic environment (by plotting rewards per episode):



- Performance of QLearning and SARSA in the stochastic environment is similar to that of the deterministic but the reward fluctuations/variance is more in a stochastic environment. We can also see that the rewards per episode is increasing for both.
- The rewards per episode at end of training is almost the same for both the algorithms, both the algorithm achieve the goal and are trying to maximise the reward per episode by following optimal policy.

Q4:

<u>Q Learning:</u>

- Here the agent tries to maximize the rewards as the training progresses and tries to obtain the optimal policy. It achieves maximum reward by taking minimum steps in an episode.

- The Q table updating formula:

$$Q(S,A) \leftarrow Q(S,A) + \alpha\Big[\underbrace{\overbrace{\underset{\uparrow\ \text{Immediate Reward}}{R} + \gamma \max_a Q(S',a)}^{\text{Target}} - \overbrace{Q(S,A)}^{\text{Prediction}}}_{\text{loss}}\Big]$$

The current state's q value is updated by considering the reward for the action taken and the max q value of the next state is selected. The action is chosen following a greedy policy.

- Q values for terminal state is 0

```
[[ 3.69800000e+01  4.83395847e+00  1.75440779e+01  1.94375966e+01]
 [ 4.22000000e+01  1.71331618e+01  1.66819216e+01  2.51750845e+01]
 [ 4.80000000e+01  3.44023970e+01  2.75808573e+01  2.97825281e+01]
 [ 4.24405747e+01  5.00000000e+01  3.84110079e+01  4.31656999e+01]
 [ 1.75357522e+01 -2.54073253e-01  3.84596750e-01  2.74614998e-01]
 [ 3.70354520e+01  1.13605221e+00  9.09421575e-01  6.79489582e+00]
 [ 4.99935227e+01  4.21221731e+00  1.20784523e+01  1.68922393e+01]
 [ 5.00000000e+01  4.31642827e+01  4.39168643e+01  4.79626617e+01]
 [ 1.25886725e+00 -6.88438531e-01 -3.03850373e-01 -7.39018574e-02]
 [ 7.52598836e-02 -3.69891190e-01 -2.98051372e-01  4.59235842e+00]
 [ 2.65637912e+00  3.59235890e-02  3.31167914e-01  2.14239187e+01]
 [ 1.06551955e+01  6.12579511e+00  4.28807173e+00  4.97758336e+01]
 [-3.26800000e-01 -4.77223668e-01 -3.94039900e-01 -3.39116650e-01]
 [-1.00000000e-01  0.00000000e+00 -4.51630900e-01  4.48243907e-01]
 [ 3.43900000e+00 -1.90000000e-01 -2.69675701e-01 -1.12293082e-02]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]]
```

Deterministic environment

```
[[36.98       21.82801699 27.62818059 29.19149676]
 [42.2        35.07621762 30.4293469  27.55819083]
 [48.         34.57142366 33.37831671 40.50367917]
 [44.61111072 50.         41.07755722 47.09407444]
 [34.23558131 -0.53615265  1.22170963  2.20673333]
 [43.89286499  7.46721623  3.08667927 22.52723381]
 [49.99999997 24.20429158 31.10479034 37.00951768]
 [50.         43.99993977 43.99999923 47.99999942]
 [ 1.45774411 -0.5870547  -0.36363573 -0.51531307]
 [-0.13165502 -0.32928172 -0.3510133  21.82591949]
 [12.85270929  1.17034542  1.54438793 40.52854461]
 [17.79818773  9.74968445 22.15016598 49.99998884]
 [-0.27146203 -0.2881     -0.36590923 -0.25873827]
 [-0.36057951 -0.13863742 -0.45240931  3.42925458]
 [ 4.68559    -0.1        -0.45873134  0.21655523]
 [ 0.          0.          0.          0.        ]]
```

Stochastic environment

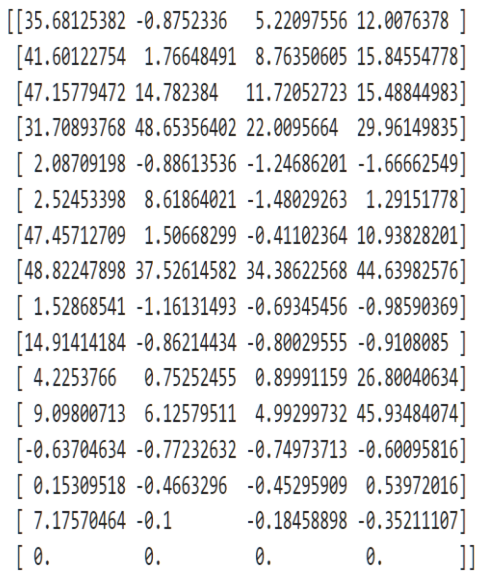
<u>SARSA:</u>

- SARSA works similar to Q learning. The agent tries to maximize the rewards as the training progresses and tries to obtain the optimal policy, the only difference is it considers the the next action. It doesn't take the max q value of the state
- The Q table updating formula:

$$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma Q(S',A') - Q(S,A)\big]$$

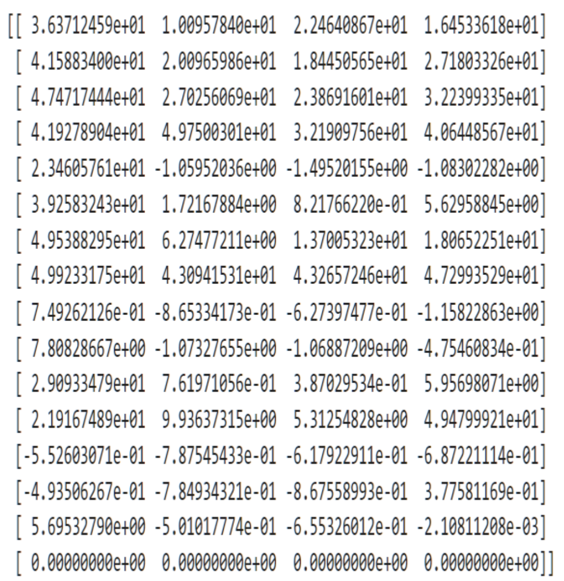
The current state q value is updated by the immediate reward and the next state's qvalue. The action is chosen by the epsilon greedy policy

- Q values for terminal state is 0



```
[[35.68125382 -0.8752336   5.22097556 12.0076378 ]
 [41.60122754  1.76648491  8.76350605 15.84554778]
 [47.15779472 14.782384   11.72052723 15.48844983]
 [31.70893768 48.65356402 22.0095664  29.96149835]
 [ 2.08709198 -0.88613536 -1.24686201 -1.66662549]
 [ 2.52453398  8.61864021 -1.48029263  1.29151778]
 [47.45712709  1.50668299 -0.41102364 10.93828201]
 [48.82247898 37.52614582 34.38622568 44.63982576]
 [ 1.52868541 -1.16131493 -0.69345456 -0.98590369]
 [14.91414184 -0.86214434 -0.80029555 -0.9108085 ]
 [ 4.2253766   0.75252455  0.89991159 26.80040634]
 [ 9.09800713  6.12579511  4.99299732 45.93484074]
 [-0.63704634 -0.77232632 -0.74973713 -0.60095816]
 [ 0.15309518 -0.4663296  -0.45295909  0.53972016]
 [ 7.17570464 -0.1        -0.18458898 -0.35211107]
 [ 0.          0.          0.          0.        ]]
```

Deterministic environment



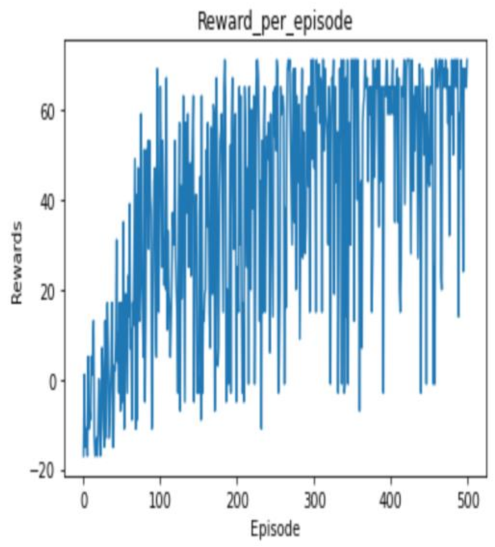
```
[[ 3.63712459e+01  1.00957840e+01  2.24640867e+01  1.64533618e+01]
 [ 4.15883400e+01  2.00965986e+01  1.84450565e+01  2.71803326e+01]
 [ 4.74717444e+01  2.70256069e+01  2.38691601e+01  3.22399335e+01]
 [ 4.19278904e+01  4.97500301e+01  3.21909756e+01  4.06448567e+01]
 [ 2.34605761e+01 -1.05952036e+00 -1.49520155e+00 -1.08302282e+00]
 [ 3.92583243e+01  1.72167884e+00  8.21766220e-01  5.62958845e+00]
 [ 4.95388295e+01  6.27477211e+00  1.37005323e+01  1.80652251e+01]
 [ 4.99233175e+01  4.30941531e+01  4.32657246e+01  4.72993529e+01]
 [ 7.49262126e-01 -8.65334173e-01 -6.27397477e-01 -1.15822863e+00]
 [ 7.80828667e+00 -1.07327655e+00 -1.06887209e+00 -4.75460834e-01]
 [ 2.90933479e+01  7.61971056e-01  3.87029534e-01  5.95698071e+00]
 [ 2.19167489e+01  9.93637315e+00  5.31254828e+00  4.94799921e+01]
 [-5.52603071e-01 -7.87545433e-01 -6.17922911e-01 -6.87221114e-01]
 [-4.93506267e-01 -7.84934321e-01 -8.67558993e-01  3.77581169e-01]
 [ 5.69532790e+00 -5.01017774e-01 -6.55326012e-01 -2.10811208e-03]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00]]
```

Stochastic environment

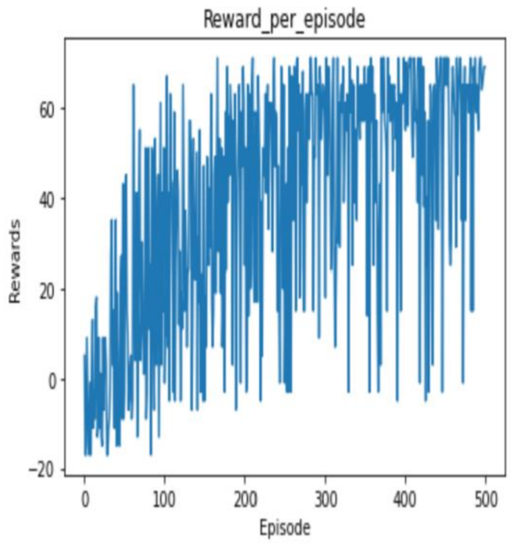
Extra Points:

Hyperparameter Tuning:

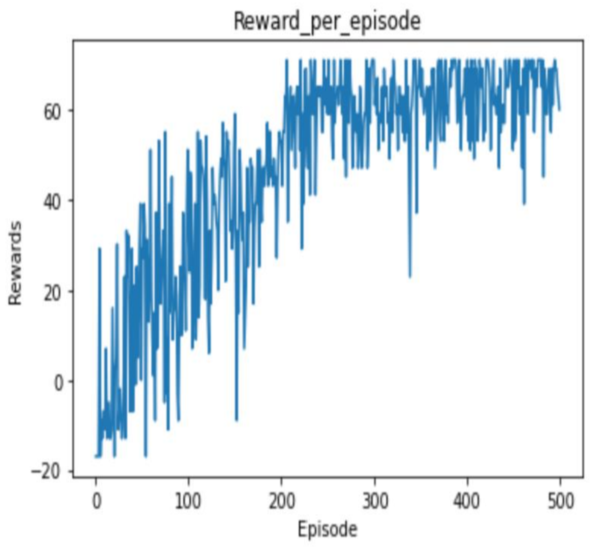
Discount factor (γ):



Gamma=0.1



Gamma=0.25



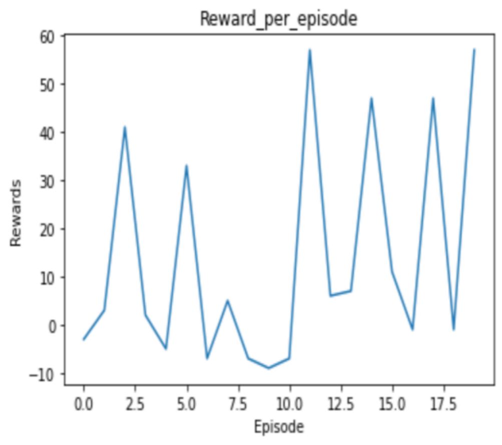
Gamma=0.9

- We can see as the gamma values increases the agent maximizes the reward as the future rewards aren't discounted that much.

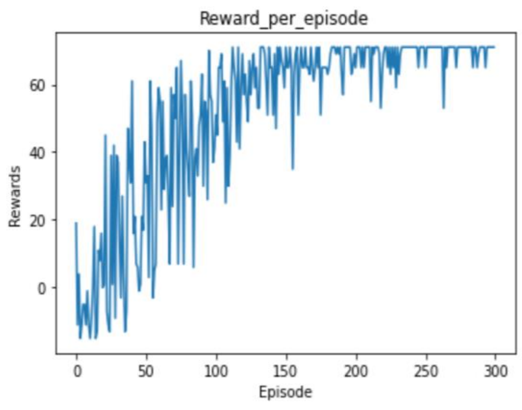
- The best Gamma value will be to keep it as high as possible around 0.9

Number of episodes:

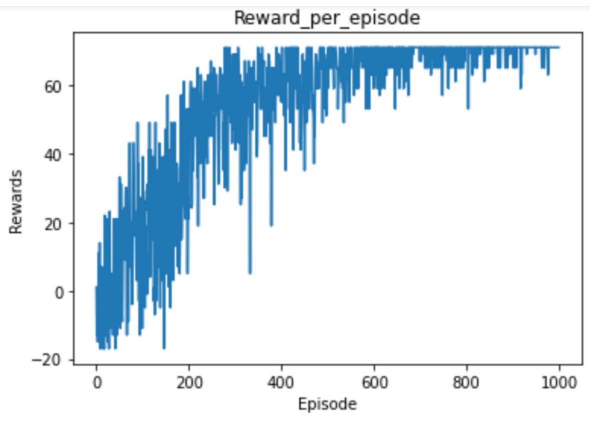
- Q learning on deterministic environment:



Episodes=20



Episodes=300



Episodes=1000

- We can see as the no of episodes increases the agent converges to the maximize the reward per episode. If the no of episodes is low the agent has less experience and isn’t trained to the fully.
- Higher the number of episodes, more accurate the results will be. After a certain number we’ll see diminishing returns and the learning will have converged to a solution. To optimize, we can check the similarity with the previous Q table.