QUESTION: *Observe what you see with the agent's behavior as it takes random actions. Does the* ***smartcab*** *eventually make it to the destination? Are there any other interesting observations to note?*

As the smartcab takes random actions, it infinitely moves on the finite map and thus sometimes make it to the destination.

QUESTION: *What states have you identified that are appropriate for modeling the* ***smartcab*** *and environment? Why do you believe each of these states to be appropriate for this problem?*

States depend on different parameters :

- next_waypoint : The waypoint the planner suggests to go to on the next step. (4 possibilities)

- light: traffic light's color (green or red for the agent) (2 possibilities)

- oncoming: direction taken by the eventual oncoming car (4 possibilities)

- left: direction taken by the eventual oncoming car on the left of the agent(4 possibilities)

- right: direction taken by the eventual oncoming car on the right of the agent (4 possibilities)

- time left to go to final direction (the number of possibilities is the initial time left)

next_waypoint is appropriate : it is important to be helped by the planner. The goal of the agent is to reach the final point. Without the planner, the agent would have no idea of where the final point is and it would go across the map until it luckily finds it. The planner gives the agent a way (not ideal) to go to its final destination. The agent should generally try to follow it (because it has not other hints about the way to take), but watch out for accidents or law infringement.

Traffic light's color is appropriate : it is important to minimize the number of negative rewards for not respecting the law.

Directions of eventual oncoming, left and right cars are appropriate: it is important to minimize the number of negative rewards for having an accident.

OPTIONAL: *How many states in total exist for the* ***smartcab*** *in this environment? Does this number seem reasonable given that the goal of Q-Learning is to learn and make informed decisions about each state? Why or why not?*

The total number of states is 4 * 2 * 4 * 4 * 4 * (initial time left). As the initial time left can be a great number, the total number of states is much too high. Indeed, to implement a Q-learning driving agent, our agent has to visit every possible state a great number of times, and learning would take a great number of time in this case.

Additionally, the time left may not be an interesting parameter to take into account at each step. It may be interesting if there was some trade-off between risk to take and time left. But here, time left should not have any influence on the driver's behavior at each step. The reward for arriving before time is over is enough to motivate the agent to play on its parameters, considering all possible states determined by : next_waypoint, light, oncoming, left, right (512 states).

QUESTION: *What changes do you notice in the agent's behavior when compared to the basic driving agent when random actions were always taken? Why is this behavior occurring?*

At the beginning of the training, the agent rarely get to the final point before time has run out. However, after a few iterations, the agent starts to arrive to the final destination in times and a few iterations more and it never misses its goal.

It means it is learning from its experiences and that our Q-learning algorithm works well. Indeed, each time our agent gets a positive or negative reward, it updates in a matrix the value for arriving in the related state, leaving via the related action, by incorporating this reward. Later, when in the same state, it will thus remember how well it went by taking this action and thus make a better decision. After visiting enough states enough times, it thus starts to have enough experience to sometimes reach the final destination in time, giving it some even bigger reward, helping it to make even better decisions and always reach the final point before time is up.

QUESTION: *Report the different values for the parameters tuned in your basic implementation of Q-Learning. For which set of parameters does the agent perform best? How well does the final driving agent perform?*

Alpha	Gamma	Epsilon	Performance
0.1	0.1	0.01	0.97
0.4	0.1	0.01	0.98
0.7	0.1	0.01	0.99
0.1	0.4	0.01	0.100
0.4	0.4	0.01	0.95
0.7	0.4	0.01	0.94
0.1	0.7	0.01	0.96
0.4	0.7	0.01	0.97
0.7	0.7	0.01	0.94
0.1	0.1	0.05	0.99
0.4	0.1	0.05	0.97
0.7	0.1	0.05	0.94
0.1	0.4	0.05	0.98
0.4	0.4	0.05	0.89
0.7	0.4	0.05	0.97
0.1	0.7	0.05	0.89
0.4	0.7	0.05	0.91
0.7	0.7	0.05	0.87
0.1	0.1	0.1	0.99

0.4	0.1	0.1	0.99
0.7	0.1	0.1	0.97
0.1	0.4	0.1	0.99
0.4	0.4	0.1	0.89
0.7	0.4	0.1	0.96
0.1	0.7	0.1	0.98
0.4	0.7	0.1	0.94
0.7	0.7	0.1	0.88

Our best perfomance is obtained with the parameters :

alpha = 0.1 ; gamma = 0.4 ; epsilon = 0.01

This performance is of 100% of success.

QUESTION: *Does your agent get close to finding an optimal policy, i.e. reach the destination in the minimum possible time, and not incur any penalties? How would you describe an optimal policy for this problem?*

The ideal policy would be : always get to the final destination in the shortest time possible, without any accident nor illegal action.

-The ideal policy doesn't always follow the direction suggested by the planner. Indeed, the planner doesn't take into account the traffic lights. There are situation where you can gain time if you don't follow the planner (ex: turn right, turn left and go forward instead of waiting, go forward, turn right and turn left).

-The learned policy doesn't always follow the suggested direction neither (often has a 0 reward where it could have gotten a 2 reward), I suppose for the same reasons as for the ideal policy.

-The ideal policy is only concerned about arriving in time (without any accident or law infringement). Having behaviors (like going in circles) to accumulate rewards is out of the question.

-The learned policy, for high numbers trials (having learned a lot), does not having any behaviors like this anymore, compared to first trials.

-The ideal policy absolutely never violates any traffic rules or have any accident.

-The learned policy (for high number trials) rarely has negative rewards for law infringement and very rarely has negative rewards for accidents. Our learned policy is thus pretty good at respecting traffic rules.

To conclude, our agent is really close to an optimal policy. It is impressive to see how fast it is learning.