Name: David Chan Chi Fung
Project: Udacity Training a Smartcab To Drive

In your report, mention what you see in the agent's behavior. Does it eventually make it to the target location?

The agent simply walks around very randomly, without eventually making it to the destination. I believe there is a probability it might make it to the destination, but this is will happen very rarely and occasionally.

Justify why you picked these set of states, and how they model the agent and its environment.

I picked the states **light, left, right, oncoming** and **next_waypoint.**
I picked up the **light, left** and **right** states so that the agent can properly model the traffic in its environment.
I also added **oncoming** because it can model whether cars are coming from in front of the agent.
And finally, I picked the **next_waypoint** variable so that the agent can get information from the planner on the next direction it should go.
I chose not to include other parameters like **deadline,** or **location** simply because they will make the state space too large, thereby forcing the agent to take too long to explore all of the state space.

What changes do you notice in the agent's behavior?

Immediately after implementing Q-learning, the agent stops behaving randomly, and it starts to find its way to the destinations.
The reason why the agent's behaviour has changed is because it uses the Q-learning algorithm, given the states of the world it's in, to choose an action which will maximize its reward and enable it reach its destination faster.
Finally, after 1 or 2 trials, the agent is able to quickly find it's way to destination, while obeying traffic rules. It's actions are not random anymore, and it has learned the best policy.

Report what changes you made to your basic implementation of Q-Learning to achieve the final version of the agent. How well does it perform?

I run it with the *alpha, gamma,* and the *q_init* variables. The *q_init* variable is used to initialise the *Q* matrix.
I obtained the values displayed in the following table:

	Rewards	**Time_to_destination**	**Reached_destination**
Alpha:0.8 Gamma:0.4 q_init:2.5	330.0	136	10
Alpha:1.0 Gamma:0.4 q_init:2.0	315.5	147	10
Alpha:1.0 Gamma:0.4 q_init:3.0	325.0	167	10
Alpha:1.0 Gamma:0.2 q_init:3.5	276.0	124	10
Alpha:0.8 Gamma:0.4 q_init:3.0	315.5	151	10
Alpha:0.8 Gamma:0.2 q_init:3.5	264.0	107	10
Alpha:0.8 Gamma:0.2 q_init:4.0	338.5	160	10
Alpha:0.6 Gamma:0.6 q_init:4.0	331.0	172	10
Alpha:0.6 Gamma:0.4 q_init:2.0	286.5	132	10

These are all parameter combinations which reached their destination 10 times. I then chose the final parameter for my agent using a greedy approach: the agent which maximised it's Rewards. This is the agent with parameters **Alpha = 0.8, Gamma = 0.4 and Init_1 = 2.0.** I chose to ignore the time_to_destination, due to my greedy nature :)
After running the trial many times, the agent finally reached the optimal policy. The results of different runs are shown in the table above, and it's clear that the optimal policy is the first one.

The ideal or optimal policy is simply one where the agent follows the next waypoint guides from the navigation, while at the same time avoiding accidents and respecting traffic rules.