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?*

ANSWER: I do see the agent occasionally reach the destination. However, most of time the agent run out of time and trial aborted. The agent's success of reaching to the destination did not improve after 100 trials. There is more work to be done.

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

ANSWER: There are states for *light*, *oncoming traffic*, *left traffic*, and *next waypoint*. The *light* is considered because it determines whether it's appropriate to turn left or go forward. The *oncoming* traffic is considered because it determines whether it's appropriate to turn left. The *left* traffic is considered because it determines whether it's appropriate to turn right. The *next waypoint* is considered because it determines which direction the agent is supposed to go. In addition, *right* traffic and *deadline* are not considered. *Right* traffic simply doesn't affect any action our agent takes and deadline has too many states that might not be informative to add values to our Q-learning. Thus, by excluding them from our states, we will get much cleaner Q table.

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?*

ANSWER: There are 2 possible states for *light*, *oncoming traffic*, and *left traffic* and 3 possible values for *next waypoint,* so the total possible states are $2^3*3 = 24$ states. Yes, this number seems reasonable because it exploits all possible situations.

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?*

ANSWER: Now the agent reached destination very frequently and only occasionally will time out. This behavior is because the agent is learning to take action at specific situation 90% of time and taking random action 10% of time. I can probably tune alpha, gamma, epsilon and epsilon decay rate to make it reach destination even more frequent.

QUESTION: *Report the different values for the parameters turned 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?*

ANSWER:
Run through each set of alpha, gamma, epsilon, and decay rate, I found many sets have a very good result. However, I feel (0.2, 0.8, 0.5, 0.05) has the best performance in destination rate and average velocity with doing well in positive reward, negative reward, and average time. All the values are computed based on all 100 runs with random seed at 1. We might want to set *n_trials* to a larger number to see a more stable result.

Alpha	Gamma	Epsilon	Decay rate	Dest. Rate	Pos. Reward	Neg. Reward	Avg. Time	Avg. Velocity
0.5	0.5	0.1	0.001	0.94	21.11	-0.345	12.83	0.4295
0.5	0.8	0.1	0.001	0.95	22.035	-0.405	13.83	0.4295
0.5	0.8	0.2	0.004	0.96	22.35	-0.45	13.45	0.4401
0.5	0.8	0.2	0.005	**0.97**	22.21	-0.335	12.48	0.4495
0.5	0.8	0.2	0.02	0.96	21.815	-0.25	13.0	0.4500
0.5	0.8	0.5	0.05	**0.97**	21.895	-0.225	**12.07**	0.4532
0.5	0.8	0.5	0.1	0.4	11.455	**-0.18**	23.86	0.1563
0.5	0.8	0.5	0.025	0.95	**22.535**	-0.515	13.78	0.4216
0.2	0.8	0.5	0.05	**0.97**	22.135	-0.24	12.92	**0.4598**
0.8	0.8	0.5	0.05	**0.97**	21.985	-0.225	**12.07**	0.4532

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?*

ANSWER: In terns of performance matrices, my final policy has a very good average time to reach the destination of 12.92, a success rate of 0.97 reach destination, and a -0.24 average penalties. I think the optimal policy is the one with the high destination rate, high positive reward, low negative reward, low average time, and high average velocity. In addition, the optimal policy should make error free decisions as to all situation it's facing. Although the performance matrices look very good, the values in q table still have some improvement. I believe if we set *n_trials* to a much larger number, all the values in q table will be optimal to each state.