QUESTION 1: *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?*

The smartcab very rarely makes it to the destination. The cab just moves in a random manner until it resets or less likely, reaches the destination.

QUESTION 2: *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?*
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 states I have identified as being appropriate for the modeling of the smartcab and environment are the inputs: light, oncoming and left as well as the deadline and next waypoint. I decided not to include 'right' for this simulation because the simulator will not allow traffic to run red lights and otherwise disobey traffic rules. With this assumption, that Dummy Agents will not break traffic laws, the 'right' input provides no value when deciding what action to take next.

Moreover, to show that 'right' is of no value I will show how the other states do provide value when deciding what action to take next. First, all actions we can take are: none, forward, left and right.

If at a green light, going straight or turning right does not require any other input other than the traffic light because we have the 'right-of-way'. If at a green light and wanting to go left, we need to know if there is any oncoming traffic to safely take that action (to obey traffic laws). That covers all possibilities for green lights.

If at a red light, we can only make a right turn when the traffic is clear coming from the left direction. This covers all possible actions and what we need to know to drive the car. Therefore, what the 'right' traffic is doing does not matter to us. However, it's important to note this only applies to this smartcab simulation because in the real world cars can make illegal traffic moves. In the real world, 'right' traffic is very important; such that a car runs a red light. This is why it is always good practice to look both ways before proceeding on a green light.

Further, the deadline is important because the car may choose a different action if the deadline is approaching. For example, it may take a right on red when the next waypoint is to go straight. Given the deadline, it may find that turning right on red is faster.

Last, the next waypoint is important because it tells us the route we should take ignoring all traffic rules. In other words, it tells us the best way to get there if we were only looking at distance.

OPTIONAL:

The total number of states that exist for the smartcab in this environment is: 512

With the knowledge I have at this point of the project, since the deadline is a discrete variable I chose to turn it into 4 ordinal values. Deadline is a represented as an integer that decrements by 1 per each time slot. The deadline varies based on how far the smartcab is from its destination. The four ordinal values I chose to represent this variable are represent by how much time is left: 1) 100%, 2) 75%, 3) 50%, 4) 25%. This is something I can tune when building my model, but am going to start here.

The reason I scaled the deadline variable is because of the large dimensionality it creates when looking at each point individually in the range of values. For example, if the deadline is 50, it will have 51 total values (including 0). This example would have 6,528 possible states. However, with my 4 point scale there are only 512 possible states.

The scaled number of possible states does seem reasonable given that Q-learning is going learn and make informed decisions about each state. However, not scaling this does not seem reasonable given it would suffer from the curse of dimensionality and would be very costly in terms of computation.

Would a different scaling be better? How about a percentage of time remaining in groups of 10 versus 4? For example, 100%, 90% time left etc. This is something I will test later. It would give us 1,280 possible states.

	Light	Oncoming	Left	Deadline	Next Waypoint
Total number values	2	4	4	4	4
Possible values	Red, green	None, left, right, forward	None, left, right, forward	1) Not rushed, 2) watching the time, 3) better hurry, 4) YIKES!	None, left, right, forward

QUESTION 3: *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?*

When we implement the Q-learning, the agent takes routes more in the direction of the destination. It is somewhat random and chaotic at first but becomes more organized as the trials continue. This behavior is occurring because it is learning what is ‘right’ and ‘wrong’ based on the rewards it gets for doing specific actions in certain states.

QUESTION 4: *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?*

The different parameter values used were (alpha, gamma, epsilon):

I started with (1,1,1), (0.5,0.5,0.5), (0.1,0.1,0.1).

Then, I thought that a higher alpha and gamma with a lower epsilon made sense so I started playing in the (1,1,0.2) range.

The (0.8, 1, 0.3) performed best. The final driving agent performs quite well most of the time but has some terrible results sometimes too.

In addition to tuning alpha, gamma and epsilon, I also tried dropping the deadline as part of the state to help with the curse of dimensionality. By dropping the deadline it would reduce to only 128 possible states the cab could be in. Since dropping the deadline performed so poorly, I decided to try a binary approach to reduce dimensionality. Since the smartcab would get 10 bonus reward points if it finished before or at the deadline I chose to make the deadline as 'met deadline' if the deadline was >=0 or 'missed deadline' if it was <0. This did perform better than my original four options for deadline, which were split into quartiles.

Last, I also tuned the initial Qvalue that was used to initialize a new state and action in the Q table. I started with 0, tried 10, 5, 3, 2 and 1. 2 was the value that performed the best here.

QUESTION 5: *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?*

My agent does get close to finding an optimal policy. Towards the end of the trials, it reaches the destination within the time but not the minimum time and does still incur penalites.

An optimal policy for driving this smart cab would be to obey all traffic rules and to reach the destination in the fastest time possible.