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?*

Since the deadline check is set to false, the learning agent navigates forever within one single trial until it reaches the destination without any training algorithm being involved. I do not notice other interesting observations at this stage.

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?*

A unique state consists of combinations of NextWaypoint, Light, Left and Oncoming (as shown in my implementation of State.__eq__) and each of such combination defines what action can be taken – traffic rule. Ex. Light is green, no oncomging traffic, then it is ok to make a left turn, any of the premise missing, then taken action might receive negative reward.

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?*

State total = light combinations * NextWaypoint combinations * Left combinations * Oncoming combinations

The amount of states in total is smaller than 100 – indicating the state spaces is small, in stochastic exploration, if we are lucky, it only takes less than 100 steps covering all the combinations.

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?*

Our test car stays at the same location more often; the reason is that the QValue for staying at the same location might be higher than other actions. After adding nextwaypoint to part of the state, the car does not stuck on the same place anymore.

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?*

I believe that epsilon greedy means that even given the best action is "forward", there is still 10% chance that the car will turning right, and 10% turning left, 80% going forward, and that is probability based. The advantage is that it is able to explore other good policy even after having a good one.

http://www.wearepop.com/articles/secret-formula-for-self-learning-computers

there is no such implementation in my code, only gamma (discount rate) and alpha (α – learning rate) are used. As discussed in class material, gamma is something that we can ignore actually, the historical original for this variable comes from economic sector. For alpha – learning rate, the higher the value is the quicker the current state qvalue gets overrided.

I run the program, using different alpha values, 0.1, 0.3, 0.5, after 60 trials for training, run another 39 trials gathering following data

total positive reward and negative reward

alpha=0.1 - total positive reward: 854.0 / total negative reward: -6.0

alpha=0.3 - total positive reward: 848.0 / total negative reward: -4.0

alpha=0.5 - total positive reward: 826.0 / total negative reward: -6.5

the difference is not much, maybe alpha 0.3 is the best fit here.

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?*

An optimal policy should behave so that no violation of traffic rules should happen. From previous classes, for both supervised and unsupervised learning, there are metrics measuring how well an algorithm performs; but for this assignment, I have no idea when it is counted as converged even I am aware that the QValue changes is supposed to get smaller and smaller.