REPORT -- SMARTCAB/ REINFORCEMENT LEARNING.

SECTION 1: IMPLEMENT A BASIC DRIVING AGENT

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

Observations on Agent'S Behavior:
 1. The cab moves randomly without any strategy or sense of direction.
 2. The cab in seldom hits the waypoint but there is no definitive pattern that can observed.
 3. The agent attempts to move forward, left, right BUT due to the rules/ constraints of the simulator when an illegal action occurs-- running a red light for example-- no movement occurs and a negative reward/penalty is recorded.

Does The Smartcab Eventually Make It To The Destination ? NO, unless unless the driving agent is enhanced to incorporate the policy for optimal driving that ensures a correct state of the agent at each step, along with the way point.

SECTION 2: INFORM THE DRIVING AGENT

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?

All states that can help plan the next set of actions are crucial state parameters that needs to be captured, we will capture the scenarios that should be included also those that needs to exlcuded.

Following states are worth capturing-- Inclusion state variables

1. next_way_point: next_way_point helps suggest the direction of travel-- forward, left, right. This is crucial otherwise the cabs wouldn't have the intent to change the course of direction it is proceeding.

2. traffic_light: The state of traffic light-- whether green or red-- influences the course of action, if the traffic light is red the cab cannot take the next step (move forward, turn right/left) until the light again turns green along the direction of travel.

3. zebra_crossing: If there are people walking in the zebra crossing the cab needs to temporarily pause unless the crossing is cleared.

4. traffic_oncoming: This is important and decide the speed at which we can drive and can interfere with our desired action. If we want to travel left,for exmaple, we can do so as long as there is no oncoming car travelling straight across.

5. traffic_approaching_from_left: If we want right, on a red light or change the course of direction, we can do so as long as there are no cars from the left travelling through.

Scenrios that can be excluded-- Exclusion of state variables

1. Traffic from right: If we want to move forward and the traffic signal is green we can go, and if red we can't. If we want to go right, then on green we just go, on red we care what the car from the left wants to do. If we want to go left, then on red we stop, and on green we care what the oncoming car wants to do.

We do not care about the intentions of the car to the right and hence including it would marely add the white noise.

2. Deadline: Regardless of how long is left on the clock we still want to take the optimal action at each step-- better late than never :).
There is another challange in modelling deadline we would have to deal with lots

of scenarios which in turn would explode the number of states we need to account for in the learning matrix.

Optional Question: 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?

This is a classic Combinatorics problem, the approach would be to first identify the independent variables. We will then calculate the possible states that each independent variables can take and then find the likely cases that we can arrive at.

There seems to be 4 independent variables.

1. Direction of travel.
2. Color of traffic light.
3. oncoming traffic.
4. State of cars to the left.

Direction of travel: This can take one of the three states-- move forward, move left, move right-- 3C1.
Color of traffic light: This can take one of the two states-- red or green-- 2C1.
Oncoming traffic: This can take one of the four states-- None, forward, left, right-- 4C1.
State of cars to the left: This can take one of the four states-- None, forward, left, right-- 4C1.

All these are independent events and hence total possible states are

total_possible_states = 3 * 2 * 4 * 4 = 96.

The smart cab based on these 96 states can take either one of the four actions -- None, forward, left, right -- 4C1, and hence total number of possible actions are

total_possible_actions = 96 * 4 = 384.

This gives us 384 State/action pairs.

Does this number seems reasonable ??

The number of state/action pair seems quite large for a simple scenario BUT with the evident of big data and superior techniques of data collection and analysis we can analyze the possible outcomes for each state, and make better informed decisions on what can potentially go wrong.

SECTION3: IMPLEMENT A Q-LEARNING DRIVING AGENT

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?

Changes in the Agent's driving behavior

Positive/Encouraging observations:

1. The Driving Agent over the course of time begins to drive more safely and is better aware of the surroundings.
2. It notices that it is being penalized when it takes a wrong action e.g. the agent moves forward when the traffic light is red.
3. It also notices that taking no action-- None state-- when the traffic light is red is a better state since there are no penalties.
4. The Driving Agent also manages to get to the waypoint before the deadline -- initially it was just meandering and appeared it would take infinite time to reach the destination.

Eccentric Behavior/Observation:

1. The Agent appears to prefer to keep driving rather than hold still, e.g if it can't drive forward like it would prefer because of a red light

it seems to do so even though it will get hit with a -0.5 reward.
2. It still takes illegal actions sometimes.

Before answering the second part of the question `Why is this behavior is observed`, let's first list down the input parameter for Q Learning-- this plays a role in the observed behavior.

The Q-learning parameters are

1. Learning rate: The rate at which we want the Agent to accumulate the knowledge. The learning rate determines to what extent the newly acquired information will override the old information
2. Discount Value: The present value of future rewards, determines the importance of future rewards
3. Initial Epsilon Value: An exploration path value.This allows the Agent to explore a path by doing random selection.
4. Epsilon Degradation Rate: Penalty imposed on the exploration path.
5. Trials: The number of trials for the experiment.

Wikipedia link, https://en.wikipedia.org/wiki/Q-learning gives details on these parameters.

The initial experiment has following sets of W-learning parameters.

Learning Rate: 0.5
Discount Value: 0.8
Initial Epsilon Value: 0.1
Epsilon Degradation Rate: 0.0
Trials: 100

The results of the experiment for this set of values is

Deadlines Missed: 23
Successful Arrivals: 77
Traffic Infractions: 71

Let's answer the second part of the question, `Why is this behavior is observed` --- 23 Deadlines missed, 71 Traffic Infractions.

1. The discount rate for future rewards to be 0.8, and it's valuing the future 2.0 rewards it will get for driving back towards the waypoint too highly. Potential solution-- Reduce the discount rate significantly.

2. Path Exploration value is set to 10% and so even when it knows the right thing to do, 10% of the time it takes a random action, often an unsafe or illegal one. We need to have some exploration path that doesn't just pick a random value; maybe we start with a higher epsilon value and reduce it to 0 over time by reducing it each time the exploration path is hit by a tiny amount.

We will control/fine-tune the values of parameters--- epsilon value, discount value, learning rate-- to check the effect of the overall performance.

CASE1: ATTEMPTS TO MAKE EPSILON VALUE DEGRADE OVER TIME:

We choose an Epsilon Degradation Rate and set it to 0.01. The input parameters for the run are

Learning Rate: 0.5
Discount Value: 0.8
Initial Epsilon Value: 0.15
Epsilon Degradation Rate: 0.01
Trials: 100

The results are

Deadlines Missed: 62
Successful Arrivals: 38
Traffic Infractions: 5

Conclusion: Traffic Infractions has improved brought down from 71 to 5 BUT we are missing more deadlines now incresed from 23 to 62 (more than 2.5 times)

CASE2: DECREASE DISCOUNT VALUE:

We decrease discount value from 0.8 to 0.3. The input parameters for the run are

Learning Rate: 0.5
Discount Value: 0.3
Initial Epsilon Value: 0.1
Epsilon Degradation Rate: 0.001
Trials: 100

The results are

Deadlines Missed: 12
Successful Arrivals: 88
Traffic Infractions: 27

Conclusion: The Agent is now better in managing the Deadlines (88% times successful compared to the 77% of the base case) the number of Traffic Infractions also is much less than initial value (27% compared to 71%)

CASE3: INCREASE LEARNING RATE

We increase the Learning Rate from 0.5 to 0.65. The input parameters for the run are

Learning Rate: 0.65
Discount Value: 0.3
Initial Epsilon Value: 0.1
Epsilon Degradation Rate: 0.001
Trials: 100

The results are

Deadlines Missed: 14
Successful Arrivals: 86
Traffic Infractions: 22

Conclusion: The success rate is 86%(compared to initial 77%) and Traffic Infractions is 22% (compared to 71% initial).

SECTION4: IMPROVE THE Q-LEARNING DRIVING AGENT

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?

The analysis is available above. We obtained the best results for following set of parameters

Learning Rate: 0.65
Discount Value: 0.3
Initial Epsilon Value: 0.1
Epsilon Degradation Rate: 0.001
Trials: 100

We can further check the veracity of our claim by increasing the number of trails and check whether this performance is further enhanced, deteriorated or maintained. We select number of trials as 5000

Learning Rate: 0.6
Discount Value: 0.25
Initial Epsilon Value: 0.1
Epsilon Degradation Rate: 0.001
Trials: 5000

The results are

```
Deadlines Missed: 331
Successful Arrivals: 4619
Traffic Infractions: 24
```

Conclusion:

1. The process has already converged and performs slightly better when the number of trials are increased -- Successful Arrivals are around 91%.
No significant increase in Traffic Infractions.

2. We encountered an edge case though, when we run into another car at an intersection in the oncoming lane, we tend to lock up. This is similar to classic multi-threading deadlock problem when two or more threads are stuck on resource lock and are waiting for other thread to release the resources. Dining philosophers problem https://en.wikipedia.org/wiki/Dining_philosophers_problem.

To solve this problem we attach a penalty/negative reward for holding still -- less than the penalty of committing an infraction. We name this new parameter as penalty_for_staying_still and assign it a value of -0.1.

The input parameters are

```
Learning Rate: 0.6
penalty_for_staying_still: -0.1
Discount Value: 0.25
Initial Epsilon Value: 0.1
Epsilon Degradation Rate: 0.001
Trials: 100
```

The results are

```
Deadlines Missed: 4
Successful Arrivals: 96
Traffic Infractions: 28
```

This is a very good lift in the number of successful arrivals (only 4% of the deadlines are missed).

Scaling the experiment to 5000 trials to check if there is any performance degradation.

```
Learning Rate: 0.6
penalty_for_staying_still: -0.1
Discount Value: 0.25
Initial Epsilon Value: 0.1
Epsilon Degradation Rate: 0.001
Trials: 500
```

The results are
```
Deadlines Missed: 38
Successful Arrivals: 462
Traffic Infractions: 26
```

The successful arrivals are able 95% which implies the method is indeed working.

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 Agent has performed quite well within the constraints. The DEADLOCK problem-- getting stuck when in the oncoming lane-- can be be improved further and we can find some more smarter way to solve it since the consequences have deep effect-- it can easily lead to traffic jams even for 1% of the cases.

There is also an scope of improvement in selecting the most OPTIMAL action for a given state-- we solved this problem by epsilon degradation but I think its still messy.

We are getting the successful arrivals for more than 90% cases which is quite a good start considering we are not breaking any laws or crashing better than a higher time-efficiency rating at the expense of causing traffic accidents occasionally.

Final Trial Results:

```
Learning Rate: 0.6
penalty_for_staying_still: -0.1
Discount Value: 0.25
Initial Epsilon Value: 0.1
Epsilon Degradation Rate: 0.001
Trials: 100
Deadlines Missed: 1
Successful Arrivals: 99
Traffic Infractions: 39
```

Note: The successful arrival are above 99% but we are having Traffic Infractions 39% of the time which I think needs improvement.