QUESTION1:Observe what you see with the agent's behavior as it takes random actions. Does the eventually make it to the destination? Are there any other interesting observations to note?

By set epsilon as 1, the cap will make action randomly.

With enforce_deadline as False, the smartcab will eventually make it to the destination.

With enforce_deadline as True, the smartcab success reach the destination 22 of 100 trials. The rate equals to 0.22.

Observation:The route from start to destination is not optimal because the cab does not handle with specific situation and does not remember or utilize previous experience. So, each time, it takes different iteration numbers to finish the route. Most iterations exceed the default limit.

QUESTION2: 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?

The states include the traffic lights (red / green), the oncoming traffic (forward/none, the right traffic(right/none), the left traffic(left/none).These selections come from the traffic rule in US. The light and traffic situation does matter the decision of the car.

Besides, I also use next_waypoint in the model. It could be used to defined the target to reach the destination. The agent can use this information to act properly to reach the destination in an efficient manner.

QUESTION3: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?

The parameter within is self.epsilon = 0.1 , self.gamma = 0.5, self.learning_rate = 0.3.

After implementing the learning, I find the car could make the right behavior meeting specific situation. And it could reach the destination faster compared to before.

With enforce_deadline as True, the smartcab success reach the destination 87 of 100 trials. The rate equals to 0.87.

The reason for that is because the system chose the larger reward when making decision. So the cab is more likely to choose activity with better result. Also, because we also have next_waypoint in the model, the whole process should be accelerated.

QUESTION4: 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 parameters I choose to change include learning rate. Gamma and Epsilon. Detailed result shown below:

Learning Rate	Gamma	Epsilon	
0.2	0.5	1	22
0.2	0.5	0.4	73
		0.2	88
		0.1	97
		0.05	97
	0.6	0.4	73
		0.2	83
		0.1	88
		0.05	97
	0.8	0.4	72
		0.2	85
		0.1	91
		0.05	94
	0.9	0.4	68
		0.2	77
		0.1	91
		0.05	94
0.1	0.5	0.4	76
		0.2	87
		0.1	96
		0.05	96
	0.6	0.4	72
		0.2	75
		0.1	91
		0.05	91
	0.8	0.4	70
		0.2	80
		0.1	88
		0.05	93
	0.9	0.4	67

		0.2	85
		0.1	81
		0.05	91
0.05	0.5	0.4	68
		0.2	88
		0.1	99
		0.05	98
	0.6	0.4	70
		0.2	76
		0.1	91
		0.05	94
	0.8	0.4	70
		0.2	84
		0.1	94
		0.05	97
	0.9	0.4	62
		0.2	79
		0.1	89
		0.05	94

The best parameter combination is {“learning rate”:0.05, “Gamma”:0.5, “Epsilon”:0.1}, it achieves 99 success in 100 trials.

I also check the Q-table values at last, part of them is shown below:

state					action	Q
Light	Oncoming	right	left	next_waypoint		
'green'	None	'left'	None	'right'	None	1.145866191
'red'	'left'	None	None	'forward'	'left'	0.55
'red'	None	None	'forward'	'forward'	None	1
'green'	None	'left'	None	'right'	'right'	1
'red'	'forward'	None	None	'right'	None	0.85
'red'	None	'forward'	None	'forward'	'left'	1
'green'	None	None	'left'	'right'	None	0.745
'red'	None	'right'	None	'forward'	None	0.825083846
'green'	None	'right'	None	'right'	'forward'	1
'red'	None	None	'left'	'right'	None	0.85
'red'	None	None	'left'	'right'	'right'	1
'green'	None	None	None	'right'	None	0.745
'green'	None	None	None	'forward'	None	0.745
'red'	None	None	None	'left'	'left'	-0.015388232
'red'	None	None	'right'	'forward'	'left'	1

'red'	'left'	None	None	'right'	'left'	1
'green'	None	'left'	None	'left'	'right'	1
'green'	'left'	None	None	'forward'	'right'	1
'red'	None	'left'	None	'forward'	None	1
'green'	'left'	None	'left'	'forward'	'left'	1
'green'	None	'forward'	None	'right'	'right'	1
'green'	None	None	'left'	'right'	'right'	1.732159659
'red'	None	None	'forward'	'forward'	'right'	1
'red'	None	'right'	None	'forward'	'forward'	1
'green'	'left'	None	None	'right'	'left'	0.720820641
'green'	None	None	'left'	'forward'	'left'	1
'green'	None	None	'left'	'right'	'forward'	0.764125
'red'	'left'	None	None	'right'	'forward'	0.541888
'green'	None	None	None	'left'	'left'	2.932120795
'red'	'left'	None	None	'forward'	None	0.544847509
'green'	'left'	None	None	'right'	'forward'	0.588532147
'green'	None	None	'forward'	'forward'	'left'	1
'green'	None	'left'	None	'right'	'forward'	1
'green'	None	None	'right'	'forward'	'left'	1
'green'	None	'forward'	None	'forward'	'forward'	2.18652109
'green'	None	'forward'	None	'forward'	'right'	1
'green'	None	'left'	None	'forward'	'left'	1.020132831
'red'	None	None	None	'right'	'right'	2.760440833
'red'	None	None	'left'	'right'	'forward'	1
'red'	'forward'	None	None	'right'	'forward'	1
'red'	None	'right'	None	'left'	'forward'	0.561313286
'green'	'left'	None	None	'right'	'right'	1
'green'	None	None	None	'left'	'forward'	1.051853013
'red'	None	'left'	None	'forward'	'forward'	1
'red'	None	'right'	None	'forward'	'right'	1
'green'	None	None	None	'right'	'right'	2.85474605
'green'	'left'	None	None	'left'	'right'	1
'red'	'forward'	None	None	'forward'	None	1.221771132
'green'	None	'right'	None	'forward'	'left'	1
'green'	None	'forward'	None	'right'	None	1
'red'	None	None	None	'left'	'right'	0.89866621
'red'	None	'forward'	None	'forward'	'forward'	1
'red'	None	None	None	'right'	'left'	0.243752538
'green'	None	None	'forward'	'right'	'forward'	0.733119207
'green'	None	'forward'	None	'right'	'forward'	1.052872737

'green'	None	None	'left'	'forward'	'right'	1
'green'	None	'left'	None	'forward'	'forward'	2.12590434
'green'	'left'	None	None	'left'	None	0.85
'green'	None	'forward'	None	'forward'	None	0.745
'red'	'left'	None	None	'right'	None	0.745
'red'	None	'right'	None	'left'	'left'	1
'green'	None	'right'	None	'right'	'right'	1
'green'	None	'forward'	None	'right'	'left'	1
'green'	None	None	'right'	'forward'	'right'	1
'green'	None	None	'right'	'forward'	None	1.118130362
'red'	None	None	'right'	'forward'	None	1.149814803
'green'	None	'left'	None	'right'	'left'	1
'red'	'left'	None	None	'left'	'left'	1
'red'	None	None	'right'	'forward'	'forward'	1
'green'	None	None	'left'	'forward'	'forward'	2.215762875
'red'	'forward'	None	None	'right'	'left'	1
'green'	None	None	None	'left'	'right'	1.051327685
'red'	None	'forward'	None	'forward'	None	0.822435919
'red'	None	None	None	'right'	None	1
'red'	None	'right'	None	'forward'	'left'	1
'red'	'left'	None	None	'left'	'forward'	0.653305987
'red'	None	None	'left'	'forward'	'right'	1
'green'	None	None	None	'left'	None	0.85
'green'	None	'right'	None	'forward'	'forward'	1.934131254
'red'	None	None	'left'	'forward'	'left'	1
'green'	None	None	None	'forward'	'right'	0.733913449
'red'	None	None	'forward'	'forward'	'left'	1
'red'	None	None	None	'left'	None	0.311538672
'green'	None	'left'	None	'left'	None	0.871395395
'green'	'left'	None	None	'forward'	None	0.745
'red'	'forward'	None	None	'forward'	'right'	0.7
'green'	None	'right'	None	'right'	'left'	1
'green'	None	None	'left'	'left'	None	1
'red'	None	'left'	None	'forward'	'left'	1
'green'	'left'	None	None	'forward'	'forward'	2.037748874
'red'	'left'	None	None	'right'	'right'	2.371561374
'red'	'forward'	None	None	'right'	'right'	1.429187013
'green'	'left'	None	'left'	'forward'	None	0.745
'green'	'left'	None	None	'left'	'left'	1
'green'	None	None	'left'	'left'	'forward'	0.987309133

'green'	None	'left'	None	'forward'	None	0.685386339
'green'	None	None	'left'	'left'	'left'	1
'green'	None	None	'forward'	'right'	'right'	1
'red'	None	'forward'	None	'forward'	'right'	1
'green'	'left'	None	'left'	'forward'	'right'	1
'red'	None	None	None	'forward'	None	0.8166454
'red'	'left'	None	None	'left'	None	0.745
'red'	'forward'	None	None	'forward'	'forward'	0.80317505
'green'	'left'	None	None	'left'	'forward'	1
'red'	None	None	None	'forward'	'right'	0.330284001
'green'	None	None	'forward'	'right'	None	0.745
'green'	None	None	None	'forward'	'forward'	3.254439639
'green'	None	'forward'	None	'forward'	'left'	1
'green'	None	'right'	None	'forward'	'right'	1
'red'	'left'	None	None	'left'	'right'	0.689313059
'red'	None	'left'	None	'forward'	'right'	0.98439244
'green'	None	None	'forward'	'forward'	'forward'	1.697217901
'green'	None	None	'forward'	'forward'	'right'	1
'green'	None	None	'left'	'left'	'right'	1
'red'	None	'right'	None	'left'	'right'	1
'green'	None	'right'	None	'right'	None	1.154620259
'red'	None	None	None	'right'	'forward'	0.655550452
'green'	None	None	None	'forward'	'left'	0.550819399
'red'	None	None	None	'forward'	'forward'	-0.2599534
'red'	'left'	None	None	'forward'	'right'	0.776435038
'green'	None	None	'right'	'forward'	'forward'	1
'green'	'left'	None	'left'	'forward'	'forward'	1.420788262
'green'	None	None	'left'	'right'	'left'	0.807626797
'green'	None	'left'	None	'left'	'left'	1
'red'	None	None	'left'	'right'	'left'	1
'green'	None	None	None	'right'	'left'	0.810033935
'red'	None	None	None	'left'	'forward'	0.235
'red'	None	None	'forward'	'forward'	'forward'	0.514591758
'red'	None	None	'left'	'forward'	None	1.032364431
'green'	'left'	None	None	'right'	None	0.745
'red'	None	None	None	'forward'	'left'	-0.603142901
'green'	None	'right'	None	'forward'	None	0.808890813

From the table, we can see q-table has high value for good action,example is shown below, when light is green and no traffic issues, if we know next waypoint is the left,,

the best action is to turn left, so this state and action has high value to encourage this behavior.

'green' None None None 'left' 'left' 2.932120795

Besides, the learning also make penalty on bad behavior. Example is shown below:

'green' None None None 'left' 'left' 2.932120795

.

I also find there are still many '1' in the table. By increase the Epsilon or the limit number of trials.This could be resolved:

(('green', 'left', None, None, 'left'), 'right'):-0.00763617049003
(('red', None, 'right', None, 'forward'), 'forward'):0.201804312957
(('green', 'forward', None, None, 'forward'), 'left'):0.27124391058
(('green', 'left', None, None, 'right'), 'left'):-0.596100461092
(('green', None, None, 'left', 'forward'), 'left'):-0.178324211063
(('red', 'forward', None, None, 'right'), 'forward'):0.0577552616481
(('red', 'left', None, None, 'right'), 'forward'):-0.698116047779
(('green', None, None, None, 'left'), 'left'):2.07372782492

Meanwhile, I also try different initialization of Q-table values:

Learning Rate	Gamma	Epsilon	Q-initialization	Success/All
0.1	0.8	0.5	-1	12/100
			1	60/100
			10	65/100

QUESTION5:: 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 find the optimal policy.

One optimal policy is described below:

Environment.reset(): Trial set up with start = (5, 6), destination = (6, 3), deadline =
RoutePlanner.route_to(): destination = (6, 3)
LearningAgent.update(): deadline = 20, inputs = {'light': 'red', 'oncoming': None, 'right': None, 'left': None}, action = None, reward = 0.0
LearningAgent.update(): deadline = 19, inputs = {'light': 'red', 'oncoming': None, 'right': None, 'left': None}, action = None, reward = 0.0
LearningAgent.update(): deadline = 18, inputs = {'light': 'red', 'oncoming': None, 'right': None, 'left': None}, action = None, reward = 0.0
LearningAgent.update(): deadline = 17, inputs = {'light': 'green', 'oncoming': None, 'right': None, 'left': None}, action = left, reward = 2.0

LearningAgent.update(): deadline = 16, inputs = {'light': 'green', 'oncoming': None, 'right': None, 'left': None}, action = left, reward = 2.0
LearningAgent.update(): deadline = 15, inputs = {'light': 'green', 'oncoming': None, 'right': None, 'left': None}, action = forward, reward = 2.0
Environment.act(): Primary agent has reached destination!
LearningAgent.update(): deadline = 14, inputs = {'light': 'green', 'oncoming': None, 'right': None, 'left': None}, action = forward, reward = 12.0

We could visualize the Penalty along the trials. We could see that after 80 trials, we only receive non-negative penalty. We are get close to the optimal result.