1. Implement a basic driving agent

Produces random action which are None, Forward, Left and Right with Random(). which will choose actions evenly.

When agent move by random action, it is likely to go around the map and sometime go and back and forward around the same place. It will eventually go around and will get to target location with some random step sometimes it is fast and sometimes it looks hundreds of steps.

2. Identify and update state

I choose state to have 4 attributes.

left	boolean to tell wether it is okay to go left according to traffic rules
right	boolean to tell wether it is okay to go right according to traffic rules
straight	boolean to tell wether it is okay to go straight according to traffic rules
lastReward	number of reward agent get from previous move.

The first 3 attributes left, right and straight is picked because it will help agent learning traffic rules. And lastReward is picked so that agent can learn while running that the direction it heading is correct or not and learn correct action with this reward information to move to target location.

3. Implement Q-Learning

I set initial Q value to be at 1.0. And choose the highest Q value as action. If they have same Q value then randomly choose one.

After action is chosen, agent get reward and use reward to update Q table below formula

*Q(S,A) = (1-learningRate)*Q(S',A') + learningRate * (reward + discountFactor * Q(S',A'))*

It will pick the highest reward action so after steps it will learn that going against traffic rule will give minus reward and that action will not be picked again.

4. Enhance the driving agent

Q-Learning in #3 will choose only best action so it is very depends on what action is chosen first and it will try to do the same actions and it does not try to explore others action. I saw agent try to turn right many times and some point it change to None action many times. So eventually agent may choose to stay in None action forever. To fix this problem I insert epsilon variable and tell it to explore others options.

Epsilon value is set to 1.0 at start and linearly decrease. Where at trial #35 it epsilon becomes 0.0 and stop to explore other actions. And from here agent will choose the best score from Q-table.

After insert exploration action, agent start to move around randomly at first, but it still come back to choose None action again because None action may give 1.0 reward but forward, left and right sometimes give 2 but mostly it gives 0.5. So agent decide that None is the best action. So I remove None action out of action list.

```
if len(actionList) > 1 :
    idx = actionList.index(None)
    qValueList.remove(qValueList[idx])
    actionList.remove(None)
```

I also try to increase reward by multiple of 2 if it it not None action.

```
if action != None :
    reward = reward * 2
```

Result of both method are similar that now agent will not choose None action and stay still. But it will move around. And continue to update other action which is not None.

Other parameters I tune are learning rate and discount factor, and final parameter is learning rate = 0.5 Discount factor = 0.2 with reward x2 for actions that is not None.

The policies that agent learn are

- go straight if it can when direction is correct
- otherwise if direction is not correct or traffic signal does not allow to go straight, it will turn right or some situation, left.

	LR 0.9 DF 0.2 XNone	LR 0.9 DF 0.5 XNone	LR 0.5 DF 0.2 XNone	LR 0.5 DF 0.2 reward x 2	LR 0.9 DF 0.2 reward x 2	LR Linear DF 0.2 Reward x 2
Sum of step left from trial #35	334	529	491	**597**	416	533
Count reach destination from trial #35	18	36	39	**39**	24	34

LR = learning rate *LR Linear means decrease Learning Rate linearly
DF = Discount factor
XNone = remove None action from action list
rewardx2 = increase reward that is not for None action by multiple of 2