```
import random
from environment import Agent, Environment
from planner import RoutePlanner
from simulator import Simulator

class LearningAgent(Agent):
    """An agent that learns to drive in the smartcab world."""

    def __init__(self, env):
        super(LearningAgent, self).__init__(env)  # sets self.env = env, sta
        self.color = 'red'  # override color
        self.planner = RoutePlanner(self.env, self)  # simple route planner
        # TODO: Initialize any additional variables here

    def reset(self, destination=None):
        self.planner.route_to(destination)
        # TODO: Prepare for a new trip; reset any variables here, if require

    def update(self, t):
        # Gather inputs
        self.next_waypoint = self.planner.next_waypoint()  # from route plan
        inputs = self.env.sense(self)
        deadline = self.env.get_deadline(self)

        # TODO: Update state
        state_to_binary =''

        # from the sense function create the corresponding dict key
        # construct state to binary
        if self.next_waypoint == 'forward':
          state_to_binary += '11'
        elif self.next_waypoint == 'left':
          state_to_binary += '10'
        elif self.next_waypoint == 'right':
          state_to_binary += '01'
        elif self.next_waypoint == None:
          state_to_binary += '00'


        if inputs.light == 'green':
          state_to_binary += '1'
        elif inputs.light == 'red':
          state_to_binary += '0'

        if inputs.oncoming_now == 'left':
          state_to_binary += '10'
        elif inputs.left_now == 'forward':
          state_to_binary += '01'
        elif inputs.oncoming_now == 'forward':
          state_to_binary += '11'
        else:
          state_to_binary += '00'


        # TODO: Select action according to your policy
        if state_count_dictionary[state_to_binary] == 0:
          action = random.choice(['left','right','forward',None])
        else:
          action_dict = state_action_dictionary[state_to_binary]
          max_val = max(action_dict, key = lambda x: action_dict[x])

          possible_actions = []

          for kee in action_dict:
            if action_dict[kee] == max_val:
              possible_actions.append(kee)

          action = random.choice(possible_actions)

        # Execute action and get reward
        reward = self.env.act(self, action)

        # TODO: Learn policy based on state, action, reward

        state_count_dictionary[state_to_binary] += 1

        state_action_dictionary[state_to_binary][action] += reward

        print "LearningAgent.update(): deadline = {}, inputs = {}, action =


state_action_dictionary = {'11111':{'right':0,'left':0,'forward':0,None:0},
                           '11100':{'right':0,'left':0,'forward':0,None:0},
                           '10110':{'right':0,'left':0,'forward':0,None:0},
                           '10011':{'right':0,'left':0,'forward':0,None:0},
                           '10000':{'right':0,'left':0,'forward':0,None:0},
                           '00010':{'right':0,'left':0,'forward':0,None:0},
                           '00111':{'right':0,'left':0,'forward':0,None:0},
                           '00100':{'right':0,'left':0,'forward':0,None:0},
                           '01010':{'right':0,'left':0,'forward':0,None:0},
                           '01111':{'right':0,'left':0,'forward':0,None:0},
                           '01100':{'right':0,'left':0,'forward':0,None:0},
                           '11010':{'right':0,'left':0,'forward':0,None:0},
                           '11101':{'right':0,'left':0,'forward':0,None:0},
                           '10001':{'right':0,'left':0,'forward':0,None:0},
                           '00101':{'right':0,'left':0,'forward':0,None:0},
                           '11001':{'right':0,'left':0,'forward':0,None:0},

state_count_dictionary = {'11111':0, '11110':0, '11101':0,
                          '11100':0, '10111':0, '10101':0,
                          '10110':0, '10100':0, '10001':0,
                          '10011':0, '10010':0, '10000':0,
                          '00001':0, '00011':0, '00010':0,
                          '00000':0, '00111':0, '00110':0,
                          '00101':0, '00100':0, '01011':0,
                          '01010':0, '01000':0, '01001':0,
                          '01111':0, '01110':0, '01101':0,
                          '01100':0, '11011':0, '11001':0,
                          '11010':0, '11000':0}


def run():
    """Run the agent for a finite number of trials."""

    # Set up environment and agent
    e = Environment()  # create environment (also adds some dummy traffic)
    a = e.create_agent(LearningAgent)  # create agent
    e.set_primary_agent(a, enforce_deadline=False)  # set agent to track

    # Now simulate it
    sim = Simulator(e, update_delay=1.0)  # reduce update_delay to speed up
    sim.run(n_trials=10)  # press Esc or close pygame window to quit


if __name__ == '__main__':
    run()

```