**Implement a basic driving agent**

Implement the basic driving agent, which processes the following inputs at each time step:

* Next waypoint location, relative to its current location and heading,
* Intersection state (traffic light and presence of cars), and,
* Current deadline value (time steps remaining),

And produces some random move/action (None, 'forward', 'left', 'right'). Don’t try to implement the correct strategy! That’s exactly what your agent is supposed to learn.

Run this agent within the simulation environment with enforce\_deadline set to False (see runfunction in agent.py), and observe how it performs. In this mode, the agent is given unlimited time to reach the destination. The current state, action taken by your agent and reward/penalty earned are shown in the simulator.

### Introduction

### Initial Behavior

*In your report, mention what you see in the agent’s behavior. Does it eventually make it to the target location?*

Prior to any updates of the agent.py file, the agent (red cab in the pygame window) doesn’t move. Regardless of the time limit, the cab does not make it to the target destination.

**Identify and update state**

Identify a set of states that you think are appropriate for modeling the driving agent. The main source of state variables are current inputs, but not all of them may be worth representing. Also, you can choose to explicitly define states, or use some combination (vector) of inputs as an implicit state.

At each time step, process the inputs and update the current state. Run it again (and as often as you need) to observe how the reported state changes through the run.

*Justify why you picked these set of states, and how they model the agent and its environment.*