# P4 – SmartCab Project Write-up

## Implement a basic driving agent

## Identify and update state

State could be defined as the inputs into the agent at this time. An example of this is:

State = {'light': 'green', 'oncoming': None, 'right': None, 'left': None}

We also have an input which is the direction that the Planner is trying to send the Smart Cab. This can be:

Left, right, forward, none

A combination of these could be used to create the state for the cab, for example:

State = {‘planner\_action’: ‘Left’, 'light': 'green', 'oncoming': None, 'right': None, 'left': None}

Consideration was given to the duration (e.g. the time left) but this has been discounted as we do not want the cab to take more risky action in order to get a higher score based on a minimal length of time. This could lead to crashes which as a cab passenger we would want to avoid! It also increases the number of states dramatically making learning harder.

However, this still leaves a potential 512 states which is too many for a potential 40-50 step journey. Therefore, the following have been identified as the key states that need modelling (this could potentially be reduced to a smaller set of states but this reflects the driving rules defined for the game in a meaningful way):

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | Inputs |  |  |  |  |  |
| State ID | Light | Oncoming | Left | Right |  | Planner\_action |
|  |  |  |  |  |  |  |
| 1 | Green | None | None | None |  | Any |
| 2 | Green | Forward | None | None |  | Left |
| 3 | Red | None | None | None |  | None |
| 4 | Red | None | None | None |  | Right |
| 5 | Red | Forward | None | None |  | Left |
| 6 | Red | Left | None | None |  | Right |
| 7 | Red | None | Forward | None |  | Any |
| 8 | Red | None | None | Forward |  | Any |

The following functions have been defined in agent.py:

1. setup\_states() creates a default state table for each of the above state ids
2. determine\_state\_id(input, planner\_action) determines an ID from the table above

## Implement Q-Learning

The learnt reward for each state are held in a Python dictionary called ‘q’. This has a key based on the state\_id from the above table.