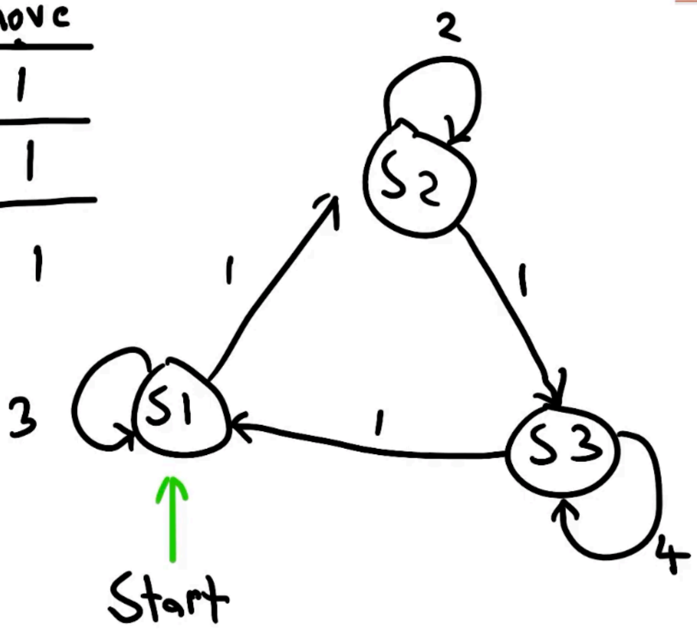
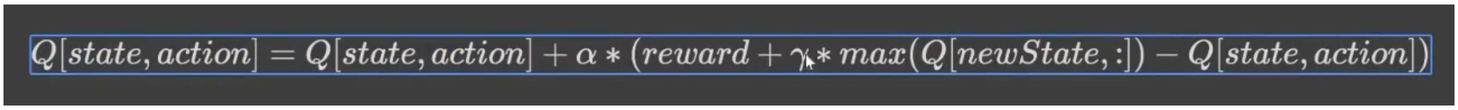
- there are three states
- in each state either the agent can stay or leave
- the agent needs to take the action
- ***the agent wants to maximise the reward in the environment***
    - -> the number of states, actions and ways the agent can interact with the states
    - -> staying in the same state can give it a reward
    - -> the agent can interact with the environment which changes its state
- ***the story of the agent***
    - -> the agent starts at state s1
    - -> it can either stay in the current state or move and receive a reward
    - -> he's making a matrix of all of the different possibilities and what the rewards are for each of them
    - -> looking at the table one time for each state
    - -> depending on where the agent starts

	Stay	Move
S1	3	1
S2	2	1
S3	4	1



    - -> it can trapped in local minima -> when it could be receiving a much larger reward elsewhere
    - -> exploring the environment and observing the rewards is how the Q-table is filled
        - you can end up with millions of these languages
        - once we have the table, then the model can produce the optimal actions depending on the state the agent is in
        - -> this is not based off of previous experiences -> where to go next is based off of the rewards of going there (which are independent of which state the agent came from)
        - -> taking random actions and being able to explore the environment more freely
- ***-> learning the Q-table <- the table of rewards for different actions***
    - -> you need the agent to go after different actions which haven't been tried before in order to see what the rewards are
    - -> the model can randomly take an action, or pick one based off of what the greatest reward would be <- the actual algorithm does a mixture of both
        - -> if this is not done then it can get stuck in a local maxima - missing out the rewad from another state
    - -> when it gets into a new state, it keeps updating the current environment



$$Q[state, action] = Q[state, action] + \alpha * (reward + \gamma * max(Q[newState, :]) - Q[state, action])$$

- -> a is the learning rate
- -> gamma is for the discount factor
  - -> the balance between taking a random action and one which would reap the highest rewards based off of where the agent was
  - -> the move action had to have a higher reward value than having it stay
    - this factor is how much the Q-values are allowed to be updated by after every single action
  - -> the learning rate means it will update slower
  - -> the value which is added is positive or negative
  - -> it's asking what the maximum reward given the new state which it's just moved into is
- -> factoring in the reward to determine the best place to move into
- -> the transition states
- -> subtracting the states and actions to prevent the agent from getting stuck in the same state
- -> the learning rate tells you how much you can update each learning value by
  - -> the balance between the future actions and the current one