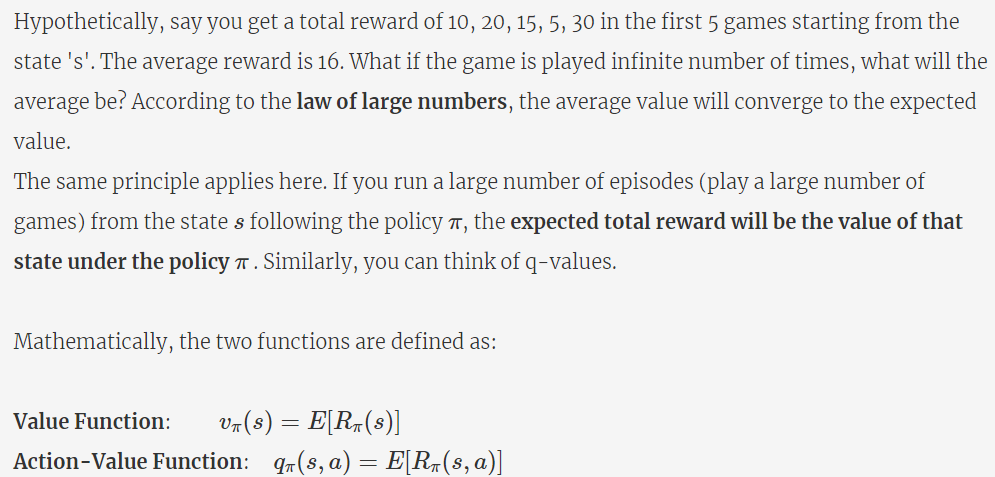
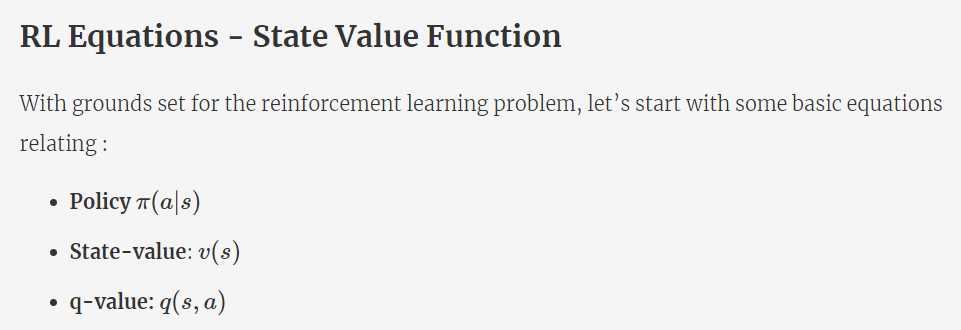
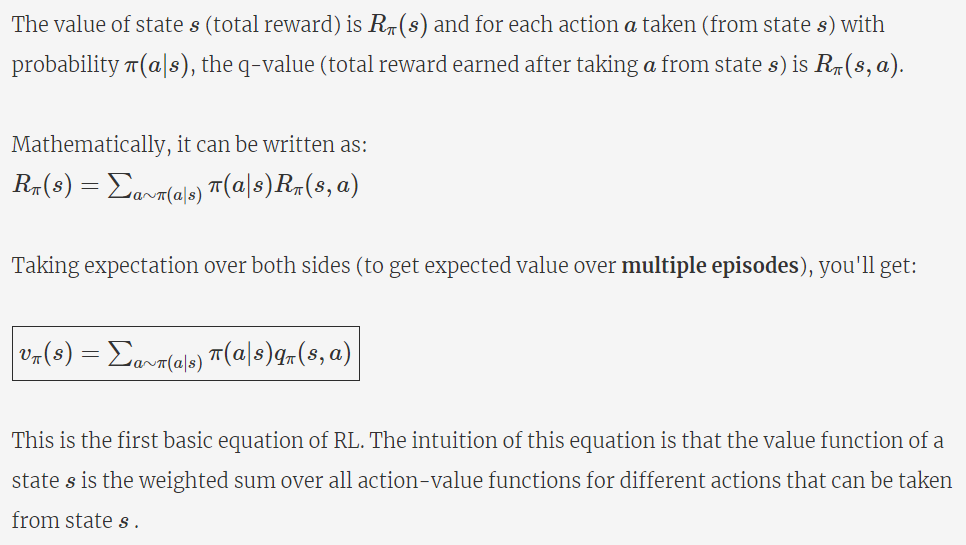
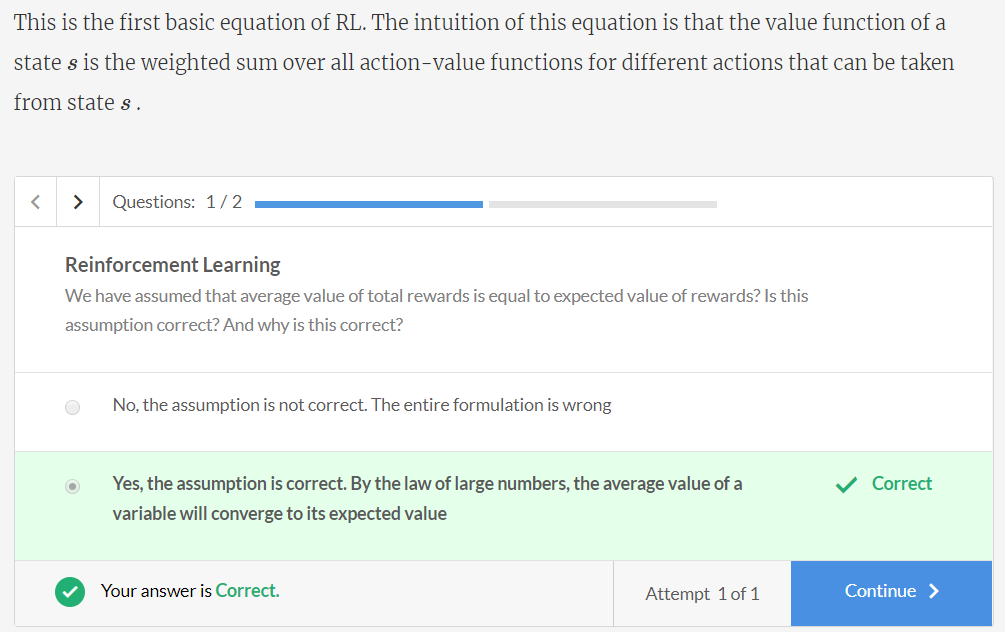
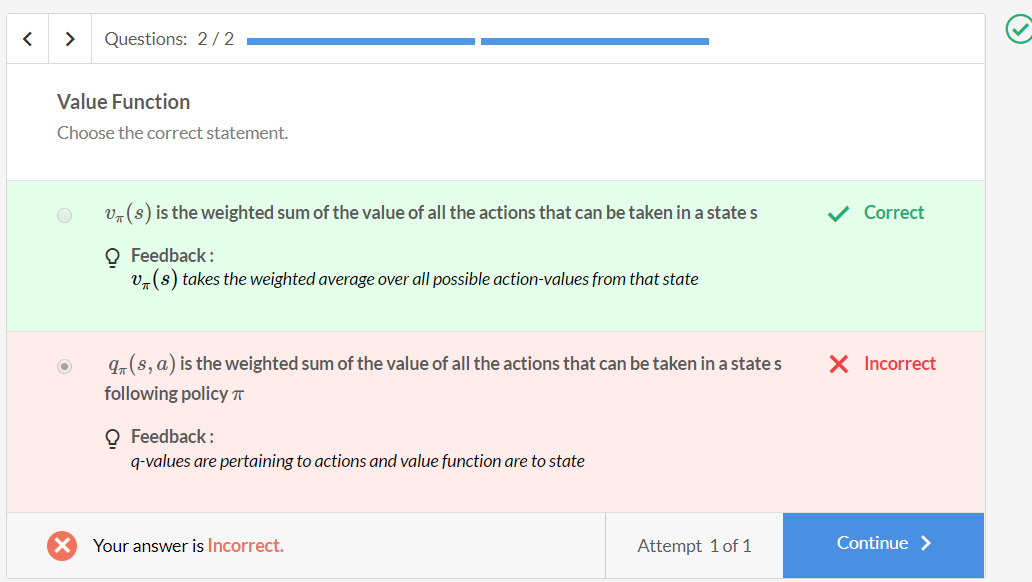
1. **State Value Function**







**Comprehension - State Value Function**

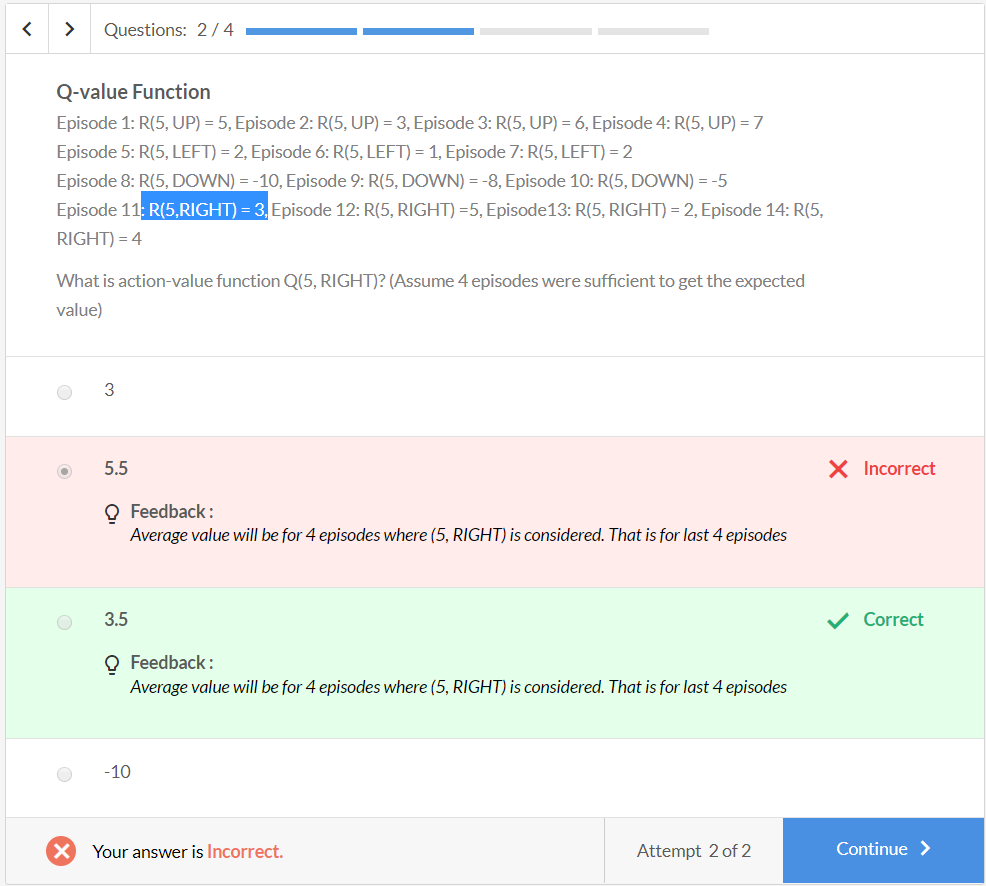
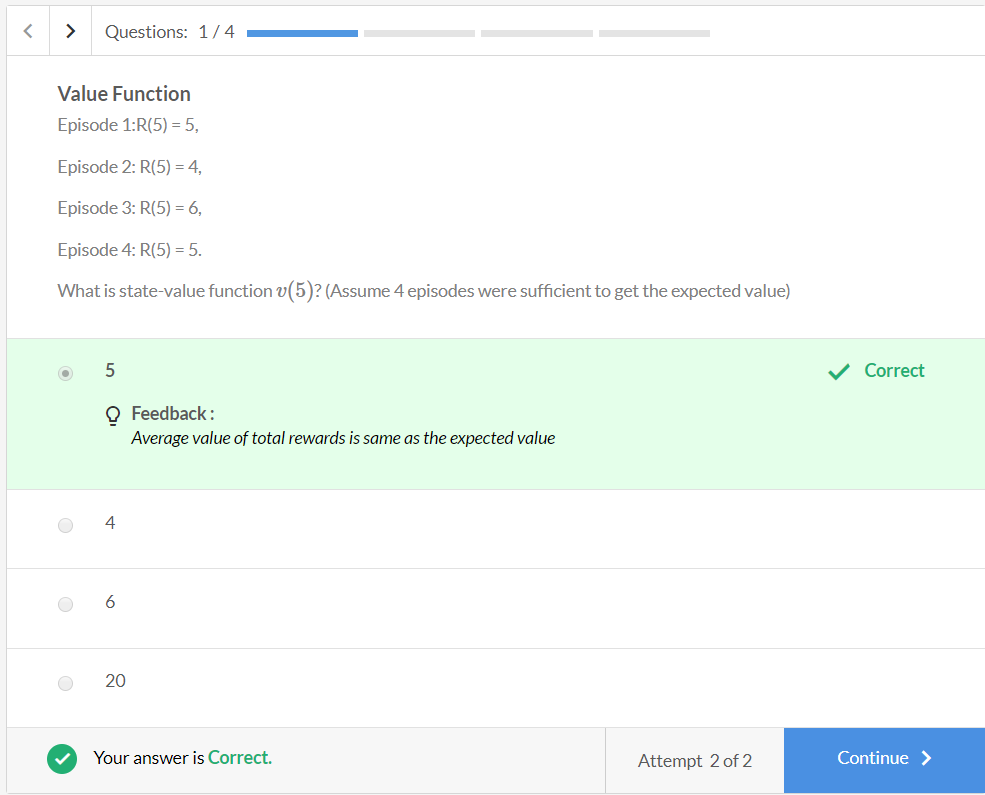
Consider the following 3x3 grid as the environment. The objective of the agent is to reach the cell named Target (9).

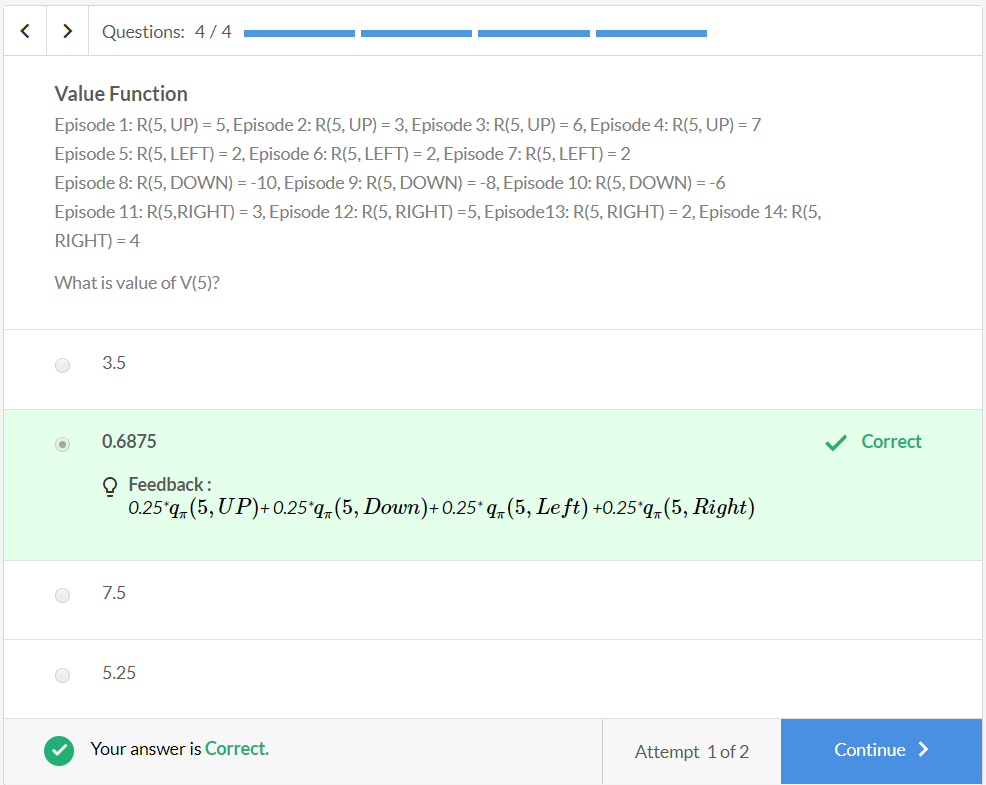
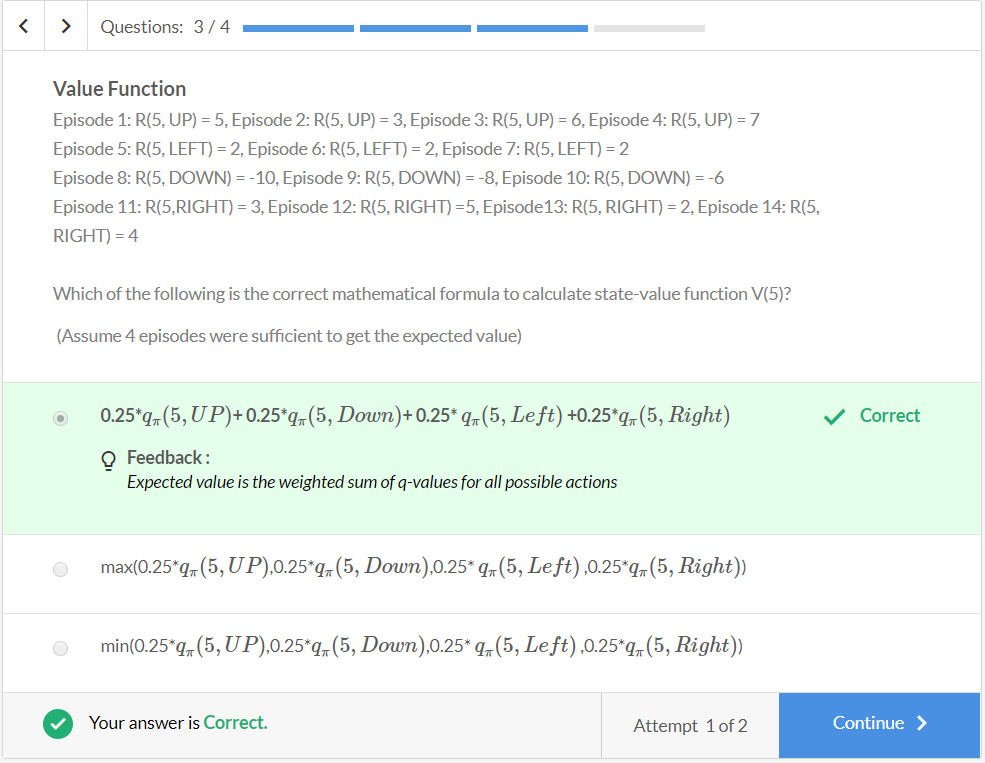
|  |  |  |
| --- | --- | --- |
| 3 | 4 | 9(Target) |
| 2 | 5(Current) | 8 |
| 1 | 6 (Fire) | 7 |

The number in each grid represents the state. The current position of the agent is 5.  There are 4 possible actions: UP, RIGHT, LEFT, DOWN.

The policy π for any state is that ‘there is a 25% chance of moving in each of the 4 directions’.

Multiple episodes were run where the starting state of the agent is always 5 and total rewards were calculated for each of the episodes. Assume that Rπ(x) represents the total reward received starting from state x, and Rπ(x,a) represents the reward of taking an action a in state x.

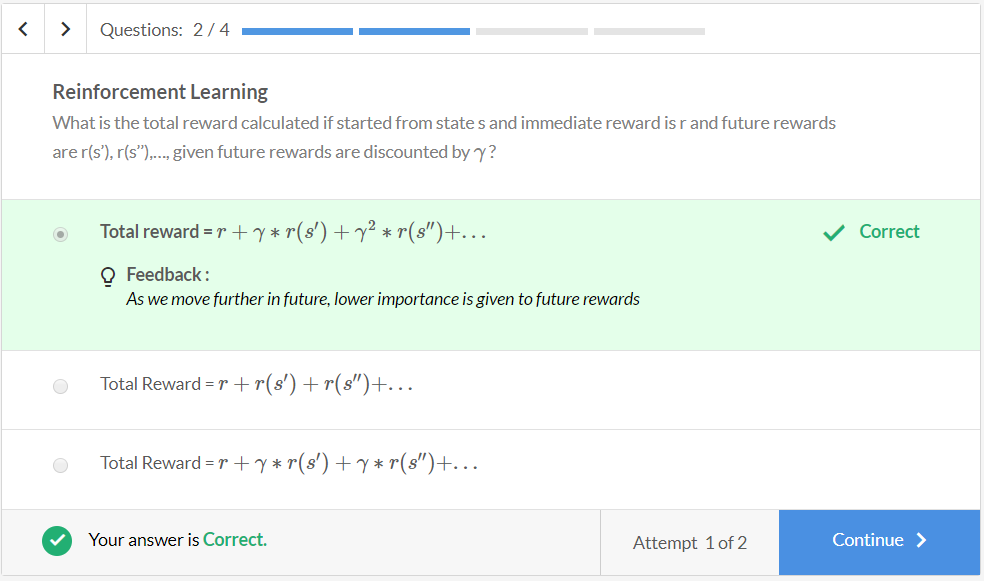
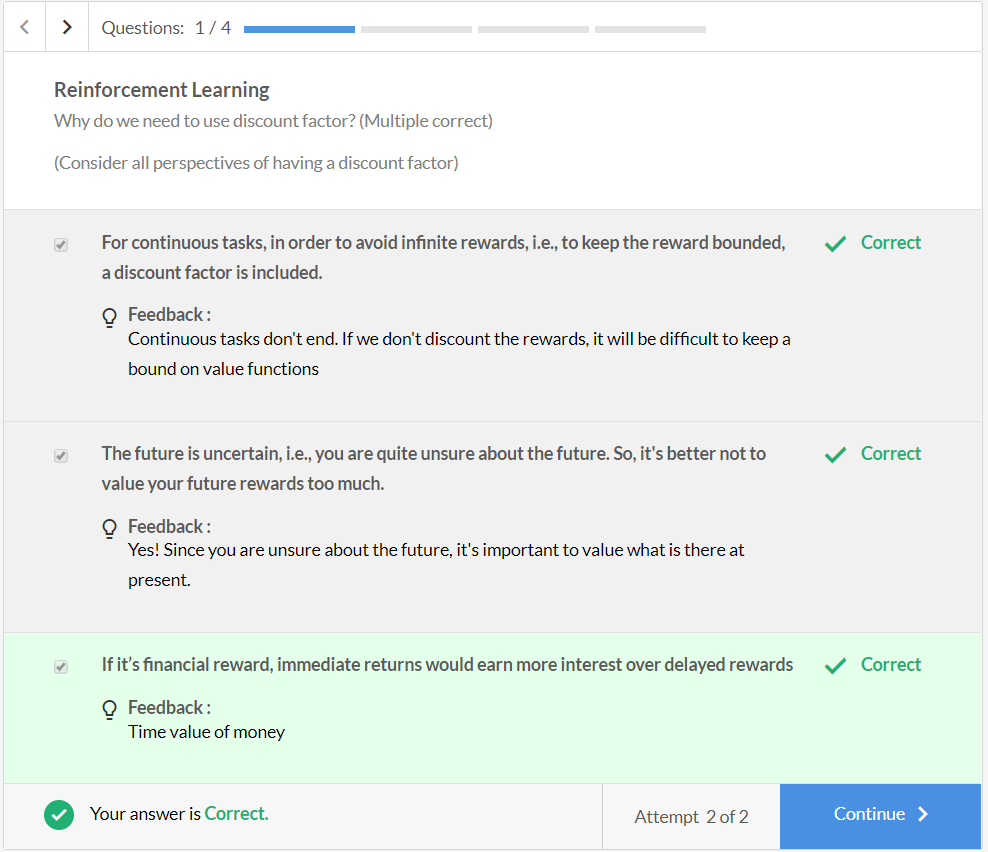


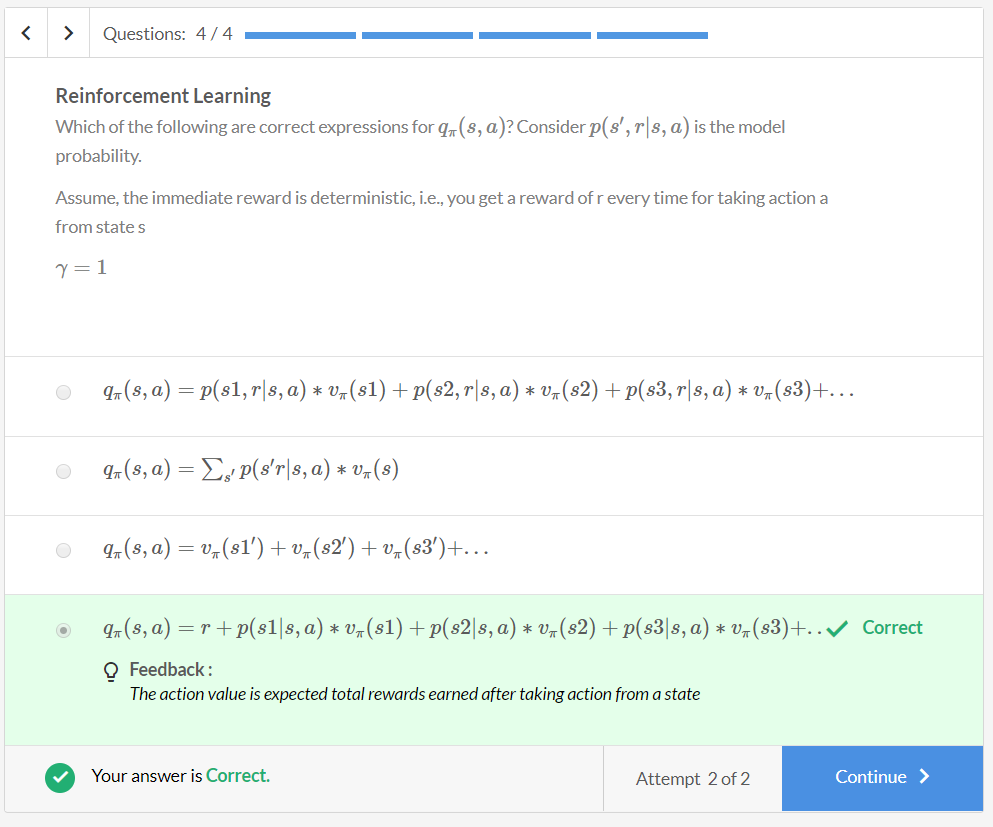
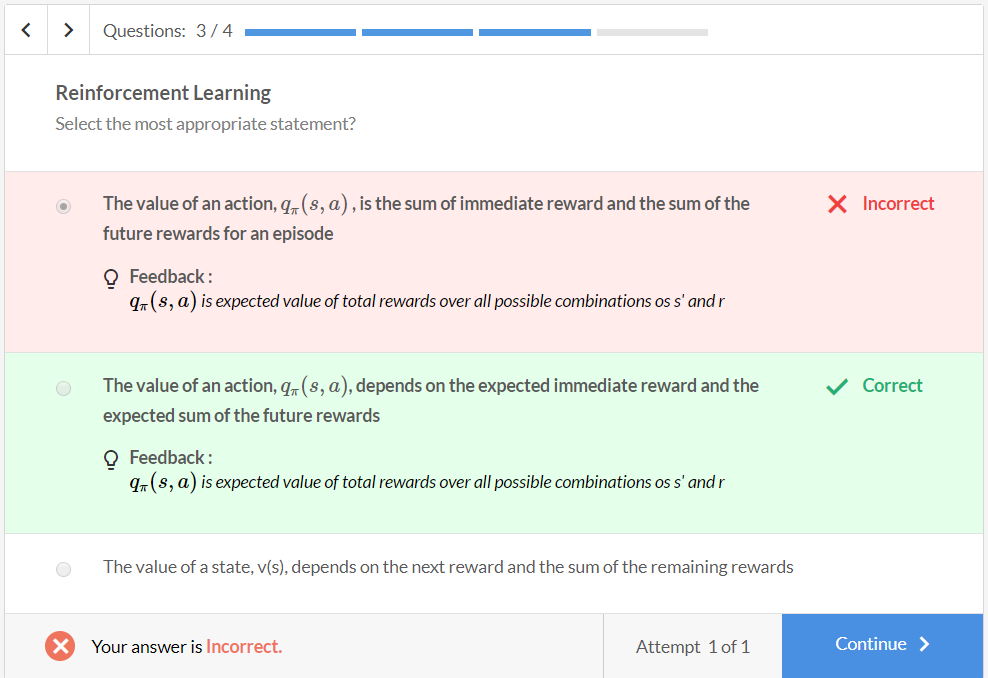


1. **Action Value Function**

To summarise:

* **q-value**- an estimate of the value of **an action**, that can be performed, in a given state.
* **state-value**- an estimate of the value of the state, by taking the weighted sum of the expected value of **all the actions**



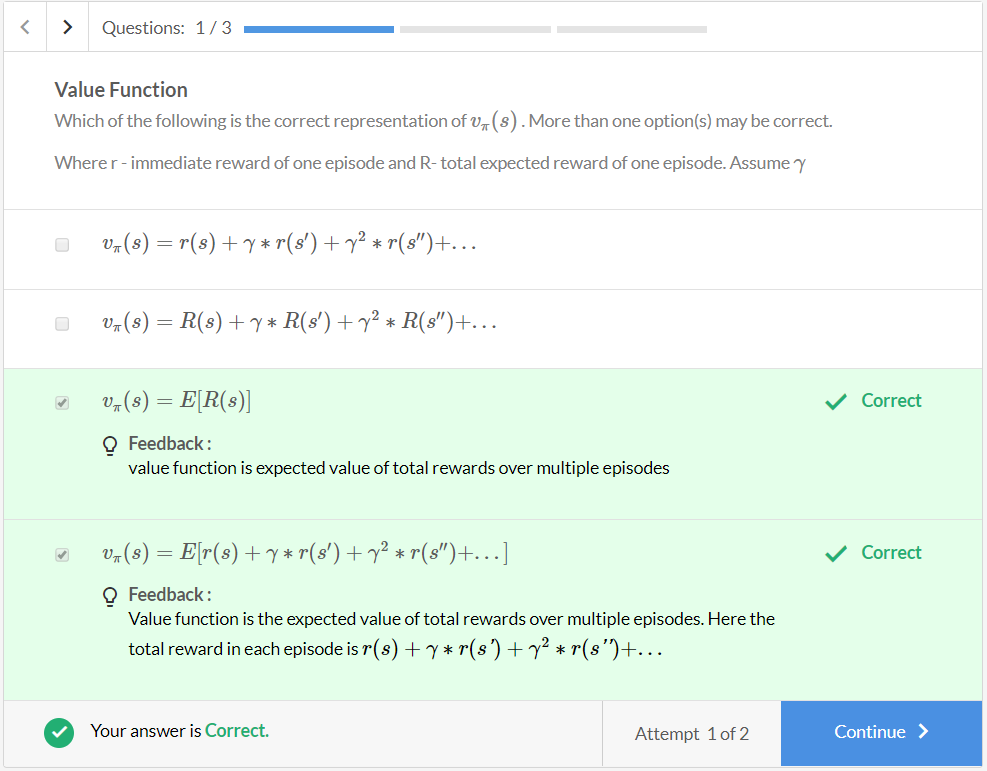
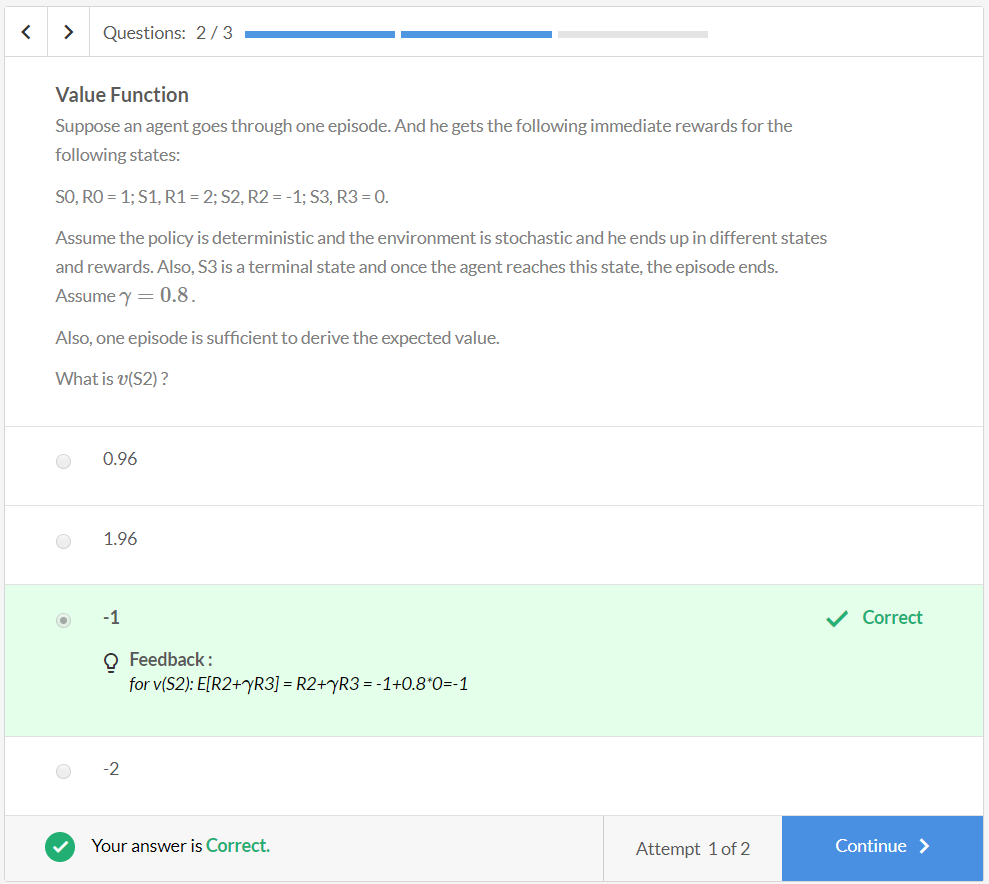


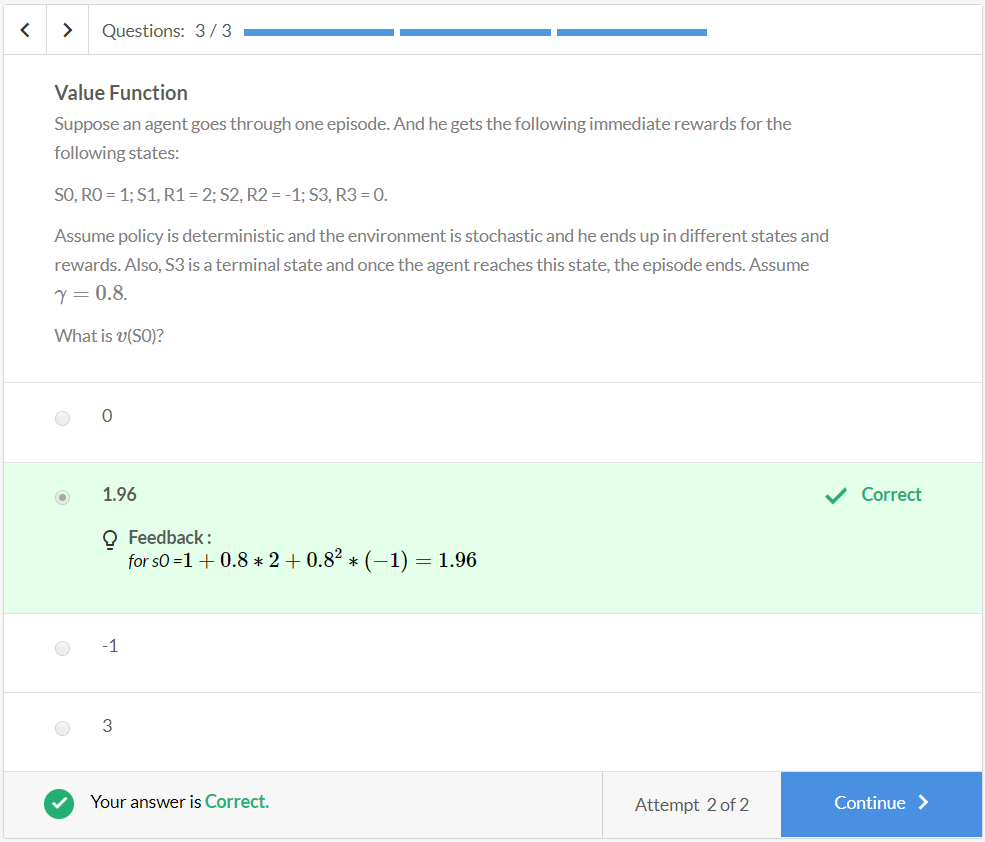
# Comprehension - Value functions

An RL agent starts from the state s and follows policy π . Answer the questions basis the RL equations of state and action value:

* vπ(s)=∑aπ(a|s)qπ(s,a)
* qπ(s,a)=∑s′∑rp(s′,r|s,a)(r+γvπ(s′))

Assume that you are doing this for one episode. The notations used in the questions are as follows: r- represents the immediate reward obtained after taking an action; R – represents the total reward obtained in one episode; s′ - represents the new state that agent lands into after taking an action a.)

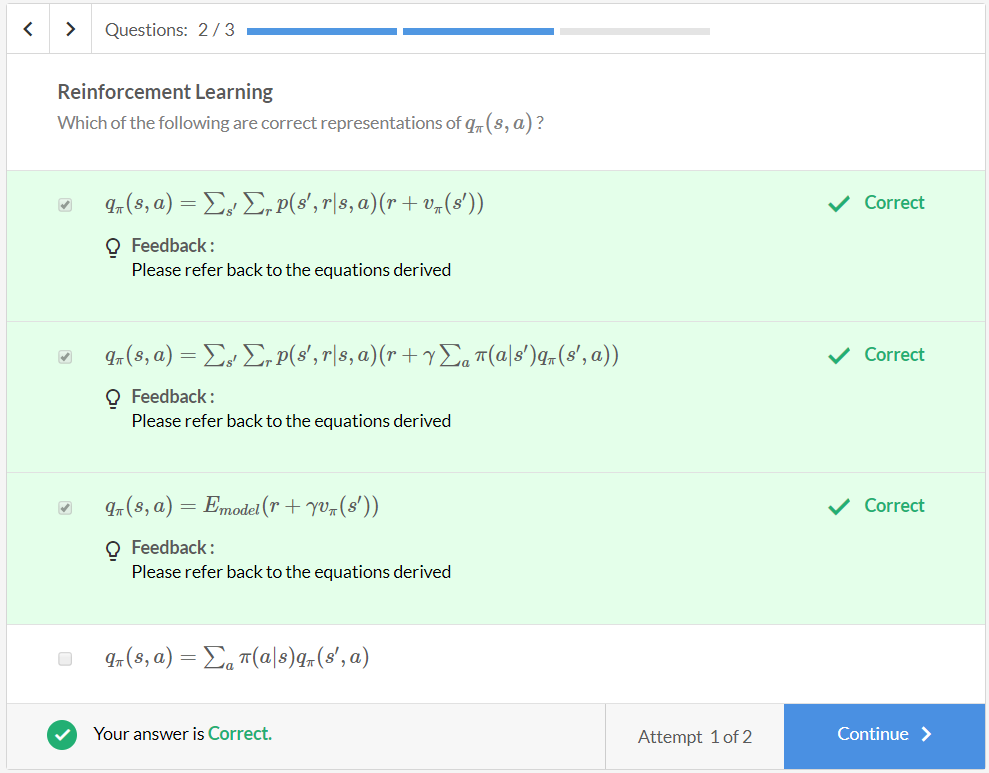
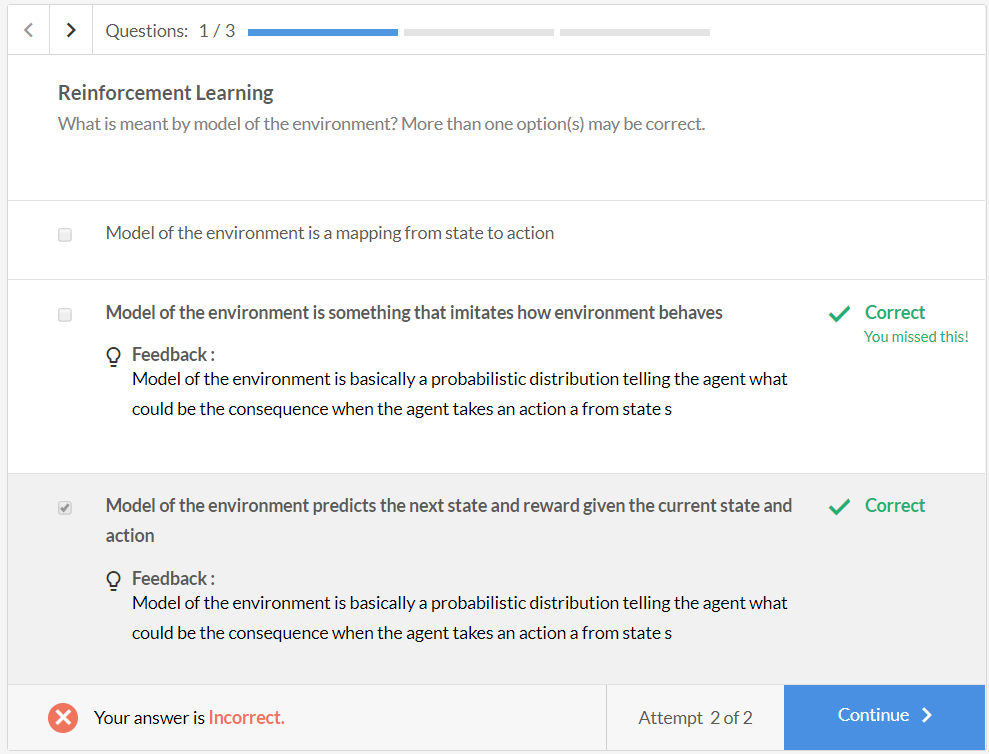


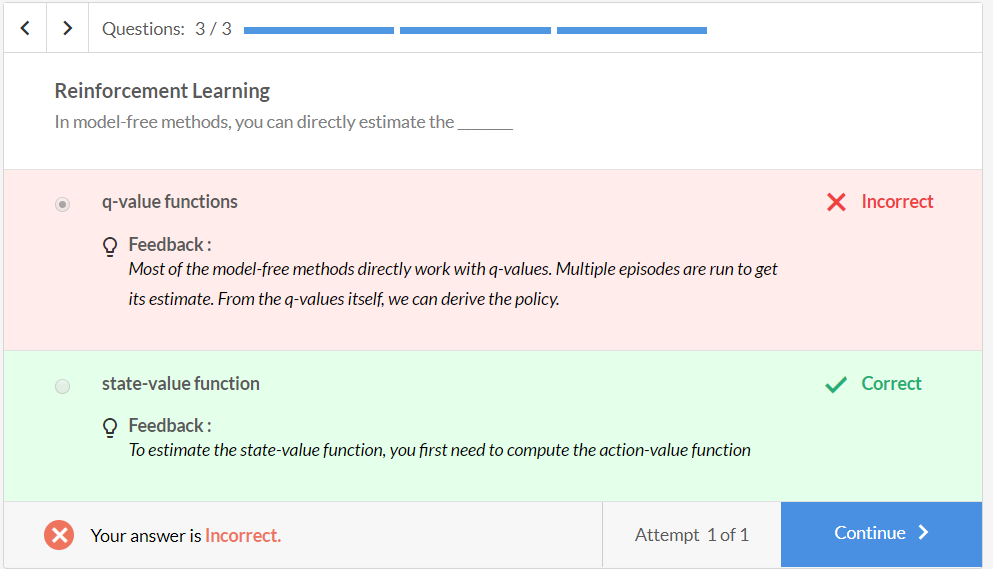
## **Model-Free Methods**

These equations can be solved if the **model of the environment**p(s′,r|s,a)**is available**. In most real-life scenarios with large state and action spaces, the model of the environment is not available (i.e. model-free methods). In such cases, you can compute the q-function using the fact that it is the expected value of the total reward:

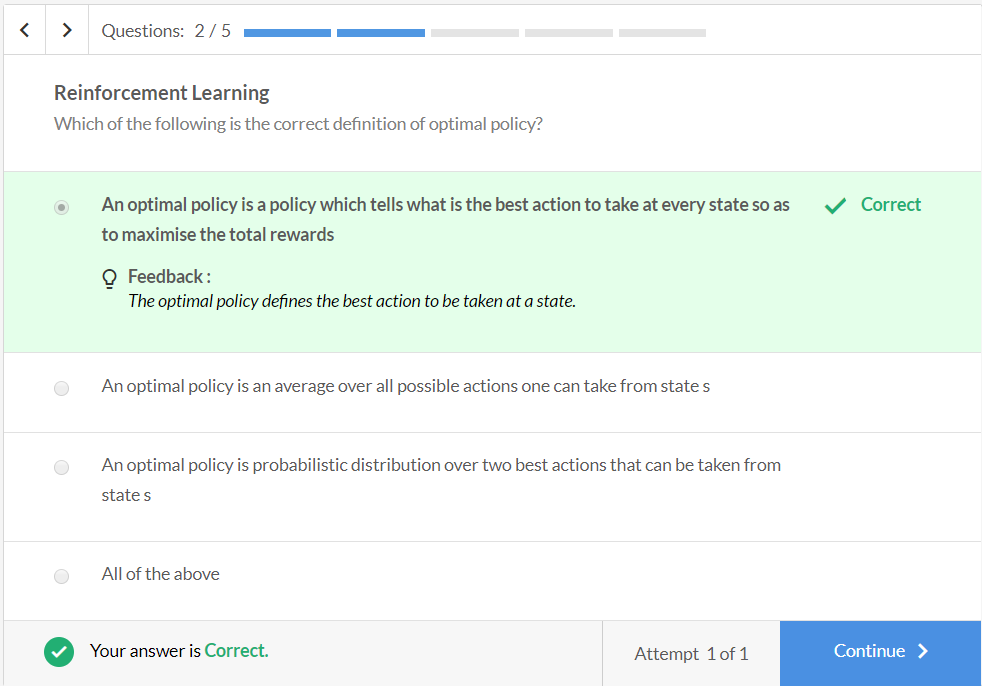
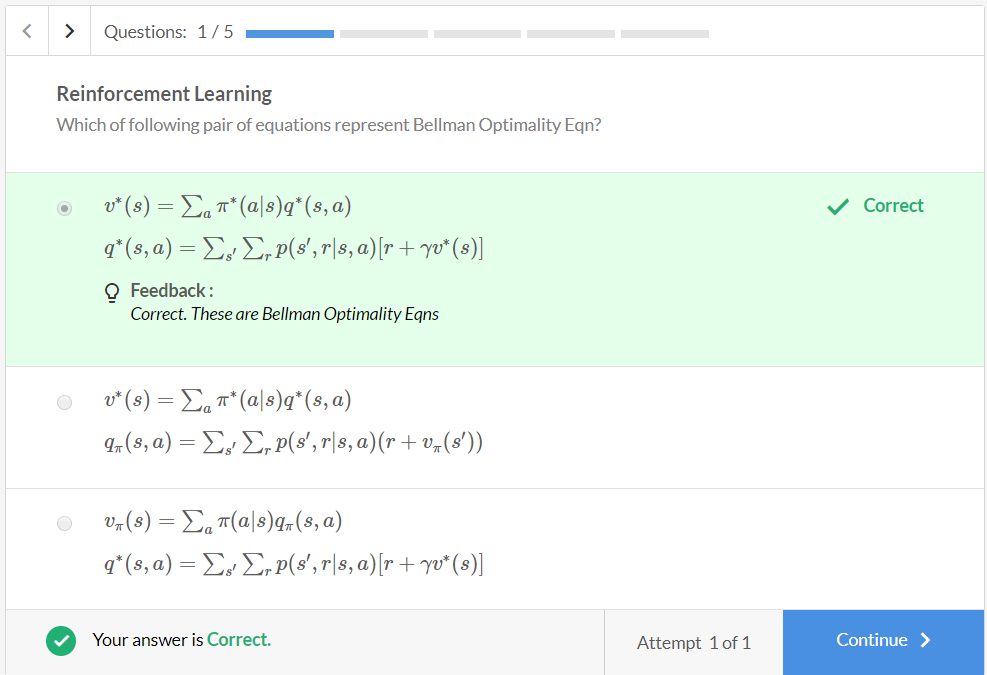
qπ(s,a)=Emodel(r+γvπ(s′))

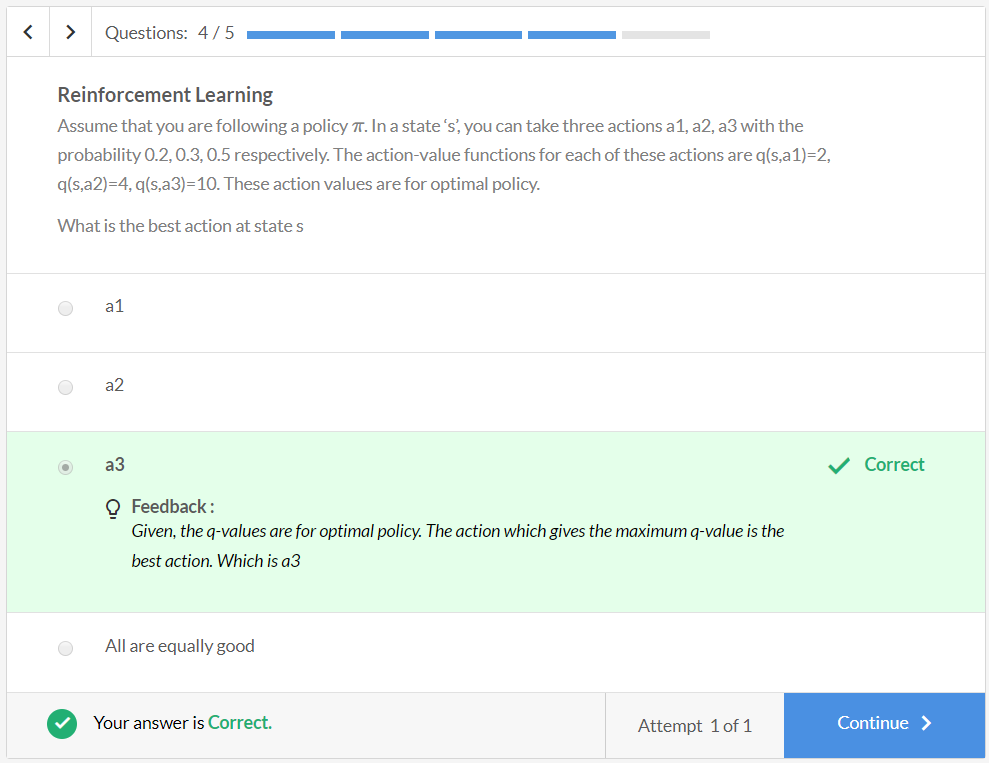
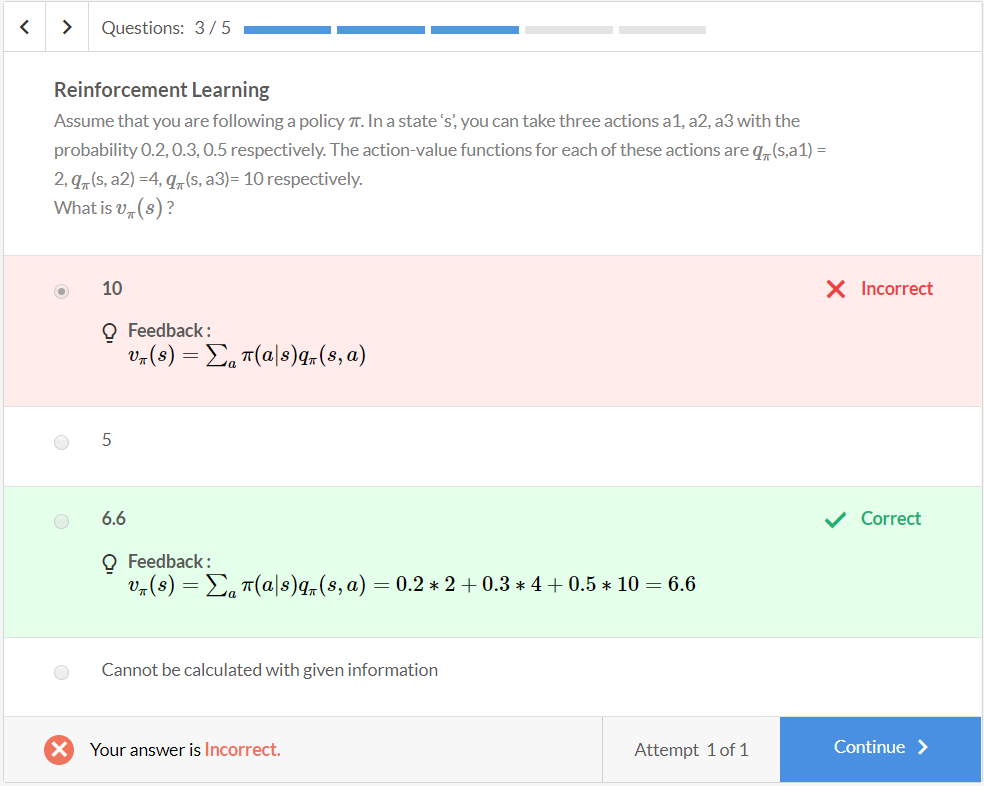
To compute the expected value, you take multiple episodes where the agent was in state ‘s’ and took the action ‘a’. For this particular state-action pair, you will get different (s', reward) pairs for different episodes. To get an estimate of the expected value of this state-action pair - qπ(s,a) - you take the average of the different rewards you get via these different episodes. We've explained how to use these equations through the cab-driver scenario.

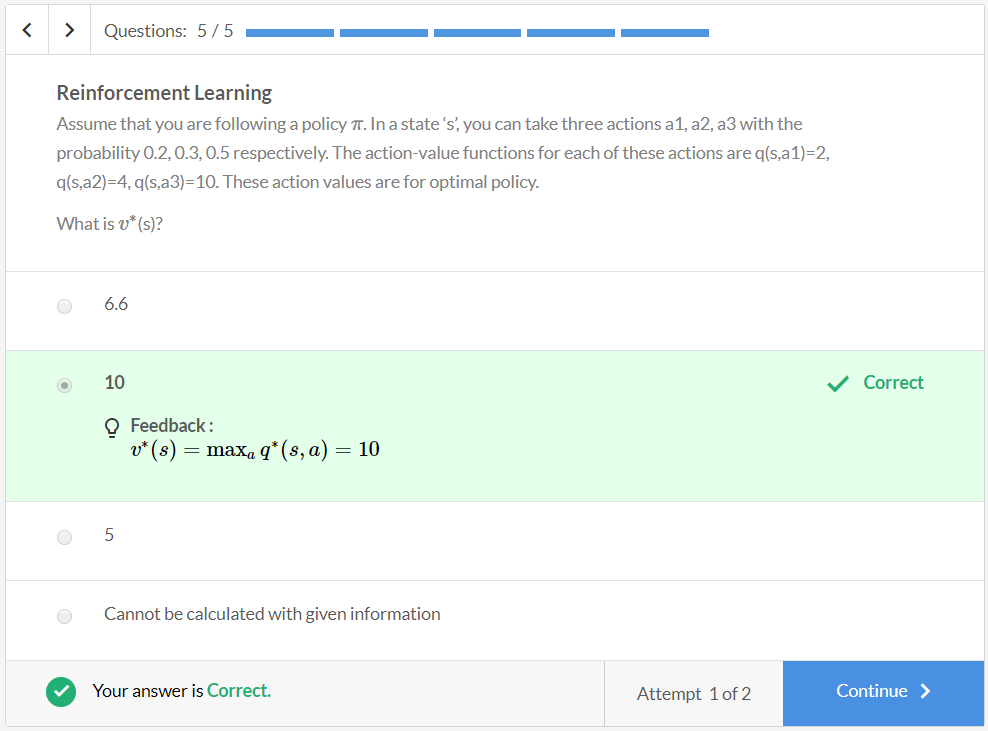




**Bellman Equations of Optimality**





Now, you’ve already picked the best value for q∗(s,a) for the policy. The right side term can be maximum only if you set the probability of picking the action for which q∗(s,a) is maximum as 1 and rest all to 0.

For reference, sample values are picked for a random variable from [3456] with a probability distribution as [0.10.30.40.2] i.e. the probability of picking 3 is 0.1, 4 is 0.3, 5 is 0.4 and 6 is 0.2. The expected value of random variable E[X] is calculated as 0.1\*3 + 0.3\*4 + 0.4\*5 + 0.2\*6 = 4.7. But you need to get the maximum value among all the values. For that, you would set the probability value for 6 as 1 and for others as zero. The same idea applies to state and action-value functions. Since v(s’) = 6 is highest out of all the v(s’) values, you make its probability = 1.

We can define optimal policy as:  
 π∗(a∗|s)={1a∗=argmaxaq∗(s,a)0otherwise}

**Note:**In this module, whenever the term optimal policy is used, it’ll by default mean a deterministic policy.

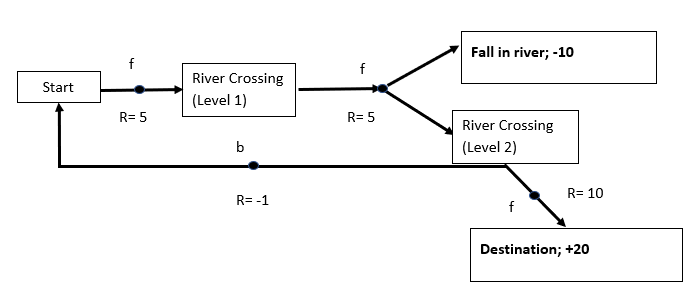
To summarise, the state-value and action-value functions for the optimal policy can be defined as:

* v∗(s)=∑aπ∗(a|s)q∗(s,a)
* q∗(s,a)=∑s′∑rp(s′,r|s,a)[r+γv∗(s)]

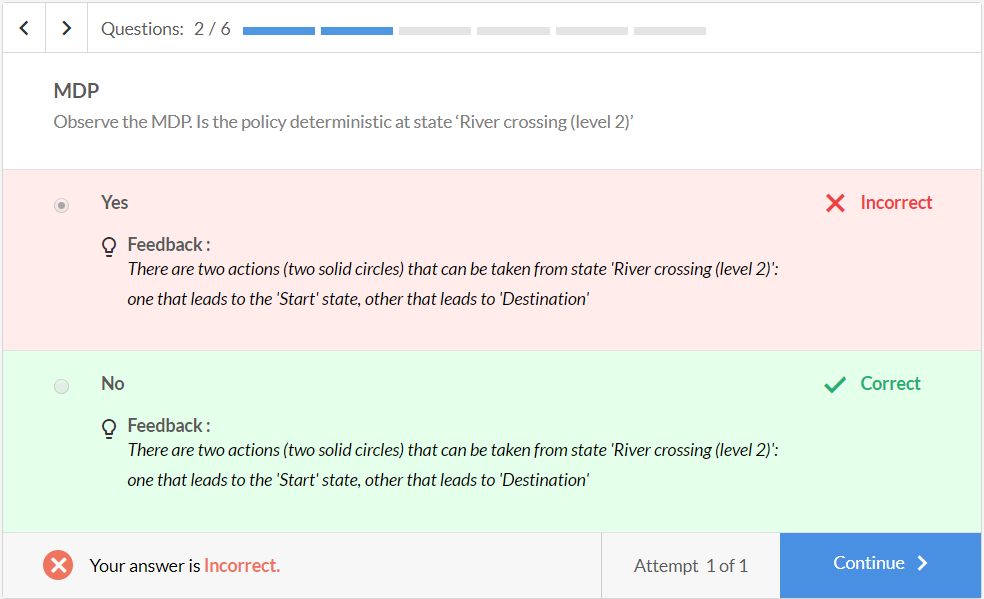
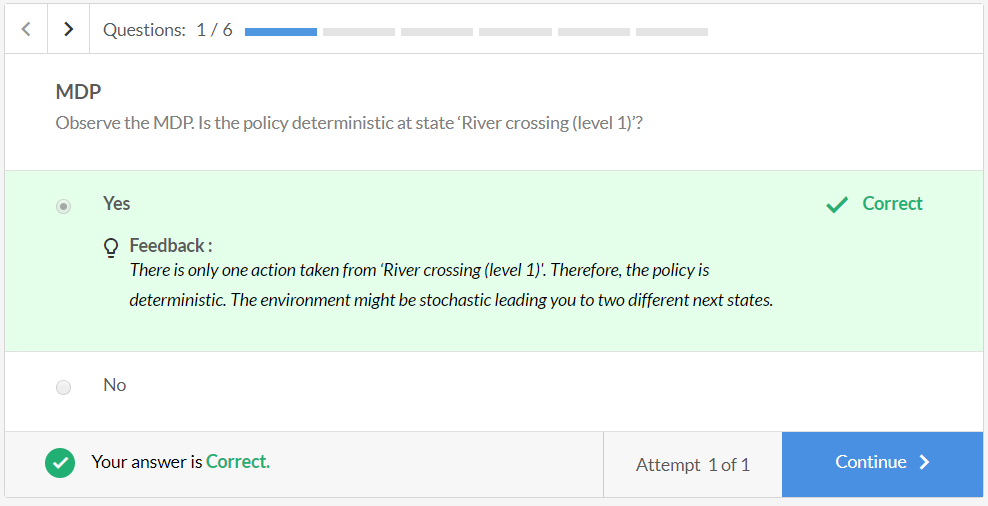
These equations relate the optimal policy to the optimal state and optimal action-value equations. These equations are popularly known as **the Bellman Equations of Optimality.**

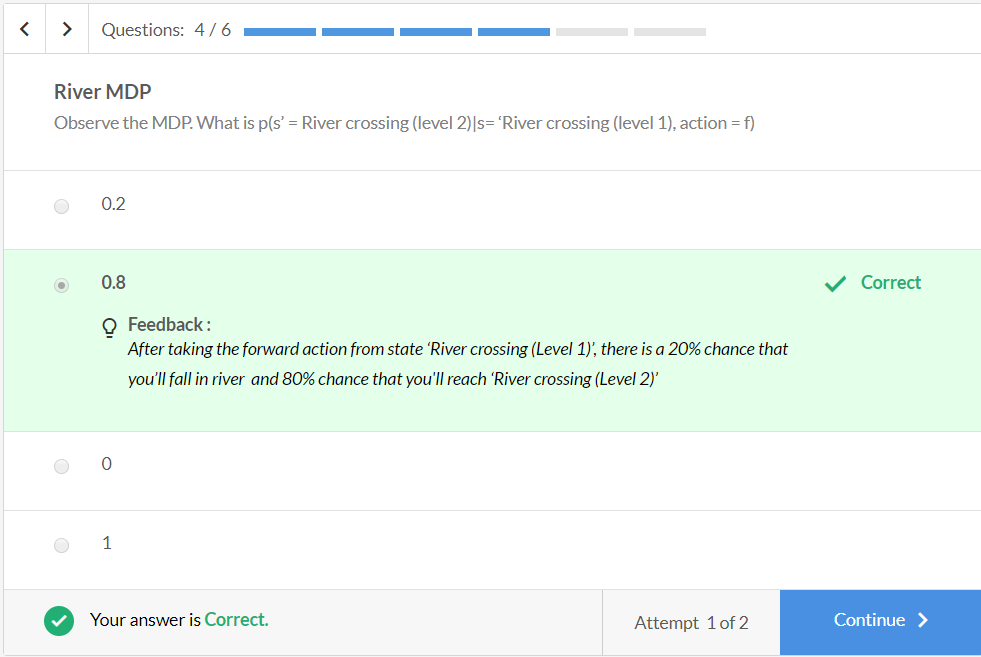
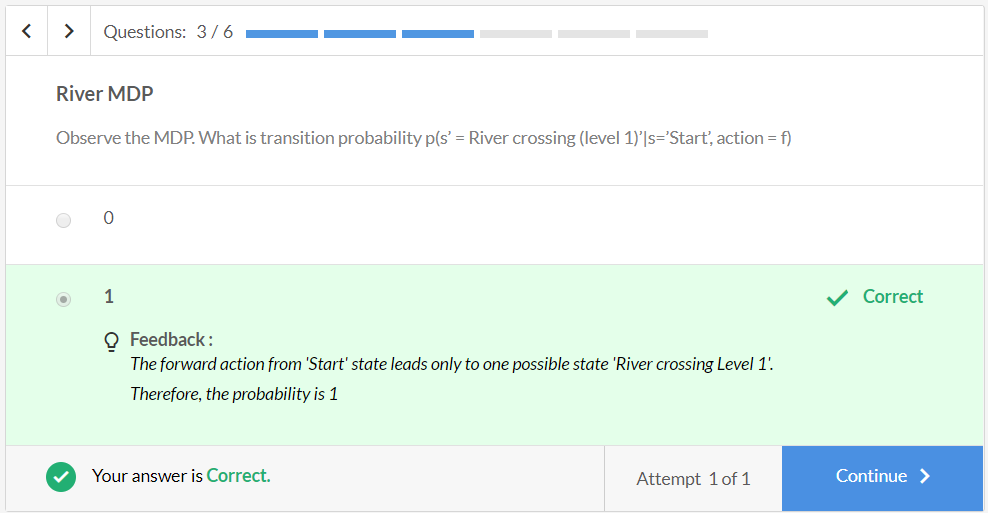
**Comprehension**

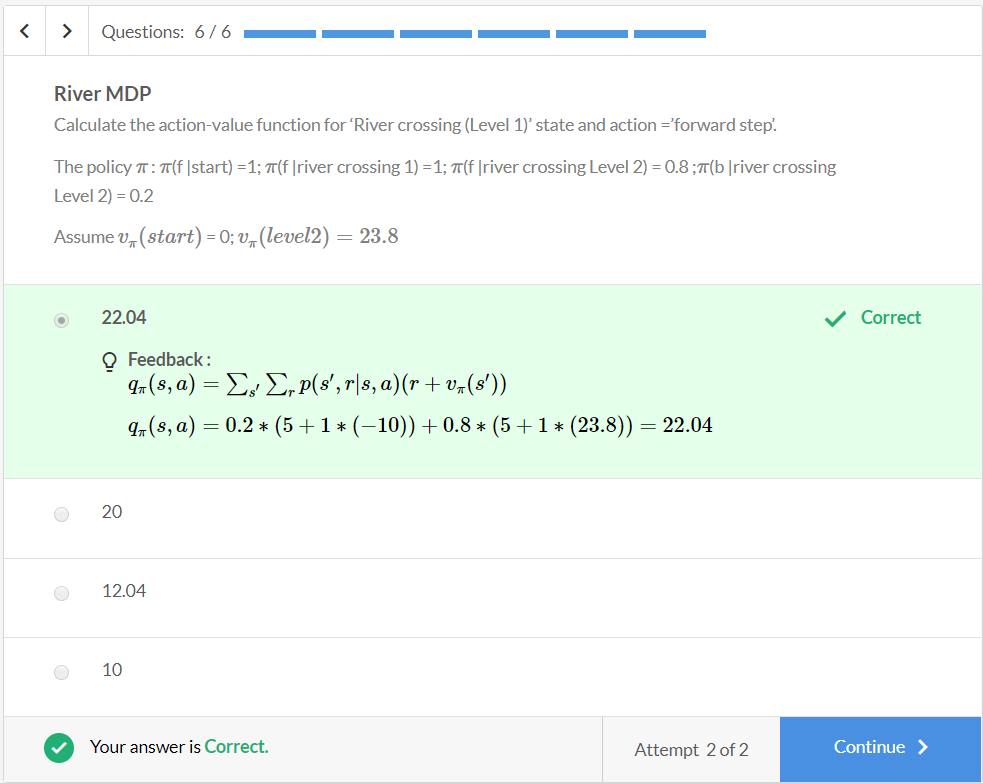
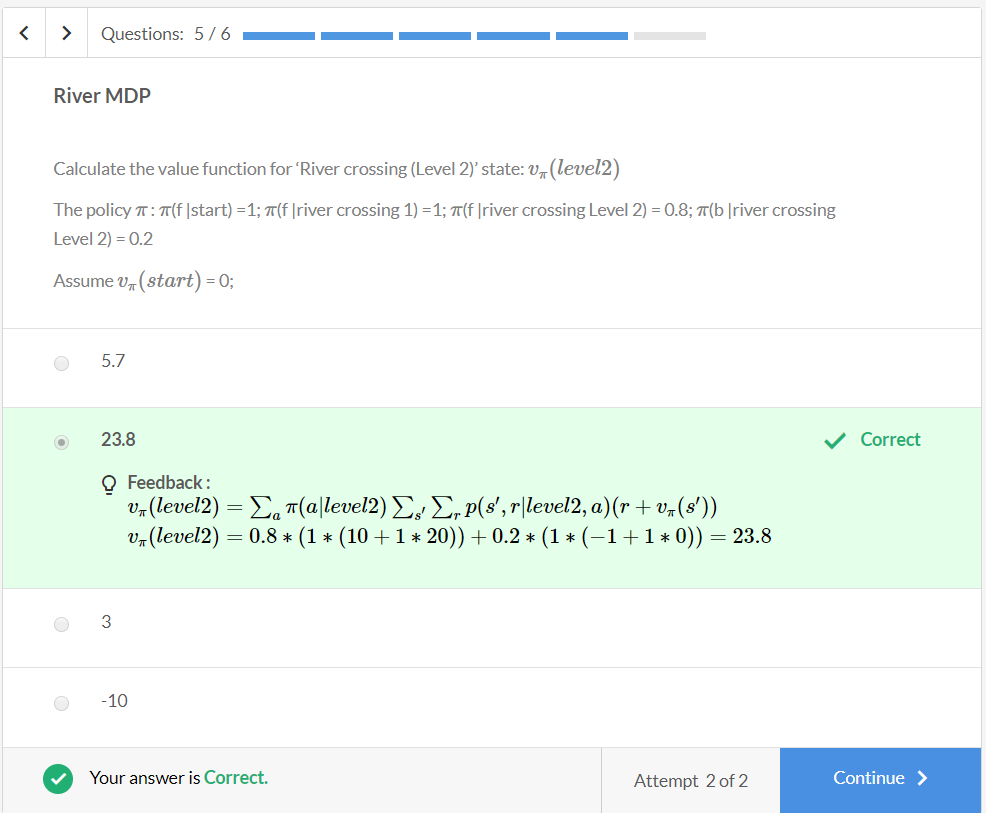
Consider the following river crossing learning activity as an MDP. Consider this as a (very) small version of Mario game, where the agent is trying to learn to reach the destination by accumulating as many rewards as possible on its way.



* There are 5 **states** in this MDP: Start, River Crossing (Level 1), River Crossing (Level 2), Fall in River, Destination
* Two **terminal states**: Fall in river and Destination. Once, the agent reaches any of these states, the episode will end (no action is taken from these states). Value of these states is defined in the diagram vdestination=20 and vfall=−10
* **Actions:** small solid circles shown in the diagram are the only possible actions an agent can take from a state. One is forward action, represented by *f*, another is backward action, represented by *b*
* **Immediate reward** for each action taken from a state is shown in the diagram. Example: from the state 'River crossing (Level 2)' and taking the ‘backward’ action will result in an immediate reward of -1.
* **Transition Probability:**after taking the forward action from state ‘River crossing (Level 1)’, there is a 20% chance that you’ll fall in the river. For the rest of (state-action) pairs shown, it is 1.
* **Discount Factor:**γ =1.



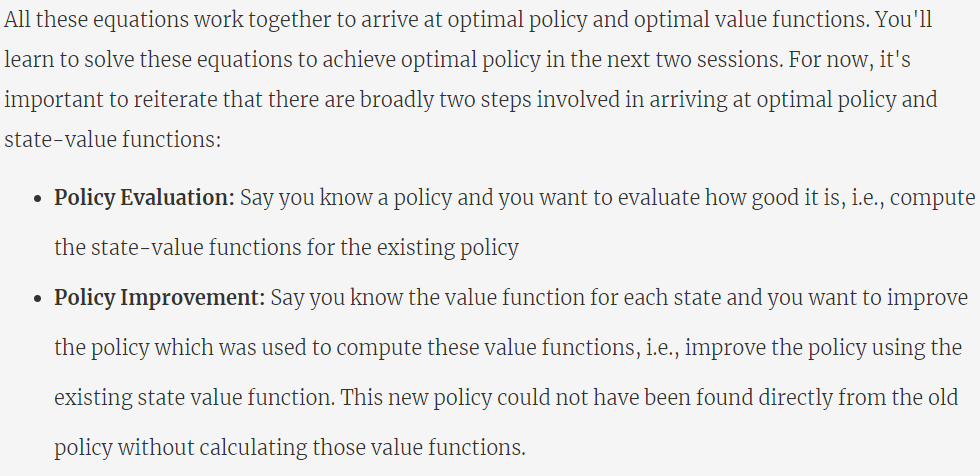


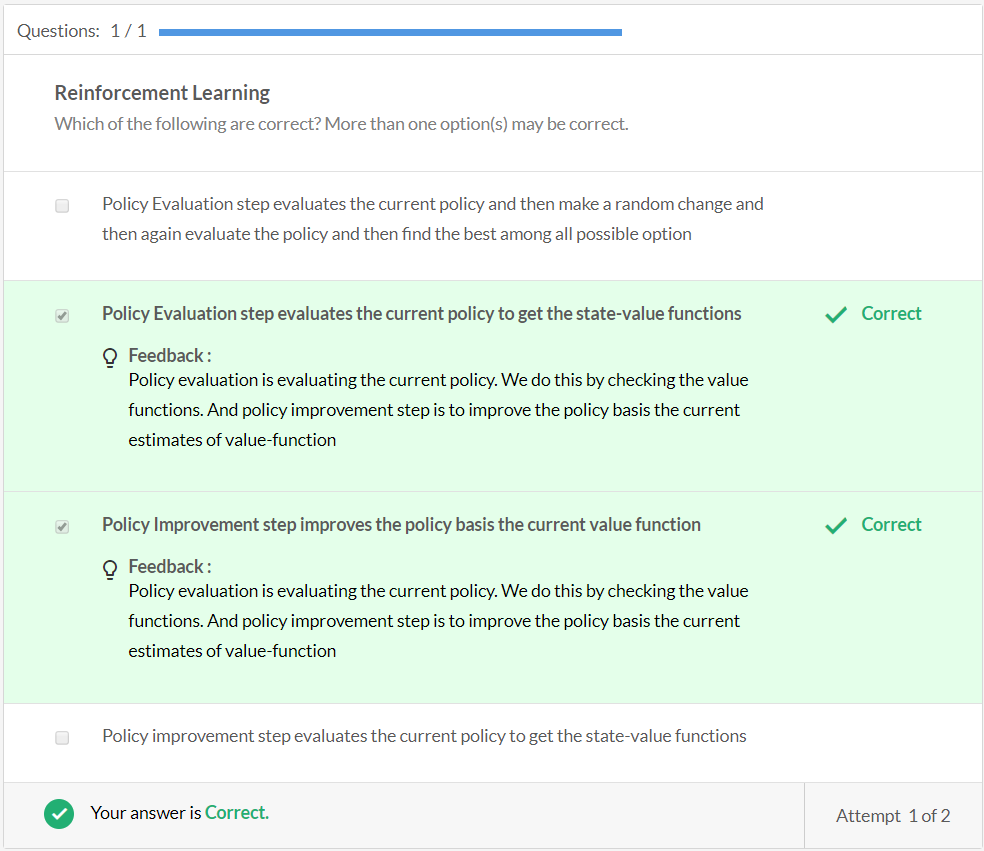


So, there are four basic equations.

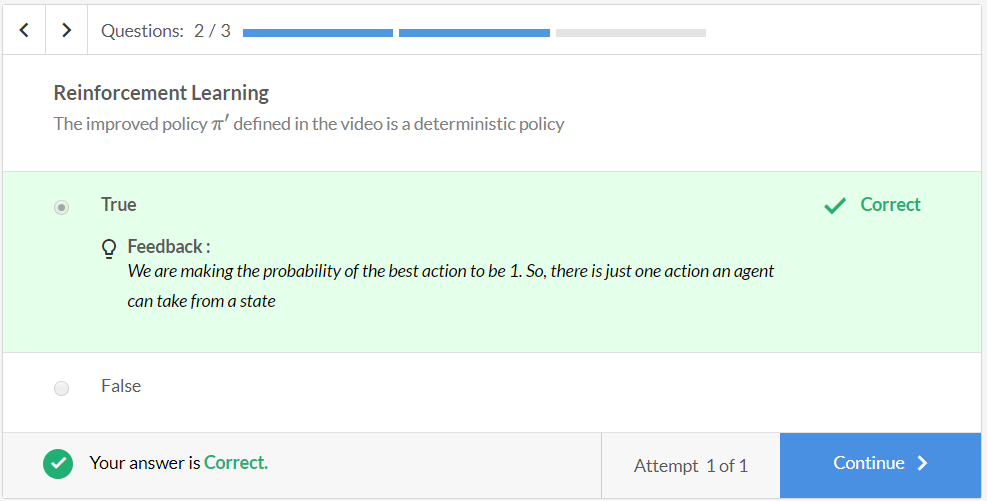
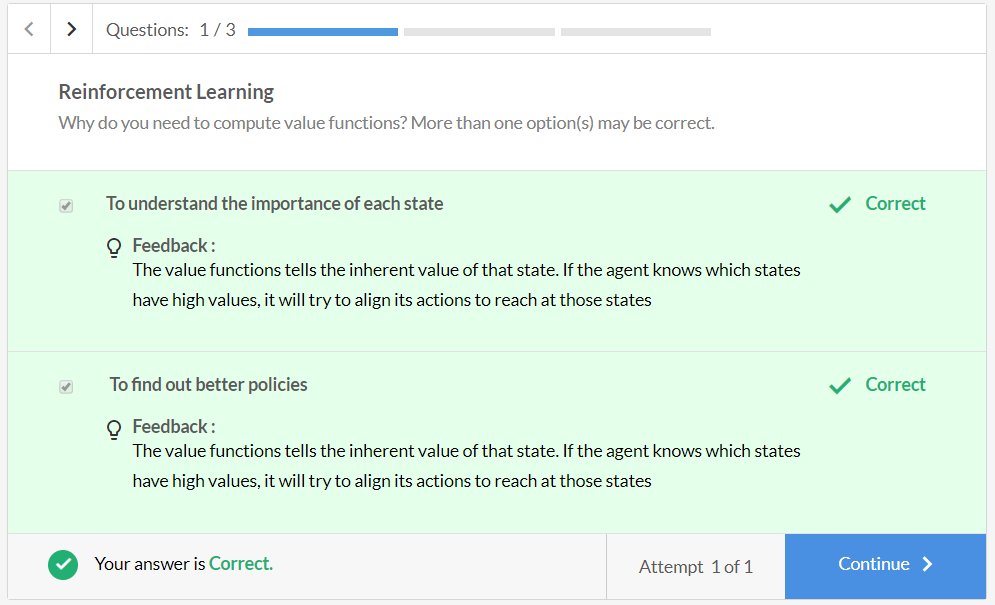
* State-value and action-value functions for policy π (These equations are known as **Bellman Expectation Equations)**
  + vπ(s)=∑aπ(a|s)qπ(s,a)
  + qπ(s,a)=∑s′∑rp(s′,r|s,a)(r+vπ(s′))
* Optimal state-value and optimal action-value function (**Bellman Optimality Equations)**
  + v∗(s)=∑aπ∗(a|s)q∗(s,a)
  + q∗(s,a)=∑s′∑rp(s′,r|s,a)[r+γv∗(s)]

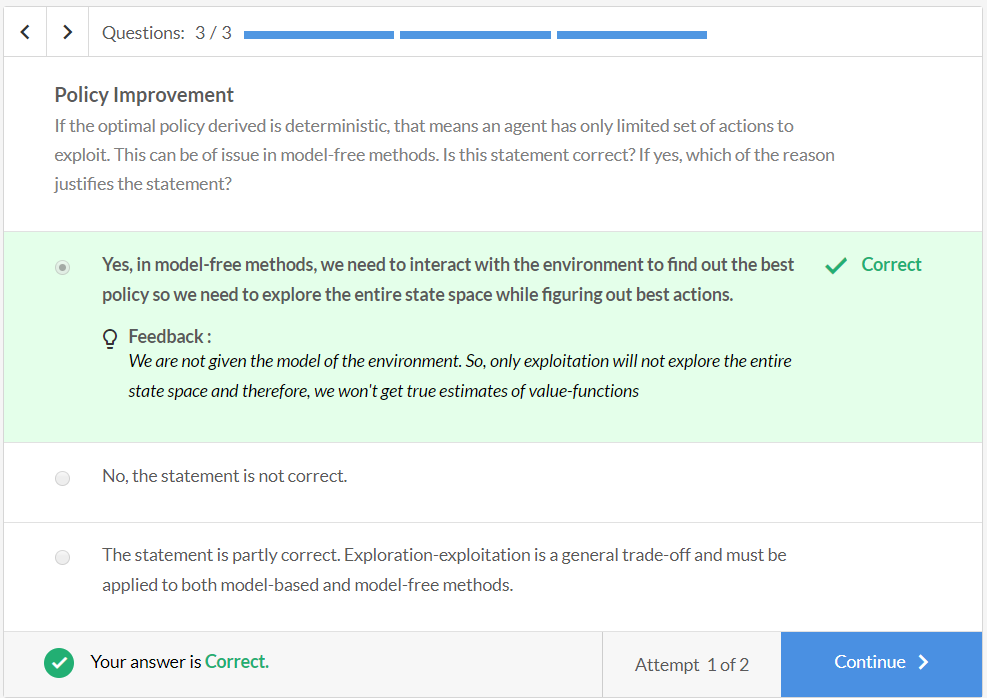
where, p(s′,r|s,a) is the model of the environment.





# Policy Improvement





**Summary**

In this session, you learnt the basic equations to solve a Reinforcement Learning problem.

There are four basic equations.

* State-value and action-value function for policy π (These equations are known as **Bellman Expectation Equations)**
  + vπ(s)=∑aπ(a|s)qπ(s,a)
  + qπ(s,a)=∑s′∑rp(s′,r|s,a)(r+vπ(s′))
* Optimal state-value and action-value function (**Bellman Optimality Equations)**
  + v∗(s)=∑aπ∗(a|s)q∗(s,a)
  + q∗(s,a)=∑s′∑rp(s′,r|s,a)[r+γv∗(s)]

Where, p(s′,r|s,a) is the model of the environment.

You learnt to calculate **value function**(of a state) and also **q-value**(of a state-action pair).You also learnt how these two terms can be related to each other for a given policy. You also learnt how to improve a policy and ultimately arrive at an optimal policy.

**Graded Questions**

