Reinforcement Learning (RL) is the field of machine learning in which an agent (i.e. the software being trained) learns to take actions to maximise some cumulative reward.

Some classical examples of RL are driverless cars, game playing agents (Chess, Go, etc.), mechanical robots in factories/warehouses etc.

**RL is a learning problem where an agent is trying to learn from its environment by taking an action and understanding the consequences; in order to achieve an objective**

**Automobile:**

[Tesla's](https://www.autotrader.com/tesla-cars.jsp)[Autopilot](https://www.tesla.com/autopilot) , Alphabet's [Waymo](https://waymo.com/), [Ford's](https://corporate.ford.com/innovation/autonomous-2021.html) self-driving car are trying for Autonomus cars.

**Robotics:**

In Robotics - finding defects in objects, carrying an object from one place to other ex: [Fanuc](https://www.technologyreview.com/s/601045/this-factory-robot-learns-a-new-job-overnight/) has deployed a robot that uses RL to pick a device from one box and put it in a container.

**Finance:**

RL is turning out to be a robust tool for evaluating trading strategies. Many companies are leveraging the "Q-Learning" algorithm of RL with the simple objective of maximising the "rewards" i.e. profits.

**Law of Effect:** Edward L. Thorndike who talked about **learning by trial and error**.

“Responses that produce a satisfying effect in a particular situation become more likely to occur again in that situation, and responses that produce a discomforting effect become less likely to occur again in that situation.”

Examples of RL :

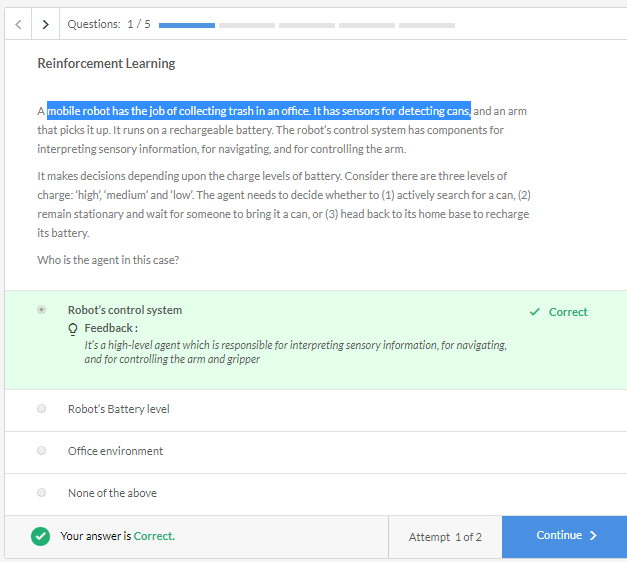
1. Solving a Maze problem (video games)
2. Managing investment portfolio
3. Deciding among the pickup requests in cab –scervice scenario
4. Process control System.

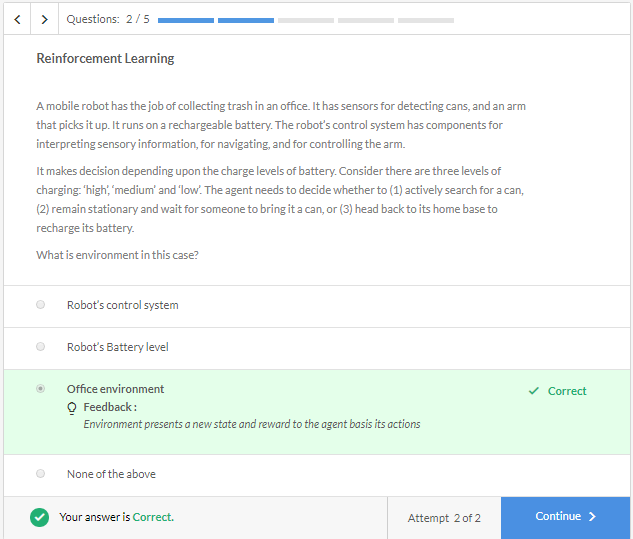
Agent is trying to solve the problem in the environment.

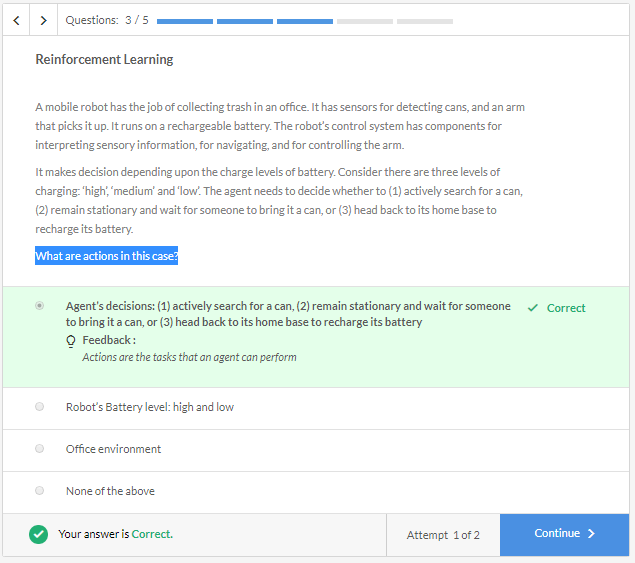
Agent: The **agent** is any robot that is trying to learn the task

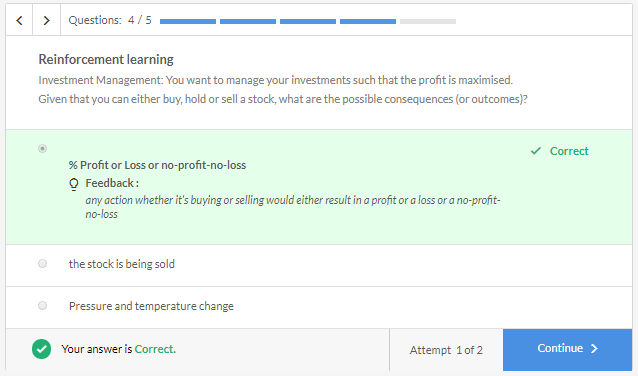
While the **environment** is the world around it that gives it the feedback.

Agent can observe the Environment and take actions . But the Consequence of the Action is not in control of Agent. It can only observe and take action next time . but cant change the consequences of the Already taken action.









**RL is a learning problem where an agent is trying to learn from its environment by taking an action and understanding the consequences; in order to achieve an objective.**

An engineer is learning to design a humanoid robot to stand and his objective is to make it stand. He can do that by turning some joints.

Here Agent is : Engineer

Environment : Robot

Actions : Turning some Joints.

Objective : Robot to Stand.

consequences : could be robot falling or standing up.

**Agent** has only control of the Actions it takes. **Environment** tells whether the Action taken by Agent is good or bad by a **Reward**. Reward only tells how well the task is performed. It doesn’t guarantee that this is the best action.

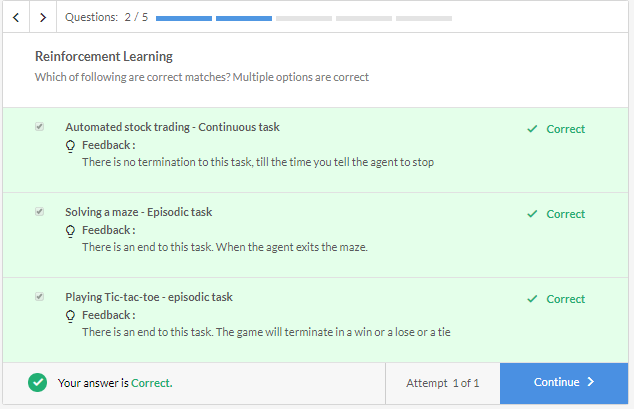
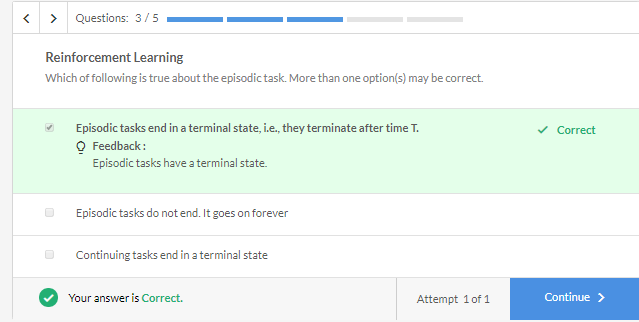
Reward is a Weak signal which indicates whether the Agent is taking decisions in right decision or not.

Unlike supervised learning, which classifies each observation as 'right' or 'wrong'; reward in reinforcement learning is just a number indicating how well you are performing the action.

**Two types of tasks:**

* **Continuous** - tasks that do not have a definite end - e.g. learning to walk, controlling a chemical plant, driving a car
* **Episodic tasks** - tasks that have a definite end - e.g. most games (videos games, Chess, Ludo) etc. are episodic since at the end of the game the agent either wins or loses.

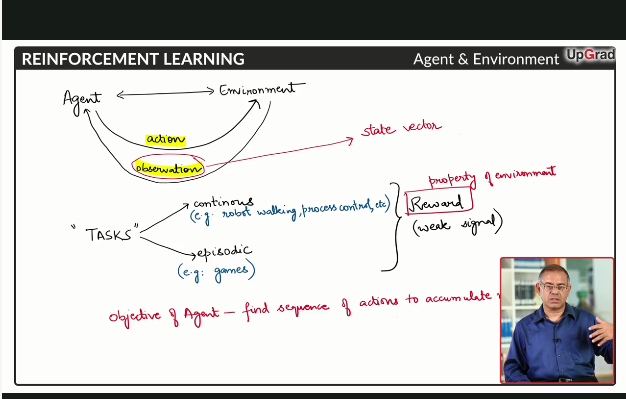
Questions:

1. Rewards are under the control of the agent, i.e., after taking an action, it can modify the rewards obtained - False
2. 
3. 

So, a **state** is a representation of the environment at any point in time. The environment will give all the signals, but how **relevant** those signals are for the agent to take an action, is what you’ve to decide. You can consider state vector as a list of features that help the agent to take an action. For each RL problem, state vector would be different.

Only information which is required to take necessary Action will be part of the state.

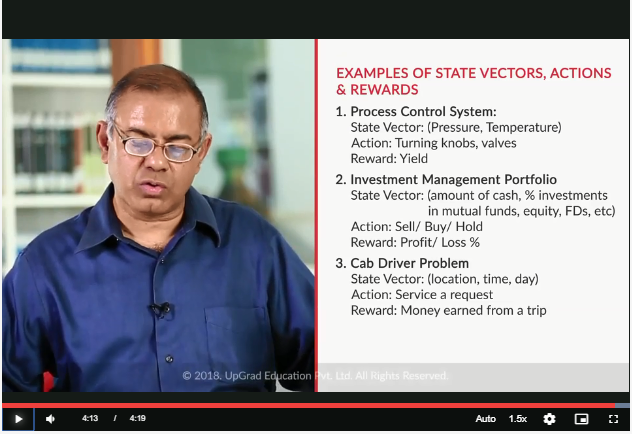
State is **your** representation of the environment. Perhaps the environment would have a lot of things, but the state that you want to take will determine which parameters in the environment really matter to you.



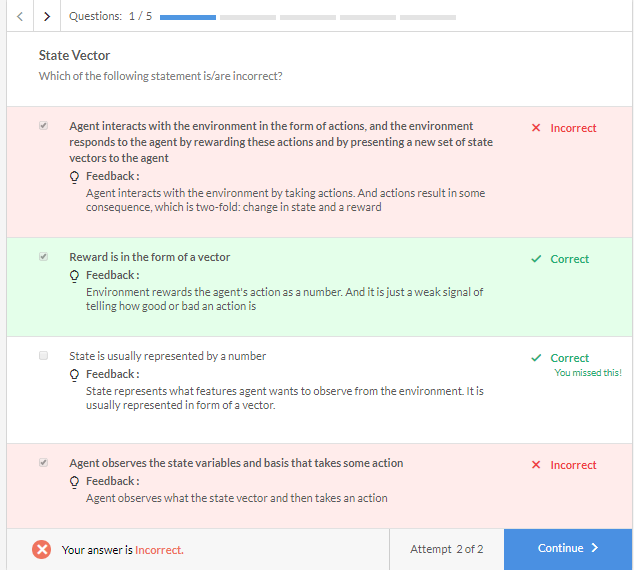
Important point is based on Task , state vector is modified.

Once the Agent takes action , Then the state vector changes.

Examples of State Vectors, Actions and Rewards:



Questions:



Netflix wants to customize the home page for each of the customer. It wants to show very relevant cover photo for each movie or series it recommends. A person watching a lot of horror movies, will be interested if he is shown some intense scene from the movie as a cover photo.

Objective of RL Agent for continuous and Episodic Tasks:

Take an episodic task, say a game of tic-tac-toe. How will you calculate the reward for each **O (or X)** you marked? You’ll get a reward after you win or lose the game. What is the agent’s end-objective in this case?

Similarly, for a continuing task, say for stock market trading, you can define your reward as how much profit you earned in a month or a day. There’s no end to continuing tasks, so how are you going to parametrize your objective?

* The objective of **episodic tasks** is to find such a sequence of actions that will make the majority of episodes successful.
* For **continuing tasks**, break it into multiple episodes and then find out actions that maximise the average rewards earned from those episodes. Ex: cab driver , We will break it for each day . Portfolio Management every Month.

Important Points about Objective :

* The objective of the RL agent is pre-decided. It doesn't depend in what state agent is in.
* RL objective is not decided by the environment
* The objective of RL agent in an episodic task is to find such sequence of actions that will make a majority of episodes successful.

