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/" \t "_blank), [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 :

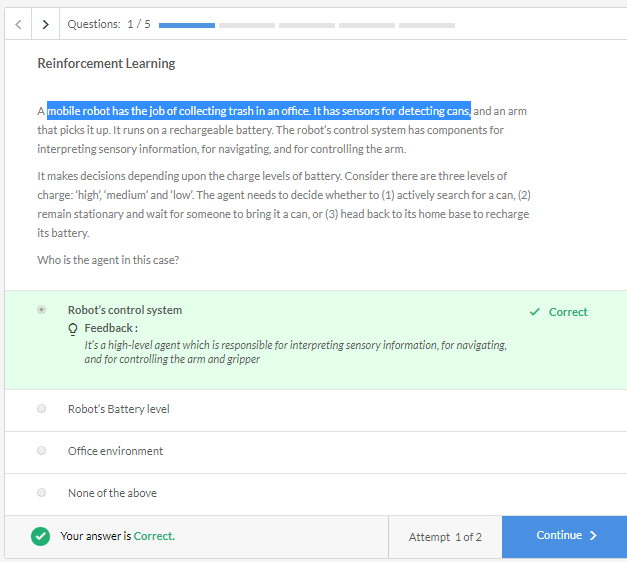
1. Solving a Maze problem (video games)
2. Managing investment portfolio
3. Deciding among the pickup requests in cab –scervice scenario
4. Process control System.

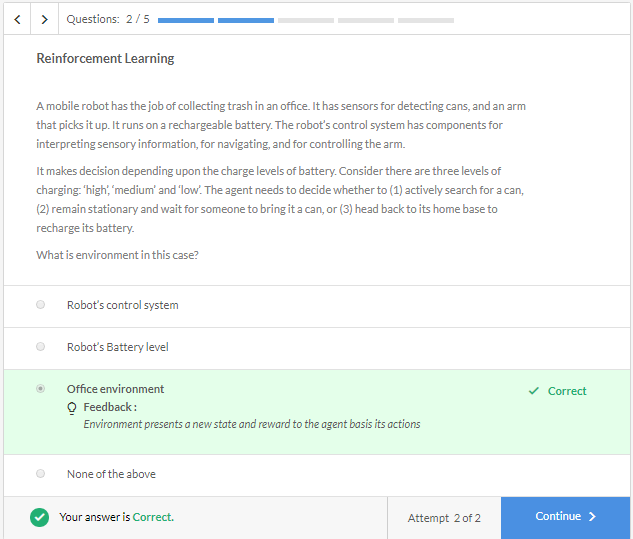
Agent is trying to solve the problem in the environment.

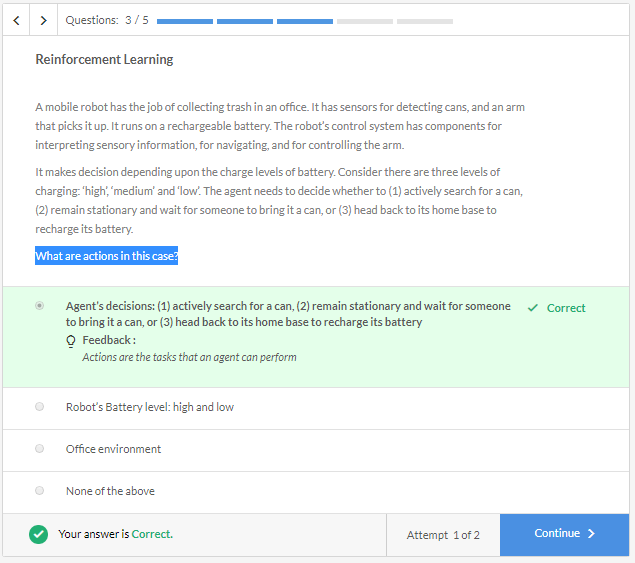
Agent: The **agent** is any robot that is trying to learn the task

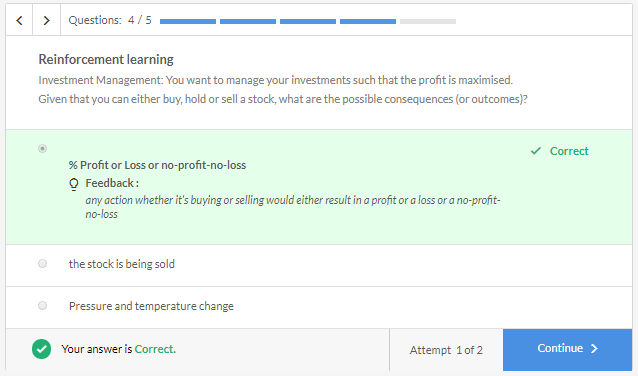
While the **environment** is the world around it that gives it the feedback.

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.

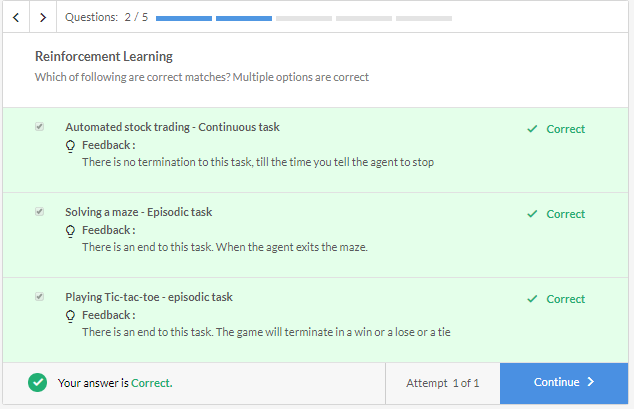
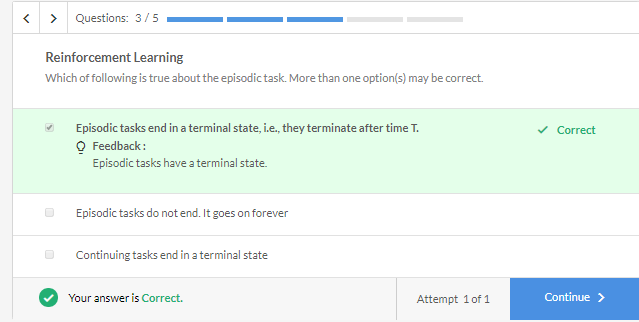
Reward is a Weak signal which indicates whether the Agent is taking decisions in right decision or not.

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.

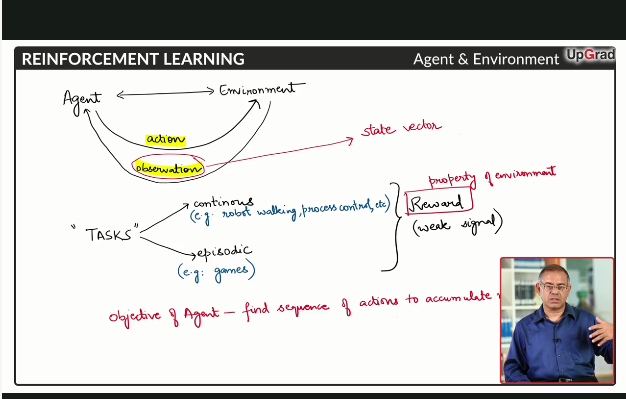
Questions:

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.

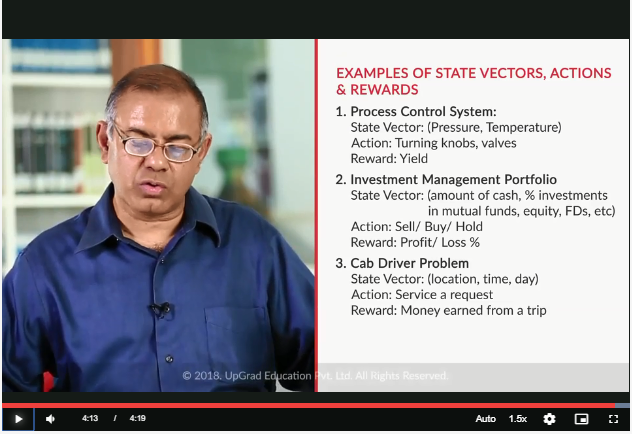
State is **your** representation of the environment. Perhaps the environment would have a lot of things, but the state that you want to take will determine which parameters in the environment really matter to you.



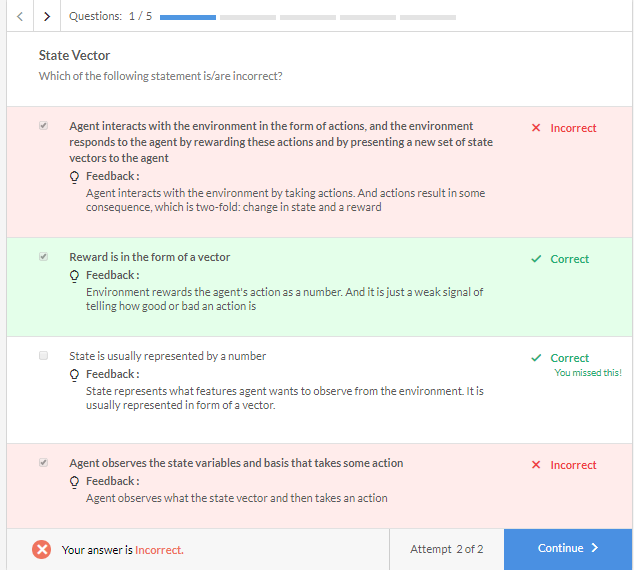
Important point is based on Task , state vector is modified.

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:

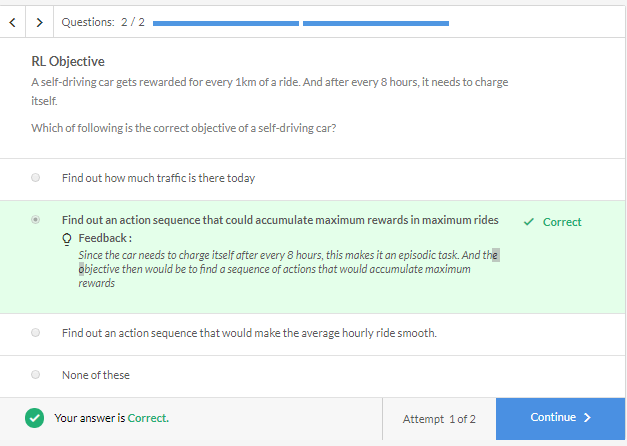
Take an episodic task, say a game of tic-tac-toe. How will you calculate the reward for each **O (or X)** you marked? You’ll get a reward after you win or lose the game. What is the agent’s end-objective in this case?

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.

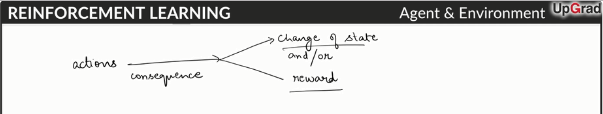
Important Points about Objective :

* 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.



**Actions :**

* An action leads to a reward and a change in state.
* The objective of the RL agent is to find the sequence of actions to maximise overall rewards.



The consequence is two-fold, one in form of **rewards**, other in form of **change of state**.

An Agent need to remember what Action it took and what consequence it lead to. This memory of action and consequence is called **knowledge-base** or **history** for an RL agent.

Instead of storing all the Action – consequence Knowledge history . We can create a Mathematical Model or Function Approximator .

In Smaller problems, We can actually Build the Knowledge base history but in actual real world problems we can’t visit all the states and can’t create knowledge base to take best Action.

# Policy :

Policy is the set of rules which help the agent to take best Action in a given State such that agent can maximise the reward in the long run.

* **Deterministic policy**: π(s)→a
* **Probabilistic policy:** π(a|s) probability of taking action a given state s

A probabilistic policy becomes deterministic when π(a|s)=1

* Deterministic means that there is exactly one action which we can take at a given state.
* Probabilistic or stochastic come into picture when we can take more than one actions at a given state. This is more realistic case. We figure out a distribution over various actions which we can take in given state. Environment is dynamic.

**Policy is also A mapping from state that helps the agent to figure out what action needs to be taken**

# Exploration vs Exploitation:

Stochastic /probabilistic policy is preferred over deterministic probability. We can explain this by understanding Exploration /Exploitation.

In real world problems, what we have experience is a small set or fraction of possible actions. may be there are better possible actions if we have explored more. But doing so we may loose out the benefits of current action if our exploration doesn’t find out to be a good option .

so we should have trade-off between exploration and exploitation .

**Exploiting the current option which you already know is reaping benefits – Exploitation**

Investment Management Portfolio: Consider you know which stocks will always reap benefits. And you invest your money there. Will you always risk your money there? Will you never explore? What if there exists some stock which is reaping 200% profits?

# Markov State :

let’s say we are in one state . It doesn’t matter how we have arrived at this state. My policy is based on the current state and not the history .This particular assumption is very important. This assumption is called Markov Assumption.

The **Markov assumption** states that the current state contains all the necessary information about all the past states the agent was in and all the past actions the agent took. It assumes that the current state is sufficient for taking the next action.

This markov assumption may not hold always good. We also need to have current position plus intent of the opponent making the move. But this Markov Assumption may hold good most of the times in practical reinforcement learning.

**Markov state :**

Markov state is some function of the knowledge base that captures all relevant information from the knowledge base. And once ‘Markov state’ is known, the knowledge base can be thrown away. What action the agent needs to take next or what possible state an agent can land on given he has taken an action - all of this can be determined from the **Markov state**.

Ex:

* Flight Manoeuvre system needs current position and speed and it doesn’t need to know how it arrived to this current position .
* Investment scenario – Current portfolio and market trend will be part of state vector and is example of Markov state.
* Plant Control System – Current Value of pressure and temperature + trend (rate of change of T and P).
* Cab Booking problem - Current position + time + where the pickup /drop of next passenger . history is not required.

# **Markov Decision** **Processes** (popularly called **MDPs**)

All the process that work in accordance of markov property is called Markov Decision Process.

Decission in MDP are the actions that are taken by the Agent in the given Markov state.

All RL problems sets its ground on MDPS.

In an RL problem,

* An agent learns how to behave in an environment by taking actions
* Then observing the consequences (rewards and next state), of the action taken.
* The control objective of the agent is to learn a policy to accumulate maximum cumulative rewards over a period of time.
* All of RL problems are based on the Markov assumption: the current state contains all relevant information to take the current action.