Monte Carlo is not specific to RL. It is we to predict values using samples
We want an expected value but we don't know the distribution that's where Monte Carlo comes in handy. We only have samples, our solution is to estimate the expected value with the sample mean

prediction: find $V_\pi(s)$ given a policy π
Control: find π^*

DP did not involve gaining any experience, only on math because we know the environment.

Monte Carlo prediction, is to play a bunch of episodes, collect G samples, average

$$V_\pi(s) = \mathbb{E}[G_t \mid S_t = s] \approx \frac{1}{N}\sum_{i=1}^{N} G_{i,s}$$

$G_{i,s}$: i^{th} sample return from state s

There are some compilation to Monte Carlo
What is the value of a state not visited by our policy? We can discard them or start each episode from a random action/state, if policy is probabilistic with non-zero probability for all actions

What if we encounter the same state more than once? First visit MC or count every visit MC. They both converge to correct answer.

What if our policy results in an infinite-cycle? Related to the second comp. MC don't apply since the episode never terminates. The solution to this is to terminate after a max number of steps.

For the control part of the problem, we already know this from policy improvement.
$V(s)$ equals to MonteCarloEvaluate(π) and then for all s in non-terminal states
we make $\pi(s) = \underset{a}{\operatorname{argmax}} \sum_{s',r} p(s',r|s,a)(r + \gamma V(s'))$ but we can't compute $p(s',r|s,a)$ so instead of $V(s)$ we use $Q(s,a)$
Our code will be slow because of $Q(s,a)$ and we will need more samples than before
Luckily we know a trick. ~~Value~~ "Value Iteration". Monte Carlo strategy is
Play one episode, update Q with the returns we sampled
Immediately update π using latest Q

We have to know Q for all the actions. In order to perform the improvement step. But we can't just use argmax because we will only collect G samples for π. We may not know all values.

The exploring method is starting with a random state - S_0 and random action - a_0

The main limitation of Exploring starts is that it can't always be done in the real world. We use Epsilon-Greedy Monte Carlo Control

What if episodes never end? For Monte Carlo, episodes must terminate, so G can be computed. Dynamic Programming uses bootstrapping and Monte Carlo uses samples. Temporal Difference uses samples and bootstrapping.

As usual we will first try to solve the prediction task and ~~[illegible]~~ we that for control tasks.

Control 1: SARSA

Control 2: Q-Learning

MC: $V(s) \leftarrow V(s) + \alpha(G - V(s))$

DP: $V_\pi(s) = E_\pi[R_{t+1} + \gamma V_\pi(S_{t+1}) \mid S_t = s]$

TD: $V(s) \leftarrow V(s) + \alpha(r + \gamma V(s') - V(s))$

How does SARSA work? SARSA = (s, a, r, s', a') a sarsa tuple.

Recall that when we're doing prediction, it's ok to use V, but when it comes to control we need to use Q

$$Q(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma Q(s',a') - Q(s,a))$$

what makes it a control problem and not a prediction problem?

Instead of being given a policy π, our policy is epsilon-greedy wrt Q

$$a^* = \underset{a}{\text{argmax}}\, Q(s,a)$$

SARSA Target: $r + \gamma Q(s', a')$

Q-Learning Target: $r + \gamma \max_{a'} Q(s', a')$

Sarsa is on policy where the Q we are learning is the Q function we're using in the environment.

Q-Learning is off-policy actions are dictated by the epsilon-greedy policy, but we are learning the Q function for the greedy policy

The behavior policy dictates how we act in the env.

The target policy is the policy we're learning.

In more deeply to RL, behavior policy can be uniform random but we'd still end up with ~~[illegible]~~ optimal target policy