Some notes:

(Reward Design)

- Any reward transformation changes the initial Markov Decision Process.
- Mean-shifting of the rewards may change the optimal policy.

- The following might alter the reward fn:
 - scale rewards: e.g. division by 10.
 - reshape rewards: add state-potential shaping fn to all rewards.

(Optimization in RL)

- The following might complicate optimization:
 - sparse reward signal; in general sparse signal is hard to find, hence extensive exploration may be required and as a result, sample efficiency of the learning methods degrades.
 - -ve feedback loops and careless reward normalization; -ve loop becomes +ve after a mean subtraction.
 - infinite length of episode; hence the exact value of value fn will be a sum of an infinite series.
 - +ve feedback loops; being stuck in such loops will capture the agent attention, slowing down the discovery of best return possible.

(Reward Discounting)

- it reduces the variance of the return estimator by decreasing the contribution of distant rewards.
- it grasps agent attention towards close rewards and reduces the value of distant ones.

(Bellman Expectation for value fn)

- the expectation in the value fn definition is taken wrt the joint distribution of all actions, states and rewards starting from current time step, till end of episode.

(Bellman Expectation for Q fn)

- it differs from Bellman Expectation for value fn in 2 main things:
1. conditioning on action at t-time step, there is no policy stochasticity as in value fn.
2. there is no need to sum over all actions as in value fn; since action is already considered.

* given Q fn & policy, value fn can be recovered.
* both are defined wrt to some policy.
* value fn can be seen as expection of q-fn wrt policy probs.

* q(s,a) can be written in terms of q(s', a')
* Bellman Expectation equations could be written as a set of linear equations,
that is; one eq per state or state-action pair.

(Bellman Optimality Eq)

- Bellman optimality is written for optimal policy. while Bellman Expection
is written for any arbitrary policy.
- changing the policy to become greedy wrt q-fn and writing Bellman Expectation eq for such policy
is literally the same as Bellman optimality eq
- maximizing value-fn under fixed current policy does NOT mean such policy is optimal.

(Randomness in RL env)

- Sources of randomness might be:
 - randomness of reward, given state & action
 - randomness of action given state
 - randomness of next state, given state & action

(Value Fn)

- is the mean reward that agent get starting from state s, acting upon some policy.

(Q-Fn)

- is the mean reward, after making some action a in state s & SUBSEQUENTLY acting upon some policy.

(Optimal Policy)

- when each episode starts in some fixed state, the agent acts upon some policy that maximizes
the value fn of such state across all possible policies. Even in case of multiple possible starting
states, the agent shall maximize value-fn in each of those states.
HOWEVER, following some policy where the agent makes action a that maximizes q-fn/value fn for some
other policy is definitely NOT necessarily optimal.

(Policy Evaluation)

- is mainly used to assess the performance of the current policy in terms of its value-fn
i.e. value-fn rep how good/bad a policy might be.
- it allows for arbitrary initialization of value-fn.
- it is a determinstic fn of the policy and the env dynamics.
If both are fixed, the output is fixed too.

(Policy Improvement)

- once value-fn of some policy is known, we improve it by acting greedily upon its value-fn
i.e. it improves policy by greedy maximization of q-fn.
- again, the agents acts greedy wrt a combination of immediate rewards & future expected return
under such policy.
- getting an optimal policy is based on whether the value-fn is optimal or not
i.e. getting the same policy as the old one, only means that their value-fn is exactly equal.

(Generalized Policy Iteration)

- the resulted policy:
 - does NOT depend on initialization.
 - is NOT susceptible to local optima.
 - guarantees to converge to global optimum.
 - does NOT need complete policy evaluation; neither in the sense of updating all states,
 nor in the sense of convergence.
- As long as the policy will be updated in each & every state once in a while, there is no need to
improve policy in all states in a particular step.
- When doing the update step in random dir -which is only correct in expectation, such alg
still guarantees to converge.

(Policy Iteration)

- unlike VI, it requires explicit storage of policy.
- unlike VI, iteration is done here till convergence.

(Value Iteration)

- it does not explicity store the probs of actions in each state,
that is; it does not require an explicit policy to perform iteration.
- policy can be retrieved from value-fn; using argmax of q-fn.
- at any step, value-fn might NOT correspond to any particular policy.
- VI performs only one step of Policy Evaluation, before each policy improvement step

* there is no need of precise soln of Bellman sys of eq; since after reaching some precision level,
further refinement of the soln will no longer change the result of subsequent policy improvement.