# Chapter 1

**Exercise 1.1:**

With Self-Play after sufficient training, the agent is expected to reach the real optimal policy of the game. The policy learned will definitely be different than the one learned with a random opponent. This is because the opposite side is also an agent which is simultaneously learning and changing its strategy. This will result in constant change in optimal actions unlike with the random opponent. One case which may arise in Self-Play is that there may be 2 somewhat-optimal policies and the agent will bounce between the two. This can happen when the opponent is playing with Policy A which leads to Policy B and in turn the opponent switches to Policy A.

**Exercise 1.2:**

To account for symmetry different states which are symmetrical can be grouped together. Grouping means that all of them have the same value. Considering symmetry will decrease the size of the state-space and therefore storing the entire state value or action value table will consume less memory.
If the opponent is not considering symmetry, then we should not too. This is because the opponent might behave differently in different states which are actually symmetrical. If we see these states as identical, this will lead to incorrect estimation of the state value. Because the opponent might act optimally in one state and not-so-optimally in the other leading

to opposite sign updates of the state for us since we see both the states as the same.

**<u>Exercise 1.3:</u>**

Greedy play will lead to a local optimal value and not to the global optima. Greedy play will not explore at all and might settle for a policy which is “just-good” rather than the “best”. Thus, greedy play will perform worse than a non greedy player. Sometimes we might be required to make a bad move (with regard to our value function) to eventually win. A greedy agent will never see this as it will always make the best move (with regard to the current value function). This is what exploration enables an agent to do. We can say that the greedy agent is therefore short-sighted.

**<u>Exercise 1.4:</u>**

It is a bad idea to update values during exploratory moves. This is because exploratory moves are not always good moves. If an exploratory move the agent makes is actually a bad move, it’s probability of winning will be low and the previous state’s value will be decreased (when it should not have been). Thus, if we update the values even during exploratory moves, the values of states will be lower than when we don’t.

## Exercise 1.5:

Instead of a probability system we can set up a reward system. The agent gets a reward of +2 if it wins, +1 if it is a draw and -1 if it loses ( the values can be different ). The main motivation behind this is to create a difference between losing and drawing a game. The agent is then expected to first prioritize winning over drawing. This will lead the agent to identify certain trap-sequences of moves which exist in tic-tac-toe and are very well known to humans. The agent will try to trap the opponent into a position where the opponent will lose no matter what. The opposite agent will also learn to counter such situations and the problem with a game like tic-tac-toe though is that the counter to such moves always leads to a draw. So with this strategy the agents will always end up drawing in an attempt to “not-lose”. This is not really an improvement but makes more sense because this method actually explores some interesting possibilities in the game of tic-tac-toe instead of random board positions and also because the idea of choosing to draw instead of losing is actually what a human player would come up with.