Q learning

Exercise 2

a) I use a python dictionary to represent the Q table, later referred to as Q. Each state, 0 to 15, represents a key that links to a list with 4 values. Each values index represents an action, 0 to 3. By choosing the highest value in the list and getting it's index, or choosing and index randomly, we get our action.

b) The greedy policy will always choose to go South because that is the best value in the action list. Unless it slides.

c) As long as you keep the Q function fixed you can't really learn anything, but e-greedy allows you to choose a random action say 1 in 10 times. This allows you to try new paths that are not necessarily optimal and let you explore the state space. So bottom line, e-greedy presents an opportunity for learning a better Q function, because it gives the possibility to find new, and possibly better, q-values .

d) If the reward is not larger than 0 the Q value will lessen at the current discounting rate. By taking the reward quantity into account when updating q-values, it is possible to estimate better q-values for any state and any action. This is also possible, but not guaranteed (instance where rewards> 0 for all (state,action)-pair), when using a greedy policy. If the greedy-policy approach generates enough negative rewards, over some iteration, the q-value might fall low enough for another action to be the greediest choice. Thus, change the behaviour of the agent.

Exercise 4

The difference is that Q-learning takes the best possible value, while the new algorithm (SARSA) uses the value for the actual action taken in the next state. Because Q-learning

pays no attention to the actual policy being followed, but rather backs up the best possible value each time, it is called an off-policy algorithm. SARSA is an on-policy algorithm because its q-values are backed by the actual action taken.

Exercise 5

This is off-policy. For it to be on-policy learning, the agent has to use the q-values of the actual action taken in state s'. This does not happen, since the agent uses a previously learned value, which means the agent ignores the current q-value of the action. Using max values for each action will ignore the 1-e times we pick a random action.

Exercise 6

If the human records the features of the transaction as a feature space our agent can mine rewards from then absolutely. I would use an on-policy learning algorithm because the reward should be based on something actually happening (a customer buying something), not on what the agent would like to happen.

Exercise 7

The data structure used is the same as in Exercise 1, but expanded with more states (keys) and more actions. This is adjusted dynamically based on state space and action space.