

* Once you are done, understand the code you copied and write a pseudo-code to explain what is happening. You need to explain:
  + \_\_init\_\_
  + getQValue
  + computeValueFromQValue
  + computeActionFromQValues
  + getAction
  + update
* Write your pseudo-code on your S24 Assignment Submission page. For example, the question you'll find is "***Write a the pseudo-code (if you paste direct code, will be awarded 0) for \_\_init\_\_***"

The main program actually calls the method runEpisode in loop. Internally the runEpisode has decision callback that calls the getAction method.

The getAction method calls the getPolicy – which actually chooses the action based on the max Score using method computeActionFromQValues on what direction to take. Also to bring in randomness and more exploration a random action is also taken sometimes based on epsilon parameter – The same is set to 0.3 which means that 30% times it will take a random choice and 70% times it will take the choice based on the highest score.

The update method is called by the environment during each transition. This method calculates the new QValues based on the formulas we discussed in the class

# Qt(s, a) = Qt-1(s, a) + alpha ( R(s, a, s') + gamma max\_a' Q(s', a') - Qt-1(s, a) )

For each run the program starts from 0,0 position which is the bottom left most corner

Then it start to navigate through the environment.

During this the Action (getAction method) to be taken is determined based on a combination of policy or a random choice. This is defined based on a parameter epsilon that can be passed as a parameter (Default set to 0.3). So 30% times random choice is made and rest times the getPolicy determines the choice based on the max value of the 4 actions. Even if the action is determined here the environment decides if that movement is allowed or not based on the boundaries and blockages in the path. So the next position is determined based on the same.

After each action completion Qvalues are calculated based on the formula as discussed in class. This is internally called using update method that is called by the environment during each transition of the state based on the earlier getAction method.

# Qt(s, a) = Qt-1(s, a) + alpha ( R(s, a, s') + gamma max\_a' Q(s', a') - Qt-1(s, a) )

* \_\_init\_\_
  + Here we initialize the variables to an empty dictionary which is of type Counter
* getQValue
  + This is to give back the current QValue based on the state and action Q(s,a)
  + This is used in multiple places to get the QValues
* computeValueFromQValues
  + This is to Return the max qvalue out of the 4
  + This is mainly used for getting the max qvalue during update as needed by the formula
  + Internally it uses the getQValue to get the current Qvalues
* computeActionFromQValues
  + Return the action and the qvalue chosen based on the current state and the max of the Qvalue
  + Called in getPolicy that is used in getAction to determine the action in case of non-random approach
* getAction
  + Main perspective here is to determine the next action to be taken
  + This action is determined from either the maxQValue or a Random choice
  + This is defined based on a parameter epsilon that can be passed as a parameter (Default set to 0.3). So 30% times random choice is made and rest times the getPolicy determines the choice based on the max value of the 4 actions.
  + If random\_action is chosen from the list of legalActions in that position randomly
  + Else the action is chosen based on Policy that is based on the Max QValue
  + Based on the epsilon value the code can be more explorative (Higher epsilon means high explorative the system is)
  + This is called as a callback for making a decision on the action to be taken in gridworld.py
  + Even if the action is determined here the environment decides if that movement is allowed or not based on the boundaries and blockages in the path. So the next position is determined based on the same.
* Update
  + After each action completion Qvalues are calculated based on the formula as discussed in class.
    - # Qt(s, a) = Qt-1(s, a) + alpha ( R(s, a, s') + gamma max\_a' Q(s', a') - Qt-1(s, a) )
  + This is internally called by the environment during each transition of the state that is determined based on the earlier getAction method.
  + This takes in inputs of current state, nextstate, action taken and the reward to make this determination