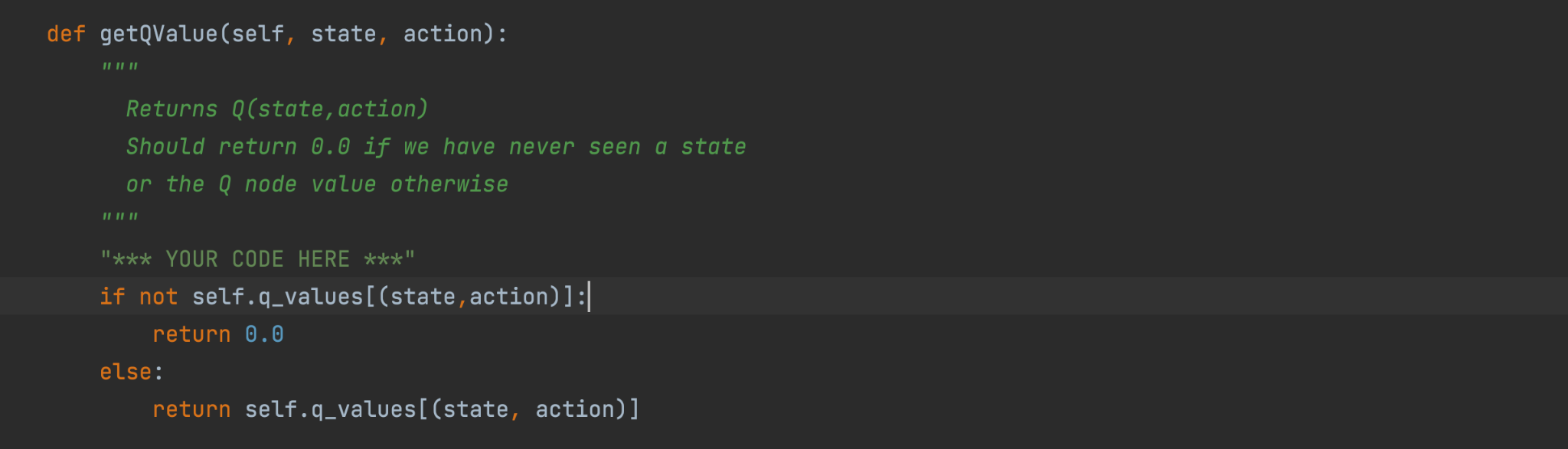
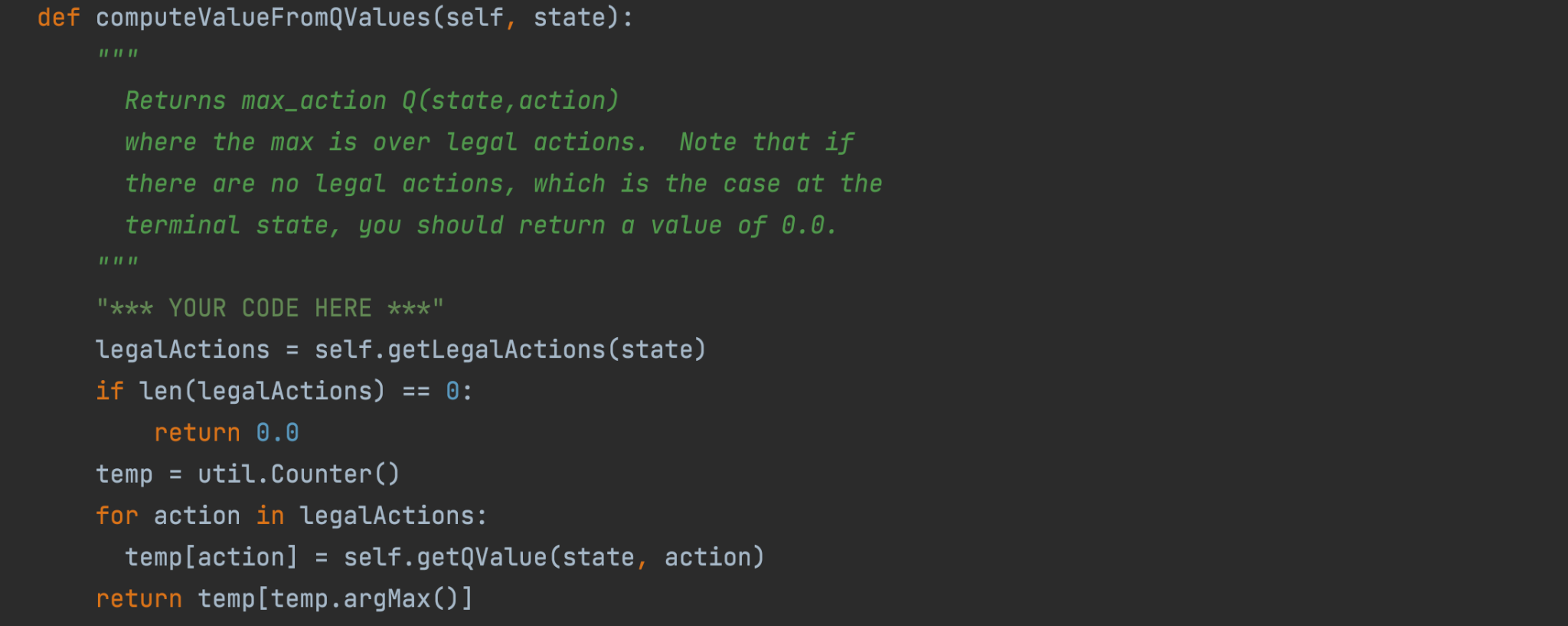
COEN 166 Artificial Intelligence

**Lab Assignment #5**

Name:Krating Khemkhon,Carlo Bilbao, Sanjana Parekh ID: 00001579161, 00001608742, 00001478244

**def getQValue(self, state, action)**

This returns the q value of the current state and action. And if we haven’t seen a state, we return 0.0

**defcomputeValueFromQValues(self,state)**

* First sets a variable legalActions which equals to the legalActions of that current state which is done by using the function self.getLegalActions(state)
* It then checks if the length of the legal actions is 0, if it is, it returns 0 since there are no legal actions available, which is the case at the terminal state
* We have initialized a counter with the name of temp, that keeps a track of the equivalent values added to it with each action, by getting the Qvalue at a certain state and action.
* It then returns the max action of the Qvalue at a specific state and with a specific action.

def computeActionFromQValues(self, state):

"""

Compute the best action to take in a state. Note that if there

are no legal actions, which is the case at the terminal state,

you should return None.

"""

"\*\*\* YOUR CODE HERE \*\*\*"

maxqval = self.getValue(state)

best\_action = []

for action in self.getLegalActions(state):

if self.getQValue(state, action) == maxqval:

best\_action.append(action)

if len(best\_action)==0:

return None

else:

return random.choice(best\_action)

**def computeActionFromQValues(self,state)**

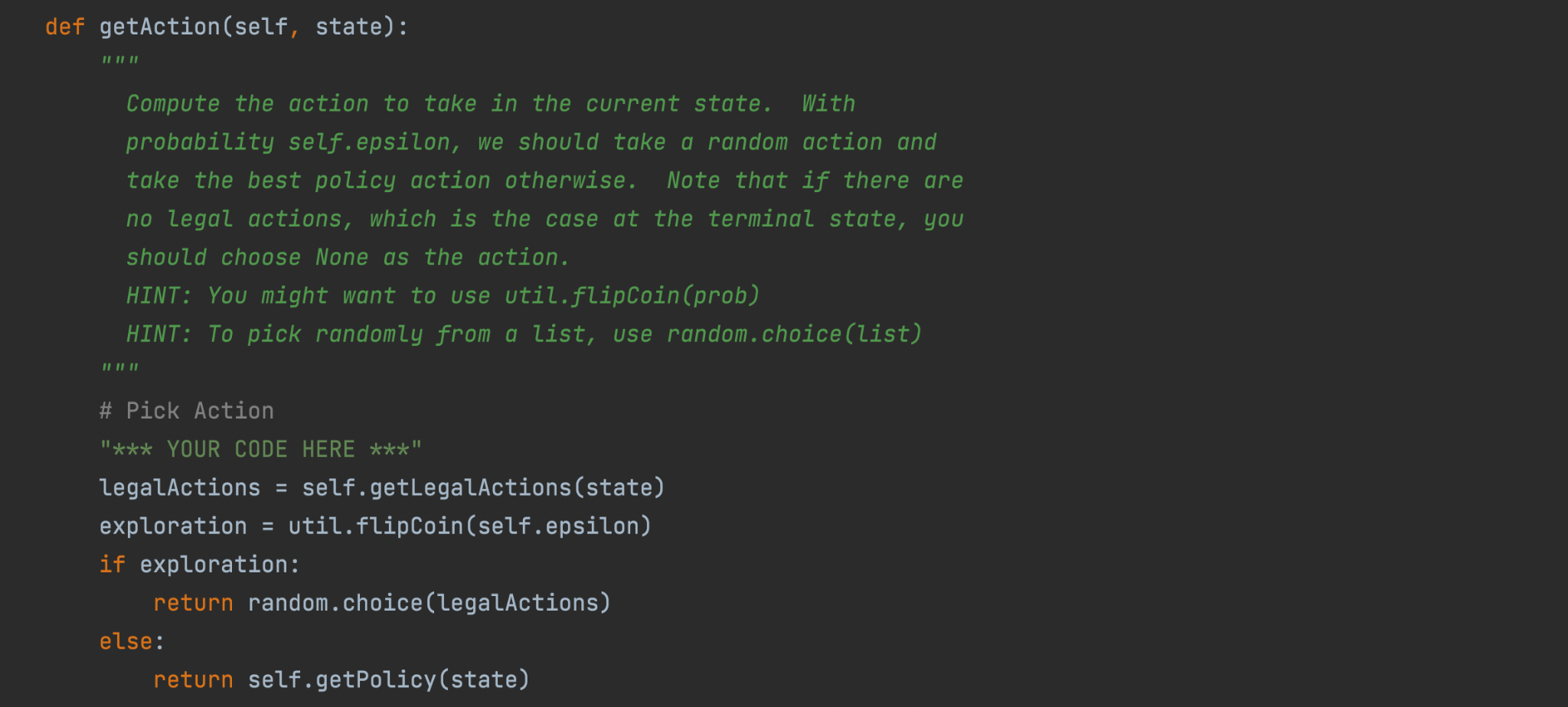
-get the best\_val by using the getValue func which calls the compute qvalue function

-create a list to store all possible best actions

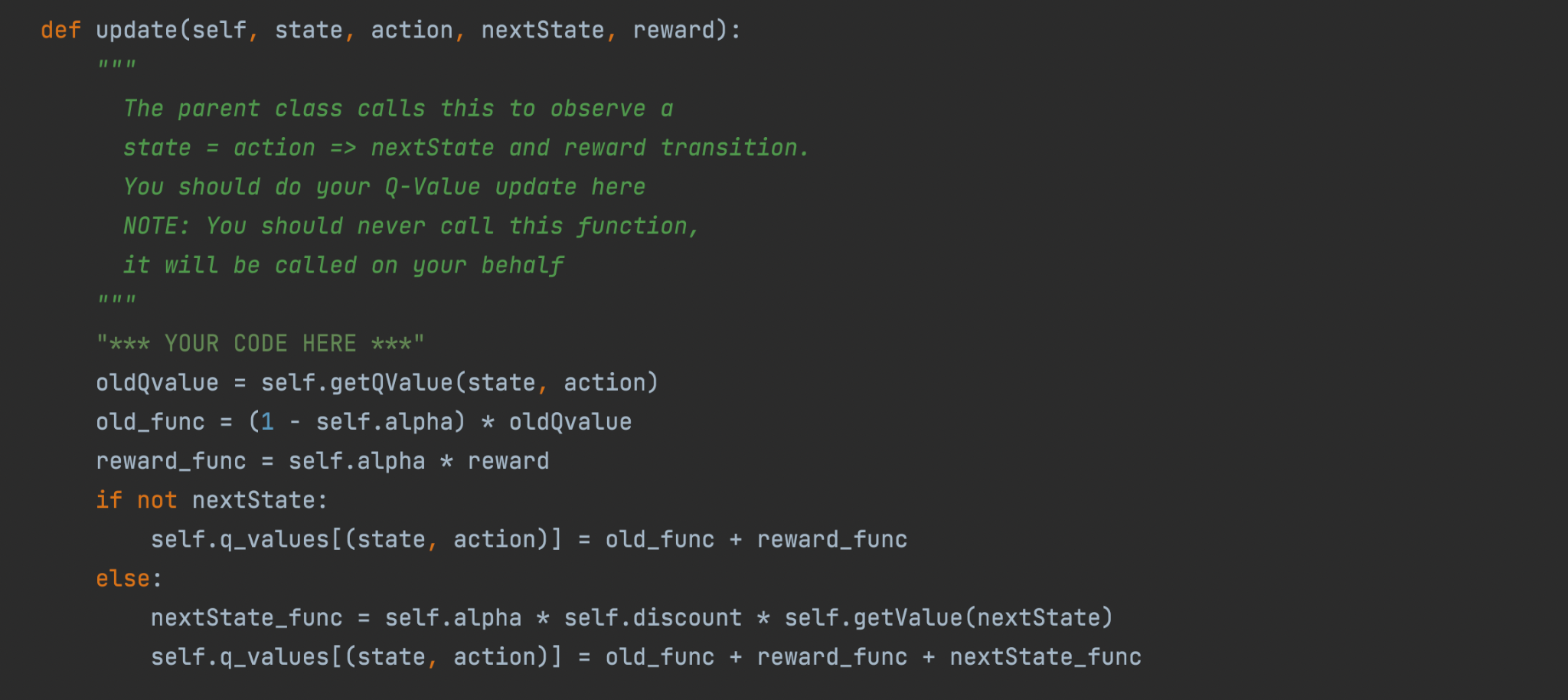
-for all legal actions if the Qvalue is equal to the max qval we add the corresponding action to the list of best action

-if there are no best actions we return None

-else we randomly select an action from the list.

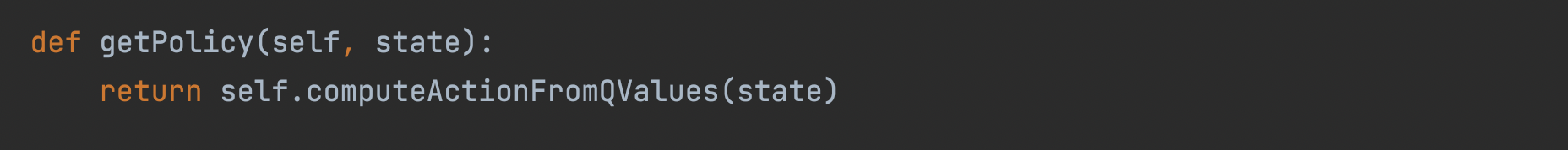
**def getAction(self,state)**

* First sets a variable legalActions which equals to the legalActions of that current state which is done by using the function self.getLegalActions(state)
* If the probability self.epsilon is true, we take a random choice for the action and otherwise take the best policy action for that state.

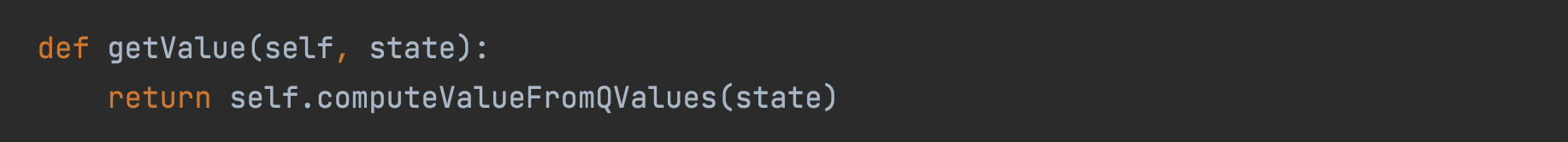
**def update(self, state, action, nextState, reward)**

This function is mainly for calculations to find the q value.

* Take the old queue value by calling getQValue for this state-action pair
* This is the old q value that is not updated yet
* Take 1 - alpha and then multiply it by our current unupdated q value
* This calculates the reward portion of the formula by taking our alpha value and multiplying it by the reward given
* If not next state
* Disregard the nextState\_part of the formula
* Set this state-action pair's q value to just be the old un updated q value + the reward
* Else, integrate the next state part of the formula
* This will take the last part of the formula: alpha \* the discount \* the max Q value of state prime (AKA next state)
* We will do the same formula above except with the next state calculation

**def getPolicy(self,state)**

It returns the best action for that state

**def getValue(self,state)**

It returns the max q value for that state

**Lab 4 Contributions of each group member (in percentage):**

Carlos: 25%

Krating: 35%

Sanjana: 40%