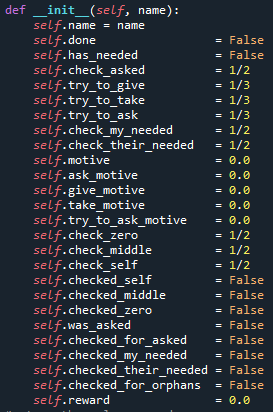
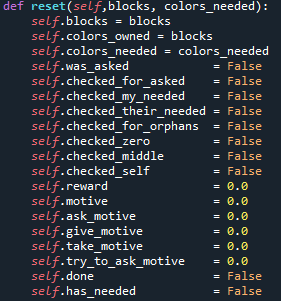
Kevin Charles Hostler

Experimental phase update Version 1:

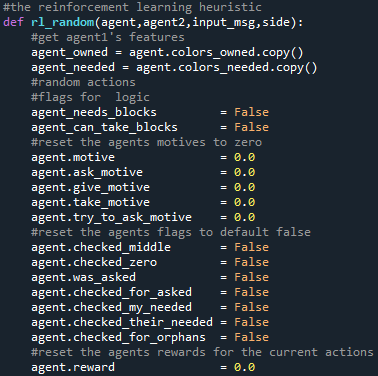
This report is to document and better help explain the plots shown in the PowerPoint slides. To start with we have two different versions of the logic code, though both have a similar premise, but the second version implements fixes to the issues found out in version one.

**Version 1**

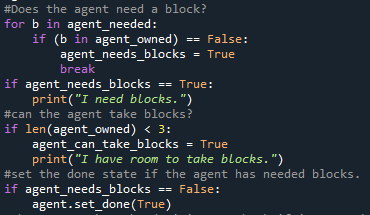
The code lay out is a system of random choices with learning weights for said choices. While many of the attribute names leave much to be desired for specifics, the code layout is organized and does work. The percentage chance to check needed blocks is set at 50% at first initialization but as the games progress thus chance increases upon successful checks which will be explained more with the RL heuristic function.

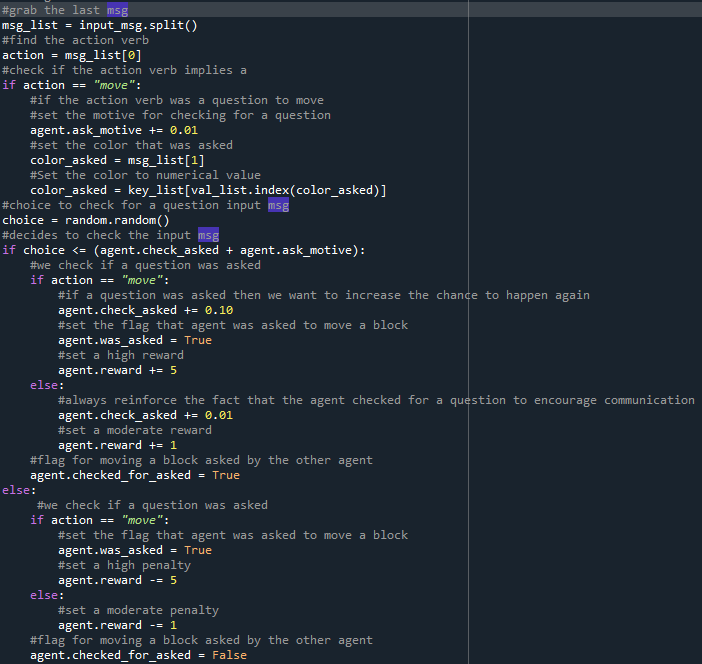
 

The function itself takes in both agents, the last message printed to the screen and which side the agent is on. The first thing done is to temporarily copy the needed and owned blocks for the agent that is taking its current turn, reset the motives and question Booleans. The motives will accumulate penalties or rewards for successful checks to add or decrease to the percentage chance of a decision happening based on the state of the board and what the agent has learned of past actions.

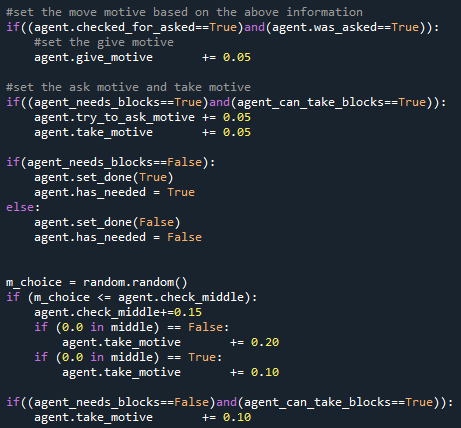


We have three assumptions for the agent, it knows if it needs blocks, it knows if it can take blocks and if it doesn’t need blocks then it knows it’s done as it doesn’t need anything.



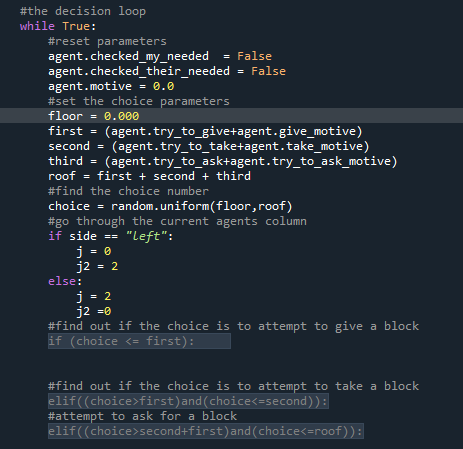


The next part of the code goes through first to check if the agent was asked a question, and if it was, we want to give a small motive to check for an asked question. The agent then runs the choice to check if it was asked a question, if it was asked a question and the agent does check then we give the reward to encourage the agent to check again next time the agent has a turn. The Booleans we saw earlier are used to track if the agent was asked a question and if the agent checked if it was asked a question.

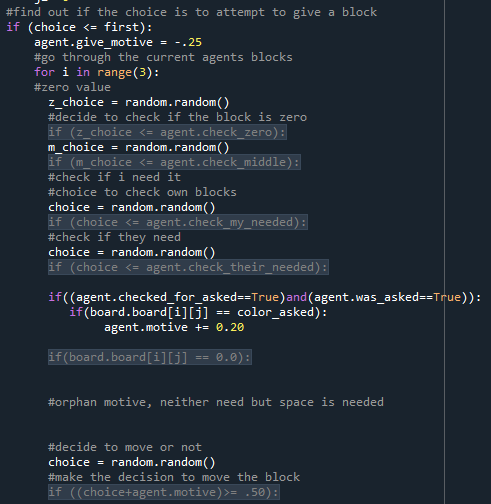


After being rewarded accordingly, we then move on to finalize which action will be more likely to be picked first. The chance to give is added to if the agent was asked a question and checked for a question. The chance to “ask to move a block” and to “take a block” are increased if the agent needs blocks and has room to take blocks. Then there is a straight decision that if the agent doesn’t need blocks, then it flags itself as done. Then the agent has the decision to check the middle for no empty spaces (zeros) or at least one zero, both decisions are rewarded and motive to take is higher. The motive to take is then set higher if the agent doesn’t need blocks but can still take them.

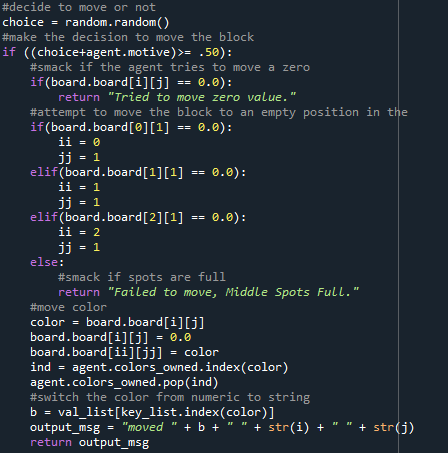
This logic shows how naïve this RL model is as I needed to add more rewards to the take motive so the agents would get stuck less often when the middle is full or slightly filled with blocks.



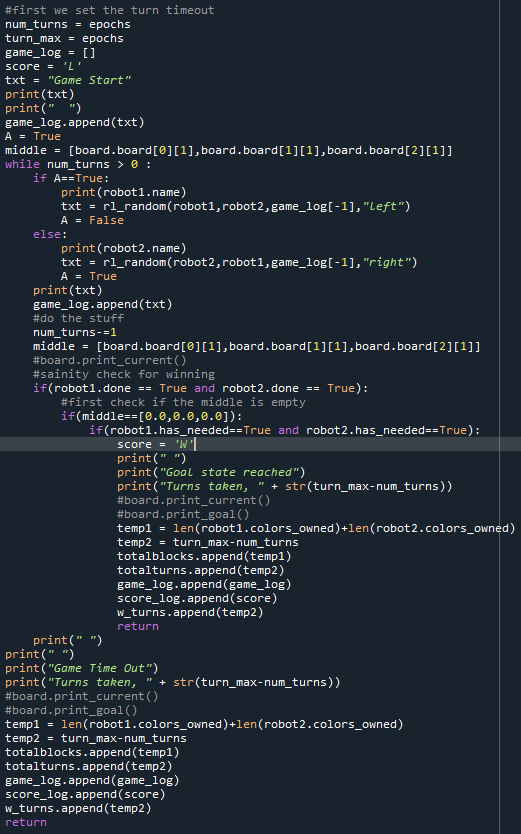
For the decision loop I had the agents loop around, with motive chances of either giving, taking or asking for a block.



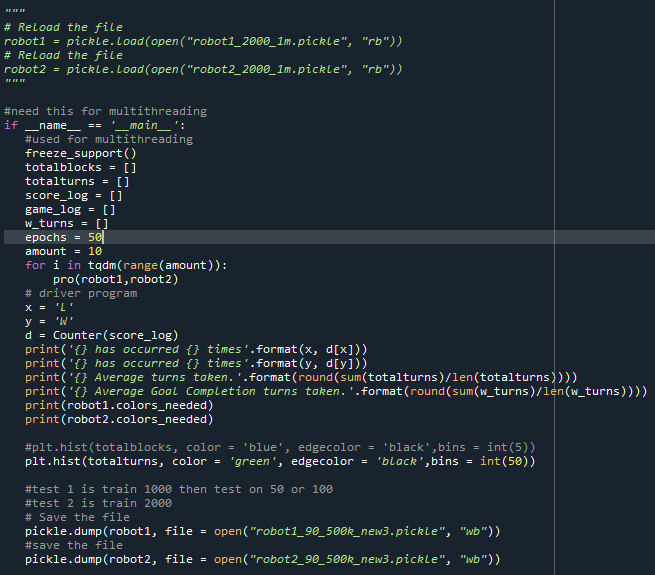
The logic for each decision runs about the same, The agent goes through either it’s blocks the middle slots or the other agents blocks. The agent the decides to check for if the current slot is empty or not then, if it needs the block in question or if the other agent needs a block. If the agent finds the block that it was asked to move then the motive to give is much higher.



Then when deciding to give, take or ask we first do a quick penalty check for if the slot is empty or if the giving spot is full. If that clears, then the agent completes the action and prints it’s message.

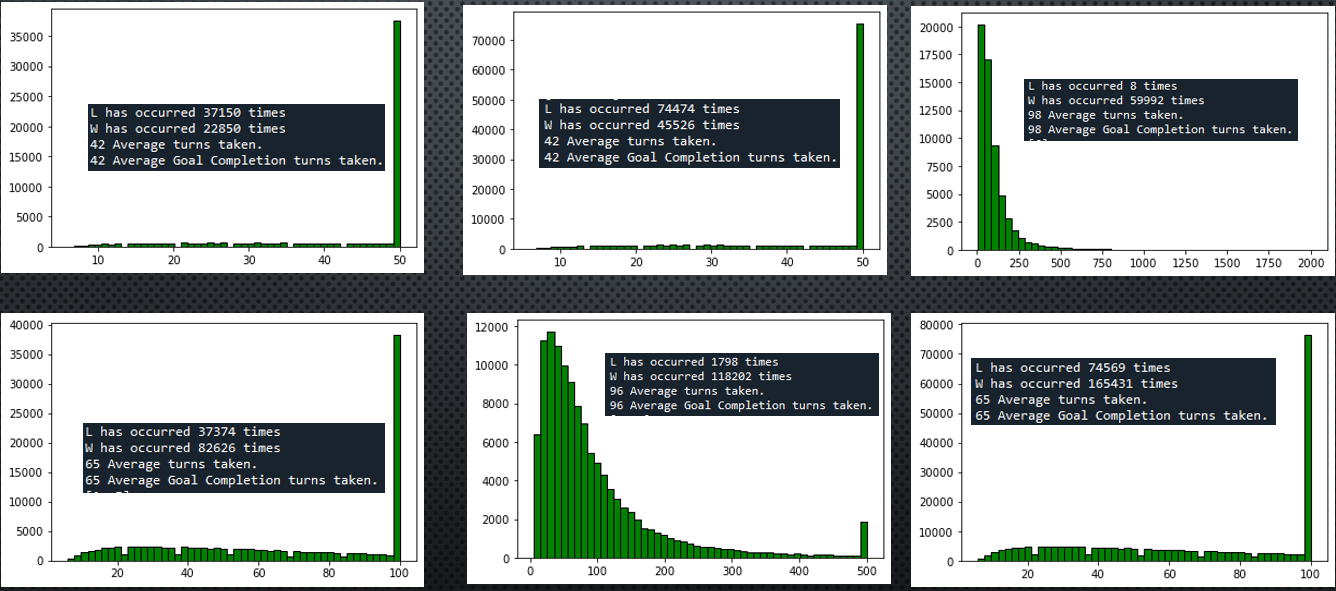


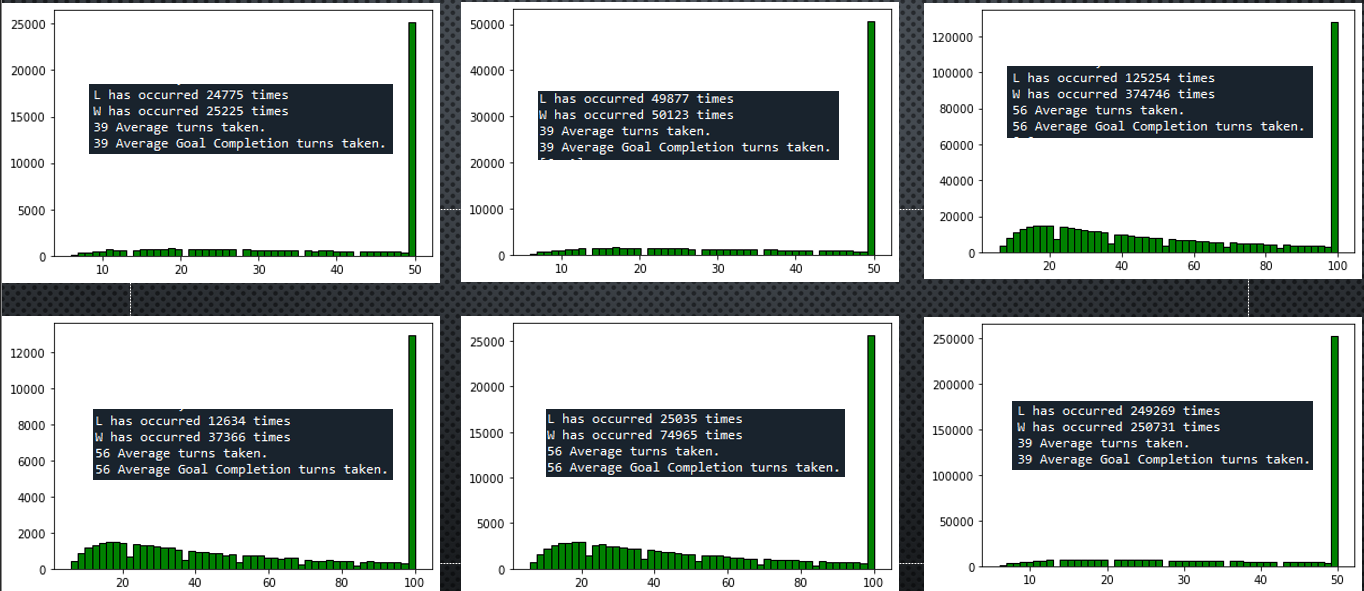
The game logic is still the same but will change slightly in the version 2.



I then have set up looping the games for a set amount of times and how long each game is allowed to be. I also saved and can load the trained models for further testing.

Version 1 Results





* 50 turn time out
  + 50 doesn’t seem enough regardless of the time given to it.
* 100 turn time out
  + Much better than 50 but needs a significant amount of time to become more than marginally better
* 500 turn time out
  + Greatly increased turn around time for allowing the agents enough time to work through the games to gain enough rewards for proper actions
* 2000 turn time out
  + Significantly increased turn around time as this gives the agents plenty of time to work out everything in order to select a proper action.