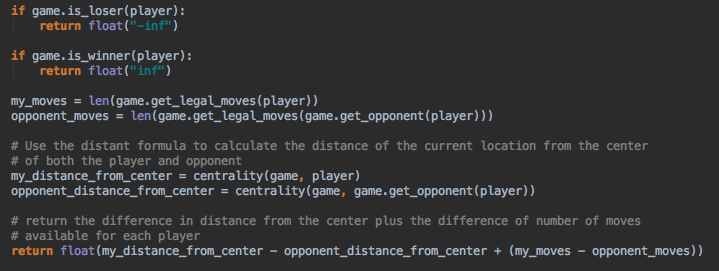
My first approach to the analysis was to try and work it out on paper for a few moves that a player could make. This was not an easy task but it helped me to further my understanding on how heuristic would play a role in the sample\_player.py After dissecting how the sample\_player.py worked, I had descent understanding about how to approach the heuristic.

As far as my understanding of chess is concerned, whoever controls the center of the board is most likely controlling the flow of the game. With that in mind there are methods two possible stances a player can do. Either an aggressive offensive position or an aggressive defense position. The three created heuristic functions were focused on a combination of distance from center and how many moves the player has. This is where I decided to create a function called centrality which calculates a player’s distance to the center. Similar to the Manhattan distance.

The following three heuristics were tested and used to play the chess game.

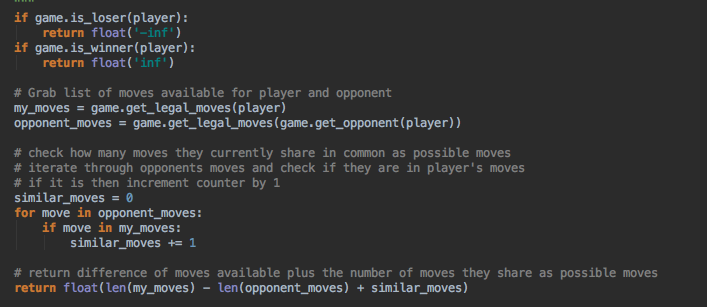
Heuristic 1



In this heuristic method I calculate the distance from the center of the board and the closer the player is to the center it means they control more of the board space, therefore have a higher score value.

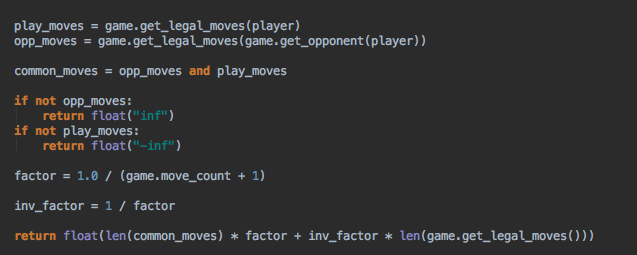
Heuristic 2

.



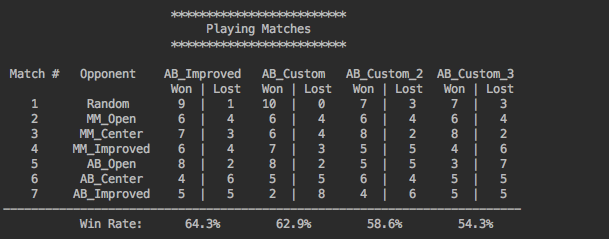
In this heuristic method I try to find the similar moves each player has and compute a difference to see how unique each players move set was. The more options a player has the more likely they are dominating the board and winning

Heuristic 3



In the last heuristic was a mix scoring of defense and offence that the player was using against the opponent. As the game comes to a closes the player has very little move set to work with their score goes down but rises as they play more better.

Below are my results for running tournament.py



It appears that the last two customs scores didn’t play drastically well, but the AB improved score was the best. The first custom score from the center was a second runner up for best score. The slowest computations were by far the last two scoring.

Overall I would recommend the AB\_Custom. The score did the best overall in term of winning rate. The computation time is faster because it has less operations to do then the other two. The depth search also seems to the faster on average as well. The AB\_Custom was also the easiest metric to come up with and implement. So my best recommendation for the best heuristic is AB\_Custom.