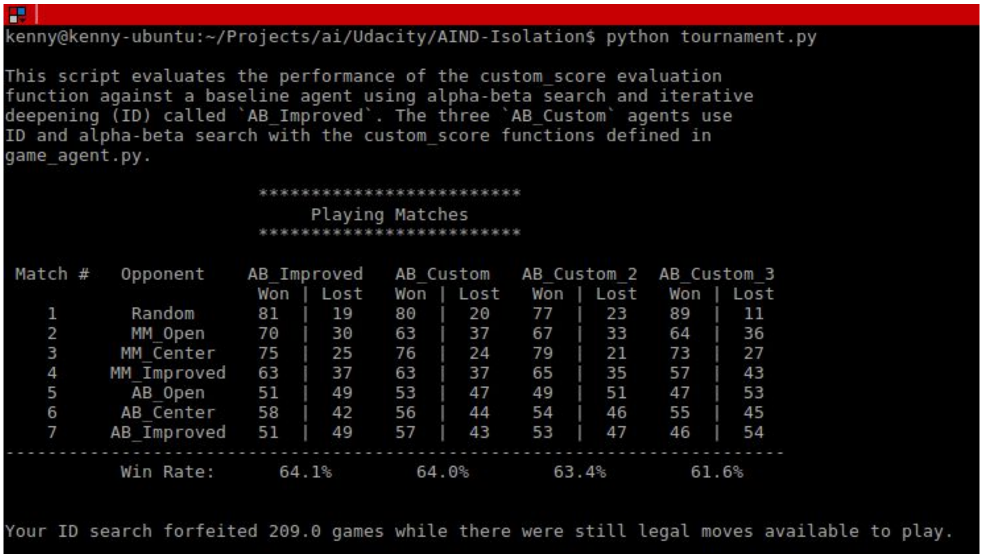
Position Evaluation Heuristic Analysis



```
kenny@kenny-ubuntu:~/Projects/ai/Udacity/AIND-Isolation$ python tournament.py

This script evaluates the performance of the custom_score evaluation
function against a baseline agent using alpha-beta search and iterative
deepening (ID) called `AB_Improved`. The three `AB_Custom` agents use
ID and alpha-beta search with the custom_score functions defined in
game_agent.py.

                        *************************
                             Playing Matches
                        *************************

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random      81  |  19    80  |  20    77  |  23    89  |  11
    2       MM_Open     70  |  30    63  |  37    67  |  33    64  |  36
    3      MM_Center    75  |  25    76  |  24    79  |  21    73  |  27
    4     MM_Improved   63  |  37    63  |  37    65  |  35    57  |  43
    5       AB_Open     51  |  49    53  |  47    49  |  51    47  |  53
    6      AB_Center    58  |  42    56  |  44    54  |  46    55  |  45
    7     AB_Improved   51  |  49    57  |  43    53  |  47    46  |  54
-----------------------------------------------------------------------
          Win Rate:      64.1%        64.0%        63.4%        61.6%

Your ID search forfeited 209.0 games while there were still legal moves available to play.
```

We were given the task to come up with custom score heuristics to evaluate board positions while searching the game tree with iterative deepening. We evaluate these custom heuristics by playing games against several pre-defined agents that use strategies like random move, mini-max, and alpha-beta pruning. Our goal is to try and beat the ID_Improved heuristic, shown as the leftmost column above, which was a differential between player available legal moves and opponent available legal moves.

My custom heuristics are as outlined:

AB_Custom: (Aggressive) player_moves - 2 * opponent_moves
AB_Custom_2: (More Aggressive) player_moves - 3 * opponent_moves
AB_Custom_3: (Defensive) 2 * player_moves - opponent_moves

Results

Because my custom heuristics were all variations of ID_Improved, there is little to no improvement in performance. While this type of heuristic is easy to interpret and compute, it is not really “game-aware” and does not take into account any type of positional advantage. If I were to continue working on this project, I would have liked to explore some additional factors:

- A player with a starting position closer to the center of the board is more likely to do better than a player near the edge, where there are less options to move later in the game
- When the player and opponent have a similar amount of available moves, we can try to evaluate the positions using another method, maybe by estimating how many more turns each player can survive in a constrained area. Given the game rule of knight-moves only, we know there exist 3x3 game areas where a player can survive for up to 7 turns (illustrated below where starting position is 0).