Heuristic Analysis

First of all, here is the result I got when I run the tournament.py file:

```
                    **************************
                        Playing Matches
                    **************************

 Match #   Opponent    MM_Improved   MM_Custom   MM_Custom_2  MM_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       10  |   0     9  |   1    10  |   0     7  |   3
    2       MM_Open       3  |   7     4  |   6     7  |   3     1  |   9
    3      MM_Center      9  |   1    10  |   0     8  |   2     5  |   5
    4     MM_Improved     5  |   5     6  |   4     3  |   7     1  |   9
    5       AB_Open       6  |   4     6  |   4     5  |   5     3  |   7
    6      AB_Center      2  |   8     4  |   6     4  |   6     3  |   7
    7     AB_Improved     6  |   4     5  |   5     6  |   4     3  |   7
-----------------------------------------------------------------------------
          Win Rate:      58.6%        62.9%        61.4%        32.9%

[Finished in 238.2s]
```

Here are my analysis on the following 3 custom_score() function.

- custom_score(): From here I use float(own_moves**2 - opp_moves**2 * 1.5). This function has a 62.9% winning rate. We can see that when the opponent plays in MM_Center, the opportunity of winning is very high. But it seems not doing very well in MM_Open and AB_Center. Even if all that, it seem that this function can do better than MM_Improved as the number of winning match is higher on almost every type of opponent.

- custom_score_2(): It is using float(own_moves - 2 * opp_moves), which is recommended in the tutorial. And it gives quite a good result of 61.4% winning rate. It also performs very well compared to MM_Improved. and if opponents are playing MM_Open, the winning rate is very high also when we use this function. From my observations, it seems that this function gives similar result to custom_score().

- custom_score_3(): It is using float(opp_moves**2/own_moves**2). From the result we can see that this function did not perform very well (32.9% winning rate) and compared to all types of opponents with MM_Improved, it loses in all situations also. It is not recommended to use this.