Heuristic 1

Scoring the move based on only the current players available moves.

```
def custom_score1(game, player):
    own_moves = game.get_legal_moves(player)
    return float(len(own_moves))
```

****
Evaluating: ID_Improved
****

Playing Matches:

Match 1: ID_Improved vs Random Result: 65 to 15
Match 2: ID_Improved vs MM_Null Result: 49 to 31
Match 3: ID_Improved vs MM_Open Result: 25 to 55
Match 4: ID_Improved vs MM_Improved Result: 25 to 55
Match 5: ID_Improved vs AB_Null Result: 68 to 12
Match 6: ID_Improved vs AB_Open Result: 47 to 33
Match 7: ID_Improved vs AB_Improved Result: 40 to 40

Results:

ID_Improved 56.96%

****
Evaluating: Student
****

Playing Matches:

Match 1: Student vs Random Result: 65 to 15
Match 2: Student vs MM_Null Result: 47 to 33
Match 3: Student vs MM_Open Result: 24 to 56
Match 4: Student vs MM_Improved Result: 21 to 59
Match 5: Student vs AB_Null Result: 69 to 11
Match 6: Student vs AB_Open Result: 40 to 40
Match 7: Student vs AB_Improved Result: 45 to 35

Results:

Student 55.54%

The purpose of this heuristic was to see if the number of opponent moves had any effect on the outcome. On a number of trials, it looks like this heuristic is very close to the improved heuristic. However it is not better than the improved heuristic.

Heuristic 2

Chase the center if open else chase the other player

```
def custom_score2(game, player):
    own_moves = game.get_legal_moves(player)
    opp_moves = game.get_legal_moves(game.get_opponent(player))
    if(len(own_moves) <= 0):
        return float("-inf")
    if len(opp_moves) <= 0:
        return float("inf")

    dist = 0

    c = game.width//2, game.height//2
    p = game.get_player_location(player)

    if c in game.get_blank_spaces():
        dist = sqrt((c[0]-p[0])**2 + (c[1]-p[1])**2)
    else:
        p1 = game.get_player_location(game.get_opponent(player))
        dist = sqrt((p1[0] - p[0])**2 + (p1[1]-p[1])**2)
    return float(len(own_moves) - len(opp_moves) - dist)
```

****
Evaluating: ID_Improved
****

Playing Matches:

Match 1: ID_Improved vs Random Result: 65 to 15
Match 2: ID_Improved vs MM_Null Result: 54 to 26
Match 3: ID_Improved vs MM_Open Result: 24 to 56
Match 4: ID_Improved vs MM_Improved Result: 20 to 60
Match 5: ID_Improved vs AB_Null Result: 69 to 11
Match 6: ID_Improved vs AB_Open Result: 45 to 35
Match 7: ID_Improved vs AB_Improved Result: 40 to 40

Results:

ID_Improved 56.61%

****

Evaluating: Student

****

Playing Matches:

Match 1:	Student	vs	Random	Result: 68 to 12
Match 2:	Student	vs	MM_Null	Result: 38 to 42
Match 3:	Student	vs	MM_Open	Result: 17 to 63
Match 4:	Student	vs	MM_Improved	Result: 13 to 67
Match 5:	Student	vs	AB_Null	Result: 68 to 12
Match 6:	Student	vs	AB_Open	Result: 20 to 60
Match 7:	Student	vs	AB_Improved	Result: 32 to 48

Results:

Student 45.71%

This seemed logically reasonable but it did not perform as expected. All such heuristics either trying to chase or run away from the opponent performed worse than the improved heuristic.

Heuristic 3

Random heuristic.

```
def custom_score(game, player):

    own_moves = game.get_legal_moves(player)
    opp_moves = game.get_legal_moves(game.get_opponent(player))

    if game.is_loser(player):
        return float("-inf")
    if game.is_winner(player):
        return float("inf")

    return float(len(own_moves)/len(opp_moves))
```

```
**************************
 Evaluating: ID_Improved
**************************

Playing Matches:
----------
  Match 1: ID_Improved vs   Random      Result: 71 to 9
  Match 2: ID_Improved vs   MM_Null     Result: 47 to 33
  Match 3: ID_Improved vs   MM_Open             Result: 21 to 59
  Match 4: ID_Improved vs MM_Improved           Result: 21 to 59
  Match 5: ID_Improved vs   AB_Null     Result: 69 to 11
  Match 6: ID_Improved vs   AB_Open     Result: 36 to 44
  Match 7: ID_Improved vs AB_Improved           Result: 40 to 40


Results:
----------
ID_Improved         54.46%

**************************
   Evaluating: Student
**************************

Playing Matches:
----------
  Match 1:   Student   vs   Random          Result: 67 to 13
  Match 2:   Student   vs   MM_Null         Result: 55 to 25
  Match 3:   Student   vs   MM_Open         Result: 33 to 47
  Match 4:   Student   vs MM_Improved   Result: 21 to 59
  Match 5:   Student   vs   AB_Null         Result: 73 to 7
  Match 6:   Student   vs   AB_Open         Result: 40 to 40
  Match 7:   Student   vs AB_Improved     Result: 42 to 38


Results:
----------
Student             59.11%
```

After struggling with various Heuristics that involved the own moves, opponent moves distances from center and opponents, all of which led to results that were not better than improved score Heuristic, I read in one of the Chris Lapollo's post that he did something random that did better than the improved score Heuristic.

From knowledge from the above two Heuristics,

1. Own moves and opponent moves both need to be considered
2. Distance from center and distance from opponent had negative effect

From various operations on the own moves and opponent moves, it was observed that this Heuristic consistently performed slightly better than the improved score.

The reason this performed better than the improved score can be given as follows:

In the improved score function,

scenario	own_moves	opp_moves	score
1	3	2	1
2	2	1	1

With the custom heuristic

scenario	own_moves	opp_moves	score
1	3	2	1.5
2	2	1	2

As can be seen above in improved score both the scenarios get the same score. However it can be seen that the scenario 2 has a chance of ending the game sooner and should be the preferred move. The custom score function clearly has an advantage in identifying scenarios that can lead to a win sooner.

Other observation,

Since the game has no draw scenario, even though both the players have no legal moves left, player_1 is declared the loser on the account that he is the first to not have any moves left. I describe this in the post https://discussions.udacity.com/t/draw-scenario-in-the-game/227390/4

This meant that if in the custom score function, we returned a "-inf" in the case where there were no legal moves present, it was not beneficial to player_2. So as player_2 it is always necessary to check if

the player is the active player and then there are no further legal moves left. This was being done in the improved score function via calling game.iswinner and game.isloser functions. This wouldn't have any impact on player_1 however, so it is always in the best interest to call the above functions which also check if the player is the active player.