Research Review:

The challenge was to build a Isolation player, which outperforms the other AI players from the sample_players module, especially an agent called "ID_Improved" that uses Iterative Deepening and the improved_score heuristic. Therefore you had to build a heuristic which delivers a good evaluation of the computed game state.
My approach was to improve the improved_score heuristic which the ID_Improved player is using:

```
return float(own_moves - opp_moves)
```

It returns the subtraction of the available moves of the current player and the available moves of the opponent player.

Therefore I tested to weight both influencing variables. In custom_score_3 I testet to give the number of available moves the opponent player a higher weight:

```
return float(own_moves - 3 * opp_moves)
```

The result was that it won 68.9 % of its matches against the other players.

In custom_score_2 I tested the opposite:

```
return float(3 * own_moves - opp_moves)
```

The result was that it won 57.1 % of its matches against the other players. So it turns out that custom_score_3 wins more games. It seems to be more important to force the opponent into a narrow gamestate!

Going this way I tested further and quickly found a heuristic which beats ID_Improved player. It looks like this:

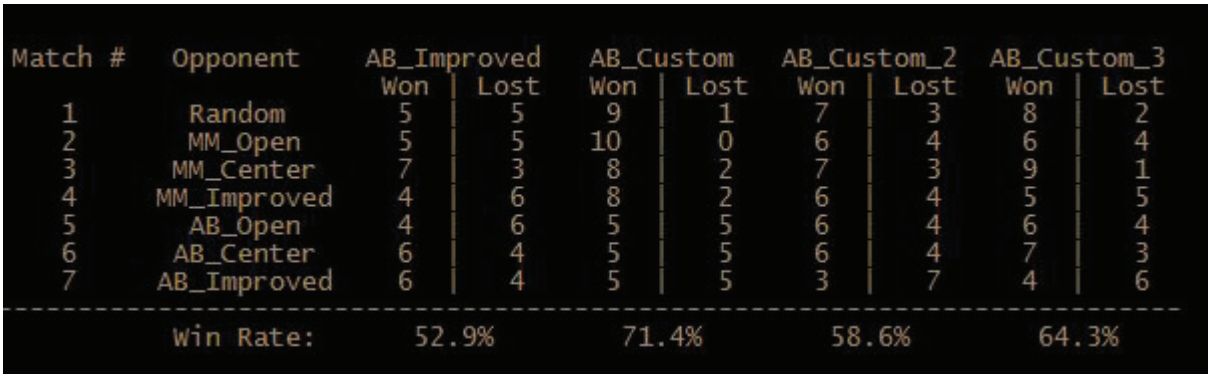
```
    own_moves = game.get_legal_moves(player)
    opp_moves = game.get_legal_moves(game.get_opponent(player))

    intersect_moves = list(set(own_moves).intersection(opp_moves))

    # If the opponent has only 2 moves left and you can block him
immediately:

    if len(opp_moves) <= 2 and len(intersect_moves) >= 1:
        return len(own_moves)*10 + 100

    return float(len(own_moves) - 3 * len(opp_moves))
```

I added a monitoring that tries to find out if you can block the opponent in an isolated position. If so the heuristic returns a very high score of this game state.

Finally here is a comparison of the heuristics:

Heuristic	Performance over all
custom_score_3 (improved score heuristic with higher weighted opp_moves factor)	64.3 % win rate
custom_score_2 (improved score heuristic with higher weighted own_moves factor)	58.6 % win rate
custom_score (improved score heuristic with higher weighted opp_moves factor and a test if you can isolate your opponent)	71.4 % win rate
AB_Improved	52.9 % win rate

Or here the output of the console:



Match #	Opponent	AB_Improved		AB_Custom		AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost	Won	Lost	Won	Lost
1	Random	5	5	9	1	7	3	8	2
2	MM_Open	5	5	10	0	6	4	6	4
3	MM_Center	7	3	8	2	7	3	9	1
4	MM_Improved	4	6	8	2	6	4	5	5
5	AB_Open	4	6	5	5	6	4	6	4
6	AB_Center	6	4	5	5	6	4	7	3
7	AB_Improved	6	4	5	5	3	7	4	6
	Win Rate:	52.9%		71.4%		58.6%		64.3%	

My custom_score heuristics seems to win more games compared to the other players even compared to AB_Improved (71.4% vs 52.9%). Direct match statistics results in 5-5 and seems to be balanced, but should be analysed by increasing the sample size.

My recommendation is to use the custom_score heuristics, it seems to be the best combination of all introduced heuristics and returns the best statistics. There was no game versus an opponent player that this heuristic lost over 10 games! During the game it even does a cheap test if it can isolate the opponent into a blind alley. It only needs one intersections of sets more than the other heuristics.