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)
```

In custom_score_2 I tested the opposite:

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

It turns out that custom_score_3 wins more games. It seems to be more important to force the opponent into a narrow gamestate!

Match #	Opponent	AB_Custom_2		AB_Custom_3	
		Won	Lost	Won	Lost
1	Random	6	4	9	1
2	MM_Open	4	6	8	2
3	MM_Center	7	3	9	1
4	MM_Improved	6	4	5	5
5	AB_Open	5	5	5	5
6	AB_Center	7	3	7	3
7	AB_Improved	5	5	5	5
	Win Rate:	57.1%		68.6%	

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.

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%	

Above you can see the results. My custom_score heuristics seems to win more games compared to AB_Improved (71.4% vs 52.9%).