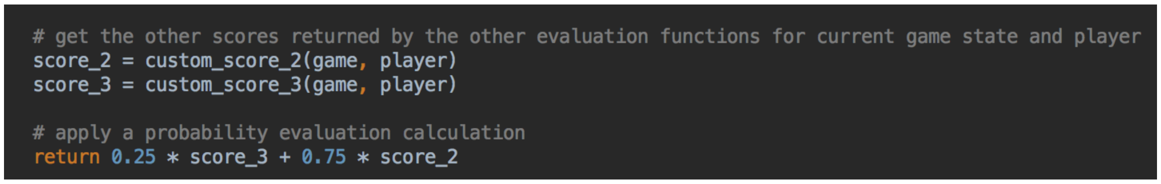
Heuristic Analysis of game_agent.py:

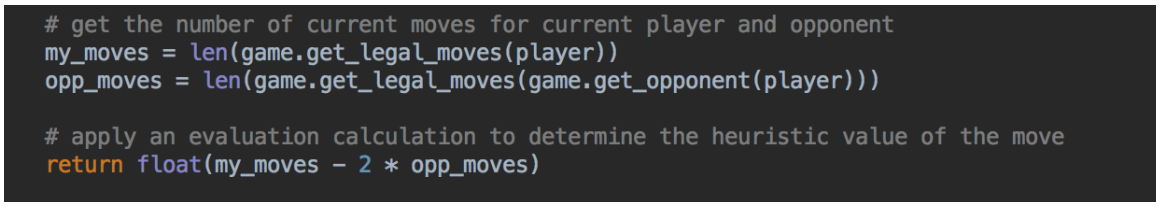
custom_score:
I ultimately chose this function as custom_score, because it was consistently the best performing custom function out of all 3 in the tournament evaluation.

This evaluation function takes the output of custom_score_2 and custom_score_3 for the current game state and player then applies a probability function to each score of 25%, and 75% probability respectively (note: this combination of probability performed better than other probability combinations, e.g. 20%/80%)



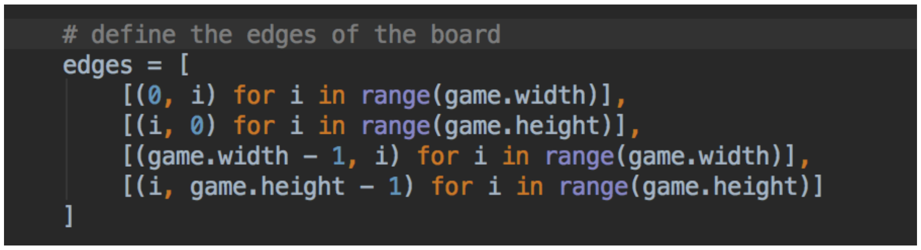
```
# get the other scores returned by the other evaluation functions for current game state and player
score_2 = custom_score_2(game, player)
score_3 = custom_score_3(game, player)

# apply a probability evaluation calculation
return 0.25 * score_3 + 0.75 * score_2
```

custom_score_2:
This function calculates the current player's available moves and the opponent's available moves and then takes a more aggressive approach, by subtracting 2 times the opponents moves from the player's moves:



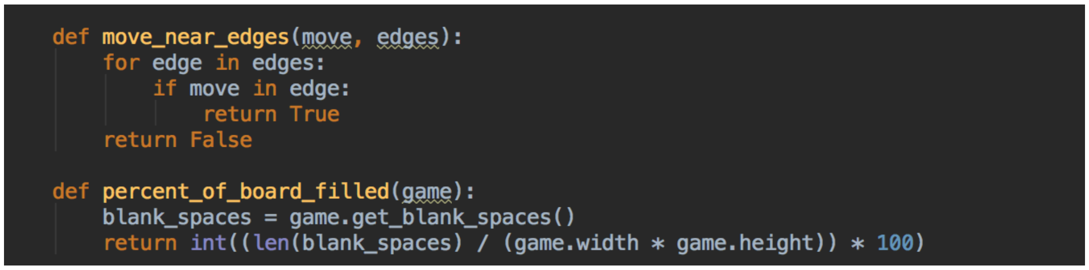
```
# get the number of current moves for current player and opponent
my_moves = len(game.get_legal_moves(player))
opp_moves = len(game.get_legal_moves(game.get_opponent(player)))

# apply an evaluation calculation to determine the heuristic value of the move
return float(my_moves - 2 * opp_moves)
```

custom_score_3:
(Heuristic based on example from AIMA Text)
I decided to implement this more complex evaluation function because it takes into account the edges of the board and # of squares taken when calculating the value of the players' moves. First the function defines the edges of the isolation board:



```
# define the edges of the board
edges = [
    [(0, i) for i in range(game.width)],
    [(i, 0) for i in range(game.height)],
    [(game.width - 1, i) for i in range(game.width)],
    [(i, game.height - 1) for i in range(game.height)]
]
```

Two helper functions are created. Move_near_edges helps identify whether a move is near the edges while percent_of_board_filled returns the percentage of the isolation board that is filled given the current game state:

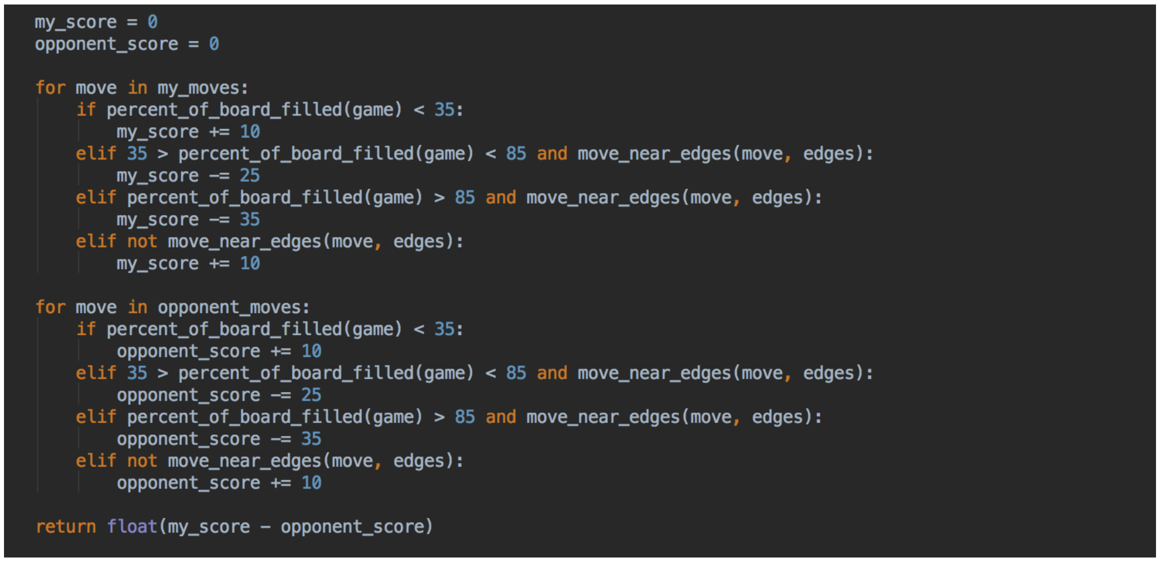


```python
def move_near_edges(move, edges):
    for edge in edges:
        if move in edge:
            return True
    return False

def percent_of_board_filled(game):
    blank_spaces = game.get_blank_spaces()
    return int((len(blank_spaces) / (game.width * game.height)) * 100)
```

Finally, the evaluation function increments or decrements the variables my_score and opponent_score based on the percentage of the board filled using a ratio as follows:

< 35 % filled
< 35 % and <85% filled (and move is near a board edge)
> 85 % filled

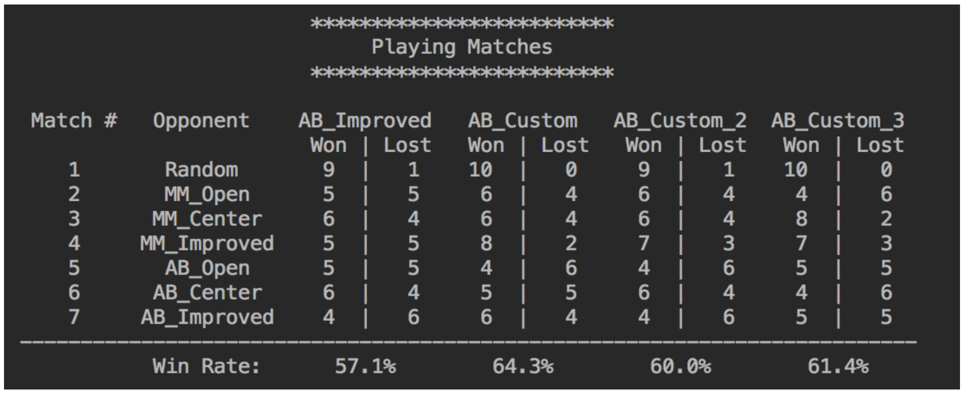
Note: I found that the values assigned to each score could be adjusted to improve the overall performance of the function.



```python
my_score = 0
opponent_score = 0

for move in my_moves:
    if percent_of_board_filled(game) < 35:
        my_score += 10
    elif 35 > percent_of_board_filled(game) < 85 and move_near_edges(move, edges):
        my_score -= 25
    elif percent_of_board_filled(game) > 85 and move_near_edges(move, edges):
        my_score -= 35
    elif not move_near_edges(move, edges):
        my_score += 10

for move in opponent_moves:
    if percent_of_board_filled(game) < 35:
        opponent_score += 10
    elif 35 > percent_of_board_filled(game) < 85 and move_near_edges(move, edges):
        opponent_score -= 25
    elif percent_of_board_filled(game) > 85 and move_near_edges(move, edges):
        opponent_score -= 35
    elif not move_near_edges(move, edges):
        opponent_score += 10

return float(my_score - opponent_score)
```

Project Summary:

All 3 heuristic functions performed with very similar win rate; however the function chosen for custom_score consistently outperformed the other functions by a small margin when playing in multiple tournaments. While my initial estimate was that custom_score_3 would perform better overall, custom_score won by a narrow margin. In this exercise I found that there are many possibilities of heuristic functions, which could be applied, given enough time and opportunity to test.



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

 Match #   Opponent    AB_Improved   AB_Custom   AB_Custom_2  AB_Custom_3
                        Won | Lost   Won | Lost   Won | Lost   Won | Lost
    1       Random       9  |   1    10  |   0     9  |   1    10  |   0
    2       MM_Open      5  |   5     6  |   4     6  |   4     4  |   6
    3      MM_Center     6  |   4     6  |   4     6  |   4     8  |   2
    4     MM_Improved    5  |   5     8  |   2     7  |   3     7  |   3
    5       AB_Open      5  |   5     4  |   6     4  |   6     5  |   5
    6      AB_Center     6  |   4     5  |   5     6  |   4     4  |   6
    7     AB_Improved    4  |   6     6  |   4     4  |   6     5  |   5
--------------------------------------------------------------------------
           Win Rate:      57.1%        64.3%        60.0%        61.4%
```