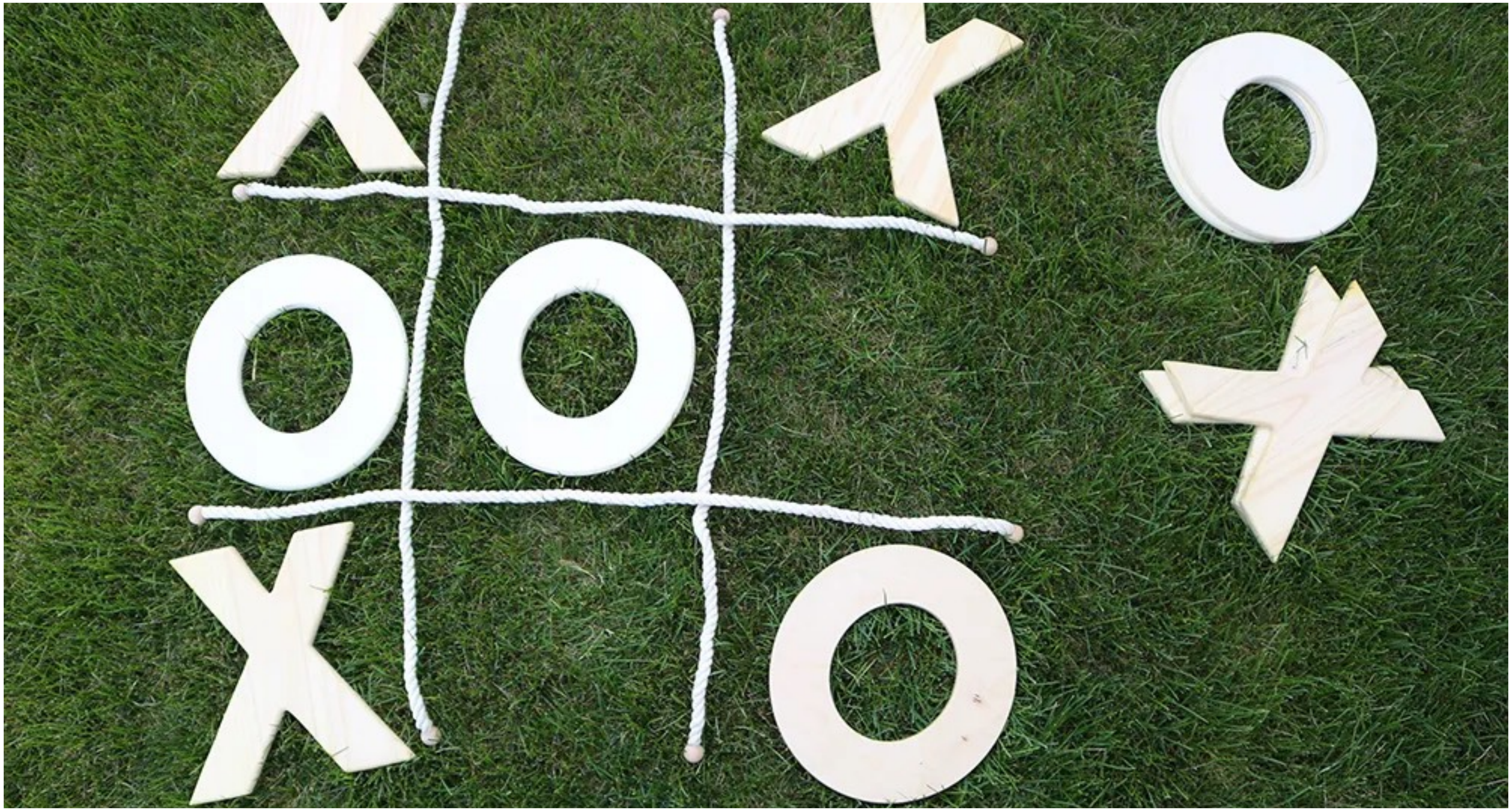
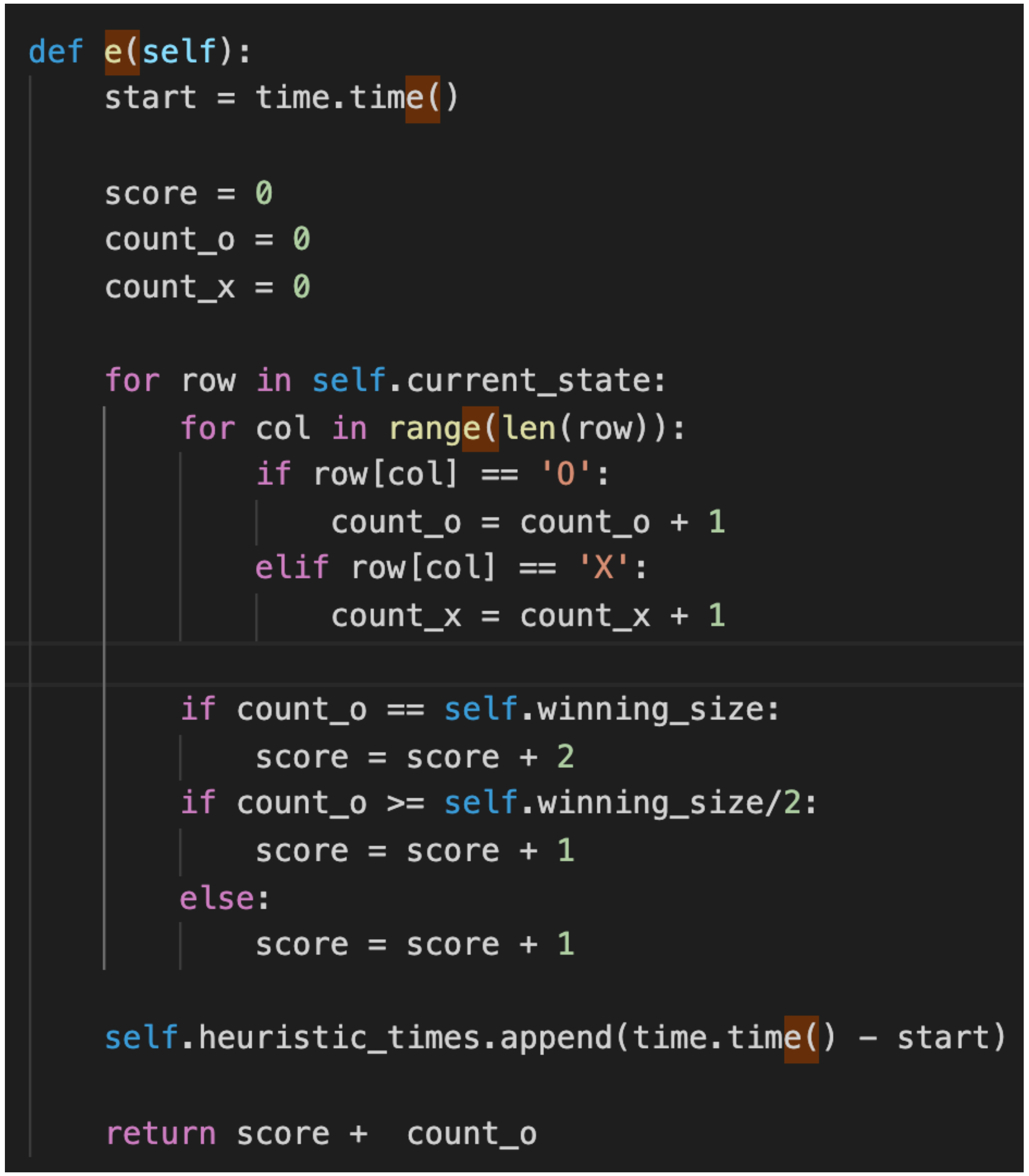
Mini Project 2

WHEN AI FIRST MET YOU



Heuristics

SIMPLE HEURISTIC E



```python
def e(self):
    start = time.time()

    score = 0
    count_o = 0
    count_x = 0

    for row in self.current_state:
        for col in range(len(row)):
            if row[col] == 'O':
                count_o = count_o + 1
            elif row[col] == 'X':
                count_x = count_x + 1


        if count_o == self.winning_size:
            score = score + 2
        if count_o >= self.winning_size/2:
            score = score + 1
        else:
            score = score + 1

    self.heuristic_times.append(time.time() - start)

    return score +  count_o
```

First Heuristic

Simple and Quick

Loops through the 2board array

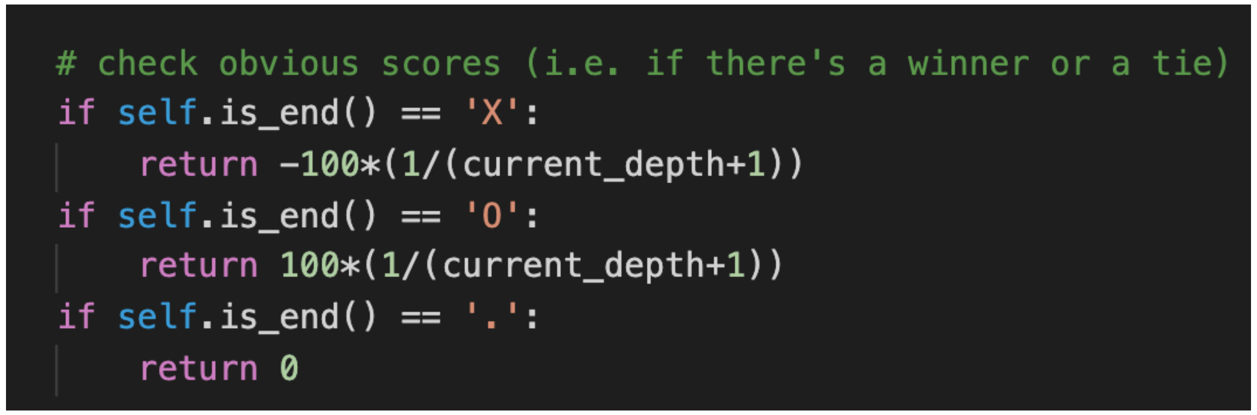
Adds one for the count of a player having played on that score

Gets extra 3 points if the total amount of squares played for that player could make him win

Gets extra point if it is has potential to win

Heuristics

COMPLEX HEURISTIC PART1



```python
# check obvious scores (i.e. if there's a winner or a tie)
if self.is_end() == 'X':
    return -100*(1/(current_depth+1))
if self.is_end() == 'O':
    return 100*(1/(current_depth+1))
if self.is_end() == '.':
    return 0
```

First part of second heuristic

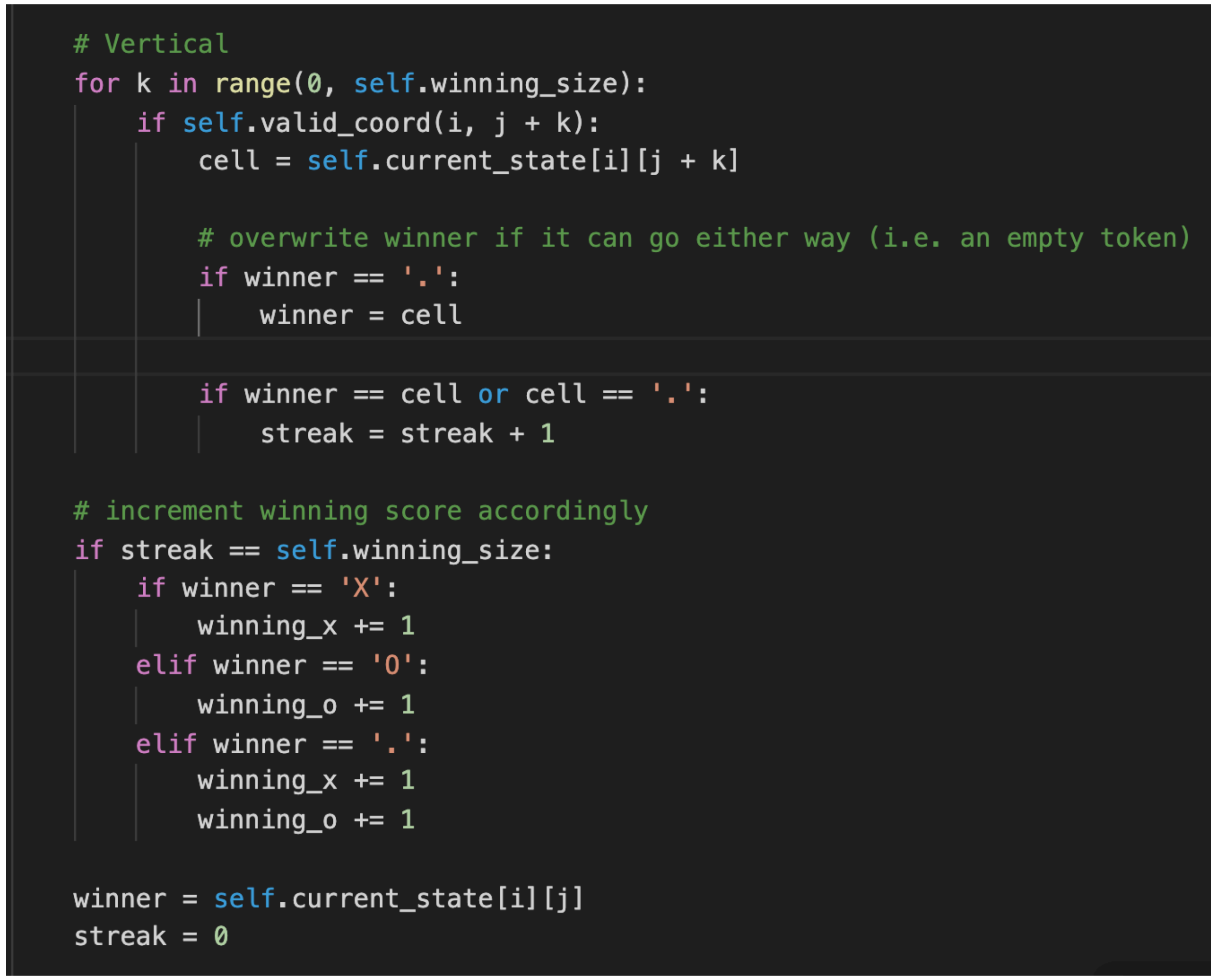
Check if there is an obvious winner or tie

Due to the skeleton we assumed to favour player ‘O’

This is why our final line is the winning_o - wining_x which is the accumulation from the possible winnings of o and the possible winnings of x.

Heuristics

COMPLEX HEURISTIC PART2 VERTICAL



```
# Vertical
for k in range(0, self.winning_size):
    if self.valid_coord(i, j + k):
        cell = self.current_state[i][j + k]

        # overwrite winner if it can go either way (i.e. an empty token)
        if winner == '.':
            winner = cell

        if winner == cell or cell == '.':
            streak = streak + 1

# increment winning score accordingly
if streak == self.winning_size:
    if winner == 'X':
        winning_x += 1
    elif winner == 'O':
        winning_o += 1
    elif winner == '.':
        winning_x += 1
        winning_o += 1

winner = self.current_state[i][j]
streak = 0
```

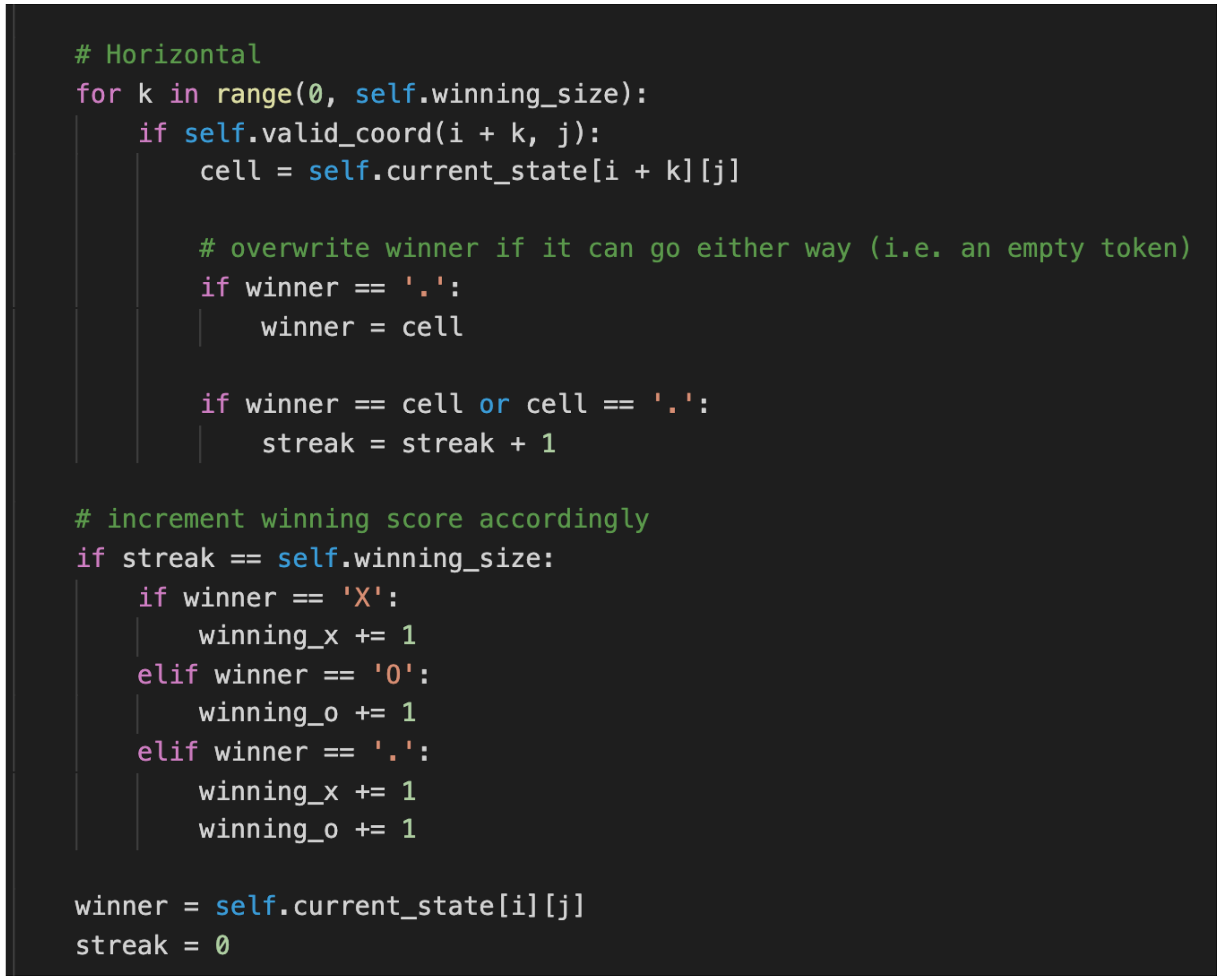
In an inner loop that goes through the 2d array

Check if coordinate is valid

For each position in the 2d array go through what is below. Will add more to the score if that row has enough consecutive items to win the game

Heuristics

COMPLEX HEURISTIC PART2 HORIZONTAL



```python
# Horizontal
for k in range(0, self.winning_size):
    if self.valid_coord(i + k, j):
        cell = self.current_state[i + k][j]

        # overwrite winner if it can go either way (i.e. an empty token)
        if winner == '.':
            winner = cell

        if winner == cell or cell == '.':
            streak = streak + 1

# increment winning score accordingly
if streak == self.winning_size:
    if winner == 'X':
        winning_x += 1
    elif winner == 'O':
        winning_o += 1
    elif winner == '.':
        winning_x += 1
        winning_o += 1

winner = self.current_state[i][j]
streak = 0
```

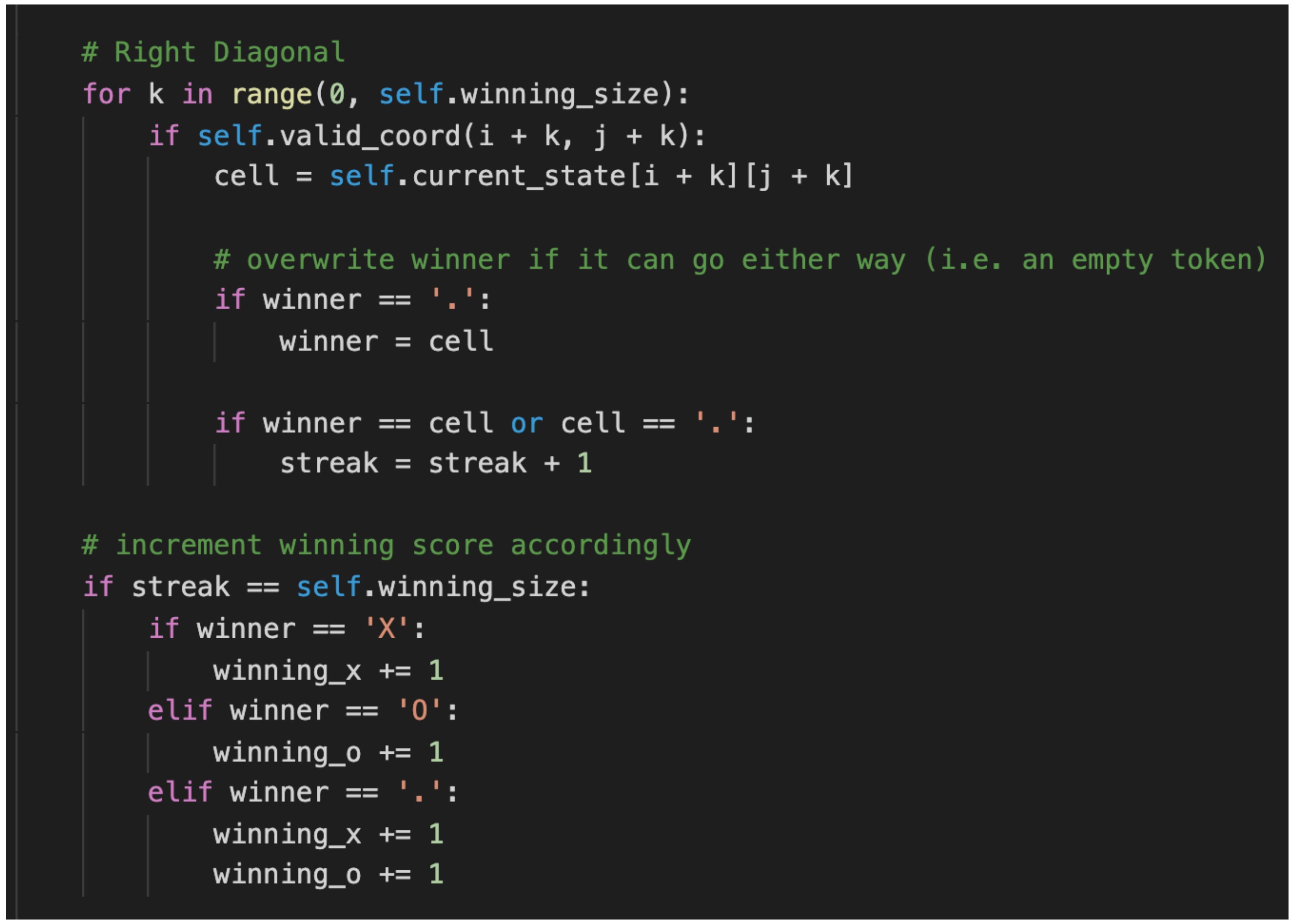
In an inner loop that goes through the 2d array

Check if coordinate is valid

For each position in the 2d array go through what is on the right. Will add more to the score if that row has enough consecutive items to win the game

Heuristics

COMPLEX HEURISTIC PART2 RIGHT TOP DIAGONAL



```
# Right Diagonal
for k in range(0, self.winning_size):
    if self.valid_coord(i + k, j + k):
        cell = self.current_state[i + k][j + k]

        # overwrite winner if it can go either way (i.e. an empty token)
        if winner == '.':
            winner = cell

        if winner == cell or cell == '.':
            streak = streak + 1

# increment winning score accordingly
if streak == self.winning_size:
    if winner == 'X':
        winning_x += 1
    elif winner == 'O':
        winning_o += 1
    elif winner == '.':
        winning_x += 1
        winning_o += 1
```

In an inner loop that goes through the 2d array

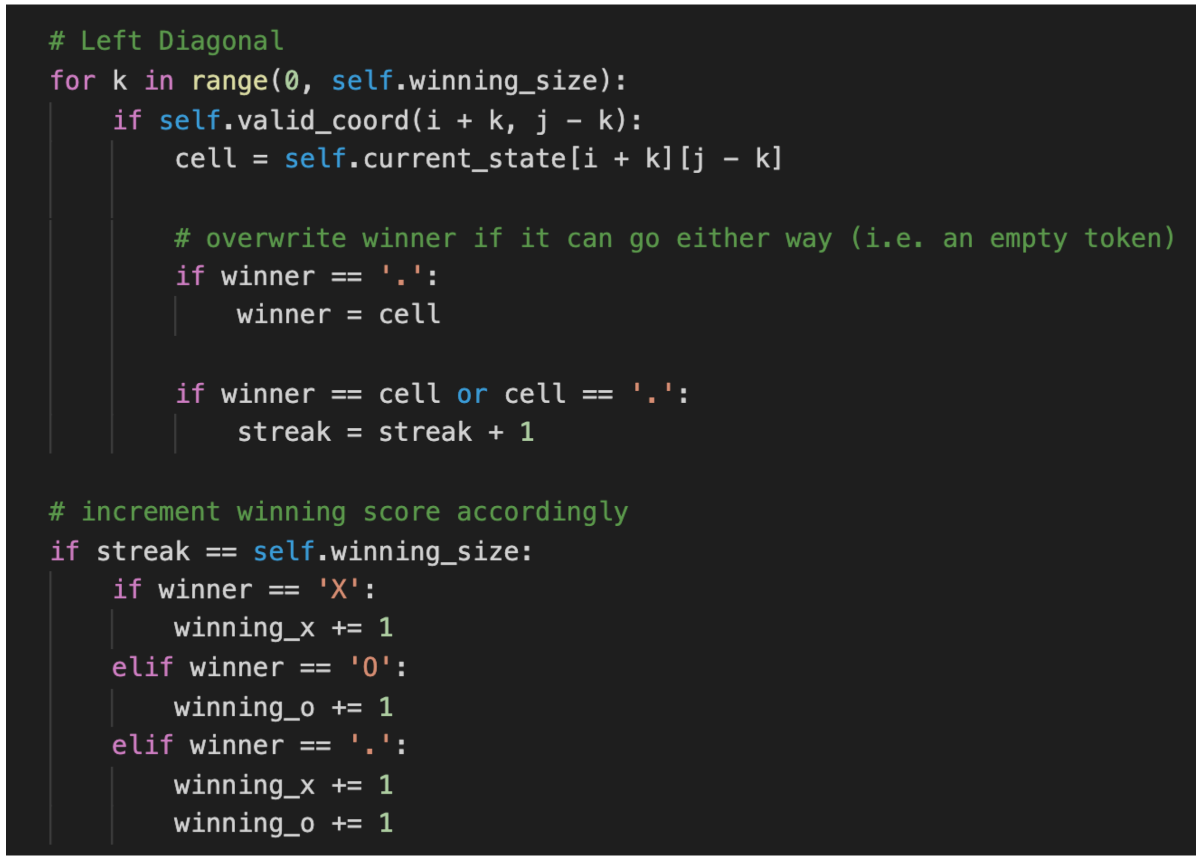
Check if coordinate is valid

For each position in the 2d array go through what is on the top right. Will add more to the score if the number of consecutive items for that player in that direction equals the winning size

Since we loop through all the items in the board, we don't need to check bottom left diagonal

Heuristics

COMPLEX HEURISTIC PART2 RIGHT TOP DIAGONAL



```
# Left Diagonal
for k in range(0, self.winning_size):
    if self.valid_coord(i + k, j - k):
        cell = self.current_state[i + k][j - k]

        # overwrite winner if it can go either way (i.e. an empty token)
        if winner == '.':
            winner = cell

        if winner == cell or cell == '.':
            streak = streak + 1

# increment winning score accordingly
if streak == self.winning_size:
    if winner == 'X':
        winning_x += 1
    elif winner == 'O':
        winning_o += 1
    elif winner == '.':
        winning_x += 1
        winning_o += 1
```

In an inner loop that goes through the 2d array

Check if coordinate is valid

For each position in the 2d array go through what is on the top left. Will add more to the score if the number of consecutive items for that player in that direction equals the winning size

Since we loop through all the items in the board, we don't need to check bottom right diagonal

Average Comparisons Analysis 1

4431	**Board Size (n) - 4**	i. Average of Average evaluation time of heuristic: 5.4758890303662644e-06
	Number Of Blocks (b) - 4	ii. Average of Total states evaluated: 468007.6
	Block Coordinates - 3	iii. Average of Average of average depths: 133.62756997350988
	Winning Size (s) - 1	iv. Average Total number of states evaluated at each depth:
	Max time (t) -1	**6**: 12947220353.3 I **5**: 1799956557.2 I **4**: 215751184.4
	Minimax/Alpha (a) - Alphabeta	**3**: 23742676.2 I **2**: 2393258.5 I **1**: 468007.6
	Player X heuristic - e	v. Average of Average ARD: 6.075939636862706
	Player O heuristic - e2	vi. Average Total Move Count: 115.5
	Max Depth p1 - 6	
	Max Depth p2 - 6	
4435	**Board Size (n) - 4**	i. Average of Average evaluation time of heuristic: 2.5828390959560204e-06
	Number Of Blocks (b) - 4	ii. Average of Total states evaluated: 3797286.0
	Block Coordinates - 4	iii. Average of Average of average depths: 64.77326152515359
	Winning Size (s) - 3	iv. Average Total number of states evaluated at each depth:
	Max time (t) - 5	**6**: 129151207446.0 I **5**: 17906267586.0 I **4**: 2138259264.0
	Minimax/Alpha (a) - Minimax	**3**: 226988529.0 I **2**: 23510214.0 I **1**: 3797286.0
	Player X heuristic - e	v. Average of Average ARD: 4.0
	Player O heuristic - e2	vi. Average Total Move Count: 55.0
	Max Depth p1 - 6	
	Max Depth p2 - 6	
8655	**Board Size (n) - 8**	i. Average of Average evaluation time of heuristic: 1.5047077945751824e-05
	Number Of Blocks (b) - 5	ii. Average of Total states evaluated: 18691679.2553386
	Block Coordinates - 5	iii. Average of Average of average depths: 645.6235752541533
	Winning Size (s) - 5	iv. Average Total number of states evaluated at each depth:
	Max time (t) -5	**6**: 15950981033.954199 I **5**: 5554538325.5050335 I **4**: 272423831.2016424
	Minimax/Alpha (a) - Alphabeta	**3**: 86839719.97524819 I **2**: 5779278.3727095 I **1**: 1336562.3926386
	Player X heuristic - e	v. Average of Average ARD: 6.6386688178277655
	Player O heuristic - e2	vi. Average Total Move Count: 2551.2182685000002
	Max Depth p1 - 6	
	Max Depth p2 - 2	

Important Facts

Look at level 1 from pruning to non pruning, almost more than 10x more states

Pruning increases amount of recursion

Average total states evaluated is only 10x more with double the board size and almost double the winning size. This shows that alpha beta pruning reduced a lot of possible states to be evaluated.

Average evaluation time goes higher with a higher winning size as there is more to iterate through

Game Simulation Analysis 1

4431

Board Size (n) - 4
Number Of Blocks (b) - 4
Block Coordinates - 3
Winning Size (s) - 1
Max time (t) -1
Minimax/Alpha (a) - Alphabeta
Player X heuristic - e
Player O heuristic - e2
Max Depth p1 - 6
Max Depth p2 - 6

Beginning good for depth being 5.94 and 5.95 for both players

The maximum depth would be 6 for both players, meaning the heuristics had the time to go to the depth when needed.

We assume that it didn't reach the full 6 due to some branches being pruned.

Evaluation time start move = 0.291 -> decent

Evaluation time end move = 4.48e-06 -> very small

Huge jump at step 3 from evaluating 13507 to 2635, most probably due to pruning

Game Simulation Analysis 2

4435

Board Size (n)	**4**
Number Of Blocks (b)	**4**
Block Coordinates	**4**
Winning Size (s)	**3**
Max time (t)	**5**
Minimax/Alpha (a)	**Minimax**
Player X heuristic	**e**
Player O heuristic	**e2**
Max Depth p1	**6**
Max Depth p2	**6**

Beginning good for depth being 5.99 and 5.98 for both players. Also shows there was not much pruning at the beginning

Evaluation time start move = 0.5008926 -> 25x more than alpha beta

Max depth is 6 and since there is no pruning it means that it mostly gets to the full depth needed, but not all the time.

Evaluation time end move = 2.08e-06 -> smaller than alpha beta. This means that there was useless overhead once we reached the end of the game

No big jumps, proportionate reductions of nodes being visited at each level.

Game Simulation Analysis 3

8655

Board Size (n) - 8
Number Of Blocks (b) - 6
Block Coordinates - 6
Winning Size (s) - 5
Max time (t) -5
Minimax/Alpha (a) - Alphabeta
Player X heuristic - e
Player O heuristic - e2
Max Depth p1 - 6
Max Depth p2 - 6

Beginning good for depth being 5.99 and 5.98 for both players. Also shows there was not much pruning at the beginning

Evaluation time start move = 5.0009637si, drastic start time increase

The max depth once again is 6 for both players and we found it interesting that at the beginning it goes down to the full depth. This is due to the time.

Evaluation time end move = 1.20e-05. This is double than the previous although since the board size is larger, it means that the function did do some amount of pruning.

Has an ARD of mostly 6,except the end of 5.99 meaning it usually goes the full depth

Game Simulation Analysis 4

8455

Board Size (n) - 8
Number Of Blocks (b) - 5
Block Coordinates - 5
Winning Size (s) - 5
Max time (t) -5
Minimax/Alpha (a) - Alphabeta
Player X heuristic - e
Player O heuristic - e2
Max Depth p1 - 6
Max Depth p2 - 2

One thing I'd like to note is that there is a max depth of 2 for player2 in this scenario.

Even though the game size and winning size are large, for the player using a depth of 2, the evaluation happens quickly.

A 0.0012021seconds time is how long it takes it to evaluate it. This makes sence due to the depth not going deep and recursion not being needed.