

Question 1:

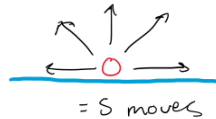
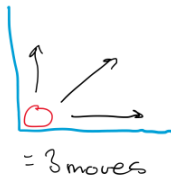
My evaluation function consisted of 6 checks: (not in any ranking order)

1. Prioritizing the middle columns.

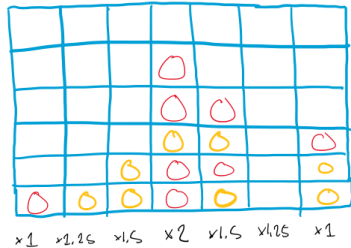
Columns towards the middle of the board have more play room. You can place more connections in more directions (e.g. diagonal, up/down, left/right) as compared to the edge columns where your connections are now limited since you can only make connections to left or right.

Therefore, it is more reasonable to prioritize middle board play

e.g.



① Prioritize middle placements



■ = me
■ = opp

Eval(Board) = sum of my tokens
- sum of opp tokens

$$\begin{aligned}
 &= [2(4) + 1.5(2) + 1(2)] \\
 &- [2(1) + 1.5(4) + 1.25(1) + 1(2)] \\
 &= [8 + 3 + 2] - [2 + 6 + 1.25 + 2] \\
 &= 13 - 11.25 = \boxed{1.75}
 \end{aligned}$$

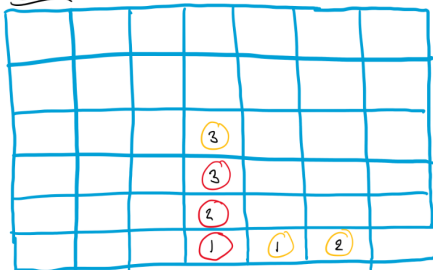
$$\begin{aligned}
 \text{Eval}(\text{Board}) &= [1(C_1 + C_7) + 1.25(C_2 + C_6) + 1.5(C_3 + C_5) + 2(C_4)] \\
 &- [1(O_1 + O_7) + 1.25(O_2 + O_6) + 1.5(O_3 + O_5) + 2(O_4)]
 \end{aligned}$$

where C_n = my coins in column n ,
 O_n = opp coins in column n

2 & 3. Check to see if the last move blocks opponents potential 4 in a row or potential 3 in a row.

This one is obvious, if the opponent has a chance to win, the player will want to eliminate the chance. Prioritizing blocking an opponent, otherwise the opponent will win the game. My code checks to see if the last played move blocks an opponent's 4 in a row, meaning if there were three consecutive same colored coins that were covered by the opposite coin color. Apply the same logic for blocking a potential three in a row. Don't want the opponent to have a three in a row because it can lead to a four in a row.

ex:



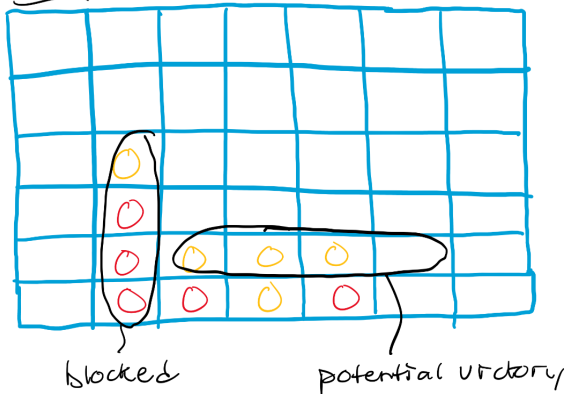
③ = last coin played

if last coin position is
above a 3 in a row:
player value += 20

4 & 5. Number of 3 in a rows or 2 in a rows with an empty slot on either side.

The more potential victories that a player has, the more likely they are to win. If a player has a three in a row that isn't blocked, then that means that in some other play, it can be filled to make a connect 4. Also, in the case where the three in a row cannot be blocked yet, the opponent will have to play around that specific column in order to not set up the player for a connect 4. This evaluation function measures who has more potential to win. My code checks to see if there is a three in a row, and if the spaces to the sides of the three in a row is empty (not yet played in). Same reasoning for the number of two in a rows with an empty slot on either side.

ex:

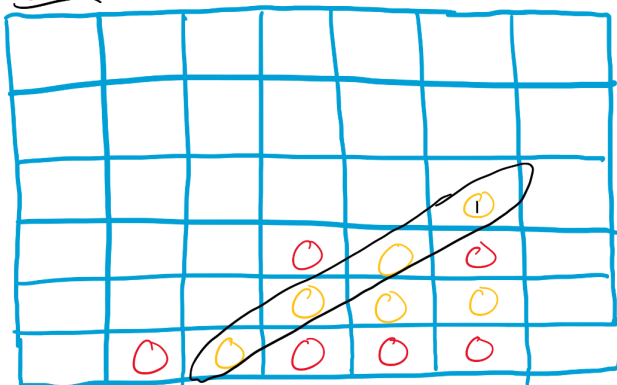


if 3 coins in a row
and empty space on a side:
player value += 20

6. Move results in connect 4

This evaluation function is pretty obvious and simple. If the move results in a victory for the player, a very good evaluation function value will be returned, signifying that the move can result in a victory later on in the game.

ex:



① = last move played

if gameover/connect 4:

player value = 1000000

Question 2:

Code:

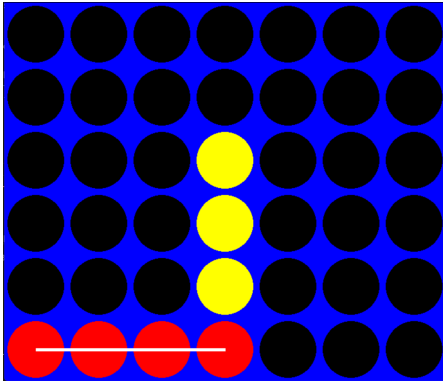
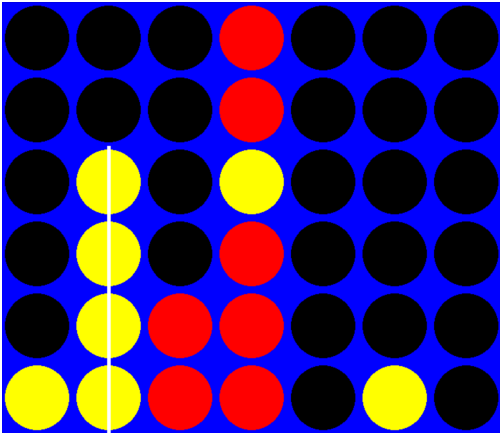
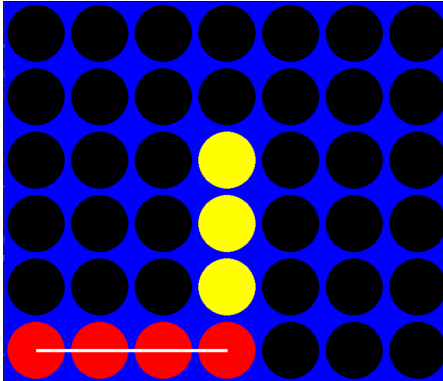
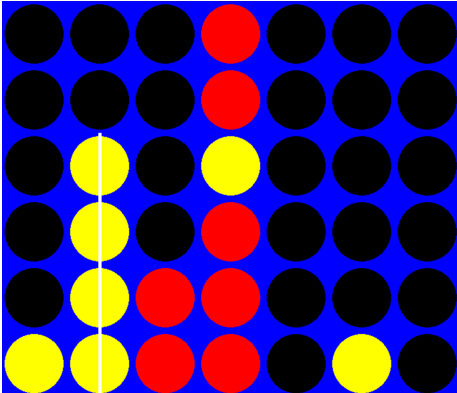
Please see the attached code.

Evaluation Function:

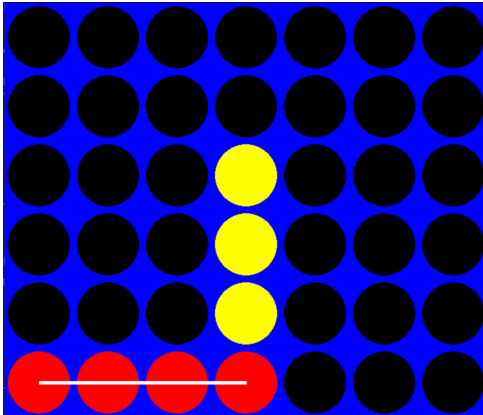
Please see the attached code.

Question 3:

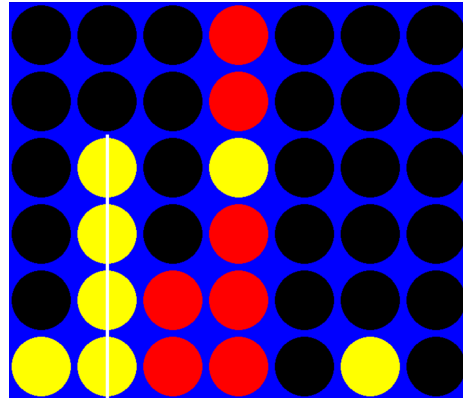
Player 1 = Red, Player 2 = Yellow

| <u>P1 MinimaxAI, P2 StupidAI</u> | <u>P1 StupidAI, P2 MinimaxAI</u> |
|--|---|
| <p>python3 main.py -p1 minimaxAI -p2 stupidAI -limit_players 1,2 -visualize True -verbose True -seed 1</p>  | <p>python3 main.py -p1 stupidAI -p2 minimaxAI -limit_players 1,2 -visualize True -verbose True -seed 1</p>  |
| <p>python3 main.py -p1 minimaxAI -p2 stupidAI -limit_players 1,2 -visualize True -verbose True -seed 2</p>  | <p>python3 main.py -p1 stupidAI -p2 minimaxAI -limit_players 1,2 -visualize True -verbose True -seed 2</p>  |
| <p>python3 main.py -p1 minimaxAI -p2 stupidAI -limit_players 1,2 -visualize True -verbose</p> | <p>python3 main.py -p1 stupidAI -p2 minimaxAI -limit_players 1,2 -visualize True -verbose</p> |

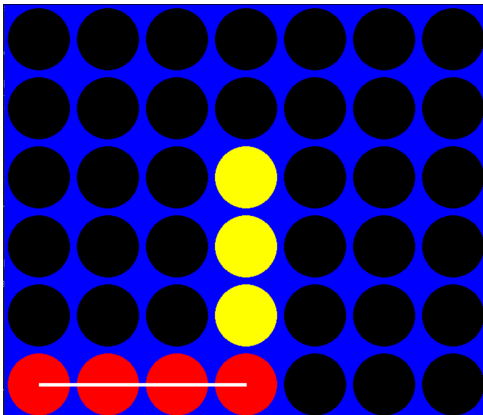
True -seed 3



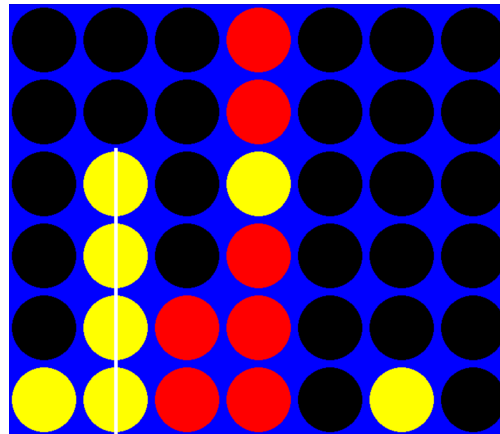
True -seed 3



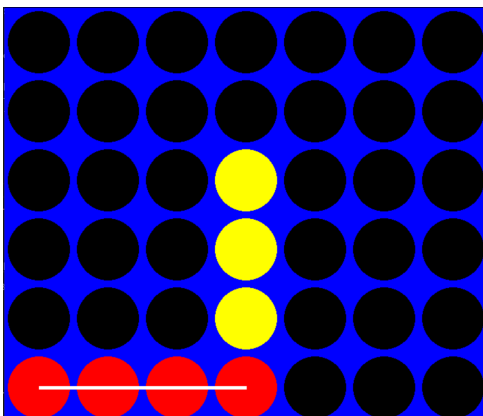
python3 main.py -p1 minimaxAI -p2 stupidAI
-limit_players 1,2 -visualize True -verbose
True -seed 4



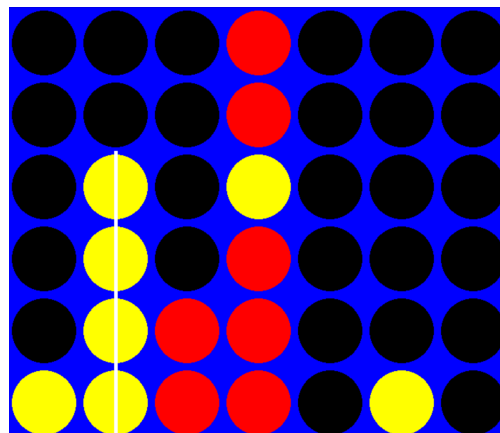
python3 main.py -p1 stupidAI -p2 minimaxAI
-limit_players 1,2 -visualize True -verbose
True -seed 4

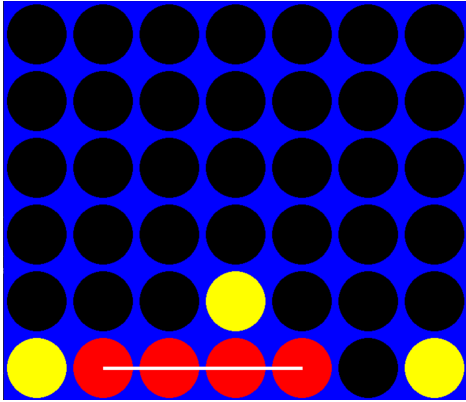
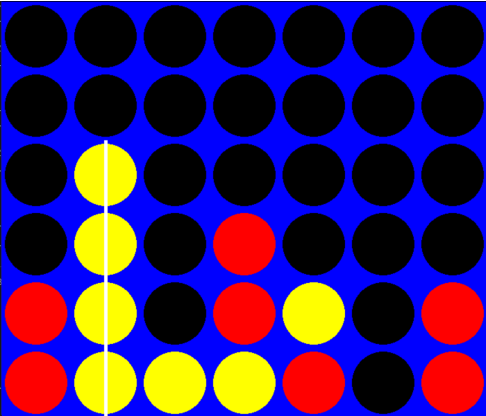
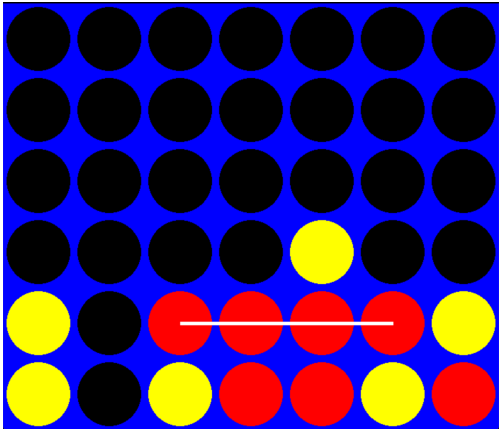
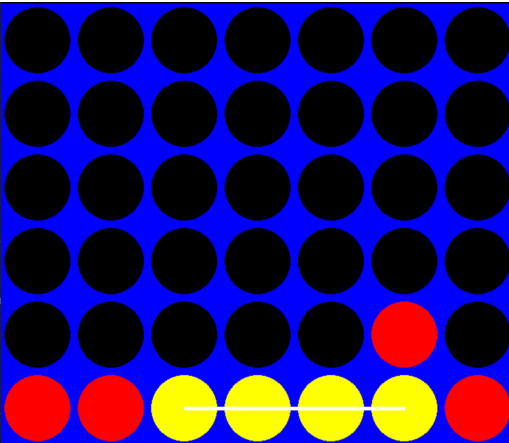


python3 main.py -p1 minimaxAI -p2 stupidAI
-limit_players 1,2 -visualize True -verbose
True -seed 5

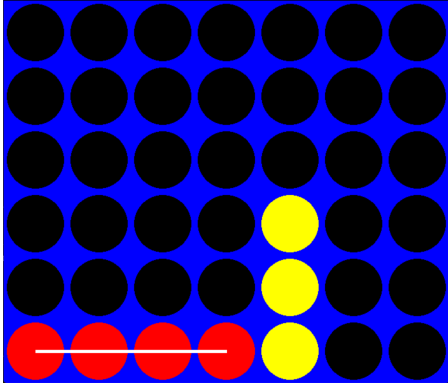


python3 main.py -p1 stupidAI -p2 minimaxAI
-limit_players 1,2 -visualize True -verbose
True -seed 5

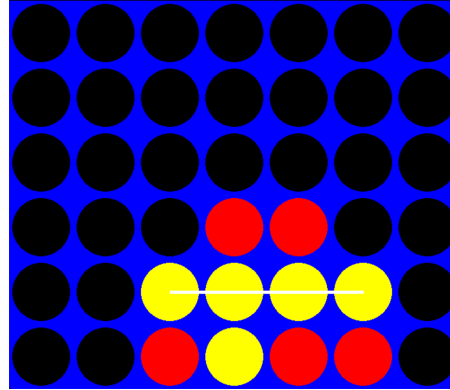


| <u>P1 MinimaxAI, P2 RandomAI</u> | <u>P1 RandomAI, P2 MinimaxAI</u> |
|---|--|
| <p>python3 main.py -p1 minimaxAI -p2 randomAI -limit_players 1,2 -visualize True -verbose True -seed 1</p>  | <p>python3 main.py -p1 randomAI -p2 minimaxAI -limit_players 1,2 -visualize True -verbose True -seed 1</p>  |
| <p>python3 main.py -p1 minimaxAI -p2 randomAI -limit_players 1,2 -visualize True -verbose True -seed 2</p>  | <p>python3 main.py -p1 randomAI -p2 minimaxAI -limit_players 1,2 -visualize True -verbose True -seed 2</p>  |

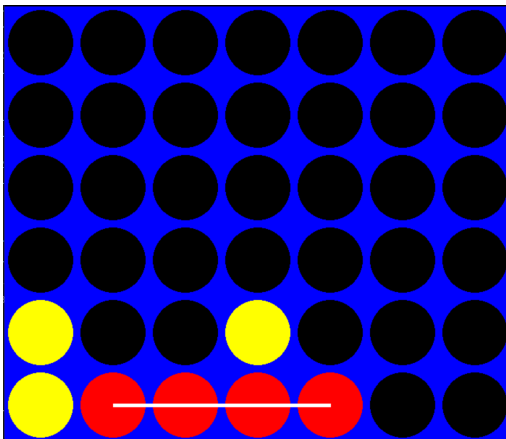
```
python3 main.py -p1 minimaxAI -p2
randomAI -limit_players 1,2 -visualize True
-verbose True -seed 3
```



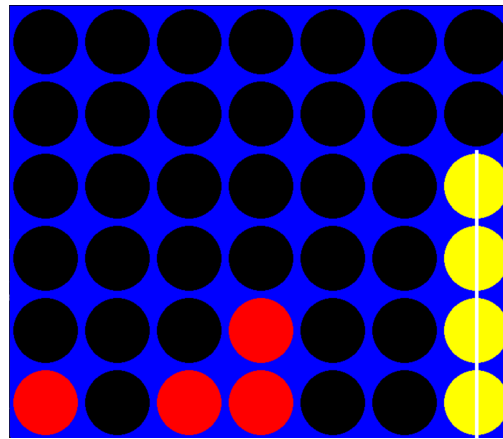
```
python3 main.py -p1 randomAI -p2
minimaxAI -limit_players 1,2 -visualize True
-verbose True -seed 3
```



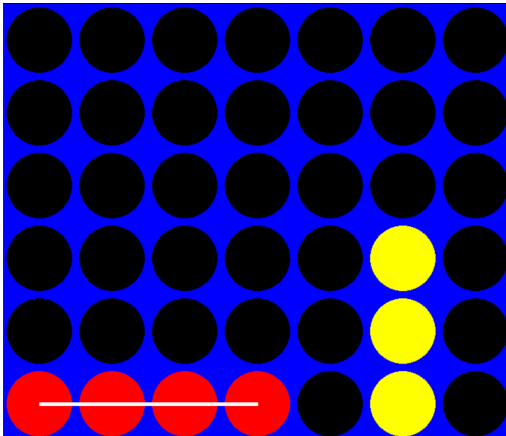
```
python3 main.py -p1 minimaxAI -p2
randomAI -limit_players 1,2 -visualize True
-verbose True -seed 4
```



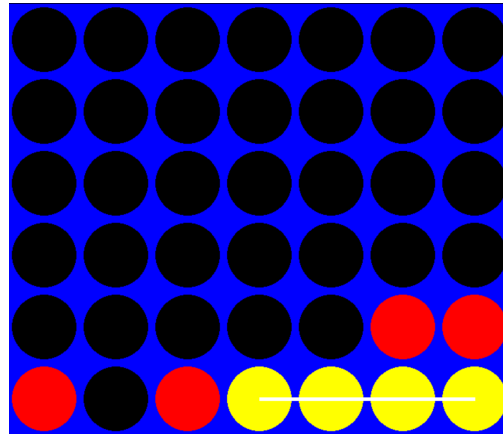
```
python3 main.py -p1 randomAI -p2
minimaxAI -limit_players 1,2 -visualize True
-verbose True -seed 4
```

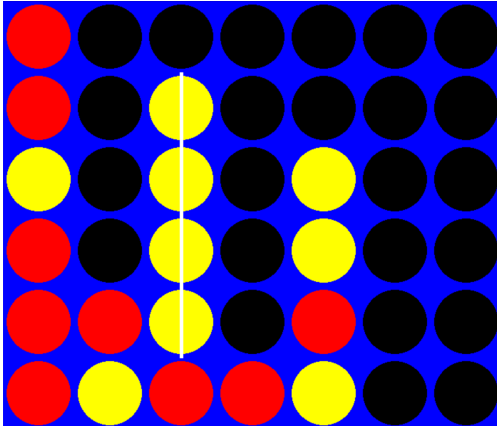
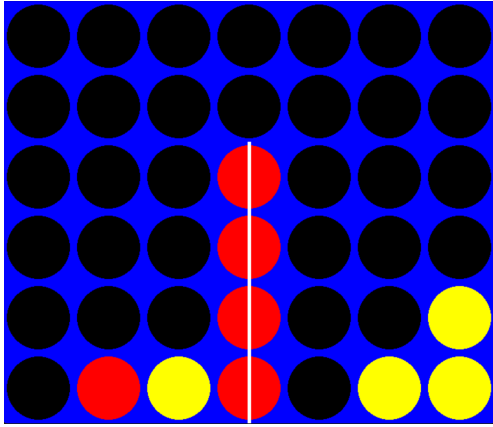
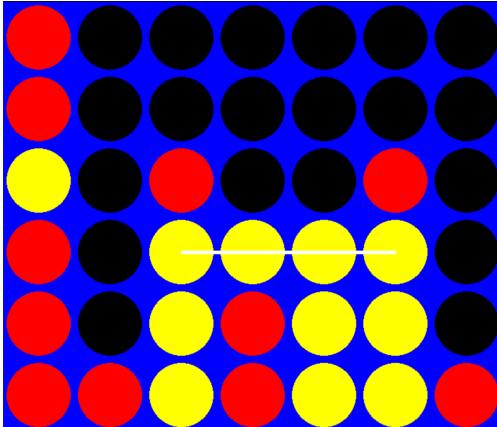
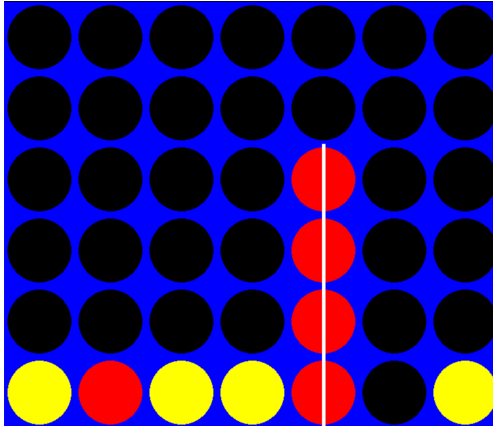


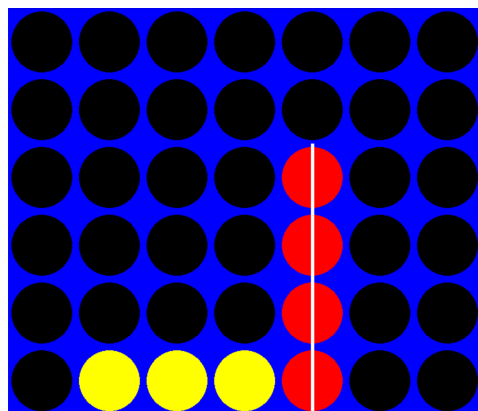
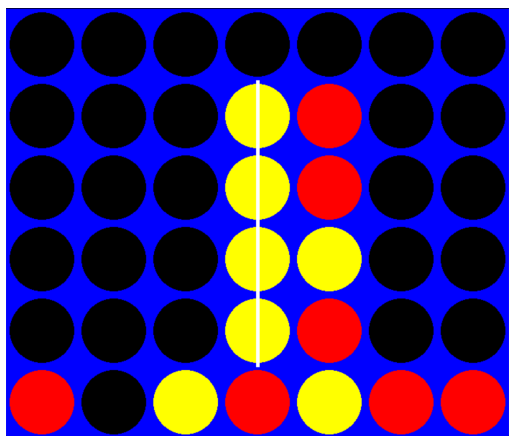
```
python3 main.py -p1 minimaxAI -p2
randomAI -limit_players 1,2 -visualize True
-verbose True -seed 5
```



```
python3 main.py -p1 randomAI -p2
minimaxAI -limit_players 1,2 -visualize True
-verbose True -seed 5
```

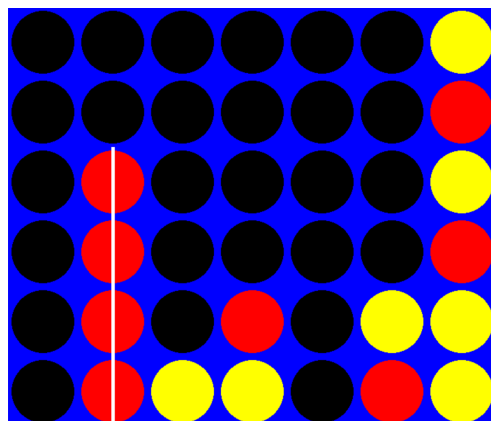
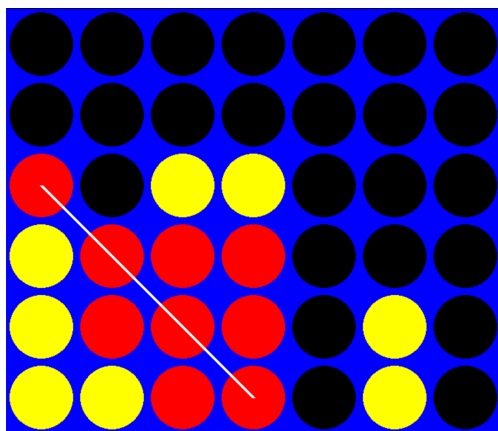


| <u>P1 MinimaxAI, P2 monteCarloAI</u> | <u>P1 monteCarloAI, P2 MinimaxAI</u> |
|--|---|
| <p>python3 main.py -p1 minimaxAI -p2 monteCarloAI -limit_players 1,2 -visualize True -verbose True -seed 1</p>  | <p>python3 main.py -p1 monteCarloAI -p2 minimaxAI -limit_players 1,2 -visualize True -verbose True -seed 1</p>  |
| <p>python3 main.py -p1 minimaxAI -p2 monteCarloAI -limit_players 1,2 -visualize True -verbose True -seed 2</p>  | <p>python3 main.py -p1 monteCarloAI -p2 minimaxAI -limit_players 1,2 -visualize True -verbose True -seed 2</p>  |
| <p>python3 main.py -p1 minimaxAI -p2 monteCarloAI -limit_players 1,2 -visualize True -verbose True -seed 3</p> | <p>python3 main.py -p1 monteCarloAI -p2 minimaxAI -limit_players 1,2 -visualize True -verbose True -seed 3</p> |



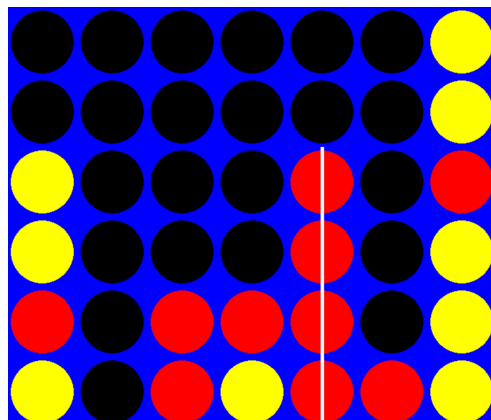
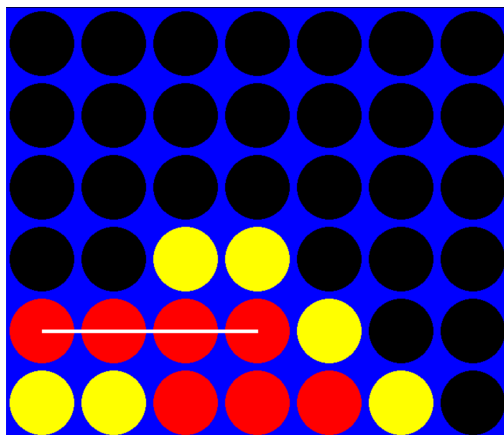
python3 main.py -p1 minimaxAI -p2
monteCarloAI -limit_players 1,2 -visualize
True -verbose True -seed 4

python3 main.py -p1 monteCarloAI -p2
minimaxAI -limit_players 1,2 -visualize True
-verbose True -seed 4



python3 main.py -p1 minimaxAI -p2
monteCarloAI -limit_players 1,2 -visualize
True -verbose True -seed 5

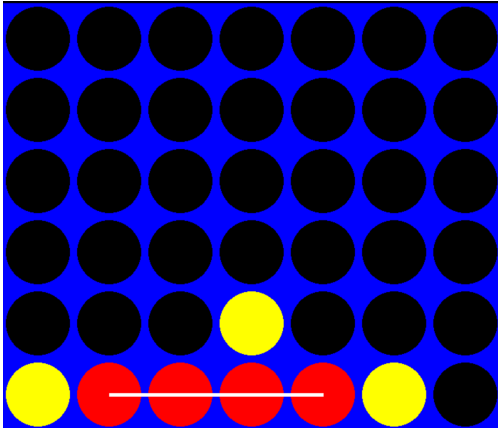
python3 main.py -p1 monteCarloAI -p2
minimaxAI -limit_players 1,2 -visualize True
-verbose True -seed 5



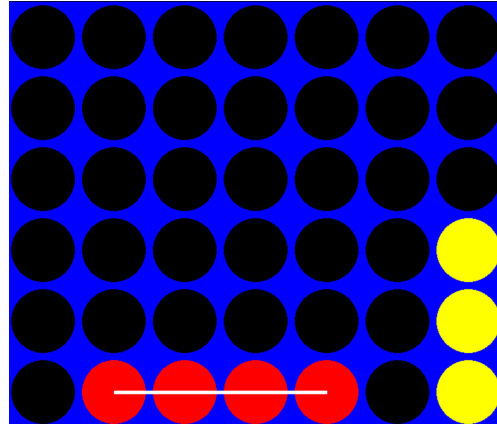
python3 main.py -p1 minimaxAI -p2

python3 main.py -p1 monteCarloAI -p2

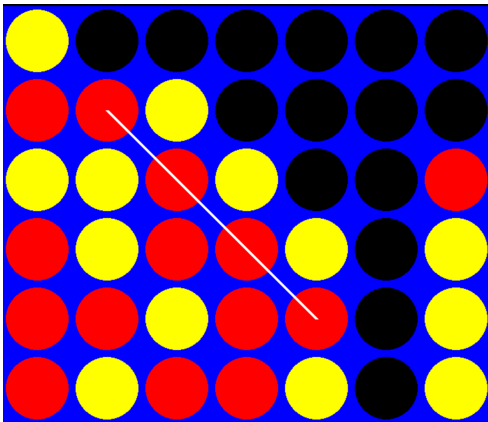
monteCarloAI -limit_players 1,2 -visualize
True -verbose True -seed 6



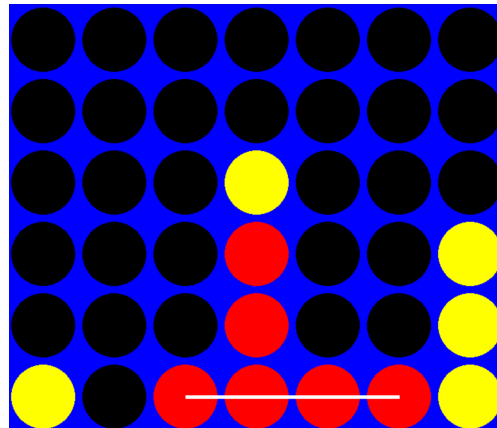
minimaxAI -limit_players 1,2 -visualize True
-verbose True -seed 6



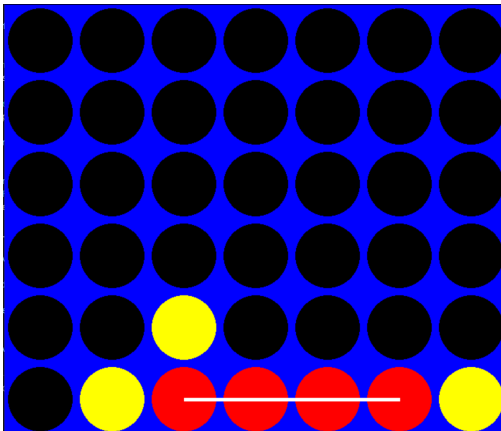
python3 main.py -p1 minimaxAI -p2
monteCarloAI -limit_players 1,2 -visualize
True -verbose True -seed 7



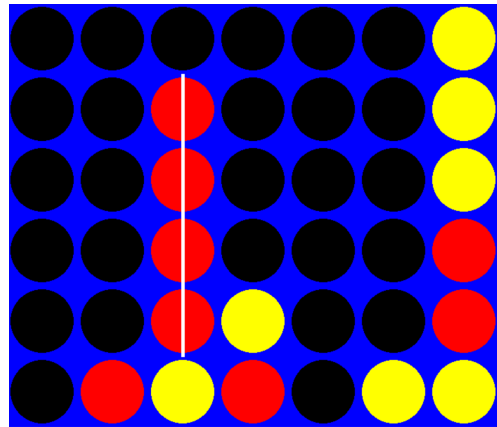
python3 main.py -p1 monteCarloAI -p2
minimaxAI -limit_players 1,2 -visualize True
-verbose True -seed 7



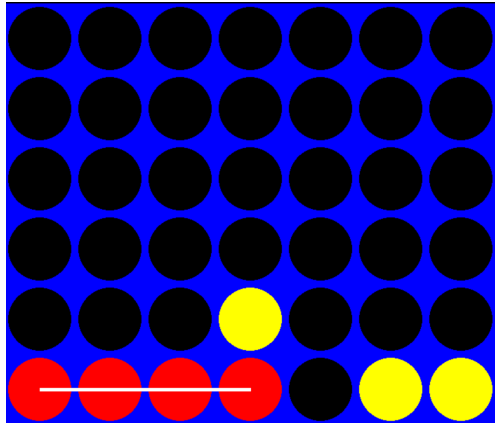
python3 main.py -p1 minimaxAI -p2
monteCarloAI -limit_players 1,2 -visualize
True -verbose True -seed 8



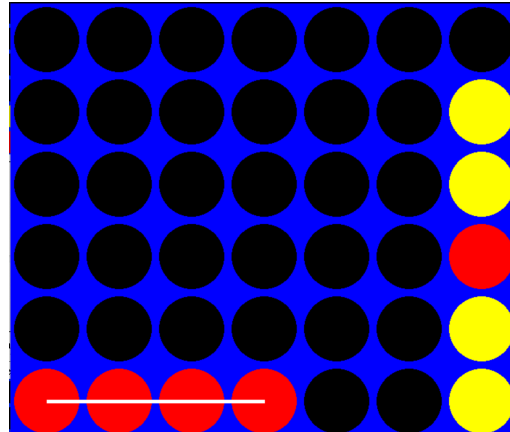
python3 main.py -p1 monteCarloAI -p2
minimaxAI -limit_players 1,2 -visualize True
-verbose True -seed 8



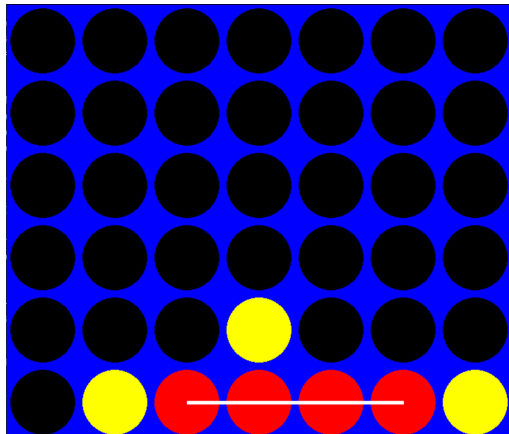
```
python3 main.py -p1 minimaxAI -p2
monteCarloAI -limit_players 1,2 -visualize
True -verbose True -seed 9
```



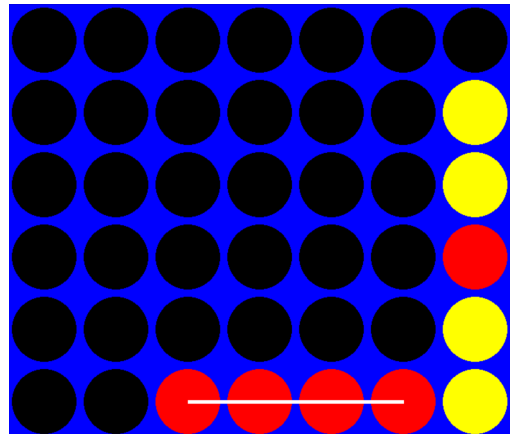
```
python3 main.py -p1 monteCarloAI -p2
minimaxAI -limit_players 1,2 -visualize True
-verbose True -seed 9
```



```
python3 main.py -p1 minimaxAI -p2
monteCarloAI -limit_players 1,2 -visualize
True -verbose True -seed 10
```



```
python3 main.py -p1 monteCarloAI -p2
minimaxAI -limit_players 1,2 -visualize True
-verbose True -seed 10
```



Question 4:

Code:

Please see the attached code.

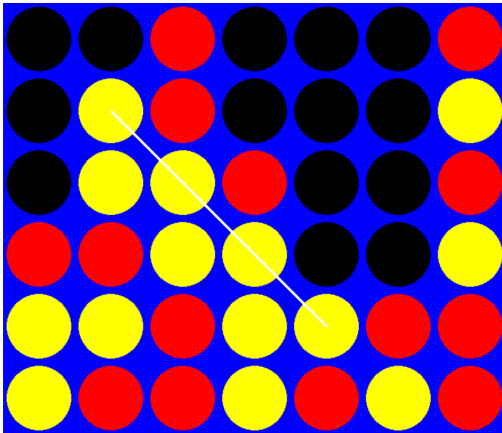
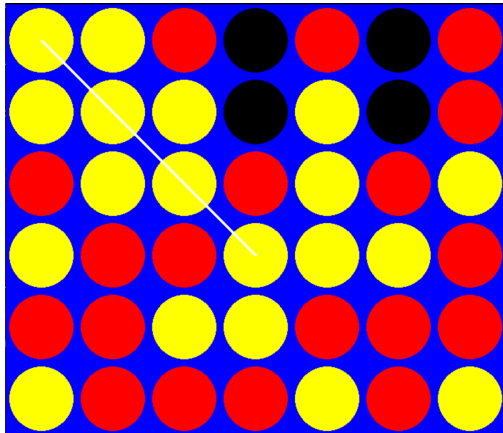
Evaluation Function:

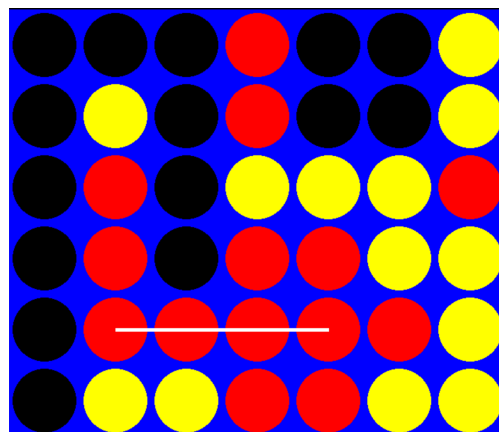
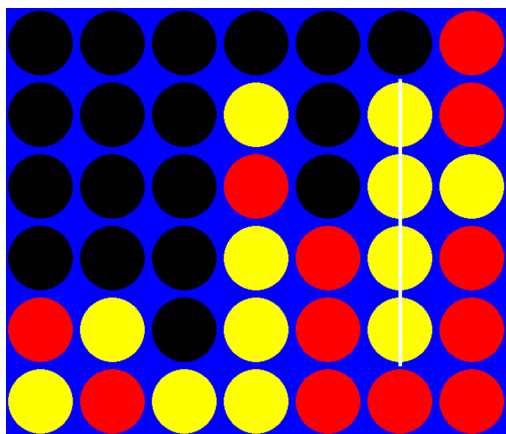
Please see the attached code.

Successor Function:

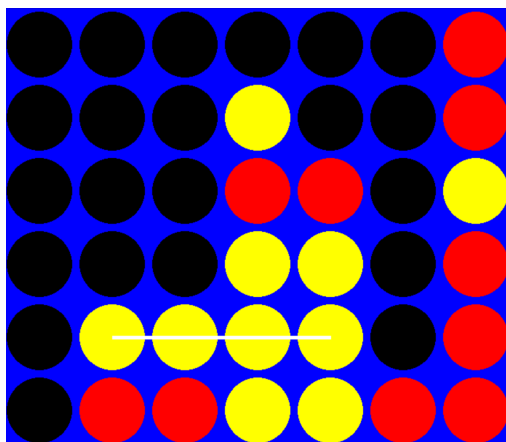
My successor function (**called rearrange()**) just orders the states expanded from plays made in the middlemost columns to the outwards columns. This is based on the evaluation function I explained in question 1. Basically, the columns in the middle of the board have more play room, moves in these columns can result in more options for a connect 4 later on in the game. Opening states where moves have been made in the middle columns first can help prune the tree more efficiently. Searching down these branches may result in finding a value where we will not have to look down other branches.

Question 5:

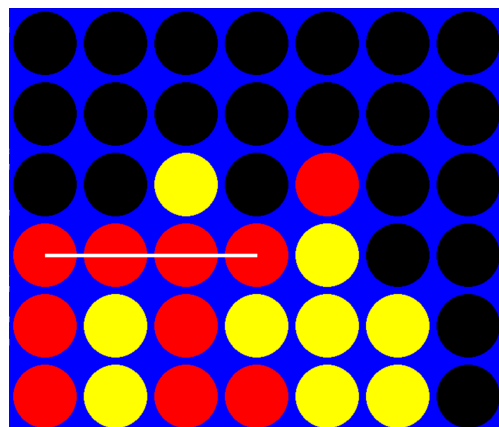
| <u>P1 alphaBetaAI, P2 monteCarloAI</u> | <u>P1 monteCarloAI, P2 alphaBetaAI</u> |
|--|---|
| <pre>python3 main.py -p1 alphaBetaAI -p2 monteCarloAI -limit_players 1,2 -visualize True -verbose True -seed 1</pre>  | <pre>python3 main.py -p1 monteCarloAI -p2 alphaBetaAI -limit_players 1,2 -visualize True -verbose True -seed 1</pre>  |
| <pre>python3 main.py -p1 alphaBetaAI -p2 monteCarloAI -limit_players 1,2 -visualize True -verbose True -seed 2</pre> | <pre>python3 main.py -p1 monteCarloAI -p2 alphaBetaAI -limit_players 1,2 -visualize True -verbose True -seed 2</pre> |



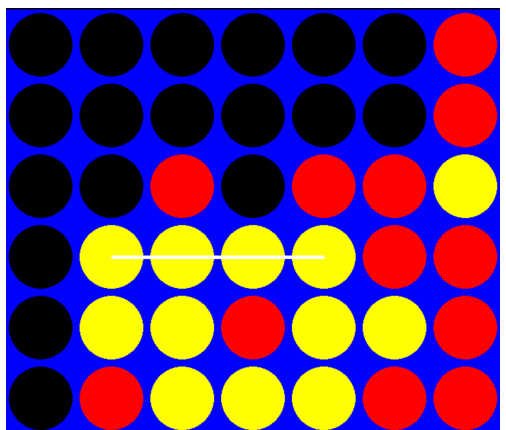
```
python3 main.py -p1 alphaBetaAI -p2
monteCarloAI -limit_players 1,2 -visualize
True -verbose True -seed 3
```



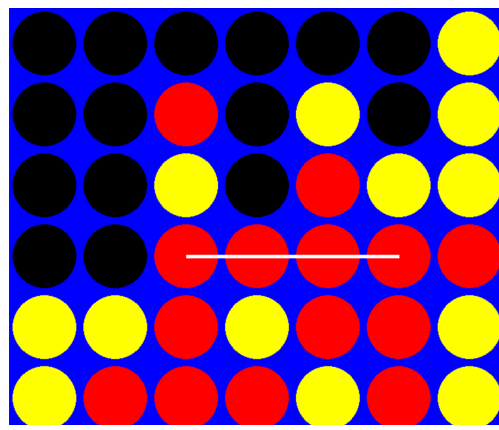
```
python3 main.py -p1 monteCarloAI -p2
alphaBetaAI -limit_players 1,2 -visualize
True -verbose True -seed 3
```



```
python3 main.py -p1 alphaBetaAI -p2
monteCarloAI -limit_players 1,2 -visualize
True -verbose True -seed 4
```



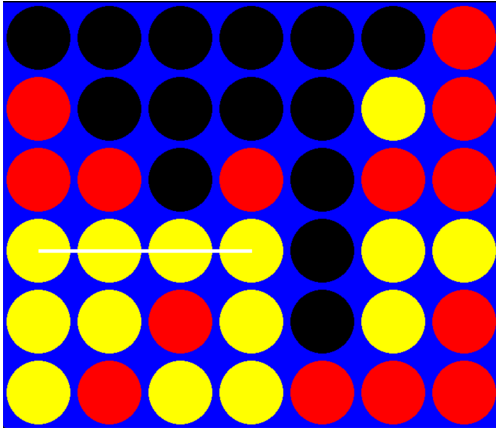
```
python3 main.py -p1 monteCarloAI -p2
alphaBetaAI -limit_players 1,2 -visualize
True -verbose True -seed 4
```



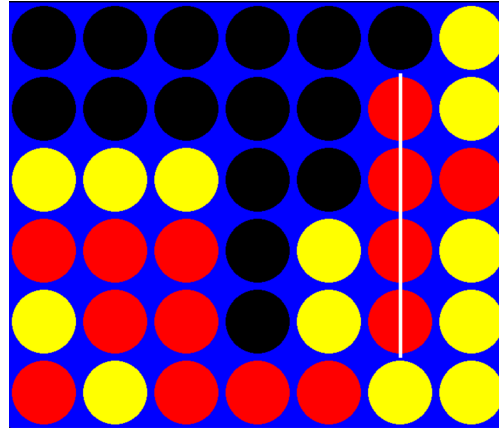
```
python3 main.py -p1 alphaBetaAI -p2
```

```
python3 main.py -p1 monteCarloAI -p2
```

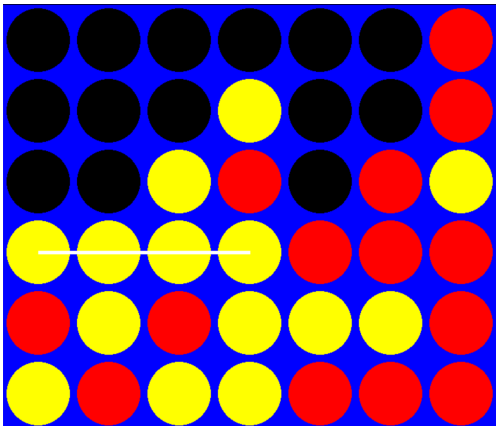
monteCarloAI -limit_players 1,2 -visualize
True -verbose True -seed 5



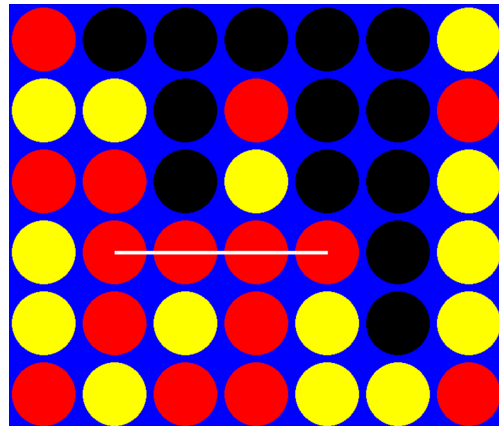
alphaBetaAI -limit_players 1,2 -visualize
True -verbose True -seed 5



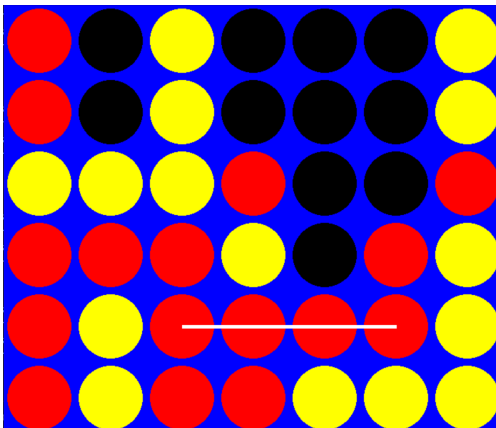
python3 main.py -p1 alphaBetaAI -p2
monteCarloAI -limit_players 1,2 -visualize
True -verbose True -seed 6



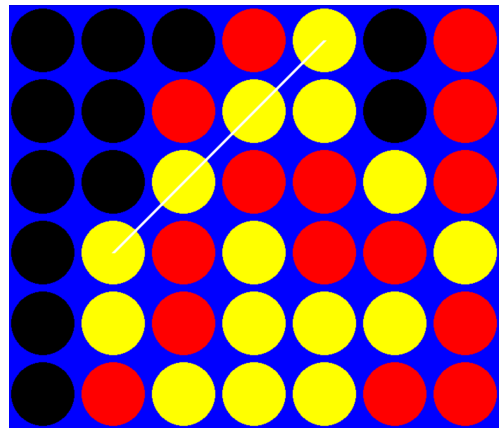
python3 main.py -p1 monteCarloAI -p2
alphaBetaAI -limit_players 1,2 -visualize
True -verbose True -seed 6



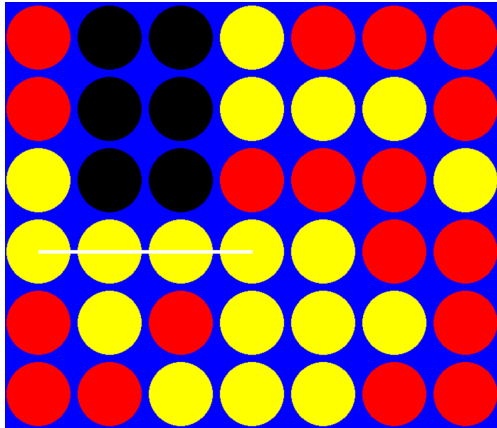
python3 main.py -p1 alphaBetaAI -p2
monteCarloAI -limit_players 1,2 -visualize
True -verbose True -seed 7



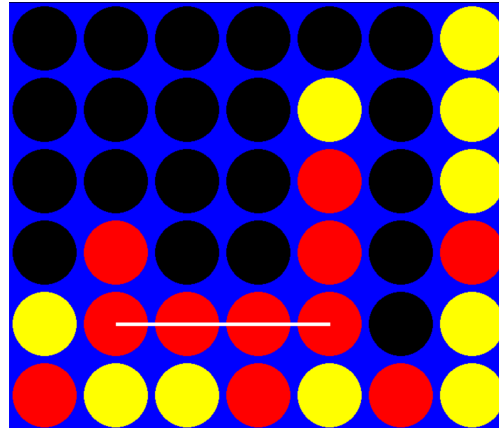
python3 main.py -p1 monteCarloAI -p2
alphaBetaAI -limit_players 1,2 -visualize
True -verbose True -seed 7



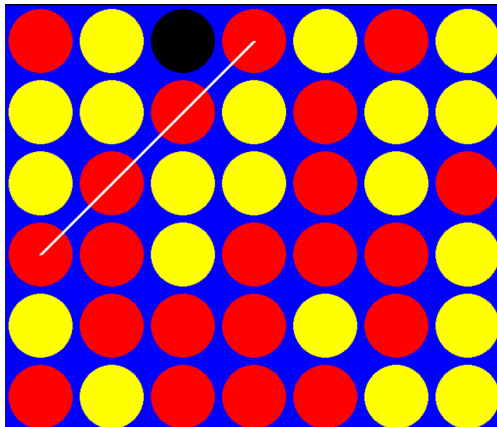
```
python3 main.py -p1 alphaBetaAI -p2
monteCarloAI -limit_players 1,2 -visualize
True -verbose True -seed 8
```



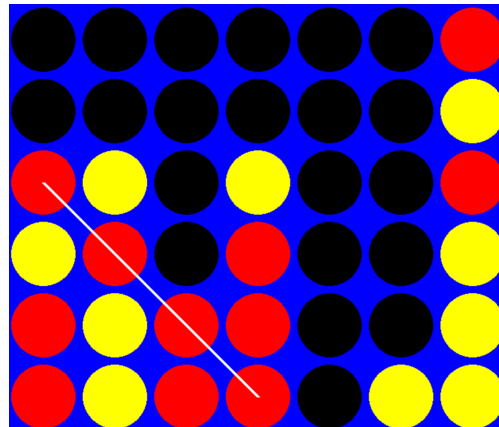
```
python3 main.py -p1 monteCarloAI -p2
alphaBetaAI -limit_players 1,2 -visualize
True verbose True -seed 8
```



```
python3 main.py -p1 alphaBetaAI -p2
monteCarloAI -limit_players 1,2 -visualize
True -verbose True -seed 9
```



```
python3 main.py -p1 monteCarloAI -p2
alphaBetaAI -limit_players 1,2 -visualize
True -verbose True -seed 9
```



```
python3 main.py -p1 alphaBetaAI -p2
monteCarloAI -limit_players 1,2 -visualize
True -verbose True -seed 10
```

```
python3 main.py -p1 monteCarloAI -p2
alphaBetaAI -limit_players 1,2 -visualize
True -verbose True -seed 10
```

