卒業研究報告書

題目

# コネクト４でのゲームAI 開発

指導教員

石水 隆　講師

報告者

15-1-037-0064

藤原 直之

近畿大学理工学部情報学科

平成 31 年 1 月 31 日提出

## 概要

今日，ゲームにおける人工知能，すなわちゲーム AI が注目されている．特に注目すべきなのはディープラーニングを用いた将棋・囲碁等のゲーム AI である．約 10 年前は，莫大な手数を推測するのが難しく，ゲーム AI はアマチュアレベルに留まっていたが，ディープラーニングの登場により，莫大な手数も推測できるようになり，人間界のトッププロに何度も勝利をした．将棋や囲碁のような複雑なゲームに対して，ディープラーニングが強力であることが分かる．では、手数が少ないゲームではどうか．従来の方法でも可能だが，ディープラーニングを用いればさらに勝率を上げることができるのか．そこで本研究では，探索範囲の狭いゲームでもディープラーニングが有効なのかを検証する．

本研究では，コネクト４と呼ばれる重力付き四目並べというボードゲームを題材に python 言語でゲーム AI を開発する．本研究テーマは 2 段階に分かれる．まずディープラーニングの学習に用いる対戦相手となるプログラムを作成する．このプログラムは，全てランダムに打つもの，リーチの時は必ず四つ目のマスに打つもの，最初の 4 手までは最善手のマスに打つものというように，ディープラーニングを行うゲーム AI の学習レベルに応じて難易度を調整できるものを作成する．次にディープライニングを行うゲーム AI を作成し，対戦相手プログラムと対戦させて学習を行うことで，学習により，コネクト４のような探索範囲の狭いゲームでも最善手にたどり着けるのか検証する．

# 目次

# 1. 序論

## 1.1 本研究の背景

近年，コンピュータの急速な進化により，人工知能の技術が急速に進化している．その中で，機械学習と呼ばれる人間の学習能力をコンピュータに実現させる技術のうち，ディープラーニング（深層学習）と呼ばれる手法が注目されている．ディープラーニングの手法を利用し，スマートフォンで人間の話す言葉を理解し，将来の自動車の自動運転の実用化の兆しも見せている．

ディープラーニングは最近では将棋やチェス，囲碁等のゲームでよく利用されており，プロを上回るものも出てきている．

## 1.2 ディープラーニング

ディープラーニング（深層学習）[1]は，機械学習と呼ばれる分野の手法の一つである．ディープラーニングは「画像認識」「音声認識」「自然言語処理」などの分野で大きな成果を挙げている．特に，「画像認識」という分野では，画像のクラス分類の競技会（ILSVRC）で毎年，画像のクラス分類の精度が向上している[1]．2012 年から本格的に画像処理に対してディープラーニングが用いられ，判定エラー率が 16.4%と，2011 年の 25.8%と比べて 9.4 ポイント減少．さらに，ニュートラルネットワークの多層化は進み，2015 年には判別エラー率 3.57%まで減少し，ヒトの眼を超える程の精度となった[1]．

## 1.3 ディープラーニングによるゲーム AI

ディープラーニングは，先述の通り音声や画像等の分野に使われているが，これらにとどまらず，2016 年 3 月に韓国のトップレベルの囲碁棋士に，ディープラーニングを用いた「AlphaGo（アルファ碁）」が勝利（4 勝 1 敗）した[10]．その AlphaGo が強化学習と呼ばれる手法で作成する．[1]．強化学習とは，試行錯誤をする中で，報酬や罰を与えることで，その高度を強化し，自らその行動を学習するという学習方法である．

将棋では，山本一成が 2008 年に開発をした ponanza[20]が挙げられる．人間界のプロと対局する将棋電王戦では ponanza は第 2 回（2013 年）[15]，第 3 回（2014 年）[16]，FINAL（2015 年）[17]に登場し，いずれも勝利を収めている．また，2016 年[18]と 2017 年の電王戦[19]でも勝利を収めている．

## 1.4 コネクト 4

「コネクト 4 (重力付き四目並べ)」とは，アメリカ生まれのボードゲームで，縦 6 段，横 7 列から成る垂直なボードに，黄色または赤色の円盤状の駒を入れていき，先に縦，横，斜めに４つ連続して並べた方が勝ちになるゲームである．[4]

コネクト 4 は，J.D.Allen により先手必勝であることが示されている[3]．1988 年，J.D.Allen はコネクト 4 の完全解析を行い，双方最善手を打った場合，21 手で先手が勝つことを示した．同じ年に，Victor Allis も同様の解析を行い，同じ結論に達している[4]．

コネクト 4 の既存の AI には，Connect 4 Solver [12]などがある．Pascal Pons が作成した Connect 4 Solver は，アルファ・ベータ法が用いられている． Connect 4 Solver は，初期局面から勝負の付いた最終局面まで短時間で探索することが可能であり，最善手を打つことができる．従って，Connect 4 Solver は先手なら必ず勝ち，後手でも先手が最善手を打たない限り勝つことができる．

## 1.5 本研究の目的

本研究ではコネクト４で，ディープラーニングが応用できるかを検証する．

1.3 節で述べたとおり，ディープラーニングは囲碁，将棋等のゲーム AI で用いられている．これら

のゲームは探索空間が非常に大きいため，局面の先読み等の従来の手法では最適解は得られない．
　一方，前節で述べた通りコネクト４は，J.D.Allen により先手必勝であることが示されている[3]．「コネクト４」は各手番での選択肢は最大７通りしか無く，従ってｎ手先まで読む場合の局面数は高々 $7^n$ 通りであり，充分な時間があれば先読みにより最適解を得ることができる．そこで本研究では，「コネクト４」のような選択肢の少ないゲームでも先読み型に比べてディープラーニングが有効となるのかを検証する．

## 1.6 本報告書の構成

　本報告の構成は以下の通りである．まず第 2 章で本研究の対象であるコネクト 4 について述べる．第 3 章ではディープラーニングで学習を行うために本研究で作成した対戦相手用プログラムについて述べ，第 4 章ではディープラーニングを用いたコネクト４AI について述べる．最後に第 5 章で結論，今後の課題について述べる．

# 2　コネクト 4

　本章では，コネクト 4 について述べる．
　コネクト４は，垂直に立てられた縦 6 段，横 7 列で構成されたマス（）に，赤色と黄色のコイン型のコマを一手ずつ交互に入れていき，相手より先に縦，横，斜めいずれか 4 マス連続で並べた方が勝ちというゲームである[4]．
　**エラー! 参照元が見つかりません。**にコネクト 4 の盤面を示し，図 2,3,4 に縦，横，斜めの勝利例を示す．各マスには，図 1 に示すように座標が割り当てられている．

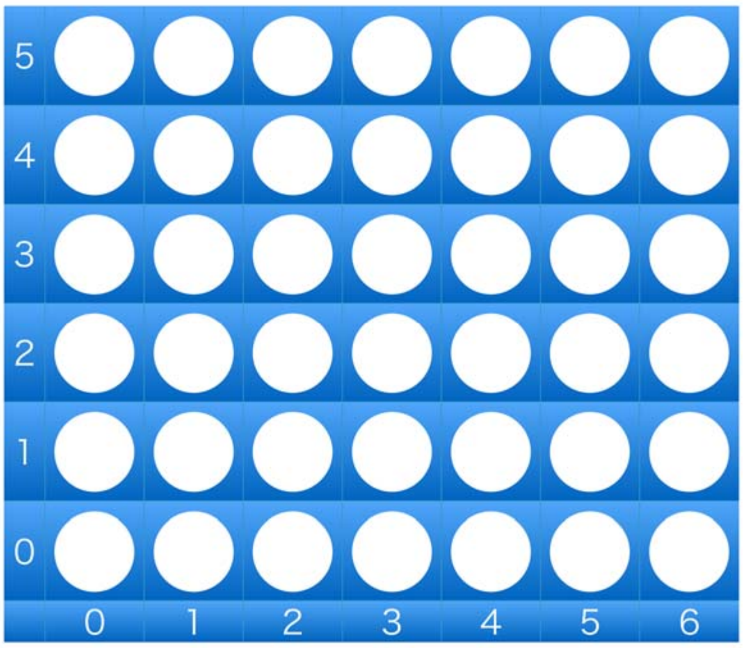

図 1 コネクト 4 の盤面

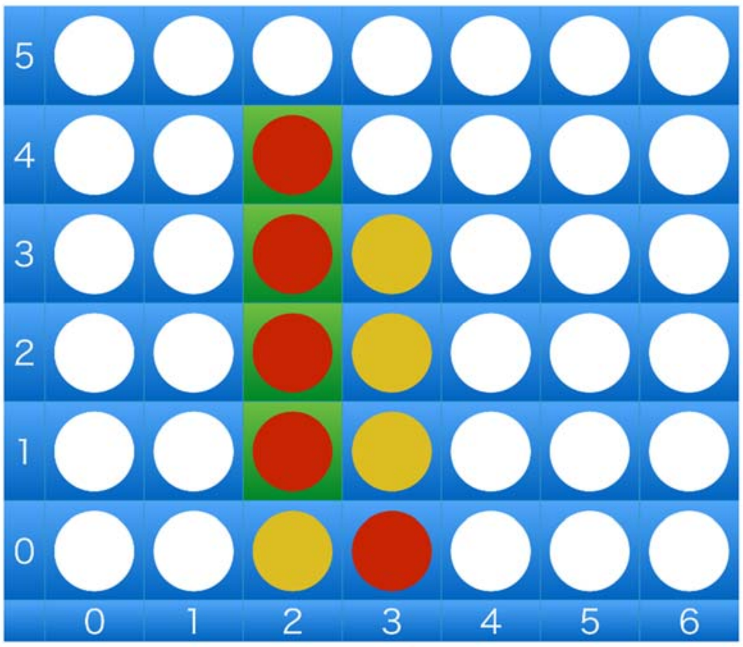

図 2 縦の列での勝利例

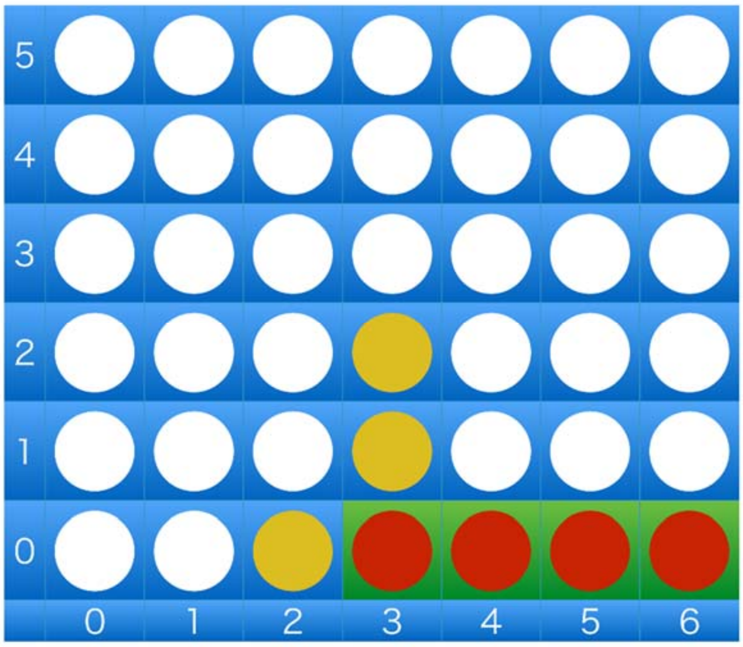

図 3 横の列での勝利例

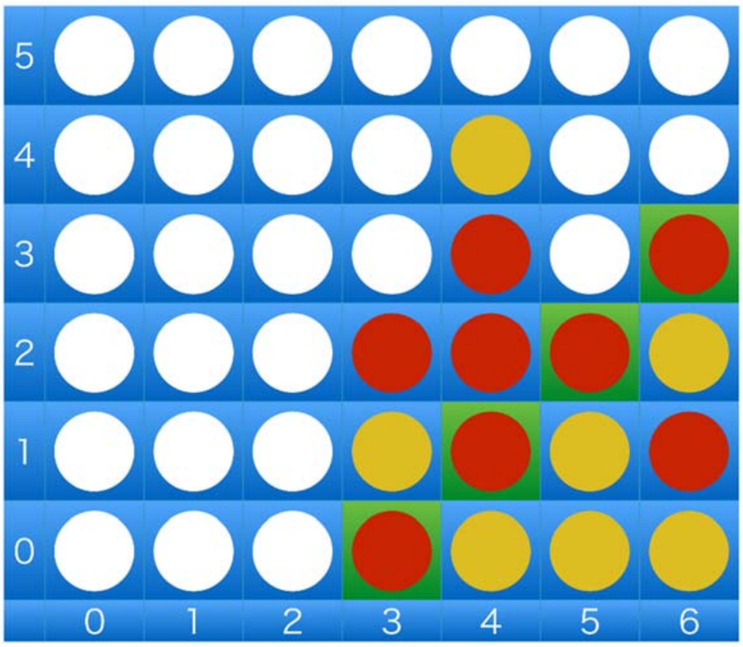

図 4斜めの列での勝利例

コネクト４の特徴として，盤面が垂直に立っている状態で行うため，横方向は自由に選べるが，縦方向は重力の関係で，一番下の段のマスもしくは，既に入っている駒の真上しか置けないことが挙げられる．なお，五目並べでいう「禁じ手」という概念は無いため，長連（縦，横，斜めいずれか５個以上連続して並べた状態）も先手後手関係なく勝ちとなる．

## 3　対戦相手用プログラムの作成

ディープラーニングでコネクト４の AI を作成するためには，学習を行うための対戦相手が必要である．そこで本研究では，対戦相手として用いることができるコネクト４のプログラムを作成する．Keith Galli 氏のコネクト４対人戦用プログラムを参考に作成．

### 3.1 対戦相手用プログラムの戦略

ディープラーニングを用いて学習させる場合，まずは弱い対戦相手で学習を行い，学習が進むにつれて順次対戦相手を強くしていく必要がある．そこで本研究では，以下の 4 つの戦略を用いる対戦相手用プログラムを作成する．

- 戦略１：ランダム
  戦略１のプログラムは，打てる場所であればランダムに打つ仕様である．

- 戦略２：リーチ時に確実に勝つ
  戦略２のプログラムは，自らが３つ並べていて，あと１マスで勝利（リーチ）となる状態で，なおかつ次のターンで勝てるマスに打てる場合，必ずそのマスに打つ仕様である．まだ勝てるマスが存在しない場合は，戦略１と同様にランダムに打つ．

- 戦略３：相手のリーチ時は防ぐ
  戦略３のプログラムは，戦略２と同様に自分にリーチが掛かっておりかつ勝てるマスに置ける場合はそのマスに置く．それ以外の場合，相手にリーチが掛かっており，相手の次の手番で相手が勝てるマスに置ける場合，相手の勝ちを防ぐためにそのマスに先に置く．これらの条件のどちらにも当てはまらない場合は戦術１と同様にランダムに打つ．

- 戦略４：序盤は最善手を打つ
  1.4 節で述べた通り，コネクト４は完全解析されており，各局面の最善手が判明している．
  戦略４のプログラムは，最初の４手目までは相手が打ったマスによって，最善手となるマスに打つという仕様である．５手目以降は戦略３と同様に打つ．
  戦略４で使用する最善手の詳細を表 １に示す．

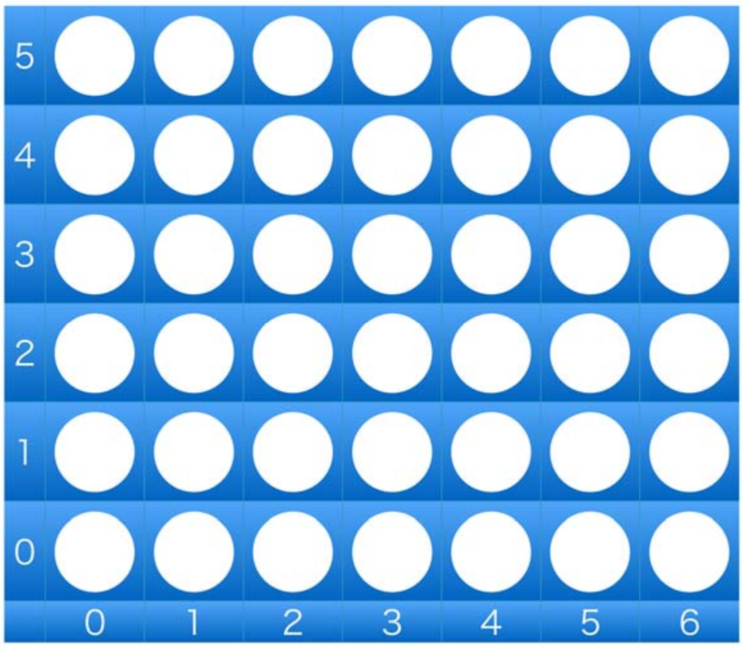

図 5盤面のマス番号

表 1 最初の 4 手の最善手

| ［段，列］マス番号は図 1 参照 | | | |
|---|---|---|---|
| 先手（1 手目） | 後手（2 手目） | 先手（3 手目） | 後手（4 手目） |
| [0,0] | [0,3] | [0,1],[0,2],[0,4],[0,5] | [1,3] |
| | | [1,0] | [0,4] |
| | | [1,3] | [2,3] |
| | | [0,6] | [0,2] |
| [0,1] | [0,2] | [0,0],[0,4],[0,5] | [1,2] |
| | | [1,1] | [2,1] |
| | | [1,2] | [2,2] |
| | | [0,3] | [1,3] |
| | | [0,6] | [0,5] |
| [0,2] | [0,3] | [0,0],[0,1],[0,4],[0,5],[0,6] | [1,3] |
| | | [1,2] | [2,2] |
| | | [1,3] | [2,3] |
| [0,3] | [1,3] | [0,0],[0,1],[0,4] | [0,2] |
| | | [0,2],[0,5],[0,6] | [0,4] |
| | | [2,3] | [3,3] |
| [0,4] | [0,3] | [0,0],[0,1],[0,2],[0,5],[0,6] | [1,3] |
| | | [1,3] | [2,3] |
| | | [1,4] | [2,4] |
| [0,5] | [0,4] | [0,1],[0,2],[0,6] | [1,4] |
| | | [0,0] | [0,1] |
| | | [0,3] | [1,3] |
| | | [1,4] | [2,4] |
| | | [1,5] | [2,5] |
| [0,6] | [0,3] | [0,1],[0,2],[0,4],[0,5] | [1,3] |
| | | [0,0] | [0,4] |
| | | [1,3] | [2,3] |
| | | [1,6] | [0,2] |

## 3.2 対戦相手用プログラム

本研究では，Python を用いて対戦相手用プログラムを作成した．付録に本研究で作成した対戦相手用プログラムを示す.

以下に本研究で作成した対戦相手用プログラムについて説明する.

- connect4aRandomAI.py
  3.1 節で述べた戦略 1 のプログラムである．マスの打ち方は，置けるマスの中からランダムに打つ．表 2 に connext4aRondom.py のメソッドを示す.

表 2 connect4aRondomAI.py のメソッド

| | |
|---|---|
| create_board( ) | ボード作成 |
| drop_piece(board, col, row, piece) | コマを打つ |
| is_varid_location(board, row) | その列が全てコマで埋まっていないか |
| get_next_open_row(board) | その列で打てるマスの中で最下段を返す |

- connect4aReachAI.py
  3.1 節で述べた戦略 2 のプログラムである．マスの打ち方は，リーチの状態で，かつ次のマスで勝利する時はすかさずそのマスに打つ．この条件以外では戦略 1 の通りに打つ．connect4aReachAI は，connect4ARondomAI.py と同様に表 2 に示すメソッドを持つ．
- connect4aReachAI2.py
  3.1 節で述べた戦略 3 のプログラムである．マスの打ち方は，相手があと 1 マスで勝利する時，相手の勝ちを阻止するために，そのマスに先に置く．この条件以外では戦略 2 の通りに打つ．connext4aRondomAI2.py は，表 2 のメソッドに加えて表 3 のメソッドを持つ．

表 3 connect4aReachAI.py のメソッド

| | |
|---|---|
| reach_1(board) | 相手のリーチ確認 |

- connect4aFirstFourAI.py
  3.1 節で述べた戦略 3 のプログラムである．マスの打ち方は，最初の 4 手（2 ターン）までは相手が打ったマスに対して最善手で対応，以降は戦略 3 の通りに打つ．connect4aFirstFourAI.py は，表 2 および表 3 のメソッドを持つ．

# 4　ディープラーニングを用いたコネクト４AI

本研究では，ディープラーニングを用いたコネクト 4 AI を作成し，前章で述べた対戦相手用プログラムを用いて学習させる予定である．

本研究では[1]のディープラーニングを用いた三目並べ AI を参考に，Python を用いてコネクト 4 AI を作成する．

## 4.1 Python のインストールおよび環境変数の設定

本研究では Python のバージョンは 2.7.15 で，その他は[1]を参考に環境変数の設定を行った．以下に環境変数の設定手順を示す．

① Anaconda 環境 main に Chainer（バージョン 1.16.0）をインストールする．
② ①に続いてグラフ描画用に matplotlib（バージョン 1.5.3）をインストールする．
③ https://code.google.com/archive/p/rl-glue-ext/downloads のサイトにある rlglue-3.04.tar.gz をダウンロードする．
④ https://code.google.com/archive/p/rl-glue-ext/downloads?page=2 のサイトにある python-codec-2.02.tar.gz をダウンロードする．

Mac のターミナルで実行を行ったが”RuntimeError:CUDA environment is not correctly set up”が表示され，対処は行ったが，改善には至らなかった．

## 4.2 発生した問題点

本節では，コネクト 4 AI 実行時に発生した問題点について述べる．

4.1 節で述べた通り，実行時に RuntimeError:CUDA environment is not correctly set up と表示されて実行できなかった．

参考文献とは違う環境下での作成もあり，改善策も調べ，対処は行ったが，同じエラーが表示され続ける．原因は不明である．

- Chainer，RL-Glue 等を一度アンインストールしたのち，もう一度インストールする．
- [1]の三目並べのディープラーニングの動作確認

## 5 結論・今後の課題

本研究ではディープラーニングを用いたゲーム AI をコネクト 4 でも有効かを検証する予定であった．しかし，原因不明なエラーの影響で作成するには至らなかった．開発の環境を整えなければならないというのが今後の課題に挙げられる．

## 謝辞

本研究を行うにあたって石水隆講師から開発に関する相談やレジュメや卒業論文の添削など様々なご指導を受けていただきました。ここに感謝の意を表します.

## 参考文献

[1] 藤田一弥，高原歩，実装ディープラーニング，オーム社 (2016).

[2] 斎藤康毅，ゼロから作る Deep Learning - Python で学ぶディープラーニングの理論と実装，オライリージャパン (2016)

[3] James Dow Allen, The Complete Book of Connect 4, Puzzle Wright Press (2010)

[4] Victor Allis, A Knowledge-based Approach of Connect-Four, The Game is Solved: White Wins, Master Thesis, Department of Mathematics and Computer Science Vrije Universiteit (1988) http://www.informatik.uni-trier.de/~fernau/DSL0607/Masterthesis-Viergewinnt.pdf

[5] Yoshiaki Yamaguchi, Kazunori Yamaguchi, Tetsuro Tanaka, and Tomoyuki Kaneko, Infinite Connect-Four Is Solved: Draw, Advances in Computer Games, pp. 208-219 (2011)

[6] Yoshiaki Yamaguchi, Kazunori Yamaguchi, Tetsuro Tanaka, Cylinder-Infinite-Connect-Four except for Widths 2, 6, and 11 is Solved: Drawn, The 8th International Conference on Computers and Games (2013).

[7] Yoshiaki Yamaguchi, Todd W. Neller, First Player's Cannot-Lose Strategies for Cylinder-Infinite-Connect-Four with Widths 2 and 6, Advances in Computer Games, pp.113-121 (2015) http://cs.gettysburg.edu/~tneller/papers/acg2015.pdf

[8] 王銘宛，棋士と AI ： アルファ碁から始まった未来，岩波新書，(2018).

[9] 斉藤康己，アルファ碁はなぜ人間に勝てたのか，ベスト新書，ベストセラーズ，(2016).

[10] 「グーグル囲碁ＡＩ「4 勝 1 敗」は人類の敗北か．プロ棋士から見た勝負の評価」，ニュースイッチ，2016 年 3 月 6 日，日刊工業新聞，(2016) https://newswitch.jp/p/3971

[11] James Dow Allen, Expert Play in Connect-Four, (1990), http://tromp.github.io/c4.html

[12] Pascal Pons, Connect 4 Solver. https://connect4.gamesolver.org/

[13] Pascal Pons. Solving Connect 4 : How To Build A Perfect AI, (2019) http://blog.gamesolver.org/

[14] 「共に電王戦出場、世界最強の“同僚”――コンピュータ将棋ソフト開発者　一丸貴則さん・山本一成さん（前編）」，2014 年 4 月 25 日，ねとらぼ，(2014) http://nlab.itmedia.co.jp/nl/articles/1404/25/news016.html

[15] 「第 2 回将棋電王戦／五番勝負」，日本将棋連盟，(2013)，https://www.shogi.or.jp/match/denou/2/index.html

[16] 「第 3 回将棋電王戦／五番勝負」，日本将棋連盟，(2014)，https://www.shogi.or.jp/match/denou/3/index.html

[17] 「将棋電王戦 FINAL ／五番勝負」，日本将棋連盟，(2015)，https://www.shogi.or.jp/match/denou/4/index.html

[18] 「第 1 期電王戦／二番勝負」，日本将棋連盟，(2016)，https://www.shogi.or.jp/match/denou/01/index.html

[19] 「電王戦　｜棋戦｜日本将棋連盟」，日本将棋連盟，(2017)，https://www.shogi.or.jp/match/denou/

[20] 山本一成，人工知能はどのようにして「名人」を超えたのか？：最強将棋 AI ポナンザの開発者が教える機械学習・深層学習・強化学習の本質，ダイヤモンド社，(2017)

[21] Connect4-Python/connect4.py at master ・KeithGalli/Connect4-Python・GitHub, GitHub, (2017), https://github.com/KeithGalli/Connect4-Python/blob/master/connect4.py

ソースプログラム

本研究で作成したプログラムのソースファイルの一部を以下に示す.

- connect4aRandomAI.py

```
import numpy as np
import random

COLUMN_COUNT = 6
ROW_COUNT = 7

#ボード作成
def create_board():
        board = np.zeros((6,7))
        return board

#コマを打つ
def drop_piece(board, col, row, piece):
        board[col][row] = piece

#その列は，コマが一番上の段まで埋まっていないか
def is_valid_location(board, row):
        return board[5][row] == 0

#その列で打てるマスのうち，最も下の段のマスを返す
def get_next_open_row(board, row):
        for c in range(COLUMN_COUNT):
         if board[c][row] == 0:
                return c

def print_board(board):
        print(np.flip(board, 0))

#決着の判定
def winning_move(board, piece):
        #ヨコで決着したか
        for r in range(ROW_COUNT-3):
                for c in range(COLUMN_COUNT):
                        if board[c][r] == piece and board[c][r+1] == piece and board[c][r+2]
== piece and board[c][r+3] == piece:
                                return True

        #タテで決着したか
        for r in range(ROW_COUNT):
                for c in range(COLUMN_COUNT-3):
                        if board[c][r] == piece and board[c+1][r] == piece and board[c+2][r]
== piece and board[c+3][r] == piece:
                                return True
        #右肩上がりナナメで決着したか
        for r in range(ROW_COUNT-3):
                for c in range(COLUMN_COUNT-3):
                        if  board[c][r]  ==  piece  and  board[c+1][r+1]  ==  piece  and
board[c+2][r+2] == piece and board[c+3][r+3] == piece:
                                return True

        #右肩下がりナナメで決着したか
        for r in range(ROW_COUNT-3):
                for c in range(3, COLUMN_COUNT):
                        if board[c][r] == piece and board[c-1][r+1] == piece and board[c-
2][r+2] == piece and board[c-3][r+3] == piece:
                                return True



  board = create_board()
  print_board(board)
  game_over = False
  turn = 0

  while not game_over:
        #Player 1の番
        if turn == 0:
                #row = int(input("Player1 Make Your Selection(0-6)"))
                while True:
                        row = int(input("Player1 Make Your Selection(0-6)"))
                        #if board[5][row] == 0 and -1 < row and row < 7: 範囲外はエラーに
なる
                        if board[5][row] == 0:
                                break

                if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 1)
                        if winning_move(board, 1):
                                print("PLAYER1 Wins!")
                                game_over = True

        #Player 2の番
        else:
                while True:
                        row = random.randrange(7)
                        print("Player2 Selected:", row)
                        if board[5][row] == 0:
                                break
                if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)

                        if winning_move(board, 2):
                                print("PLAYER2 Wins!")
                                game_over = True

        print_board(board)

        turn += 1
        turn = turn % 2
```

- connect4aReachAI.py

```
import numpy as np
import random

COLUMN_COUNT = 6
ROW_COUNT = 7

#ボード作成
def create_board():
        board = np.zeros((6,7))
        return board

#コマを打つ
def drop_piece(board, col, row, piece):
        board[col][row] = piece

#その列は，コマが一番上の段まで埋まっていないか
def is_valid_location(board, row):
        return board[5][row] == 0

#その列で打てるマスのうち，最も下の段のマスを返す
def get_next_open_row(board, row):
        for c in range(COLUMN_COUNT):
         if board[c][row] == 0:
                 return c

def print_board(board):
        print(np.flip(board, 0))

#決着の判定
def winning_move(board, piece):
        #ヨコで決着したか
        for r in range(ROW_COUNT-3):
                for c in range(COLUMN_COUNT):
                        if board[c][r] == piece and board[c][r+1] == piece and board[c][r+2]
== piece and board[c][r+3] == piece:
                                return True

        #タテで決着したか
        for r in range(ROW_COUNT):
                for c in range(COLUMN_COUNT-3):
                        if board[c][r] == piece and board[c+1][r] == piece and board[c+2][r]
== piece and board[c+3][r] == piece:
                                return True

        #右肩上がりナナメで決着したか
        for r in range(ROW_COUNT-3):
                for c in range(COLUMN_COUNT-3):
                        if  board[c][r]  ==  piece  and  board[c+1][r+1]  ==  piece  and
board[c+2][r+2] == piece and board[c+3][r+3] == piece:
                                return True

        #右肩下がりナナメで決着したか
        for r in range(ROW_COUNT-3):
                for c in range(3, COLUMN_COUNT):
                        if board[c][r] == piece and board[c-1][r+1] == piece and board[c-
2][r+2] == piece and board[c-3][r+3] == piece:
                                return True

board = create_board()
print_board(board)
game_over = False
turn = 0

while not game_over:
    #Player 1 の番
    if turn == 0:
        while True:
            row = int(input("Player1 Make Your Selection(0-6)"))
            if board[5][row] == 0:
                break

        if is_valid_location(board, row):
            col = get_next_open_row(board, row)
            drop_piece(board, col, row, 1)
            if winning_move(board, 1):
                print("Player1 Wins!")
                game_over = True

    #Player 2 の番
    else:
        #ヨコ(1 段目マス絡みの決着のみ)
        for r in range(ROW_COUNT-3):
            if board[0][r] == 0 and board[0][r+1] == 2 and board[0][r+2] == 2 and
board[0][r+3] == 2:
                row = r
                col = 0
                drop_piece(board, col, row, 2)
                winning_move(board, 2)
                print("Player2 Wins!")
                game_over = True
                break
            elif board[0][r] == 2 and board[0][r+1] == 0 and board[0][r+2] == 2 and
board[0][r+3] == 2:
                row = r+1
                col = 0
                drop_piece(board, col, row, 2)
                winning_move(board, 2)
                print("Player2 Wins!")
                game_over = True
                break
            elif board[0][r] == 2 and board[0][r+1] == 2 and board[0][r+2] == 0 and
board[0][r+3] == 2:
                row = r+2
                col = 0
                drop_piece(board, col, row, 2)
                winning_move(board, 2)
                print("Player2 Wins!")
                game_over = True
                break
            elif board[0][r] == 2 and board[0][r+1] == 2 and board[0][r+2] == 2 and
board[0][r+3] == 0:
                row = r+3
                col = 0
                drop_piece(board, col, row, 2)
                winning_move(board, 2)
                print("Player2 Wins!")
                game_over = True
                break

        #ヨコ(2 段目以降)
        #4 つ目のマスの真下にコマがあるかを判断(以降も「1 段目マス絡みの決着のみ」を除いて同
様に)
        for r in range(ROW_COUNT-3):
            for c in range(1, COLUMN_COUNT):
                if board[c][r] == 0 and board[c][r+1] == 2 and board[c][r+2] == 2 and
board[c][r+3] == 2 and board[c-1][r] != 0:
                    row = r
                    col = c
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[c][r] == 2 and board[c][r+1] == 0 and board[c][r+2] == 2 and
board[c][r+3] == 2 and board[c-1][r+1] != 0:
                    row = r+1
                    col = c
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[c][r] == 2 and board[c][r+1] == 2 and board[c][r+2] == 0 and
board[c][r+3] == 2 and board[c-1][r+2] != 0:
                    row = r+2
                    col = c
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[c][r] == 2 and board[c][r+1] == 2 and board[c][r+2] == 2 and
board[c][r+3] == 0 and board[c-1][r+3] != 0:
                    row = r+3
                    col = c
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break

        #タテ
        for r in range(ROW_COUNT):
            for c in range(COLUMN_COUNT-3):
                if board[c][r] == 2 and board[c+1][r] == 2 and board[c+2][r] == 2 and
board[c+3][r] == 0:
                    row = r
                    col = c+3
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break

        #右肩上がりナナメ(1段目マス絡みの決着のみ)
        for r in range(ROW_COUNT-3):
            if board[0][r] == 0 and board[1][r+1] == 2 and board[2][r+2] == 2 and
```

```
board[3][r+3] == 2:
                row = r
                col = 0
                drop_piece(board, col, row, 2)
                winning_move(board, 2)
                print("Player2 Wins!")
                game_over = True
                break
            elif  board[0][r]  ==  2  and  board[1][r+1]  ==  0  and  board[2][r+2]  ==  2  and
board[3][r+3] == 2 and board[0][r+1] != 0:
                row = r+1
                col = 1
                drop_piece(board, col, row, 2)
                winning_move(board, 2)
                print("Player2 Wins!")
                game_over = True
                break
            elif  board[0][r]  ==  2  and  board[1][r+1]  ==  2  and  board[2][r+2]  ==  0  and
board[3][r+3] == 2 and board[1][r+2] != 0:
                row = r+2
                col = 2
                drop_piece(board, col, row, 2)
                winning_move(board, 2)
                print("Player2 Wins!")
                game_over = True
                break
            elif  board[0][r]  ==  2  and  board[1][r+1]  ==  2  and  board[2][r+2]  ==  2  and
board[3][r+3] == 0 and board[2][r+3] != 0:
                row = r+3
                col = 3
                drop_piece(board, col, row, 2)
                winning_move(board, 2)
                print("Player2 Wins!")
                game_over = True
                break

        #右肩上がりナナメ(1段目マス絡み以外)
        for r in range(ROW_COUNT-3):
            for c in range(1,COLUMN_COUNT-3):
                if board[c][r] == 0 and board[c+1][r+1] == 2 and board[c+2][r+2] == 2 and
board[c+3][r+3] == 2 and board[c-1][r] != 0:
                    row = r
                    col = c
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[c][r] == 2 and board[c+1][r+1] == 0 and board[c+2][r+2] == 2 and board[c+3][r+3] == 2 and board[c][r+1] != 0:
                    row = r+1
                    col = c+1
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[c][r] == 2 and board[c+1][r+1] == 2 and board[c+2][r+2] == 0 and board[c+3][r+3] == 2 and board[c+1][r+2] != 0:
                    row = r+2
                    col = c+2
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[c][r] == 2 and board[c+1][r+1] == 2 and board[c+2][r+2] == 2 and board[c+3][r+3] == 0 and board[c+2][r+3] != 0:
                    row = r+3
                    col = c+3
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break

        #右肩下がりナナメ(1段目マス絡みの決着のみ)
        for r in range(ROW_COUNT-3):
            if board[3][r] == 0 and board[2][r+1] == 2 and board[1][r+2] == 2 and board[0][r+3] == 2 and board[2][r] != 0:
                row = r
                col = 3
                drop_piece(board, col, row, 2)
                winning_move(board, 2)
                print("Player2 Wins!")
                game_over = True
                break
            elif board[3][r] == 2 and board[2][r+1] == 0 and board[1][r+2] == 2 and board[0][r+3] == 2 and board[1][r+1] != 0:
                row = r+1
                col = 2
                drop_piece(board, col, row, 2)
                winning_move(board, 2)
                print("Player2 Wins!")
                game_over = True
                break
            elif  board[3][r]  ==  2  and  board[2][r+1]  ==  2  and  board[1][r+2]  ==  0  and
board[0][r+3] == 2 and board[0][r+2] != 0:
                row = r+2
                col = 1
                drop_piece(board, col, row, 2)
                winning_move(board, 2)
                print("Player2 Wins!")
                game_over = True
                break
            elif  board[3][r]  ==  2  and  board[2][r+1]  ==  2  and  board[1][r+2]  ==  2  and
board[0][r+3] == 0:
                row = r+3
                col = 0
                drop_piece(board, col, row, 2)
                winning_move(board, 2)
                print("Player2 Wins!")
                game_over = True
                break

        #右肩下がりナナメ(1段目マス絡み以外)
        for r in range(ROW_COUNT-3):
            for c in range(4, COLUMN_COUNT):
                if board[c][r] == 0 and board[c-1][r+1] == 2 and board[c-2][r+2] == 2 and
board[c-3][r+3] == 2 and board[c-1][r] != 0:
                    row = r
                    col = c
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[c][r] == 2 and board[c-1][r+1] == 0 and board[c-2][r+2] == 2 and
board[c-3][r+3] == 2 and board[c-2][r+1] != 0:
                    row = r+1
                    col = c-1
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[c][r] == 2 and board[c-1][r+1] == 2 and board[c-2][r+2] == 0 and
board[c-3][r+3] == 2 and board[c-3][r+2] != 0:
                    row = r+2
                    col = c-2
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[c][r] == 2 and board[c-1][r+1] == 2 and board[c-2][r+2] == 2 and
board[c-3][r+3] == 0 and board[c-4][r+3] != 0:
                    row = r+3
                    col = c-3
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break


                 if not winning_move(board, 2):
                        #次で勝てる時以外はランダム
            while True:
                row = random.randrange(7)
                print("Player2 Selected:", row)
                if board[5][row] == 0:
                    break
            if is_valid_location(board, row):
                col = get_next_open_row(board, row)
                drop_piece(board, col, row, 2)

                if reach_move(board, 2) and not winning_move(board, 2):
                    print("PLAYER2 Reach")

    print_board(board)

    turn += 1
    turn = turn % 2
```

- connect4aReachAI2.py

```
import numpy as np
import random

COLUMN_COUNT = 6
ROW_COUNT = 7

#ボード作成
def create_board():
        board = np.zeros((6,7))
        return board

#コマを打つ
def drop_piece(board, col, row, piece):
        board[col][row] = piece
```

```
#その列は，コマが一番上の段まで埋まっていないか
def is_valid_location(board, row):
        return board[5][row] == 0

#その列で打てるマスのうち，最も下の段のマスを返す
def get_next_open_row(board, row):
        for c in range(COLUMN_COUNT):
         if board[c][row] == 0:
                 return c

def print_board(board):
        print(np.flip(board, 0))

#決着の判定
def winning_move(board, piece):
        #ヨコで決着したか
        for r in range(ROW_COUNT-3):
                for c in range(COLUMN_COUNT):
                        if board[c][r] == piece and board[c][r+1] == piece and board[c][r+2]
== piece and board[c][r+3] == piece:
                                return True

        #タテで決着したか
        for r in range(ROW_COUNT):
                for c in range(COLUMN_COUNT-3):
                        if board[c][r] == piece and board[c+1][r] == piece and board[c+2][r]
== piece and board[c+3][r] == piece:
                                return True

        #右肩上がりナナメで決着したか
        for r in range(ROW_COUNT-3):
                for c in range(COLUMN_COUNT-3):
                        if   board[c][r]   ==   piece   and   board[c+1][r+1]   ==   piece   and
board[c+2][r+2] == piece and board[c+3][r+3] == piece:
                                return True

        #右肩下がりナナメで決着したか
        for r in range(ROW_COUNT-3):
                for c in range(3, COLUMN_COUNT):
                        if board[c][r] == piece and board[c-1][r+1] == piece and board[c-
2][r+2] == piece and board[c-3][r+3] == piece:
                                return True


def reach_1(board):
    #相手のリーチ確認
    #ヨコ１段目のみ
    for r in range(ROW_COUNT-3):
        if board[0][r] == 0 and board[0][r+1] == 1 and board[0][r+2] == 1 and board[0][r+3]
== 1:
            return True
        elif board[0][r] == 1 and board[0][r+1] == 0 and board[0][r+2] == 1 and board[0][r+3]
== 1:
            return True
        elif board[0][r] == 1 and board[0][r+1] == 1 and board[0][r+2] == 0 and board[0][r+3]
== 1:
            return True
        elif board[0][r] == 1 and board[0][r+1] == 1 and board[0][r+2] == 1 and board[0][r+3]
== 0:
            return True

    #ヨコ２段目以降
    for r in range(ROW_COUNT-3):
        for c in range(1, COLUMN_COUNT):
            if board[c][r] == 0 and board[c][r+1] == 1 and board[c][r+2] == 1 and
board[c][r+3] == 1 and board[c-1][r] != 0:
                return True
            elif board[c][r] == 1 and board[c][r+1] == 0 and board[c][r+2] == 1 and
board[c][r+3] == 1 and board[c-1][r+1] != 0:
                return True
            elif board[c][r] == 1 and board[c][r+1] == 1 and board[c][r+2] == 0 and
board[c][r+3] == 1 and board[c-1][r+2] != 0:
                return True
            elif board[c][r] == 1 and board[c][r+1] == 1 and board[c][r+2] == 1 and
board[c][r+3] == 0 and board[c-1][r+3] != 0:
                return True

    #タテ
    for r in range(ROW_COUNT):
        for c in range(COLUMN_COUNT-3):
            if board[c][r] == 1 and board[c+1][r] == 1 and board[c+2][r] == 1 and
board[c+3][r] == 0:
                return True

    #右肩上がりナナメ(1段目マス絡みのみ)
    for r in range(ROW_COUNT-3):
        if board[0][r] == 0 and board[1][r+1] == 1 and board[2][r+2] == 1 and board[3][r+3]
== 1:
            return True
        elif board[0][r] == 1 and board[1][r+1] == 0 and board[2][r+2] == 1 and board[3][r+3]
== 1 and board[0][r+1] != 0:
            return True
        elif board[0][r] == 1 and board[1][r+1] == 1 and board[2][r+2] == 0 and board[3][r+3]
== 1 and board[1][r+2] != 0:
            return True
        elif board[0][r] == 1 and board[1][r+1] == 1 and board[2][r+2] == 1 and board[3][r+3]
```

```
== 0 and board[2][r+3] != 0:
            return True

    #右肩上がりナナメ(1段目マス絡み以外)
    for r in range(ROW_COUNT-3):
        for c in range(1,COLUMN_COUNT-3):
            if board[c][r] == 0 and board[c+1][r+1] == 1 and board[c+2][r+2] == 1 and
board[c+3][r+3] == 1 and board[c-1][r] != 0:
                return True
            elif board[c][r] == 1 and board[c+1][r+1] == 0 and board[c+2][r+2] == 1 and
board[c+3][r+3] == 1 and board[c][r+1] != 0:
                return True
            elif board[c][r] == 1 and board[c+1][r+1] == 1 and board[c+2][r+2] == 0 and
board[c+3][r+3] == 1 and board[c+1][r+2] != 0:
                return True
            elif board[c][r] == 1 and board[c+1][r+1] == 1 and board[c+2][r+2] == 1 and
board[c+3][r+3] == 0 and board[c+2][r+3] != 0:
                return True

    #右肩下がりナナメ(1段目マス絡みのみ)
    for r in range(ROW_COUNT-3):
        if board[3][r] == 0 and board[2][r+1] == 1 and board[1][r+2] == 1 and board[0][r+3]
== 1 and board[2][r] != 0:
            return True
        elif board[3][r] == 1 and board[2][r+1] == 0 and board[1][r+2] == 1 and board[0][r+3]
== 1 and board[1][r+1] != 0:
            return True
        elif board[3][r] == 1 and board[2][r+1] == 1 and board[1][r+2] == 0 and board[0][r+3]
== 1 and board[0][r+2] != 0:
            return True
        elif board[3][r] == 2 and board[2][r+1] == 2 and board[1][r+2] == 2 and board[0][r+3]
== 0:
            return True

    #右肩下がりナナメ(1段目マス絡み以外)
    for r in range(ROW_COUNT-3):
        for c in range(4, COLUMN_COUNT):
            if board[c][r] == 0 and board[c-1][r+1] == 1 and board[c-2][r+2] == 1 and board[c-
3][r+3] == 1 and board[c-1][r] != 0:
                return True
            elif board[c][r] == 1 and board[c-1][r+1] == 0 and board[c-2][r+2] == 1 and
board[c-3][r+3] == 1 and board[c-2][r+1] != 0:
                return True
            elif board[c][r] == 1 and board[c-1][r+1] == 1 and board[c-2][r+2] == 0 and
board[c-3][r+3] == 1 and board[c-3][r+2] != 0:
                return True
            elif board[c][r] == 1 and board[c-1][r+1] == 1 and board[c-2][r+2] == 1 and
board[c-3][r+3] == 0 and board[c-4][r+3] != 0:
```

```
                return True


board = create_board()
print_board(board)
game_over = False
turn = 0

while not game_over:
    #Player 1の番
    if turn == 0:
        while True:
            row = int(input("Player1 Make Your Selection(0-6)"))
            if board[5][row] == 0:
                break

        if is_valid_location(board, row):
            col = get_next_open_row(board, row)
            drop_piece(board, col, row, 1)
            if winning_move(board, 1):
                print("Player1 Wins!")
                game_over = True
                break

    #Player 2の番
    else:
        #ヨコ(1段目マス絡みの決着のみ)
        for r in range(ROW_COUNT-3):
            if board[0][r] == 0 and board[0][r+1] == 2 and board[0][r+2] == 2 and
board[0][r+3] == 2:
                row = r
                col = 0
                drop_piece(board, col, row, 2)
                winning_move(board, 2)
                print("Player2 Wins!")
                game_over = True
                break
            elif board[0][r] == 2 and board[0][r+1] == 0 and board[0][r+2] == 2 and
board[0][r+3] == 2:
                row = r+1
                col = 0
                drop_piece(board, col, row, 2)
                winning_move(board, 2)
                print("Player2 Wins!")
                game_over = True
                break
            elif board[0][r] == 2 and board[0][r+1] == 2 and board[0][r+2] == 0 and
board[0][r+3] == 2:
```

```
                row = r+2
                col = 0
                drop_piece(board, col, row, 2)
                winning_move(board, 2)
                print("Player2 Wins!")
                game_over = True
                break
            elif board[0][r] == 2 and board[0][r+1] == 2 and board[0][r+2] == 2 and
board[0][r+3] == 0:
                row = r+3
                col = 0
                drop_piece(board, col, row, 2)
                winning_move(board, 2)
                print("Player2 Wins!")
                game_over = True
                break

        #ヨコ(2 段目以降)
        #4 つ目のマスの真下にコマがあるかを判断(以降も「1 段目マス絡みの決着のみ」を除いて同
様に)
        for r in range(ROW_COUNT-3):
            for c in range(1, COLUMN_COUNT):
                if board[c][r] == 0 and board[c][r+1] == 2 and board[c][r+2] == 2 and
board[c][r+3] == 2 and board[c-1][r] != 0:
                    row = r
                    col = c
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[c][r] == 2 and board[c][r+1] == 0 and board[c][r+2] == 2 and
board[c][r+3] == 2 and board[c-1][r+1] != 0:
                    row = r+1
                    col = c
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[c][r] == 2 and board[c][r+1] == 2 and board[c][r+2] == 0 and
board[c][r+3] == 2 and board[c-1][r+2] != 0:
                    row = r+2
                    col = c
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[c][r] == 2 and board[c][r+1] == 2 and board[c][r+2] == 2 and
board[c][r+3] == 0 and board[c-1][r+3] != 0:
                    row = r+3
                    col = c
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break

        #vertical タテ
        for r in range(ROW_COUNT):
            for c in range(COLUMN_COUNT-3):
                if board[c][r] == 2 and board[c+1][r] == 2 and board[c+2][r] == 2 and
board[c+3][r] == 0:
                    row = r
                    col = c+3
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break

        #右肩上がりナナメ(1段目マス絡みの決着のみ)
        for r in range(ROW_COUNT-3):
            if board[0][r] == 0 and board[1][r+1] == 2 and board[2][r+2] == 2 and
board[3][r+3] == 2:
                row = r
                col = 0
                drop_piece(board, col, row, 2)
                winning_move(board, 2)
                print("Player2 Wins!")
                game_over = True
                break
            elif board[0][r] == 2 and board[1][r+1] == 0 and board[2][r+2] == 2 and
board[3][r+3] == 2 and board[0][r+1] != 0:
                row = r+1
                col = 1
                drop_piece(board, col, row, 2)
                winning_move(board, 2)
                print("Player2 Wins!")
                game_over = True
                break
            elif board[0][r] == 2 and board[1][r+1] == 2 and board[2][r+2] == 0 and
board[3][r+3] == 2 and board[1][r+2] != 0:
                row = r+2
                col = 2
                drop_piece(board, col, row, 2)
                winning_move(board, 2)
                print("Player2 Wins!")
                game_over = True
                break
            elif  board[0][r]  ==  2  and  board[1][r+1]  ==  2  and  board[2][r+2]  ==  2  and
board[3][r+3] == 0 and board[2][r+3] != 0:
                row = r+3
                col = 3
                drop_piece(board, col, row, 2)
                winning_move(board, 2)
                print("Player2 Wins!")
                game_over = True
                break

        #右肩上がりナナメ(1段目マス絡み以外)
        for r in range(ROW_COUNT-3):
            for c in range(1,COLUMN_COUNT-3):
                if board[c][r] == 0 and board[c+1][r+1] == 2 and board[c+2][r+2] == 2 and
board[c+3][r+3] == 2 and board[c-1][r] != 0:
                    row = r
                    col = c
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[c][r] == 2 and board[c+1][r+1] == 0 and board[c+2][r+2] == 2 and
board[c+3][r+3] == 2 and board[c][r+1] != 0:
                    row = r+1
                    col = c+1
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[c][r] == 2 and board[c+1][r+1] == 2 and board[c+2][r+2] == 0 and
board[c+3][r+3] == 2 and board[c+1][r+2] != 0:
                    row = r+2
                    col = c+2
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[c][r] == 2 and board[c+1][r+1] == 2 and board[c+2][r+2] == 2 and
board[c+3][r+3] == 0 and board[c+2][r+3] != 0:
                    row = r+3
                    col = c+3
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break

        #右肩下がりナナメ(1段目マス絡みの決着のみ)
        for r in range(ROW_COUNT-3):
            if board[3][r] == 0 and board[2][r+1] == 2 and board[1][r+2] == 2 and board[0][r+3] == 2 and board[2][r] != 0:
                row = r
                col = 3
                drop_piece(board, col, row, 2)
                winning_move(board, 2)
                print("Player2 Wins!")
                game_over = True
                break
            elif board[3][r] == 2 and board[2][r+1] == 0 and board[1][r+2] == 2 and board[0][r+3] == 2 and board[1][r+1] != 0:
                row = r+1
                col = 2
                drop_piece(board, col, row, 2)
                winning_move(board, 2)
                print("Player2 Wins!")
                game_over = True
                break
            elif board[3][r] == 2 and board[2][r+1] == 2 and board[1][r+2] == 0 and board[0][r+3] == 2 and board[0][r+2] != 0:
                row = r+2
                col = 1
                drop_piece(board, col, row, 2)
                winning_move(board, 2)
                print("Player2 Wins!")
                game_over = True
                break
            elif board[3][r] == 2 and board[2][r+1] == 2 and board[1][r+2] == 2 and board[0][r+3] == 0:
                row = r+3
                col = 0
                drop_piece(board, col, row, 2)
                winning_move(board, 2)
                print("Player2 Wins!")
                game_over = True
                break

        #右肩下がりナナメ(1段目マス絡み以外)
        for r in range(ROW_COUNT-3):
            for c in range(4, COLUMN_COUNT):
                if board[c][r] == 0 and board[c-1][r+1] == 2 and board[c-2][r+2] == 2 and
board[c-3][r+3] == 2 and board[c-1][r] != 0:
                    row = r
                    col = c
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[c][r] == 2 and board[c-1][r+1] == 0 and board[c-2][r+2] == 2 and
board[c-3][r+3] == 2 and board[c-2][r+1] != 0:
                    row = r+1
                    col = c-1
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[c][r] == 2 and board[c-1][r+1] == 2 and board[c-2][r+2] == 0 and
board[c-3][r+3] == 2 and board[c-3][r+2] != 0:
                    row = r+2
                    col = c-2
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[c][r] == 2 and board[c-1][r+1] == 2 and board[c-2][r+2] == 2 and
board[c-3][r+3] == 0 and board[c-4][r+3] != 0:
                    row = r+3
                    col = c-3
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break

        if not winning_move(board, 2) and reach_1(board): #相手がリーチかけられた時, 先に置
いて防ぐ
            #ヨコ１段目のみ
            for r in range(ROW_COUNT-3):
                if board[0][r] == 0 and board[0][r+1] == 1 and board[0][r+2] == 1 and
board[0][r+3] == 1:
                    row = r
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                        break
                elif board[0][r] == 1 and board[0][r+1] == 0 and board[0][r+2] == 1 and
board[0][r+3] == 1:
                    row = r+1
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                        break
                elif board[0][r] == 1 and board[0][r+1] == 1 and board[0][r+2] == 0 and
board[0][r+3] == 1:
                    row = r+2
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                        break
                elif board[0][r] == 1 and board[0][r+1] == 1 and board[0][r+2] == 1 and
board[0][r+3] == 0:
                    row = r+3
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                        break

            #ヨコ２段目以降
            for r in range(ROW_COUNT-3):
                for c in range(1, COLUMN_COUNT):
                    if board[c][r] == 0 and board[c][r+1] == 1 and board[c][r+2] == 1 and
board[c][r+3] == 1 and board[c-1][r] != 0:
                        row = r
                        if is_valid_location(board, row):
                            col = get_next_open_row(board, row)
                            drop_piece(board, col, row, 2)
                            break
                    elif board[c][r] == 1 and board[c][r+1] == 0 and board[c][r+2] == 1 and
board[c][r+3] == 1 and board[c-1][r+1] != 0:
                        row = r+1
                        if is_valid_location(board, row):
                            col = get_next_open_row(board, row)
                            drop_piece(board, col, row, 2)
                            break
                    elif board[c][r] == 1 and board[c][r+1] == 1 and board[c][r+2] == 0 and
board[c][r+3] == 1 and board[c-1][r+2] != 0:
                        row = r+2
                        if is_valid_location(board, row):
                            col = get_next_open_row(board, row)
                            drop_piece(board, col, row, 2)
                            break
                    elif board[c][r] == 1 and board[c][r+1] == 1 and board[c][r+2] == 1 and
```

```
board[c][r+3] == 0 and board[c-1][r+3] != 0:
                        row = r+3
                        if is_valid_location(board, row):
                            col = get_next_open_row(board, row)
                            drop_piece(board, col, row, 2)
                            break

            #タテ
            for r in range(ROW_COUNT):
                for c in range(COLUMN_COUNT-3):
                    if board[c][r] == 1 and board[c+1][r] == 1 and board[c+2][r] == 1 and
board[c+3][r] == 0:
                        row = r
                        if is_valid_location(board, row):
                            col = get_next_open_row(board, row)
                            drop_piece(board, col, row, 2)
                            break

            #右肩上がりナナメ(1段目マス絡みのみ)
            for r in range(ROW_COUNT-3):
                if board[0][r] == 0 and board[1][r+1] == 1 and board[2][r+2] == 1 and
board[3][r+3] == 1:
                    row = r
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                        break
                elif board[0][r] == 1 and board[1][r+1] == 0 and board[2][r+2] == 1 and
board[3][r+3] == 1 and board[0][r+1] != 0:
                    row = r+1
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                        break
                elif board[0][r] == 1 and board[1][r+1] == 1 and board[2][r+2] == 0 and
board[3][r+3] == 1 and board[1][r+2] != 0:
                    row = r+2
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                        break
                elif board[0][r] == 1 and board[1][r+1] == 1 and board[2][r+2] == 1 and
board[3][r+3] == 0 and board[2][r+3] != 0:
                    row = r+3
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                        break
            #右肩上がりナナメ(1段目マス絡み以外)
            for r in range(ROW_COUNT-3):
                for c in range(1,COLUMN_COUNT-3):
                    if board[c][r] == 0 and board[c+1][r+1] == 1 and board[c+2][r+2] == 1
and board[c+3][r+3] == 1 and board[c-1][r] != 0:
                        row = r
                        if is_valid_location(board, row):
                            col = get_next_open_row(board, row)
                            drop_piece(board, col, row, 2)
                            break
                    elif board[c][r] == 1 and board[c+1][r+1] == 0 and board[c+2][r+2] == 1
and board[c+3][r+3] == 1 and board[c][r+1] != 0:
                        row = r+1
                        if is_valid_location(board, row):
                            col = get_next_open_row(board, row)
                            drop_piece(board, col, row, 2)
                            break
                    elif board[c][r] == 1 and board[c+1][r+1] == 1 and board[c+2][r+2] == 0
and board[c+3][r+3] == 1 and board[c+1][r+2] != 0:
                        row = r+2
                        if is_valid_location(board, row):
                            col = get_next_open_row(board, row)
                            drop_piece(board, col, row, 2)
                            break
                    elif board[c][r] == 1 and board[c+1][r+1] == 1 and board[c+2][r+2] == 1
and board[c+3][r+3] == 0 and board[c+2][r+3] != 0:
                        row = r+3
                        if is_valid_location(board, row):
                            col = get_next_open_row(board, row)
                            drop_piece(board, col, row, 2)
                            break

            #右肩下がりナナメ(1段目マス絡みのみ)
            for r in range(ROW_COUNT-3):
                if board[3][r] == 0 and board[2][r+1] == 1 and board[1][r+2] == 1 and
board[0][r+3] == 1 and board[2][r] != 0:
                    row = r
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                        break
                elif board[3][r] == 1 and board[2][r+1] == 0 and board[1][r+2] == 1 and
board[0][r+3] == 1 and board[1][r] != 0:
                    row = r+1
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                        break
                elif board[3][r] == 1 and board[2][r+1] == 1 and board[1][r+2] == 0 and
board[0][r+3] == 1 and board[0][r+2] != 0:
                    row = r+2
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                        break
                elif board[3][r] == 2 and board[2][r+1] == 2 and board[1][r+2] == 2 and
board[0][r+3] == 0:
                    row = r+3
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                        break

            #右肩下がりナナメ(1段目マス絡み以外)
            for r in range(ROW_COUNT-3):
                for c in range(4, COLUMN_COUNT):
                    if board[c][r] == 0 and board[c-1][r+1] == 1 and board[c-2][r+2] == 1
and board[c-3][r+3] == 1 and board[c-1][r] != 0:
                        row = r
                        if is_valid_location(board, row):
                            col = get_next_open_row(board, row)
                            drop_piece(board, col, row, 2)
                            break
                    elif board[c][r] == 1 and board[c-1][r+1] == 0 and board[c-2][r+2] == 1
and board[c-3][r+3] == 1 and board[c-2][r+1] != 0:
                        row = r+1
                        if is_valid_location(board, row):
                            col = get_next_open_row(board, row)
                            drop_piece(board, col, row, 2)
                            break
                    elif board[c][r] == 1 and board[c-1][r+1] == 1 and board[c-2][r+2] == 0
and board[c-3][r+3] == 1 and board[c-3][r+2] != 0:
                        row = r+2
                        if is_valid_location(board, row):
                            col = get_next_open_row(board, row)
                            drop_piece(board, col, row, 2)
                            break
                    elif board[c][r] == 1 and board[c-1][r+1] == 1 and board[c-2][r+2] == 1
and board[c-3][r+3] == 0 and board[c-4][r+3] != 0:
                        row = r+3
                        if is_valid_location(board, row):
                            col = get_next_open_row(board, row)
                            drop_piece(board, col, row, 2)
                            break
```

```
        elif not winning_move(board, 2) and not reach_1(board):
            #次で勝てる or 負ける時以外はランダム
            while True:
                row = random.randrange(7)
                print("Player2 Selected:", row)
                if board[5][row] == 0:
                    break
            if is_valid_location(board, row):
                col = get_next_open_row(board, row)
                drop_piece(board, col, row, 2)

    print_board(board)

    turn += 1
    turn = turn % 2
```

- connect4aFirstFour.py

```
import numpy as np
import random

COLUMN_COUNT = 6
ROW_COUNT = 7

#ボード作成
def create_board():
        board = np.zeros((6,7))
        return board

#コマを打つ
def drop_piece(board, col, row, piece):
        board[col][row] = piece

#その列は，コマが一番上の段まで埋まっていないか
def is_valid_location(board, row):
        return board[5][row] == 0

#その列で打てるマスのうち，最も下の段のマスを返す
def get_next_open_row(board, row):
        for c in range(COLUMN_COUNT):
         if board[c][row] == 0:
                  return c

def print_board(board):
        print(np.flip(board, 0))

#決着の判定
```

```
def winning_move(board, piece):
        #ヨコで決着したか
        for r in range(ROW_COUNT-3):
                for c in range(COLUMN_COUNT):
                        if board[c][r] == piece and board[c][r+1] == piece and board[c][r+2]
== piece and board[c][r+3] == piece:
                                return True

        #タテで決着したか
        for r in range(ROW_COUNT):
                for c in range(COLUMN_COUNT-3):
                        if board[c][r] == piece and board[c+1][r] == piece and board[c+2][r]
== piece and board[c+3][r] == piece:
                                return True

        #右肩上がりナナメで決着したか
        for r in range(ROW_COUNT-3):
                for c in range(COLUMN_COUNT-3):
                        if  board[c][r]  ==  piece  and  board[c+1][r+1]  ==  piece  and
board[c+2][r+2] == piece and board[c+3][r+3] == piece:
                                return True

        #右肩下がりナナメで決着したか
        for r in range(ROW_COUNT-3):
                for c in range(3, COLUMN_COUNT):
                        if board[c][r] == piece and board[c-1][r+1] == piece and board[c-
2][r+2] == piece and board[c-3][r+3] == piece:
                                return True


def reach_1(board):
    #相手のリーチ確認
    #ヨコ１段目のみ
    for r in range(ROW_COUNT-3):
        if board[0][r] == 0 and board[0][r+1] == 1 and board[0][r+2] == 1 and board[0][r+3]
== 1:
            return True
        elif board[0][r] == 1 and board[0][r+1] == 0 and board[0][r+2] == 1 and board[0][r+3]
== 1:
            return True
        elif board[0][r] == 1 and board[0][r+1] == 1 and board[0][r+2] == 0 and board[0][r+3]
== 1:
            return True
        elif board[0][r] == 1 and board[0][r+1] == 1 and board[0][r+2] == 1 and board[0][r+3]
== 0:
            return True

    #ヨコ２段目以降
    for r in range(ROW_COUNT-3):
        for c in range(1, COLUMN_COUNT):
            if board[c][r] == 0 and board[c][r+1] == 1 and board[c][r+2] == 1 and
board[c][r+3] == 1 and board[c-1][r] != 0:
                return True
            elif board[c][r] == 1 and board[c][r+1] == 0 and board[c][r+2] == 1 and
board[c][r+3] == 1 and board[c-1][r+1] != 0:
                return True
            elif board[c][r] == 1 and board[c][r+1] == 1 and board[c][r+2] == 0 and
board[c][r+3] == 1 and board[c-1][r+2] != 0:
                return True
            elif board[c][r] == 1 and board[c][r+1] == 1 and board[c][r+2] == 1 and
board[c][r+3] == 0 and board[c-1][r+3] != 0:
                return True

    #タテ
    for r in range(ROW_COUNT):
        for c in range(COLUMN_COUNT-3):
            if board[c][r] == 1 and board[c+1][r] == 1 and board[c+2][r] == 1 and
board[c+3][r] == 0:
                return True

    #右肩上がりナナメ(1段目マス絡みのみ)
    for r in range(ROW_COUNT-3):
        if board[0][r] == 0 and board[1][r+1] == 1 and board[2][r+2] == 1 and board[3][r+3]
== 1:
            return True
        elif board[0][r] == 1 and board[1][r+1] == 0 and board[2][r+2] == 1 and board[3][r+3]
== 1 and board[0][r+1] != 0:
            return True
        elif board[0][r] == 1 and board[1][r+1] == 1 and board[2][r+2] == 0 and board[3][r+3]
== 1 and board[1][r+2] != 0:
            return True
        elif board[0][r] == 1 and board[1][r+1] == 1 and board[2][r+2] == 1 and board[3][r+3]
== 0 and board[2][r+3] != 0:
            return True

    #右肩上がりナナメ(1段目マス絡み以外)
    for r in range(ROW_COUNT-3):
        for c in range(1,COLUMN_COUNT-3):
            if board[c][r] == 0 and board[c+1][r+1] == 1 and board[c+2][r+2] == 1 and
board[c+3][r+3] == 1 and board[c-1][r] != 0:
                return True
            elif board[c][r] == 1 and board[c+1][r+1] == 0 and board[c+2][r+2] == 1 and
board[c+3][r+3] == 1 and board[c][r+1] != 0:
                return True
            elif board[c][r] == 1 and board[c+1][r+1] == 1 and board[c+2][r+2] == 0 and
board[c+3][r+3] == 1 and board[c+1][r+2] != 0:
```

```
                return True
            elif board[c][r] == 1 and board[c+1][r+1] == 1 and board[c+2][r+2] == 1 and
board[c+3][r+3] == 0 and board[c+2][r+3] != 0:
                return True

    #右肩下がりナナメ(1 段目マス絡みのみ)
    for r in range(ROW_COUNT-3):
        if board[3][r] == 0 and board[2][r+1] == 1 and board[1][r+2] == 1 and board[0][r+3]
== 1 and board[2][r] != 0:
            return True
        elif board[3][r] == 1 and board[2][r+1] == 0 and board[1][r+2] == 1 and board[0][r+3]
== 1 and board[1][r+1] != 0:
            return True
        elif board[3][r] == 1 and board[2][r+1] == 1 and board[1][r+2] == 0 and board[0][r+3]
== 1 and board[0][r+2] != 0:
            return True
        elif board[3][r] == 2 and board[2][r+1] == 2 and board[1][r+2] == 2 and board[0][r+3]
== 0:
            return True

    #右肩下がりナナメ(1 段目マス絡み以外)
    for r in range(ROW_COUNT-3):
        for c in range(4, COLUMN_COUNT):
            if board[c][r] == 0 and board[c-1][r+1] == 1 and board[c-2][r+2] == 1 and board[c-
3][r+3] == 1 and board[c-1][r] != 0:
                return True
            elif board[c][r] == 1 and board[c-1][r+1] == 0 and board[c-2][r+2] == 1 and
board[c-3][r+3] == 1 and board[c-2][r+1] != 0:
                return True
            elif board[c][r] == 1 and board[c-1][r+1] == 1 and board[c-2][r+2] == 0 and
board[c-3][r+3] == 1 and board[c-3][r+2] != 0:
                return True
            elif board[c][r] == 1 and board[c-1][r+1] == 1 and board[c-2][r+2] == 1 and
board[c-3][r+3] == 0 and board[c-4][r+3] != 0:
                return True



board = create_board()
print_board(board)
game_over = False
turn = 0

turn_count = 0 #ターン数

while not game_over:
    #Player 1 の番
    if turn == 0:
        while True:
            row = int(input("Player1 Make Your Selection(0-6)"))
            if board[5][row] == 0:
                break

        if is_valid_location(board, row):
            col = get_next_open_row(board, row)
            drop_piece(board, col, row, 1)
            if winning_move(board, 1):
                print("Player1 Wins!")
                game_over = True
                break

    #Player 2 の番
    else:
        #1 ターン目のコマの置き方
        if turn_count == 1:
            if board[0][0] == 1 or board[0][2] == 1 or board[0][3] == 1 or board[0][4] == 1
or board[0][6] == 1:
                row = 3
                print("Player2 Selected:", row)
                if is_valid_location(board, row):
                    col = get_next_open_row(board, row)
                    drop_piece(board, col, row, 2)
            elif board[0][1] == 1:
                row = 2
                print("Player2 Selected:", row)
                if is_valid_location(board, row):
                    col = get_next_open_row(board, row)
                    drop_piece(board, col, row, 2)
            elif board[0][5] == 1:
                row = 4
                print("Player2 Selected:", row)
                if is_valid_location(board, row):
                    col = get_next_open_row(board, row)
                    drop_piece(board, col, row, 2)

        #2 ターン目のコマの置き方
        elif turn_count == 3:
            #1 ターン目に先手が置いたコマ
            if board[0][0] == 1:
                #2 ターン目に先手が置いたコマ
                if board[0][1] == 1 or board[0][2] == 1 or board[1][3] == 1 or board[0][4]
== 1 or board[0][5] == 1:
                    row = 3
                    print("Player2 Selected:", row)
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                elif board[1][0] == 1:
                    row = 4
                    print("Player2 Selected:", row)
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                elif board[0][6] == 1:
                    row = 2
                    print("Player2 Selected:", row)
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)

            #1 ターン目に先手が置いたコマ
            elif board[0][1] == 1:
                #2 ターン目に先手が置いたコマ
                if board[0][0] == 1 or board[1][2] == 1 or board[0][4] == 1 or board[0][5]
== 1:
                    row = 2
                    print("Player2 Selected:", row)
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                elif board[1][1] == 1:
                    row = 1
                    print("Player2 Selected:", row)
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                elif board[0][3] == 1:
                    row = 3
                    print("Player2 Selected:", row)
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                elif board[0][6] == 1:
                    row = 5
                    print("Player2 Selected:", row)
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)

            #1 ターン目に先手が置いたコマ
            elif board[0][2] == 1:
                #2 ターン目に先手が置いたコマ
                if board[0][0] == 1 or board[0][1] == 1 or board[1][3] == 1 or board[0][4]
== 1 or board[0][5] == 1 or board[0][6] == 1:
                    row = 3
                    print("Player2 Selected:", row)
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                elif board[1][2] == 1:
                    row = 2
                    print("Player2 Selected:", row)
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)

            #1 ターン目に先手が置いたコマ
            elif board[0][3] == 1:
                #2 ターン目に先手が置いたコマ
                if board[0][0] == 1 or board[0][1] == 1 or board[0][4] == 1:
                    row = 2
                    print("Player2 Selected:", row)
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                elif board[0][2] == 1 or board[0][5] == 1 or board[0][6] == 1:
                    row = 4
                    print("Player2 Selected:", row)
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                elif board[2][3] == 1:
                    row = 3
                    print("Player2 Selected:", row)
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)

            #1 ターン目に先手が置いたコマ
            elif board[0][4] == 1:
                #2 ターン目に先手が置いたコマ
                if board[0][0] == 1 or board[0][1] == 1 or board[0][2] == 1 or board[1][3]
== 1 or board[0][5] == 1 or board[0][6] == 1:
                    row = 3
                    print("Player2 Selected:", row)
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                elif board[1][4] == 1:
                    row = 4
                    print("Player2 Selected:", row)
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)

            #1 ターン目に先手が置いたコマ
            elif board[0][5] == 1:
                #2 ターン目に先手が置いたコマ
                if board[0][1] == 1 or board[0][2] == 1 or board[1][4] == 1 or board[0][6]
== 1:
                    row = 4
                    print("Player2 Selected:", row)
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                elif board[0][0] == 1:
                    row = 1
                    print("Player2 Selected:", row)
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                elif board[0][3] == 1:
                    row = 3
                    print("Player2 Selected:", row)
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                elif board[1][5] == 1:
                    row = 5
                    print("Player2 Selected:", row)
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)

            #1 ターン目に先手が置いたコマ
            elif board[0][6] == 1:
                #2 ターン目に先手が置いたコマ
                if board[0][1] == 1 or board[0][2] == 1 or board[1][3] == 1 or board[0][4]
== 1 or board[0][5] == 1:
                    row = 3
                    print("Player2 Selected:", row)
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                elif board[0][0] == 1:
                    row = 4
                    print("Player2 Selected:", row)
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)
                elif board[1][6] == 1:
                    row = 2
                    print("Player2 Selected:", row)
                    if is_valid_location(board, row):
                        col = get_next_open_row(board, row)
                        drop_piece(board, col, row, 2)

        elif 3 < turn_count: #最初の2ターン以外の時
            #ヨコ(1段目マス絡みの決着のみ)
            for r in range(ROW_COUNT-3):
                if board[0][r] == 0 and board[0][r+1] == 2 and board[0][r+2] == 2 and
board[0][r+3] == 2:
                    row = r
                    col = 0
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[0][r] == 2 and board[0][r+1] == 0 and board[0][r+2] == 2 and
board[0][r+3] == 2:
                    row = r+1
                    col = 0
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[0][r] == 2 and board[0][r+1] == 2 and board[0][r+2] == 0 and
board[0][r+3] == 2:
                    row = r+2
                    col = 0
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[0][r] == 2 and board[0][r+1] == 2 and board[0][r+2] == 2 and
board[0][r+3] == 0:
                    row = r+3
                    col = 0
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break

            #ヨコ(2段目以降)
            #4つ目のマスの真下にコマがあるかを判断(以降も「1段目マス絡みの決着のみ」を除い
```

て同様に)

```
            for r in range(ROW_COUNT-3):
                for c in range(1, COLUMN_COUNT):
                    if board[c][r] == 0 and board[c][r+1] == 2 and board[c][r+2] == 2 and
board[c][r+3] == 2 and board[c-1][r] != 0:
                        row = r
                        col = c
                        drop_piece(board, col, row, 2)
                        winning_move(board, 2)
                        print("Player2 Wins!")
                        game_over = True
                        break
                    elif board[c][r] == 2 and board[c][r+1] == 0 and board[c][r+2] == 2 and
board[c][r+3] == 2 and board[c-1][r+1] != 0:
                        row = r+1
                        col = c
                        drop_piece(board, col, row, 2)
                        winning_move(board, 2)
                        print("Player2 Wins!")
                        game_over = True
                        break
                    elif board[c][r] == 2 and board[c][r+1] == 2 and board[c][r+2] == 0 and
board[c][r+3] == 2 and board[c-1][r+2] != 0:
                        row = r+2
                        col = c
                        drop_piece(board, col, row, 2)
                        winning_move(board, 2)
                        print("Player2 Wins!")
                        game_over = True
                        break
                    elif board[c][r] == 2 and board[c][r+1] == 2 and board[c][r+2] == 2 and
board[c][r+3] == 0 and board[c-1][r+3] != 0:
                        row = r+3
                        col = c
                        drop_piece(board, col, row, 2)
                        winning_move(board, 2)
                        print("Player2 Wins!")
                        game_over = True
                        break

            #vertical タテ
            for r in range(ROW_COUNT):
                for c in range(COLUMN_COUNT-3):
                    if board[c][r] == 2 and board[c+1][r] == 2 and board[c+2][r] == 2 and
board[c+3][r] == 0:
                        row = r
                        col = c+3
                        drop_piece(board, col, row, 2)
                        winning_move(board, 2)
                        print("Player2 Wins!")
                        game_over = True
                        break

            #右肩上がりナナメ(1段目マス絡みの決着のみ)
            for r in range(ROW_COUNT-3):
                if board[0][r] == 0 and board[1][r+1] == 2 and board[2][r+2] == 2 and board[3][r+3] == 2:
                    row = r
                    col = 0
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[0][r] == 2 and board[1][r+1] == 0 and board[2][r+2] == 2 and board[3][r+3] == 2 and board[0][r+1] != 0:
                    row = r+1
                    col = 1
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[0][r] == 2 and board[1][r+1] == 2 and board[2][r+2] == 0 and board[3][r+3] == 2 and board[1][r+2] != 0:
                    row = r+2
                    col = 2
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[0][r] == 2 and board[1][r+1] == 2 and board[2][r+2] == 2 and board[3][r+3] == 0 and board[2][r+3] != 0:
                    row = r+3
                    col = 3
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break

            #右肩上がりナナメ(1段目マス絡み以外)
            for r in range(ROW_COUNT-3):
                for c in range(1,COLUMN_COUNT-3):
                    if board[c][r] == 0 and board[c+1][r+1] == 2 and board[c+2][r+2] == 2
```

```
and board[c+3][r+3] == 2 and board[c-1][r] != 0:
                        row = r
                        col = c
                        drop_piece(board, col, row, 2)
                        winning_move(board, 2)
                        print("Player2 Wins!")
                        game_over = True
                        break
                    elif board[c][r] == 2 and board[c+1][r+1] == 0 and board[c+2][r+2] == 2
and board[c+3][r+3] == 2 and board[c][r+1] != 0:
                        row = r+1
                        col = c+1
                        drop_piece(board, col, row, 2)
                        winning_move(board, 2)
                        print("Player2 Wins!")
                        game_over = True
                        break
                    elif board[c][r] == 2 and board[c+1][r+1] == 2 and board[c+2][r+2] == 0
and board[c+3][r+3] == 2 and board[c+1][r+2] != 0:
                        row = r+2
                        col = c+2
                        drop_piece(board, col, row, 2)
                        winning_move(board, 2)
                        print("Player2 Wins!")
                        game_over = True
                        break
                    elif board[c][r] == 2 and board[c+1][r+1] == 2 and board[c+2][r+2] == 2
and board[c+3][r+3] == 0 and board[c+2][r+3] != 0:
                        row = r+3
                        col = c+3
                        drop_piece(board, col, row, 2)
                        winning_move(board, 2)
                        print("Player2 Wins!")
                        game_over = True
                        break

            #右肩下がりナナメ(1段目マス絡みの決着のみ)
            for r in range(ROW_COUNT-3):
                if board[3][r] == 0 and board[2][r+1] == 2 and board[1][r+2] == 2 and
board[0][r+3] == 2 and board[2][r] != 0:
                    row = r
                    col = 3
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[3][r] == 2 and board[2][r+1] == 0 and board[1][r+2] == 2 and
```

```
board[0][r+3] == 2 and board[1][r+1] != 0:
                    row = r+1
                    col = 2
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[3][r] == 2 and board[2][r+1] == 2 and board[1][r+2] == 0 and
board[0][r+3] == 2 and board[0][r+2] != 0:
                    row = r+2
                    col = 1
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break
                elif board[3][r] == 2 and board[2][r+1] == 2 and board[1][r+2] == 2 and
board[0][r+3] == 0:
                    row = r+3
                    col = 0
                    drop_piece(board, col, row, 2)
                    winning_move(board, 2)
                    print("Player2 Wins!")
                    game_over = True
                    break

            #右肩下がりナナメ(1段目マス絡み以外)
            for r in range(ROW_COUNT-3):
                for c in range(4, COLUMN_COUNT):
                    if board[c][r] == 0 and board[c-1][r+1] == 2 and board[c-2][r+2] == 2
and board[c-3][r+3] == 2 and board[c-1][r] != 0:
                        row = r
                        col = c
                        drop_piece(board, col, row, 2)
                        winning_move(board, 2)
                        print("Player2 Wins!")
                        game_over = True
                        break
                    elif board[c][r] == 2 and board[c-1][r+1] == 0 and board[c-2][r+2] == 2
and board[c-3][r+3] == 2 and board[c-2][r+1] != 0:
                        row = r+1
                        col = c-1
                        drop_piece(board, col, row, 2)
                        winning_move(board, 2)
                        print("Player2 Wins!")
                        game_over = True
                        break
                    elif board[c][r] == 2 and board[c-1][r+1] == 2 and board[c-2][r+2] == 0
and board[c-3][r+3] == 2 and board[c-3][r+2] != 0:
                        row = r+2
                        col = c-2
                        drop_piece(board, col, row, 2)
                        winning_move(board, 2)
                        print("Player2 Wins!")
                        game_over = True
                        break
                    elif board[c][r] == 2 and board[c-1][r+1] == 2 and board[c-2][r+2] == 2
and board[c-3][r+3] == 0 and board[c-4][r+3] != 0:
                        row = r+3
                        col = c-3
                        drop_piece(board, col, row, 2)
                        winning_move(board, 2)
                        print("Player2 Wins!")
                        game_over = True
                        break

            if not winning_move(board, 2) and reach_1(board): #相手がリーチかけられた時，先
に置いて防ぐ
                #ヨコ１段目のみ
                for r in range(ROW_COUNT-3):
                    if board[0][r] == 0 and board[0][r+1] == 1 and board[0][r+2] == 1 and
board[0][r+3] == 1:
                        row = r
                        if is_valid_location(board, row):
                            col = get_next_open_row(board, row)
                            drop_piece(board, col, row, 2)
                            break
                    elif board[0][r] == 1 and board[0][r+1] == 0 and board[0][r+2] == 1 and
board[0][r+3] == 1:
                        row = r+1
                        if is_valid_location(board, row):
                            col = get_next_open_row(board, row)
                            drop_piece(board, col, row, 2)
                            break
                    elif board[0][r] == 1 and board[0][r+1] == 1 and board[0][r+2] == 0 and
board[0][r+3] == 1:
                        row = r+2
                        if is_valid_location(board, row):
                            col = get_next_open_row(board, row)
                            drop_piece(board, col, row, 2)
                            break
                    elif board[0][r] == 1 and board[0][r+1] == 1 and board[0][r+2] == 1 and
board[0][r+3] == 0:
                        row = r+3
                        if is_valid_location(board, row):
                            col = get_next_open_row(board, row)
                            drop_piece(board, col, row, 2)
                            break

                #ヨコ2段目以降
                for r in range(ROW_COUNT-3):
                    for c in range(1, COLUMN_COUNT):
                        if board[c][r] == 0 and board[c][r+1] == 1 and board[c][r+2] == 1
and board[c][r+3] == 1 and board[c-1][r] != 0:
                            row = r
                            if is_valid_location(board, row):
                                col = get_next_open_row(board, row)
                                drop_piece(board, col, row, 2)
                                break
                        elif board[c][r] == 1 and board[c][r+1] == 0 and board[c][r+2] == 1
and board[c][r+3] == 1 and board[c-1][r+1] != 0:
                            row = r+1
                            if is_valid_location(board, row):
                                col = get_next_open_row(board, row)
                                drop_piece(board, col, row, 2)
                                break
                        elif board[c][r] == 1 and board[c][r+1] == 1 and board[c][r+2] == 0
and board[c][r+3] == 1 and board[c-1][r+2] != 0:
                            row = r+2
                            if is_valid_location(board, row):
                                col = get_next_open_row(board, row)
                                drop_piece(board, col, row, 2)
                                break
                        elif board[c][r] == 1 and board[c][r+1] == 1 and board[c][r+2] == 1
and board[c][r+3] == 0 and board[c-1][r+3] != 0:
                            row = r+3
                            if is_valid_location(board, row):
                                col = get_next_open_row(board, row)
                                drop_piece(board, col, row, 2)
                                break

                #タテ
                for r in range(ROW_COUNT):
                    for c in range(COLUMN_COUNT-3):
                        if board[c][r] == 1 and board[c+1][r] == 1 and board[c+2][r] == 1
and board[c+3][r] == 0:
                            row = r
                            if is_valid_location(board, row):
                                col = get_next_open_row(board, row)
                                drop_piece(board, col, row, 2)
                                break

                #右肩上がりナナメ(1段目マス絡みのみ)
                for r in range(ROW_COUNT-3):
                    if board[0][r] == 0 and board[1][r+1] == 1 and board[2][r+2] == 1 and
board[3][r+3] == 1:
                        row = r
                        if is_valid_location(board, row):
                            col = get_next_open_row(board, row)
                            drop_piece(board, col, row, 2)
                            break
                    elif board[0][r] == 1 and board[1][r+1] == 0 and board[2][r+2] == 1 and
board[3][r+3] == 1 and board[0][r+1] != 0:
                        row = r+1
                        if is_valid_location(board, row):
                            col = get_next_open_row(board, row)
                            drop_piece(board, col, row, 2)
                            break
                    elif board[0][r] == 1 and board[1][r+1] == 1 and board[2][r+2] == 0 and
board[3][r+3] == 1 and board[1][r+2] != 0:
                        row = r+2
                        if is_valid_location(board, row):
                            col = get_next_open_row(board, row)
                            drop_piece(board, col, row, 2)
                            break
                    elif board[0][r] == 1 and board[1][r+1] == 1 and board[2][r+2] == 1 and
board[3][r+3] == 0 and board[2][r+3] != 0:
                        row = r+3
                        if is_valid_location(board, row):
                            col = get_next_open_row(board, row)
                            drop_piece(board, col, row, 2)
                            break

                #右肩上がりナナメ(1段目マス絡み以外)
                for r in range(ROW_COUNT-3):
                    for c in range(1,COLUMN_COUNT-3):
                        if board[c][r] == 0 and board[c+1][r+1] == 1 and board[c+2][r+2] ==
1 and board[c+3][r+3] == 1 and board[c-1][r] != 0:
                            row = r
                            if is_valid_location(board, row):
                                col = get_next_open_row(board, row)
                                drop_piece(board, col, row, 2)
                                break
                        elif board[c][r] == 1 and board[c+1][r+1] == 0 and board[c+2][r+2]
== 1 and board[c+3][r+3] == 1 and board[c][r+1] != 0:
                            row = r+1
                            if is_valid_location(board, row):
                                col = get_next_open_row(board, row)
                                drop_piece(board, col, row, 2)
                                break
                        elif board[c][r] == 1 and board[c+1][r+1] == 1 and board[c+2][r+2]
```

```
== 0 and board[c+3][r+3] == 1 and board[c+1][r+2] != 0:
                            row = r+2
                            if is_valid_location(board, row):
                                col = get_next_open_row(board, row)
                                drop_piece(board, col, row, 2)
                                break
                        elif board[c][r] == 1 and board[c+1][r+1] == 1 and board[c+2][r+2]
== 1 and board[c+3][r+3] == 0 and board[c+2][r+3] != 0:
                            row = r+3
                            if is_valid_location(board, row):
                                col = get_next_open_row(board, row)
                                drop_piece(board, col, row, 2)
                                break

                #右肩下がりナナメ(1段目マス絡みのみ)
                for r in range(ROW_COUNT-3):
                    if board[3][r] == 0 and board[2][r+1] == 1 and board[1][r+2] == 1 and
board[0][r+3] == 1 and board[2][r] != 0:
                        row = r
                        if is_valid_location(board, row):
                            col = get_next_open_row(board, row)
                            drop_piece(board, col, row, 2)
                            break
                    elif board[3][r] == 1 and board[2][r+1] == 0 and board[1][r+2] == 1 and
board[0][r+3] == 1 and board[1][r] != 0:
                        row = r+1
                        if is_valid_location(board, row):
                            col = get_next_open_row(board, row)
                            drop_piece(board, col, row, 2)
                            break
                    elif board[3][r] == 1 and board[2][r+1] == 1 and board[1][r+2] == 0 and
board[0][r+3] == 1 and board[0][r+2] != 0:
                        row = r+2
                        if is_valid_location(board, row):
                            col = get_next_open_row(board, row)
                            drop_piece(board, col, row, 2)
                            break
                    elif board[3][r] == 2 and board[2][r+1] == 2 and board[1][r+2] == 2 and
board[0][r+3] == 0:
                        row = r+3
                        if is_valid_location(board, row):
                            col = get_next_open_row(board, row)
                            drop_piece(board, col, row, 2)
                            break

                #右肩下がりナナメ(1段目マス絡み以外)
                for r in range(ROW_COUNT-3):
                    for c in range(4, COLUMN_COUNT):
                        if board[c][r] == 0 and board[c-1][r+1] == 1 and board[c-2][r+2] ==
1 and board[c-3][r+3] == 1 and board[c-1][r] != 0:
                            row = r
                            if is_valid_location(board, row):
                                col = get_next_open_row(board, row)
                                drop_piece(board, col, row, 2)
                                break
                        elif board[c][r] == 1 and board[c-1][r+1] == 0 and board[c-2][r+2]
== 1 and board[c-3][r+3] == 1 and board[c-2][r+1] != 0:
                            row = r+1
                            if is_valid_location(board, row):
                                col = get_next_open_row(board, row)
                                drop_piece(board, col, row, 2)
                                break
                        elif board[c][r] == 1 and board[c-1][r+1] == 1 and board[c-2][r+2]
== 0 and board[c-3][r+3] == 1 and board[c-3][r+2] != 0:
                            row = r+2
                            if is_valid_location(board, row):
                                col = get_next_open_row(board, row)
                                drop_piece(board, col, row, 2)
                                break
                        elif board[c][r] == 1 and board[c-1][r+1] == 1 and board[c-2][r+2]
== 1 and board[c-3][r+3] == 0 and board[c-4][r+3] != 0:
                            row = r+3
                            if is_valid_location(board, row):
                                col = get_next_open_row(board, row)
                                drop_piece(board, col, row, 2)
                                break

            elif not winning_move(board, 2) and not reach_1(board):
                #次で勝てる or 負ける時以外はランダム
                while True:
                    row = random.randrange(7)
                    print("Player2 Selected:", row)
                    if board[5][row] == 0:
                        break
                if is_valid_location(board, row):
                    col = get_next_open_row(board, row)
                    drop_piece(board, col, row, 2)

    print_board(board)

    turn += 1
    turn = turn % 2
    turn_count += 1
```