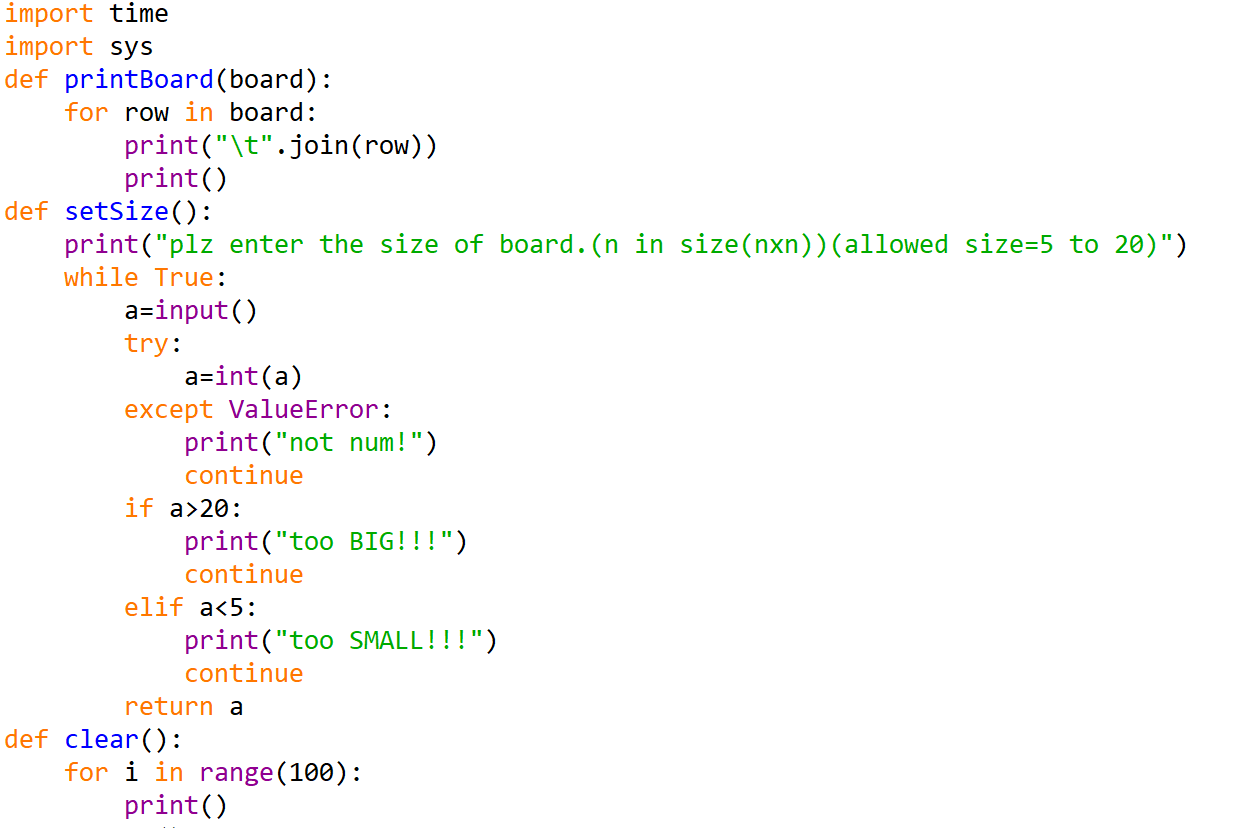
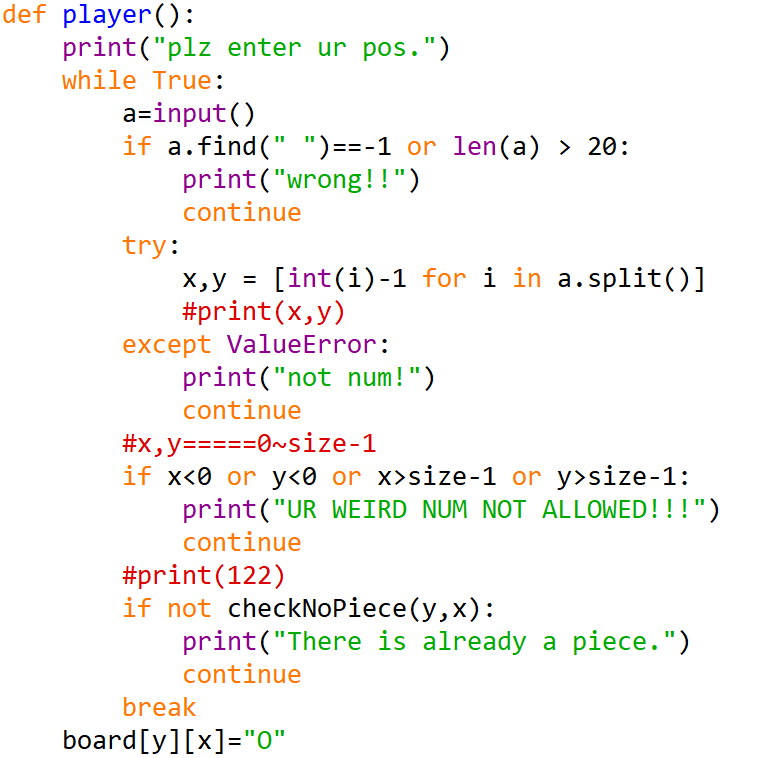
**五子棋遊戲及下棋AI的實作**

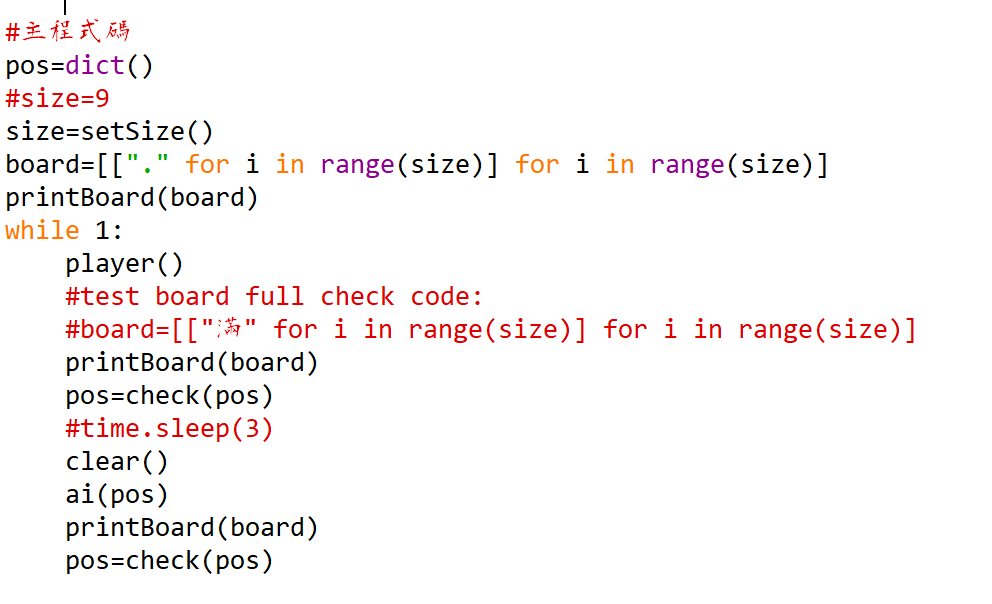
**成品連結:** [**https://drive.google.com/file/d/10qZWmcq1QD9OySQ8SyYWdYT-jZsmGb1d/view?usp=sharing**](https://drive.google.com/file/d/10qZWmcq1QD9OySQ8SyYWdYT-jZsmGb1d/view?usp=sharing)

1. 實作動機

因有位同學僅利用一天，就寫出了五子棋的遊戲，使當時正在學pygame而備感困難的我知道了專題可以做什麼-----不用pygame的棋類遊戲。點開他的程式碼後，我赫然發現他的程式居然會自動下棋！而從來沒有寫過太複雜的程式，甚至是AI的我，便好奇我是否有能力作出下五子棋的AI?於是，我利用三天的時間，先作出基本的五子棋遊戲(因我不求遊戲品質)，再試著寫出下棋AI的程式，並求更聰明。

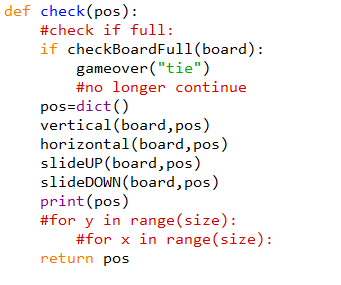
1. 實作過程
2. 我先寫出基本遊戲架構，確保其可執行



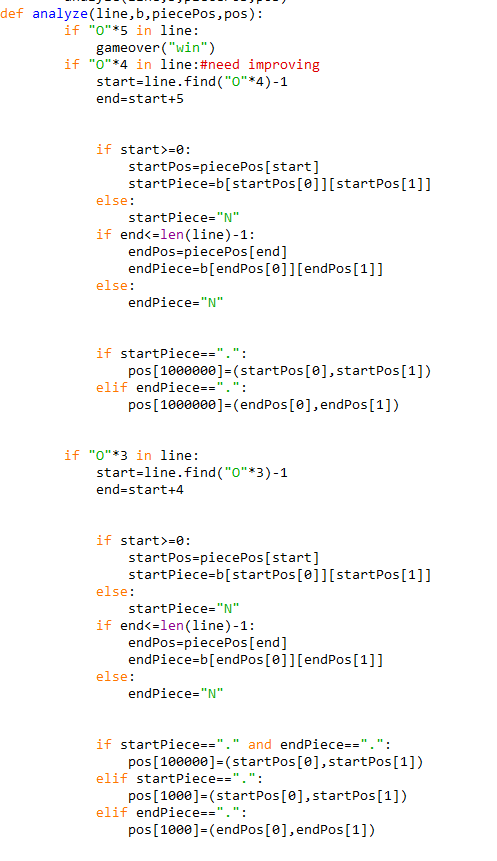


1. 利用判斷玩家下完旗後的棋盤狀況，來決定電腦要下在哪。

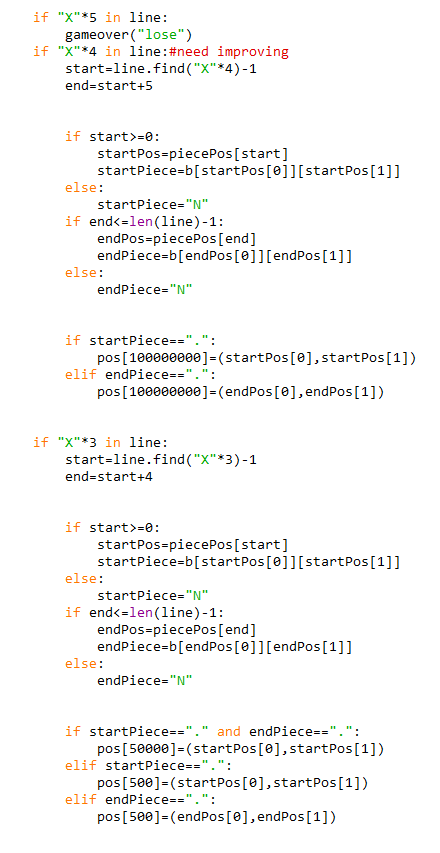
缺點:電腦只能針對每一排個別作分析，並不能同時看很多排(例如當電腦下在棋盤上某一個位置時，就可形成兩組三連線，此時電腦不一定會下在那麼好的位置)





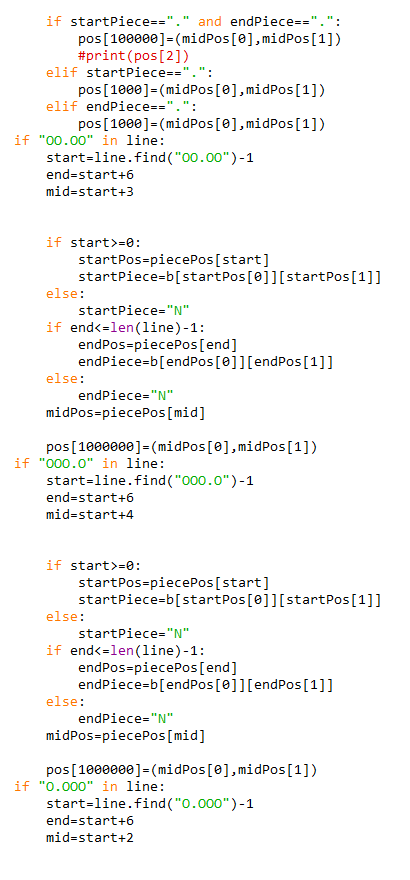


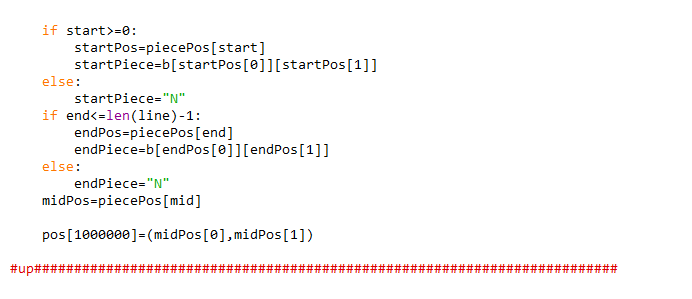


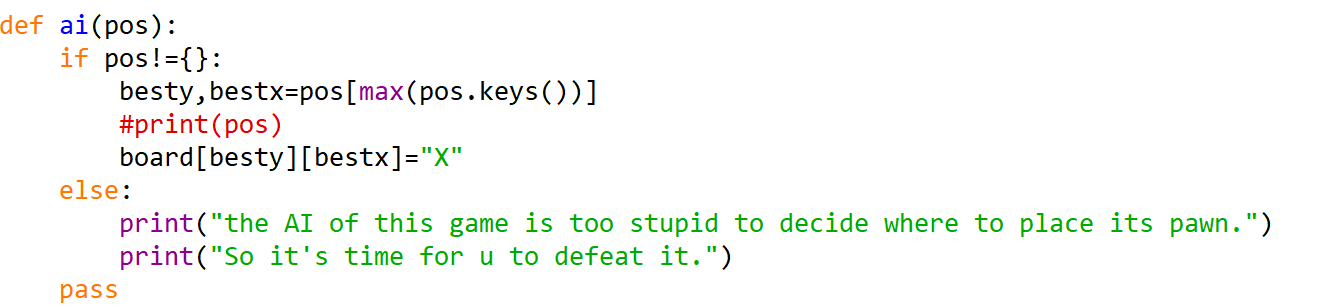












1. 改善下棋AI:判斷當電腦下在每一個可下之處時能得到幾分(一樣，分析下棋後的棋盤)，並將分數及位置存入Dictionary，最後電腦挑出最高分的位置下棋。

缺點:AI只能預測表面情況，並不能發現(預測)未來可能的盤面情形，故只要有智商就不會輸給這個爛AI。

1. 改善下棋AI:判斷當電腦下在每一個可下之處時能得到幾分(也一樣，分析下棋後的棋盤)，再由電腦模擬玩家下棋(挑出表面上(也就是利用剛剛的AI)來找出最好下的點並下棋)，再由電腦下在每一個可下之處，如此一層一層重複。最後將每一層的分數依比例算出總分，並將總分及位置存入Dictionary，最後電腦挑出最高分的位置下棋。

缺點:AI模擬玩家下棋時，是利用前一次(3)的爛AI，不保證下的位置對於AI最不利。

1. 利用minimax演算法找出最好的位置，並利用alpha-beta剪枝法去掉多餘的遞迴。

缺點:檢查深度越深，AI就越聰明，但計算量越大(迴圈叫太多次)，目前利用統計剩餘下棋格數(反映電腦分析時的計算量)，去在AI每次分析時決定其分析深度，當剩餘越多空格，就分析地越淺，雖電腦較笨，但運算資料量不會爆滿。

1. 加上顏色和行數之標示，更方便遊玩。
2. 一張含有 螢幕擷取畫面 的圖片

   自動產生的描述更改玩家輸入系統：玩家剛開始會需要輸入難度(改變計算AI分析深度之程式的一個常數)，因為AI分析深度越淺，所花費之時間越少，玩家會較願意遊玩。同時，令玩家使用上下左右鍵輸入下棋位置，並修改印出棋盤的程式使其可以特別改變背景色標示特定格子(如下圖)。
3. 程式遊玩方式
4. 先輸入遊戲難度(0或1)。
5. 再輸入棋盤大小(如要17x17就輸入17)。
6. 使用上下左右鍵調整想要下的位置。若錯誤訊息太多導致看不見棋盤，可以按”p”鍵。最後按ENTER確定位置。
7. 耐心等待1秒~5分鐘不等，取決於其空格數量及分析深度(受開始遊戲時輸入之遊戲難度影響)
8. 耐心等待程式再次出現下圖，即表示可以自步驟2重複執行(又輪到玩家下棋)。
9. 心得

我藉由改進我的AI，學會了minimax的皮毛。而希望未來的繼續透過改善程式或利用其他演算法，不只讓AI更聰明，也要更快!

我也可以嘗試利用pygame來製作此遊戲，改善圖像介面。

1. 最終程式碼

import time

import sys

import colored

import keyboard

#印出棋盤，包括要特別標記的格子(以其橫排數chosenROWNUM和其直排數chosenPIECENUM表示)

def printBoard(board,width,chosenROWNUM=-1,chosenPIECENUM=-1):#若不須標記格子，則原本要特別標記的格子之座標預設為(-1,-1)，程式就不會找到對應的格子特別標示了

print("\t",end="")

for i in range(size):

print(" "+str(i+1)+" ",end=" "\*(width-(len(str(i+1))-1)))#印出直排行數

print("\n")#換兩行==print()print()

#跑遍各排的各顆棋子

for rowNum in range(size):

print(str(rowNum+1),end="\t")#印出橫排行數

for pieceNum in range(size):

piece=board[rowNum][pieceNum]

#針對玩家棋子、AI棋子、或空格，印出不同的前景色

if piece=="O":

THISfcolor=colored.fg("green")

elif piece=="X":

THISfcolor=colored.fg("red\_1")

elif piece==".":

THISfcolor=fcolor

#若該格是要特別標記的格子，則改變其背景色

if rowNum==chosenROWNUM and pieceNum==chosenPIECENUM:

THISbcolor=markedcolor

else:

THISbcolor=bcolor

print(THISfcolor+THISbcolor+" "+piece+" "+reset,end="")#印出一格棋子

if not pieceNum==size-1: #不是最後一個棋子就再加上空白

print(fcolor+bcolor+" " \* width+reset,end="")

print()

if not rowNum==size-1: #不是最後一排就再加上一行空白!

print("\t"+fcolor+bcolor+" "\*(width\*(size-1)+size\*(1+2))+reset)

#設定難度，即AI的搜尋深度

#0:搜尋深度淺，花費時間少;1:搜尋深度深，花費時間多

def setDifficulty():

print("plz choose the difficulty.(0:AI's stupid while spending shorter time\t/\t1:AI's smart while spending more time)")

while True:

a=input()

try:

a=int(a)

except ValueError:

print("not num!")

continue

if a==0:

return 6

elif a==1:

return 7

else:

print("plz enter 0 or 1.")

continue

#設定棋盤大小

def setSize():

print("plz enter the size of board.(n in size(nxn))(allowed size=5 to 20)")

while True:

a=input()

try:

a=int(a)

except ValueError:

print("not num!")

continue

if a>20:

print("too BIG!!!")

continue

elif a<5:

print("too SMALL!!!")

continue

return a

#清空畫面(感謝徐晧倫)

def clear():

print("\n"\*100)

#玩家下棋

def player():

def getPlayerInput(x,y):#讓玩家調整要下棋的位置並等待按下enter

while 1:

print("plz choose ur pos.(Press the arrow keys to choose the position u like. Then submit it by pressing enter.)\n\t\t(press P should the board need reprinting.)")

while 1:#等待玩家按下按鍵

if keyboard.is\_pressed(77) and x+1<size: #if right arrow is pressed

x+=1

break

if keyboard.is\_pressed(75) and x-1>=0: #if left arrow is pressed

x-=1

break

if keyboard.is\_pressed(72) and y-1>=0: #if up arrow is pressed

y-=1

break

if keyboard.is\_pressed(80) and y+1<size: #if down arrow is pressed

y+=1

break

if keyboard.is\_pressed(28): #if enter is pressed

time.sleep(0.1)

return x,y

if keyboard.is\_pressed(25): #if p is pressed

break

clear()

printBoard(board,boardWidth,y,x)

time.sleep(0.1)#延遲以避免按一次就執行多次

x=int(size/2)

y=int(size/2)

clear()

printBoard(board,boardWidth,y,x)#標記棋盤中心點作為預設下棋位置

#print("plz choose ur pos.")

while True:

x,y=getPlayerInput(x,y)#等待玩家案enter輸入

if not checkNoPiece(y,x):#若下在已經有棋子的地方

print(colored.fg("white")+colored.bg("red\_1")+"There is already a piece."+reset)

continue

break

board[y][x]="O"#下棋

#檢查是否下在沒棋子之處

def checkNoPiece(y,x):

return board[y][x]=="."

#檢查棋盤是否已滿

def checkBoardFull(board):

isFull = True

for row in board:

for piece in row:

if piece==".":

isFull=False

return isFull

#以minimax搜索最佳下棋位置

def DeepCheck(checkDepth,onlyCheck):

if checkBoardFull(board):

gameover("tie")

positionsss=dict()

depth=0

newBoard=[[i for i in row] for row in board]

positionsss=AllCheck("AI",depth,newBoard,checkDepth,onlyCheck,0,False)

return positionsss

#檢查一層的所有可下棋位置

def AllCheck(who,depth,newBoard,checkDepth,onlyCheck,alpha,checkAplhaBeta):

cut=False

beta="no"

positions=dict()

#所有可下棋位置

for y in range(size):

for x in range(size):

if newBoard[y][x]==".":

#計算下在各格的分數

score=check(x,y,positions,who,depth,checkDepth,newBoard,onlyCheck,beta,beta!="no")

#存入dictionary

positions[score]=(y,x)

#存較大層之最大or最小分數

if who=="AI":

beta=max(positions.keys())

elif who=="playerAI":

beta=min(positions.keys())

#alpha-beta剪枝

if checkAplhaBeta:

if who=="AI" and beta>alpha and checkAplhaBeta:

cut=True

break

elif who=="playerAI" and beta<alpha and checkAplhaBeta:

cut=True

break

if cut:

break

#回傳總分

if depth==0:

#print(positions)

return positions

else:

#回傳各層最大or最小分數

if positions!=dict():

if who=="playerAI":

return min(positions.keys())

if who=="AI":

return max(positions.keys())

return 0

#檢查各格分數

def check(x,y,positions,who,depth,checkDepth,newBoard,onlyCheck,beta,checkAplhaBeta):

end=False

#試著下一顆棋

aWholeNewBoard=[[i for i in row]for row in newBoard]

if not onlyCheck:

if who=="playerAI":

aWholeNewBoard[y][x]="O"

elif who=="AI":

aWholeNewBoard[y][x]="X"

#計算直排、橫排、斜排分數

pos=0

pos,end1=vertical(aWholeNewBoard,pos,who,depth,onlyCheck)

pos,end2=horizontal(aWholeNewBoard,pos,who,depth,onlyCheck)

pos,end3=slideUP(aWholeNewBoard,pos,who,depth,onlyCheck)

pos,end4=slideDOWN(aWholeNewBoard,pos,who,depth,onlyCheck)

#是否下滿

if end1 or end2 or end3 or end4:

end=True

#遞迴

if who=="playerAI":

if end or checkBoardFull(aWholeNewBoard) or depth+1>=checkDepth:#結束搜尋，傳回分數

return pos

else:

return AllCheck("AI",depth+1,aWholeNewBoard,checkDepth,onlyCheck,beta,checkAplhaBeta)

elif who=="AI":

if end or checkBoardFull(aWholeNewBoard) or depth+1>=checkDepth:#結束搜尋，傳回分數

return pos

else:

return AllCheck("playerAI",depth+1,aWholeNewBoard,checkDepth,onlyCheck,beta,checkAplhaBeta)

#直排

def vertical(b,pos,who,depth,onlyCheck):

for x in range(size):

line = [b[y][x] for y in range(size)]

piecePos=[(y,x) for y in range(size)]

line = "".join(line)

pos,end=analyze(line,b,piecePos,pos,who,depth,onlyCheck)

return pos,end

#橫排

def horizontal(b,pos,who,depth,onlyCheck):

for y in range(size):

line = [b[y][x] for x in range(size)]

piecePos=[(y,x) for x in range(size)]

line = "".join(line)

pos,end=analyze(line,b,piecePos,pos,who,depth,onlyCheck)

return pos,end

#斜排

def slideUP(b,pos,who,depth,onlyCheck):

for k in range(0+4,(size\*2-1)-4):

line = [b[y][x] for x in range(size) for y in range(size) if x+y==k]

piecePos=[(y,x) for x in range(size) for y in range(size) if x+y==k]

line = "".join(line)

pos,end=analyze(line,b,piecePos,pos,who,depth,onlyCheck)

return pos,end

#斜排

def slideDOWN(b,pos,who,depth,onlyCheck):

for k in range((-(size-1))+4,(size)-4):

line = [b[y][x] for x in range(size) for y in range(size) if x-y==k]

piecePos=[(y,x) for x in range(size) for y in range(size) if x-y==k]

line = "".join(line)

pos,end=analyze(line,b,piecePos,pos,who,depth,onlyCheck)

return pos,end

#分析情勢再加減分數

def analyze(line,b,piecePos,pos,who,depth,onlyCheck):

#playerAI = O AI = X

s,e="X","O"

how=(1/10)\*\*depth#越下層之分數越接近0，較不會影響總分

checkend=False

#判斷情勢

if e\*5 in line:

if depth==0 and onlyCheck:

gameover("win")#玩家贏了

pos-=10000000000 \* how

print("too bad")

checkend=True

if e\*4 in line:

start=line.find(e\*4)-1

end=start+5

if start>=0:

startPos=piecePos[start]

startPiece=b[startPos[0]][startPos[1]]

else:

startPiece="N"

if end<=len(line)-1:

endPos=piecePos[end]

endPiece=b[endPos[0]][endPos[1]]

else:

endPiece="N"

if startPiece=="." or endPiece==".":

pos-=10000000 \* how

if e\*3 in line:

start=line.find(e\*3)-1

end=start+4

if start>=0:

startPos=piecePos[start]

startPiece=b[startPos[0]][startPos[1]]

else:

startPiece="N"

if end<=len(line)-1:

endPos=piecePos[end]

endPiece=b[endPos[0]][endPos[1]]

else:

endPiece="N"

if startPiece=="." and endPiece==".":

pos-=1000000 \* how

elif startPiece=="." or endPiece==".":

pos-=5000 \* how

if e\*2 in line:

start=line.find(e\*2)-1

end=start+3

if start>=0:

startPos=piecePos[start]

startPiece=b[startPos[0]][startPos[1]]

else:

startPiece="N"

if end<=len(line)-1:

endPos=piecePos[end]

endPiece=b[endPos[0]][endPos[1]]

else:

endPiece="N"

if startPiece=="." and endPiece==".":

pos-=100 \* how

elif startPiece=="." or endPiece==".":

pos-=50 \* how

if e\*1 in line:

start=line.find(e\*1)-1

end=start+2

if start>=0:

startPos=piecePos[start]

startPiece=b[startPos[0]][startPos[1]]

else:

startPiece="N"

if end<=len(line)-1:

endPos=piecePos[end]

endPiece=b[endPos[0]][endPos[1]]

else:

endPiece="N"

if startPiece=="." and endPiece==".":

pos-=3 \* how

elif startPiece==".":

pos-=1 \* how

elif endPiece==".":

pos-=1 \* how

if s\*5 in line:

if depth==0 and onlyCheck:

gameover("lose")#AI贏了

pos+=10000000000 \* how

print("too good")

checkend=True

if s\*4 in line:

start=line.find(s\*4)-1

end=start+5

if start>=0:

startPos=piecePos[start]

startPiece=b[startPos[0]][startPos[1]]

else:

startPiece="N"

if end<=len(line)-1:

endPos=piecePos[end]

endPiece=b[endPos[0]][endPos[1]]

else:

endPiece="N"

if startPiece=="." and endPiece==".":

pos+=10000000 \* how

elif startPiece=="." or endPiece==".":

pos+=5000 \* how

if s\*3 in line:

start=line.find(s\*3)-1

end=start+4

if start>=0:

startPos=piecePos[start]

startPiece=b[startPos[0]][startPos[1]]

else:

startPiece="N"

if end<=len(line)-1:

endPos=piecePos[end]

endPiece=b[endPos[0]][endPos[1]]

else:

endPiece="N"

if startPiece=="." and endPiece==".":

pos+=1000 \* how

elif startPiece=="." or endPiece==".":

pos+=10 \* how

if s\*2 in line:

start=line.find(s\*2)-1

end=start+3

if start>=0:

startPos=piecePos[start]

startPiece=b[startPos[0]][startPos[1]]

else:

startPiece="N"

if end<=len(line)-1:

endPos=piecePos[end]

endPiece=b[endPos[0]][endPos[1]]

else:

endPiece="N"

if startPiece=="." and endPiece==".":

pos+=5 \* how

elif startPiece=="." or endPiece==".":

pos+=2 \* how

if s\*1 in line:

start=line.find(s\*1)-1

end=start+2

if start>=0:

startPos=piecePos[start]

startPiece=b[startPos[0]][startPos[1]]

else:

startPiece="N"

if end<=len(line)-1:

endPos=piecePos[end]

endPiece=b[endPos[0]][endPos[1]]

else:

endPiece="N"

if startPiece=="." and endPiece==".":

pos+=2 \* how

elif startPiece=="." or endPiece==".":

pos+=1 \* how

if e+"."+e in line:

start=line.find(e+"."+e)-1

end=start+4

if start>=0:

startPos=piecePos[start]

startPiece=b[startPos[0]][startPos[1]]

else:

startPiece="N"

if end<=len(line)-1:

endPos=piecePos[end]

endPiece=b[endPos[0]][endPos[1]]

else:

endPiece="N"

if startPiece=="." and endPiece==".":

pos-=500000 \* how

elif startPiece==".":

pos-=5000 \* how

elif endPiece==".":

pos-=5000 \* how

if e+e+"."+e in line:

start=line.find(e+e+"."+e)-1

end=start+5

if start>=0:

startPos=piecePos[start]

startPiece=b[startPos[0]][startPos[1]]

else:

startPiece="N"

if end<=len(line)-1:

endPos=piecePos[end]

endPiece=b[endPos[0]][endPos[1]]

else:

endPiece="N"

if startPiece=="." and endPiece==".":

pos-=1000000 \* how

elif startPiece==".":

pos-=10000 \* how

elif endPiece==".":

pos-=10000 \* how

if e+"."+e+e in line:

start=line.find(e+"."+e+e)-1

end=start+5

if start>=0:

startPos=piecePos[start]

startPiece=b[startPos[0]][startPos[1]]

else:

startPiece="N"

if end<=len(line)-1:

endPos=piecePos[end]

endPiece=b[endPos[0]][endPos[1]]

else:

endPiece="N"

if startPiece=="." and endPiece==".":

pos-=1000000 \* how

elif startPiece==".":

pos-=10000 \* how

elif endPiece==".":

pos-=10000 \* how

if e+e+"."+e+e in line:

pos-=10000000 \* how

if e+e+e+"."+e in line:

pos-=10000000 \* how

if e+"."+e+e+e in line:

pos-=10000000 \* how

return pos,checkend

#AI下棋

def ai(positions,board):

if positions!=dict():

besty,bestx=positions[max(positions.keys())]

board[besty][bestx]="X"

else:

print("the AI of this game is too stupid to decide where to place its pawn.")

print("So it's time for u to defeat it.")

return board

#遊戲結束

def gameover(text):

if text=="win":

print("U win!!!")

elif text=="lose":

print("haha u lose!!!")

elif text=="tie":

print("the board is full!!!")

print("LOOK WHAT U'VE DONE!!!")

time.sleep(5)

print("end!")

sys.exit()#end script

#初始化

k=7#default

k=setDifficulty()

size=setSize()

CheckDepth=1

board=[["." for i in range(size)] for i in range(size)]

boardWidth=1

#定顏色

fcolor=colored.fg("black")

bcolor=colored.bg("white")

reset=colored.attr("reset")

markedcolor=colored.bg("orange\_1")

boardSpaceAmount=size\*\*2

print(f"boardSpaceAmount:{boardSpaceAmount}")

printBoard(board,boardWidth)

#循環進行遊戲

while 1:

#玩家下棋

player()

boardSpaceAmount-=1

clear()

printBoard(board,boardWidth)

positions=DeepCheck(1,True)#檢查是否有人贏了

#訂定AI搜尋深度

CheckDepth=int(-0.05 \* boardSpaceAmount +k)#k==7or6，由設定的難度所決定

if CheckDepth<1:

CheckDepth=1

print(f"CheckDepth : {CheckDepth}")

#AI下棋

print("loading...")

positions=DeepCheck(CheckDepth,False)

clear()

board=ai(positions,board)

boardSpaceAmount-=1

printBoard(board,boardWidth)

positions=DeepCheck(1,True)#檢查是否有人贏了