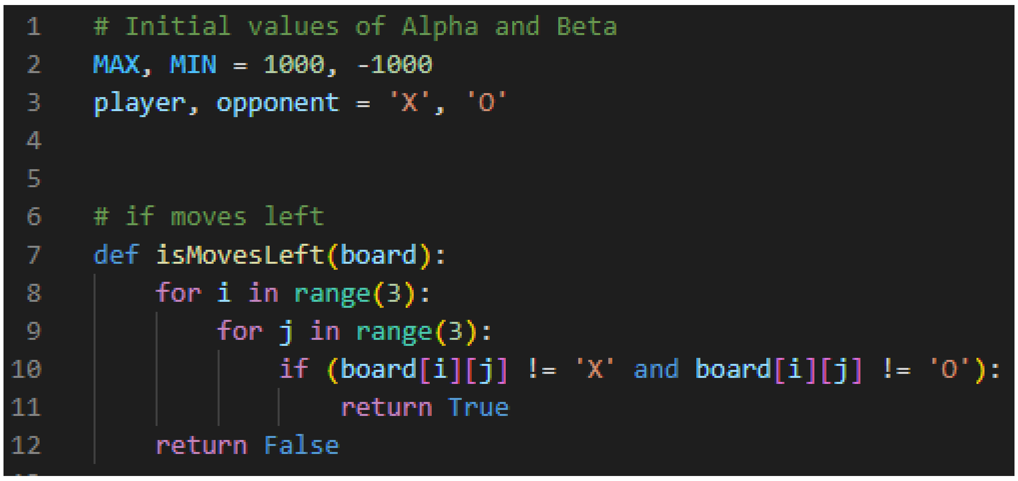
程式碼

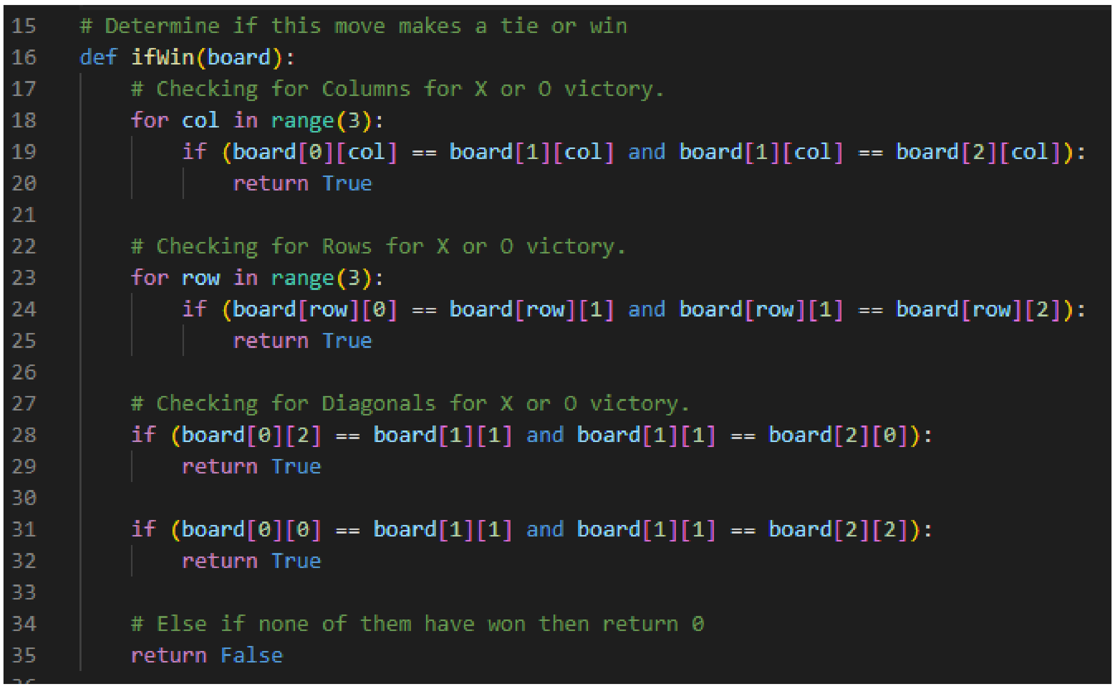
1.

檢查還有沒有空格可以下



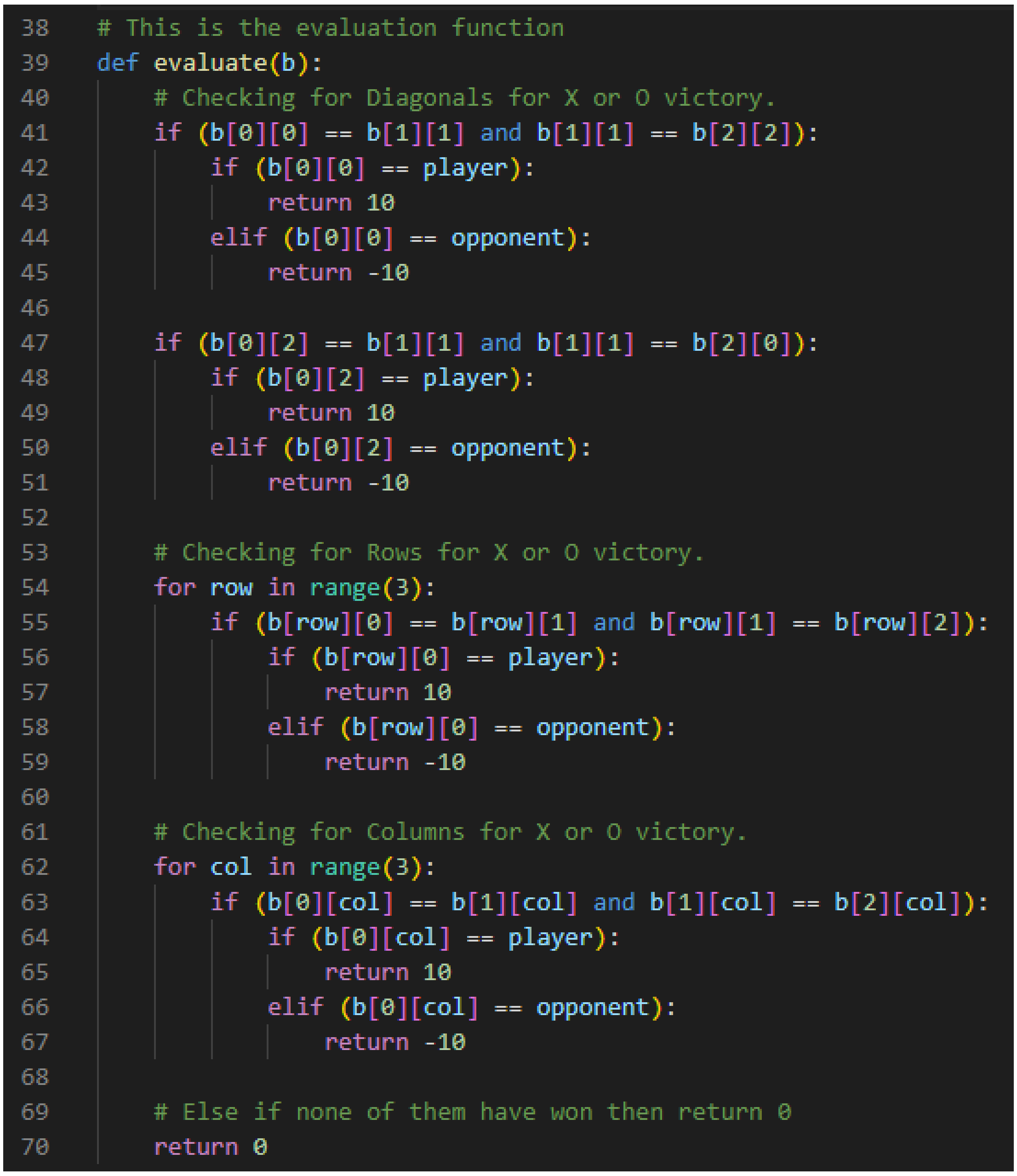
```python
# Initial values of Alpha and Beta
MAX, MIN = 1000, -1000
player, opponent = 'X', 'O'


# if moves left
def isMovesLeft(board):
    for i in range(3):
        for j in range(3):
            if (board[i][j] != 'X' and board[i][j] != 'O'):
                return True
    return False
```

2.檢查(這步棋之後)是不是贏了



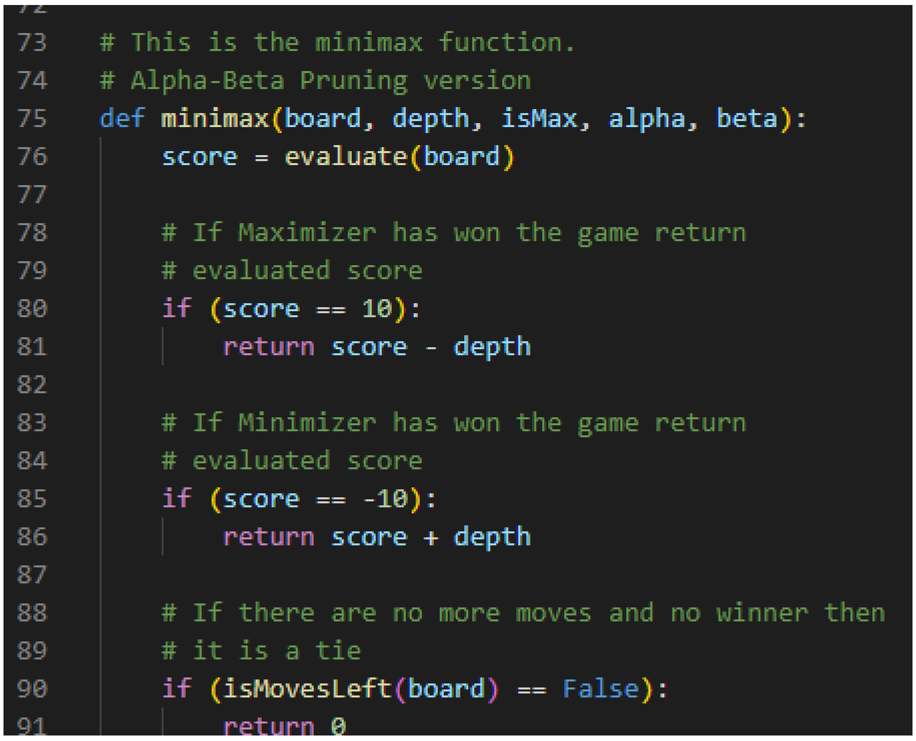
```python
# Determine if this move makes a tie or win
def ifWin(board):
    # Checking for Columns for X or O victory.
    for col in range(3):
        if (board[0][col] == board[1][col] and board[1][col] == board[2][col]):
            return True

    # Checking for Rows for X or O victory.
    for row in range(3):
        if (board[row][0] == board[row][1] and board[row][1] == board[row][2]):
            return True

    # Checking for Diagonals for X or O victory.
    if (board[0][2] == board[1][1] and board[1][1] == board[2][0]):
        return True

    if (board[0][0] == board[1][1] and board[1][1] == board[2][2]):
        return True

    # Else if none of them have won then return 0
    return False
```

3.minimax 的算分法，若是自己贏回傳 10 分，對方贏回傳-10 分



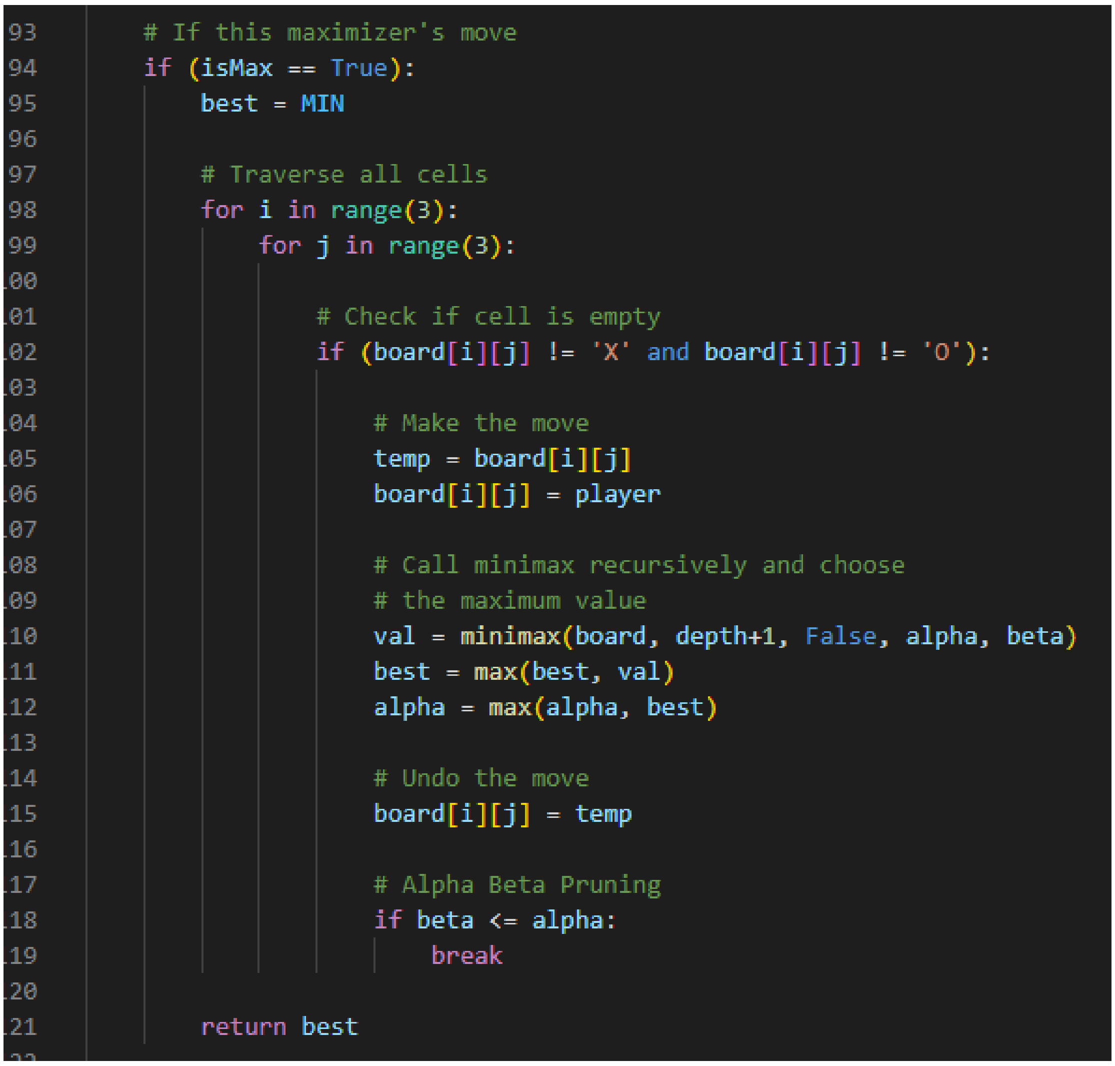
```python
# This is the evaluation function
def evaluate(b):
    # Checking for Diagonals for X or O victory.
    if (b[0][0] == b[1][1] and b[1][1] == b[2][2]):
        if (b[0][0] == player):
            return 10
        elif (b[0][0] == opponent):
            return -10

    if (b[0][2] == b[1][1] and b[1][1] == b[2][0]):
        if (b[0][2] == player):
            return 10
        elif (b[0][2] == opponent):
            return -10

    # Checking for Rows for X or O victory.
    for row in range(3):
        if (b[row][0] == b[row][1] and b[row][1] == b[row][2]):
            if (b[row][0] == player):
                return 10
            elif (b[row][0] == opponent):
                return -10

    # Checking for Columns for X or O victory.
    for col in range(3):
        if (b[0][col] == b[1][col] and b[1][col] == b[2][col]):
            if (b[0][col] == player):
                return 10
            elif (b[0][col] == opponent):
                return -10

    # Else if none of them have won then return 0
    return 0
```

4.minimax function with α-β pruning，第一張圖的回傳設計能讓 ai player 選擇步驟數少的那部棋，



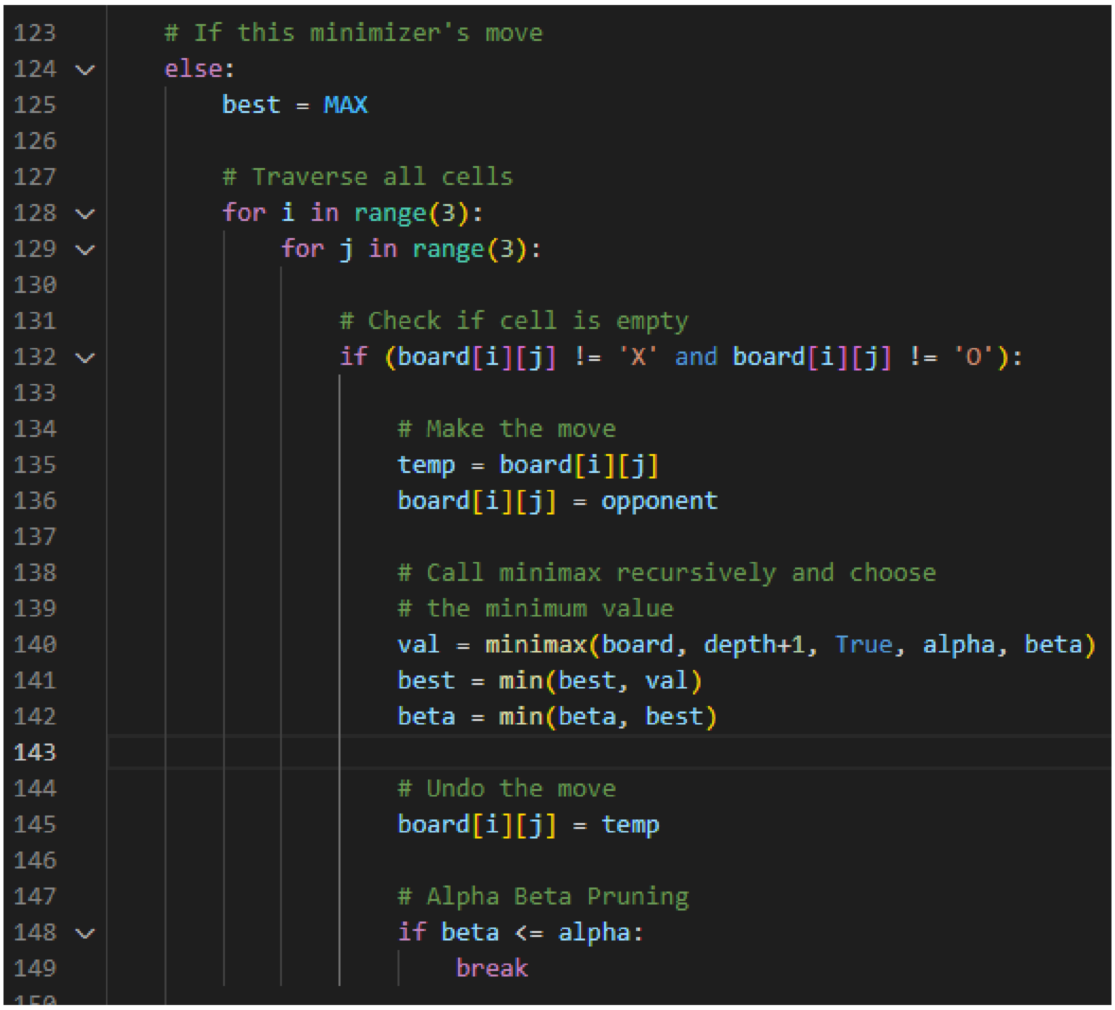
```python
# This is the minimax function.
# Alpha-Beta Pruning version
def minimax(board, depth, isMax, alpha, beta):
    score = evaluate(board)

    # If Maximizer has won the game return
    # evaluated score
    if (score == 10):
        return score - depth

    # If Minimizer has won the game return
    # evaluated score
    if (score == -10):
        return score + depth

    # If there are no more moves and no winner then
    # it is a tie
    if (isMovesLeft(board) == False):
        return 0
```

Maximizer 時，若是 beta <= alpha 就退出



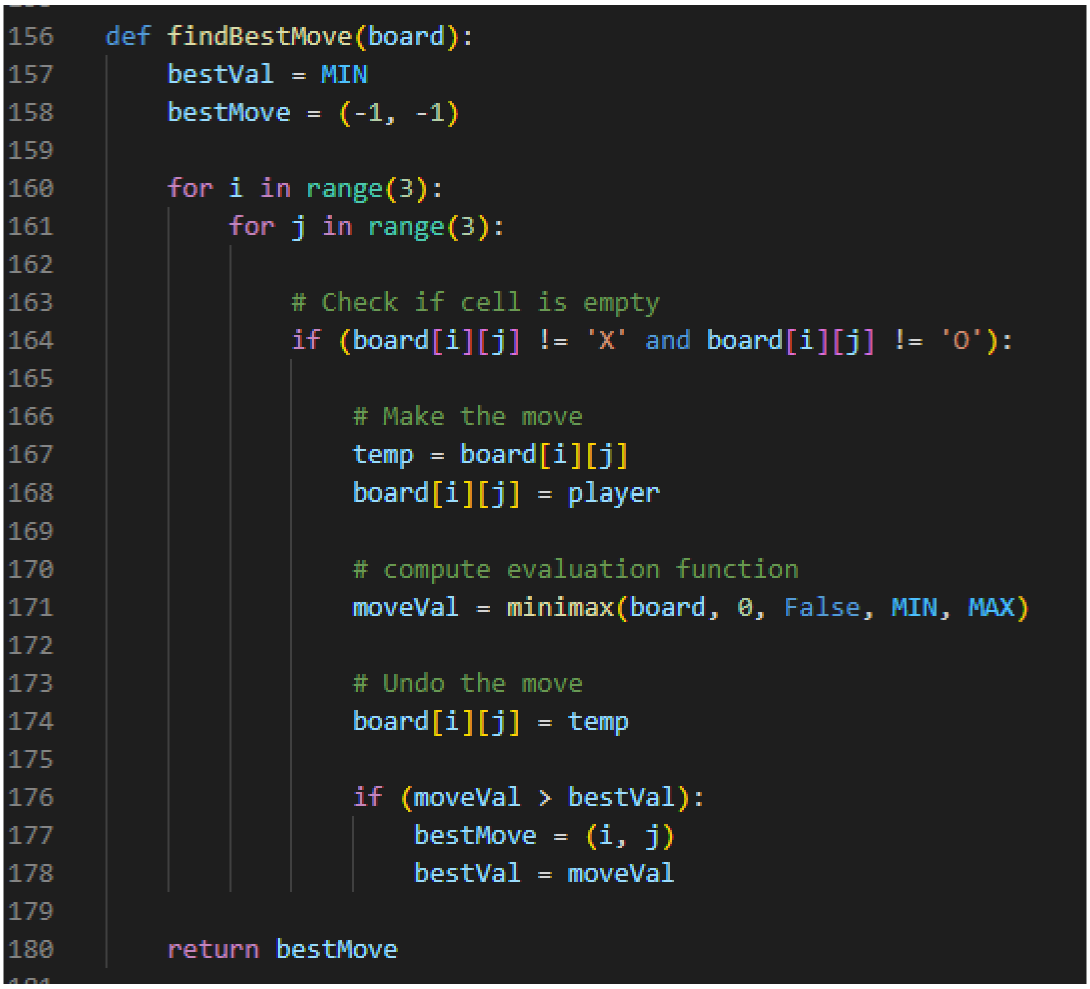
```python
    # If this maximizer's move
    if (isMax == True):
        best = MIN

        # Traverse all cells
        for i in range(3):
            for j in range(3):

                # Check if cell is empty
                if (board[i][j] != 'X' and board[i][j] != 'O'):

                    # Make the move
                    temp = board[i][j]
                    board[i][j] = player

                    # Call minimax recursively and choose
                    # the maximum value
                    val = minimax(board, depth+1, False, alpha, beta)
                    best = max(best, val)
                    alpha = max(alpha, best)

                    # Undo the move
                    board[i][j] = temp

                    # Alpha Beta Pruning
                    if beta <= alpha:
                        break

        return best
```

Minimizer 時，若是 beta <= alpha 就退出



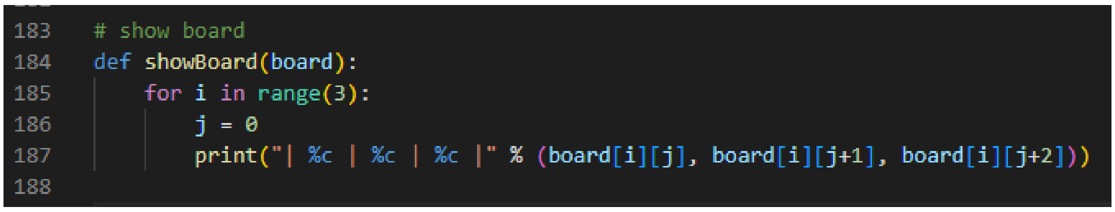
```
    # If this minimizer's move
    else:
        best = MAX

        # Traverse all cells
        for i in range(3):
            for j in range(3):

                # Check if cell is empty
                if (board[i][j] != 'X' and board[i][j] != 'O'):

                    # Make the move
                    temp = board[i][j]
                    board[i][j] = opponent

                    # Call minimax recursively and choose
                    # the minimum value
                    val = minimax(board, depth+1, True, alpha, beta)
                    best = min(best, val)
                    beta = min(beta, best)

                    # Undo the move
                    board[i][j] = temp

                    # Alpha Beta Pruning
                    if beta <= alpha:
                        break
```

5.找出 ai player 最適合的一步



```python
def findBestMove(board):
    bestVal = MIN
    bestMove = (-1, -1)

    for i in range(3):
        for j in range(3):

            # Check if cell is empty
            if (board[i][j] != 'X' and board[i][j] != 'O'):

                # Make the move
                temp = board[i][j]
                board[i][j] = player

                # compute evaluation function
                moveVal = minimax(board, 0, False, MIN, MAX)

                # Undo the move
                board[i][j] = temp

                if (moveVal > bestVal):
                    bestMove = (i, j)
                    bestVal = moveVal

    return bestMove
```

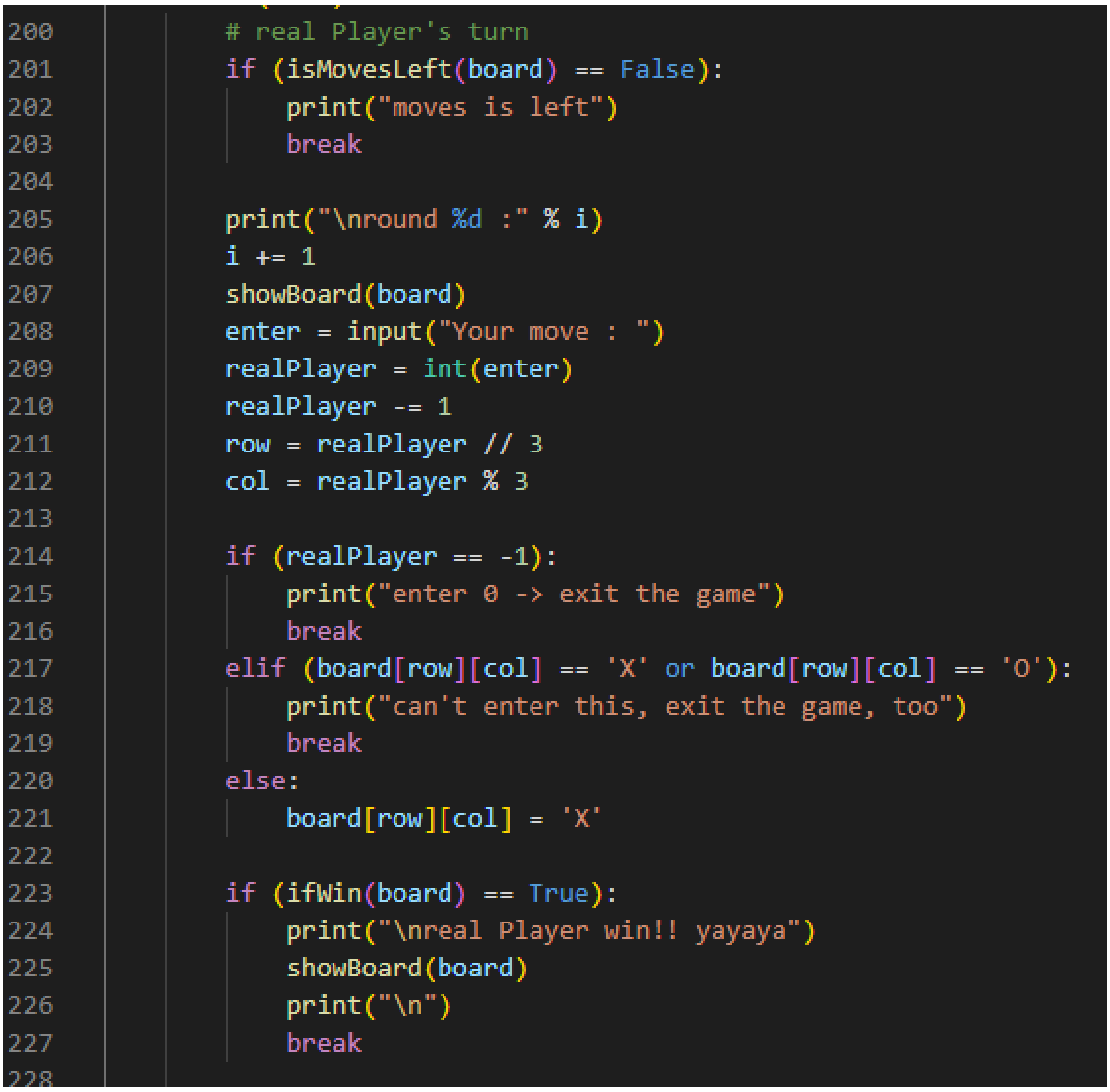
6.印棋盤



```python
# show board
def showBoard(board):
    for i in range(3):
        j = 0
        print("| %c | %c | %c |" % (board[i][j], board[i][j+1], board[i][j+2]))
```

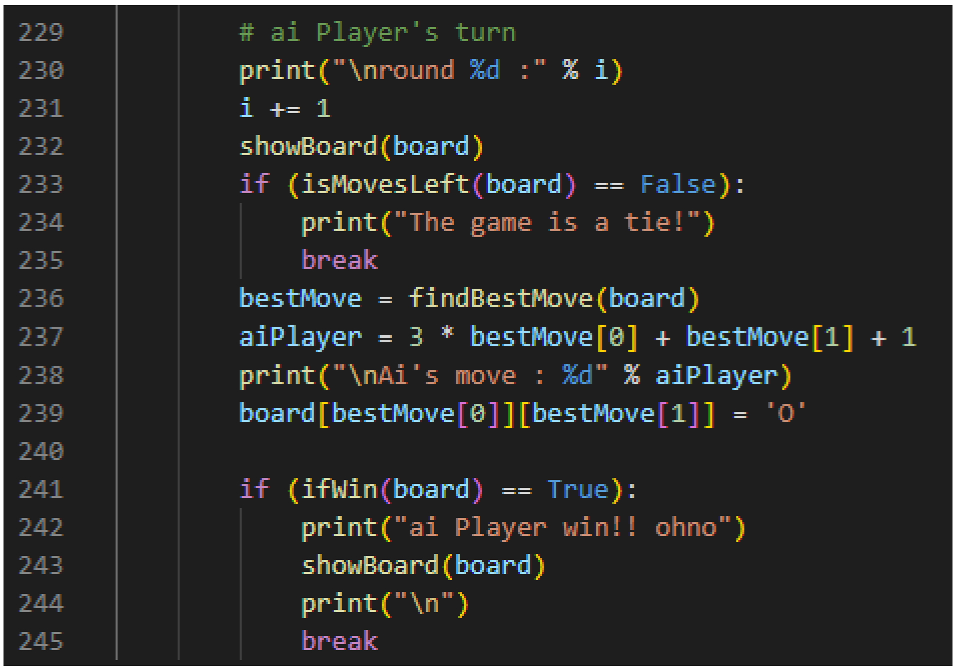
7.main

Real player，下 0 的話結束遊戲，按到出過的地方的話也會跳出，贏的話會慶祝



```python
        # real Player's turn
        if (isMovesLeft(board) == False):
            print("moves is left")
            break

        print("\nround %d :" % i)
        i += 1
        showBoard(board)
        enter = input("Your move : ")
        realPlayer = int(enter)
        realPlayer -= 1
        row = realPlayer // 3
        col = realPlayer % 3

        if (realPlayer == -1):
            print("enter 0 -> exit the game")
            break
        elif (board[row][col] == 'X' or board[row][col] == 'O'):
            print("can't enter this, exit the game, too")
            break
        else:
            board[row][col] = 'X'

        if (ifWin(board) == True):
            print("\nreal Player win!! yayaya")
            showBoard(board)
            print("\n")
            break
```

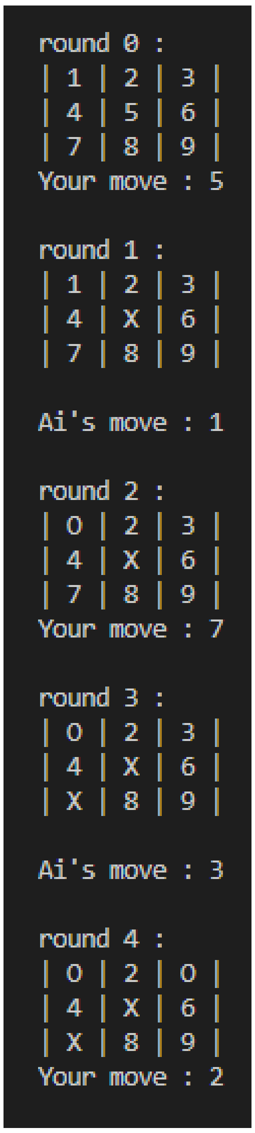
Ai player，贏的話會幫玩家惋惜



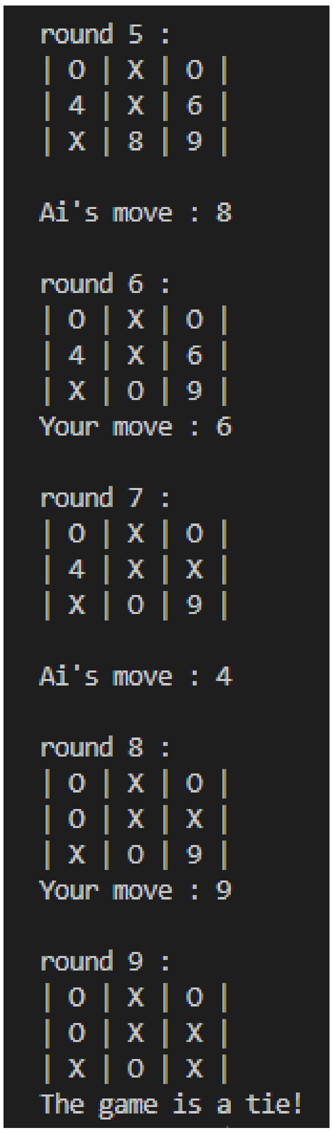
```python
        # ai Player's turn
        print("\nround %d :" % i)
        i += 1
        showBoard(board)
        if (isMovesLeft(board) == False):
            print("The game is a tie!")
            break
        bestMove = findBestMove(board)
        aiPlayer = 3 * bestMove[0] + bestMove[1] + 1
        print("\nAi's move : %d" % aiPlayer)
        board[bestMove[0]][bestMove[1]] = 'O'

        if (ifWin(board) == True):
            print("ai Player win!! ohno")
            showBoard(board)
            print("\n")
            break
```

成果

1. 平局

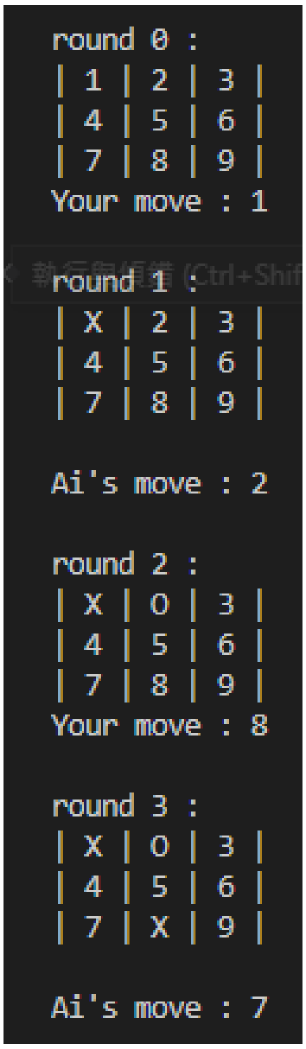


```
round 0 :
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |
Your move : 5

round 1 :
| 1 | 2 | 3 |
| 4 | X | 6 |
| 7 | 8 | 9 |

Ai's move : 1

round 2 :
| O | 2 | 3 |
| 4 | X | 6 |
| 7 | 8 | 9 |
Your move : 7

round 3 :
| O | 2 | 3 |
| 4 | X | 6 |
| X | 8 | 9 |

Ai's move : 3

round 4 :
| O | 2 | O |
| 4 | X | 6 |
| X | 8 | 9 |
Your move : 2
```

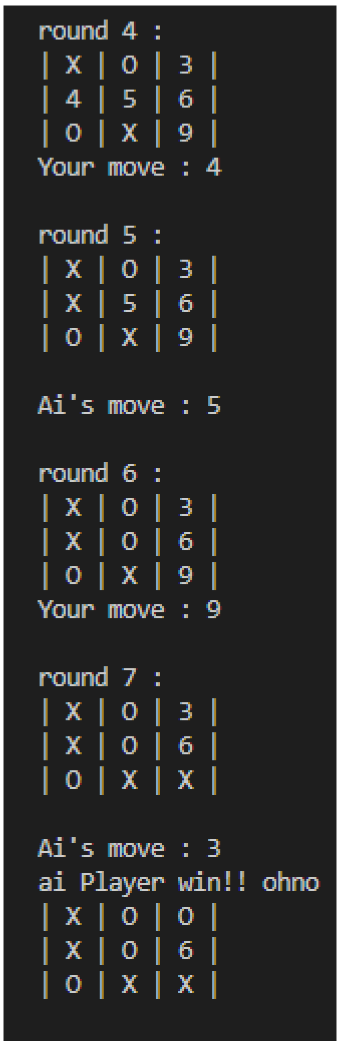


```
round 5 :
| O | X | O |
| 4 | X | 6 |
| X | 8 | 9 |

Ai's move : 8

round 6 :
| O | X | O |
| 4 | X | 6 |
| X | O | 9 |
Your move : 6

round 7 :
| O | X | O |
| 4 | X | X |
| X | O | 9 |

Ai's move : 4

round 8 :
| O | X | O |
| O | X | X |
| X | O | 9 |
Your move : 9

round 9 :
| O | X | O |
| O | X | X |
| X | O | X |
The game is a tie!
```

2. Ai player 獲勝

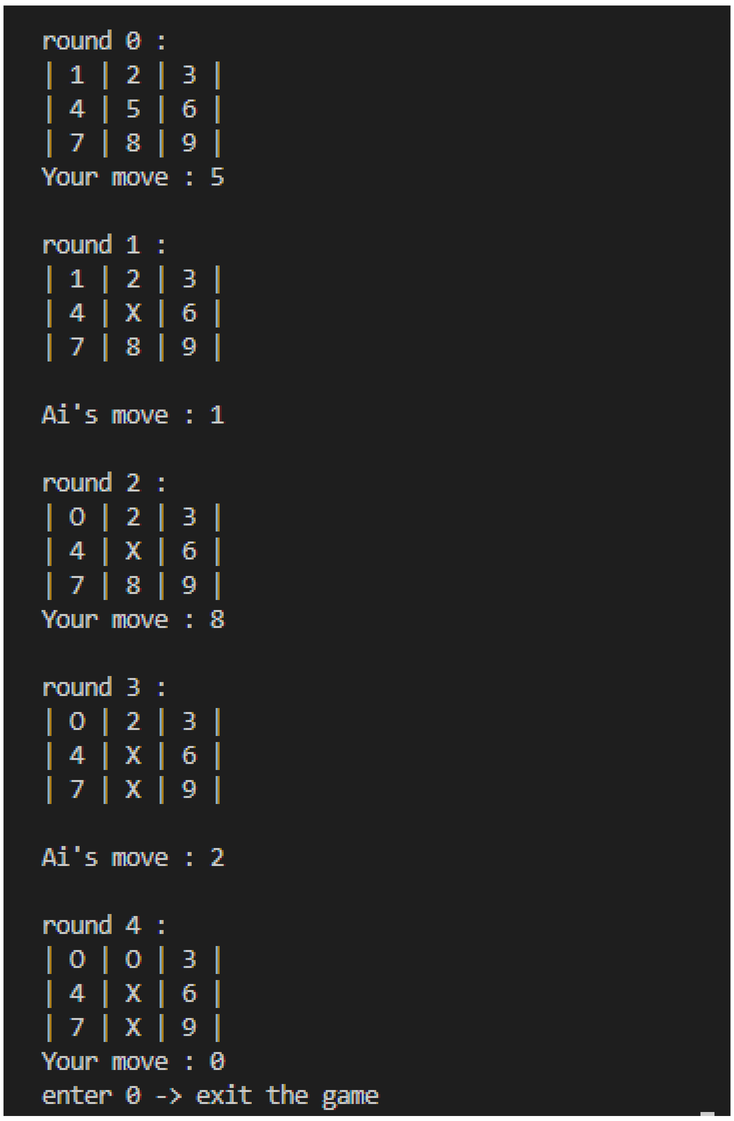


```
round 0 :
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |
Your move : 1

round 1 :
| X | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

Ai's move : 2

round 2 :
| X | O | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |
Your move : 8

round 3 :
| X | O | 3 |
| 4 | 5 | 6 |
| 7 | X | 9 |

Ai's move : 7
```



```
round 4 :
| X | O | 3 |
| 4 | 5 | 6 |
| O | X | 9 |
Your move : 4

round 5 :
| X | O | 3 |
| X | 5 | 6 |
| O | X | 9 |

Ai's move : 5

round 6 :
| X | O | 3 |
| X | O | 6 |
| O | X | 9 |
Your move : 9

round 7 :
| X | O | 3 |
| X | O | 6 |
| O | X | X |

Ai's move : 3
ai Player win!! ohno
| X | O | O |
| X | O | 6 |
| O | X | X |
```

3. 按 0



```
round 0 :
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |
Your move : 5

round 1 :
| 1 | 2 | 3 |
| 4 | X | 6 |
| 7 | 8 | 9 |

Ai's move : 1

round 2 :
| O | 2 | 3 |
| 4 | X | 6 |
| 7 | 8 | 9 |
Your move : 8

round 3 :
| O | 2 | 3 |
| 4 | X | 6 |
| 7 | X | 9 |

Ai's move : 2

round 4 :
| O | O | 3 |
| 4 | X | 6 |
| 7 | X | 9 |
Your move : 0
enter 0 -> exit the game
```