

程式碼

1.

檢查還有沒有空格可以下

```
1  # Initial values of Alpha and Beta
2  MAX, MIN = 1000, -1000
3  player, opponent = 'X', 'O'
4
5
6  # if moves left
7  def isMovesLeft(board):
8      for i in range(3):
9          for j in range(3):
10             if (board[i][j] != 'X' and board[i][j] != 'O'):
11                 return True
12     return False
```

2.檢查(這步棋之後)是不是贏了

```
15 # Determine if this move makes a tie or win
16 def ifWin(board):
17     # Checking for Columns for X or O victory.
18     for col in range(3):
19         if (board[0][col] == board[1][col] and board[1][col] == board[2][col]):
20             return True
21
22     # Checking for Rows for X or O victory.
23     for row in range(3):
24         if (board[row][0] == board[row][1] and board[row][1] == board[row][2]):
25             return True
26
27     # Checking for Diagonals for X or O victory.
28     if (board[0][2] == board[1][1] and board[1][1] == board[2][0]):
29         return True
30
31     if (board[0][0] == board[1][1] and board[1][1] == board[2][2]):
32         return True
33
34     # Else if none of them have won then return 0
35     return False
```

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

```

38 # This is the evaluation function
39 def evaluate(b):
40     # Checking for Diagonals for X or O victory.
41     if (b[0][0] == b[1][1] and b[1][1] == b[2][2]):
42         if (b[0][0] == player):
43             return 10
44         elif (b[0][0] == opponent):
45             return -10
46
47     if (b[0][2] == b[1][1] and b[1][1] == b[2][0]):
48         if (b[0][2] == player):
49             return 10
50         elif (b[0][2] == opponent):
51             return -10
52
53     # Checking for Rows for X or O victory.
54     for row in range(3):
55         if (b[row][0] == b[row][1] and b[row][1] == b[row][2]):
56             if (b[row][0] == player):
57                 return 10
58             elif (b[row][0] == opponent):
59                 return -10
60
61     # Checking for Columns for X or O victory.
62     for col in range(3):
63         if (b[0][col] == b[1][col] and b[1][col] == b[2][col]):
64             if (b[0][col] == player):
65                 return 10
66             elif (b[0][col] == opponent):
67                 return -10
68
69     # Else if none of them have won then return 0
70     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，

```

72
73 # This is the minimax function.
74 # Alpha-Beta Pruning version
75 def minimax(board, depth, isMax, alpha, beta):
76     score = evaluate(board)
77
78     # If Maximizer has won the game return
79     # evaluated score
80     if (score == 10):
81         return score - depth
82
83     # If Minimizer has won the game return
84     # evaluated score
85     if (score == -10):
86         return score + depth
87
88     # If there are no more moves and no winner then
89     # it is a tie
90     if (isMovesLeft(board) == False):
91         return 0

```

Maximizer 時，若是 $\beta \leq \alpha$ 就退出

```

93     # If this maximizer's move
94     if (isMax == True):
95         best = MIN
96
97         # Traverse all cells
98         for i in range(3):
99             for j in range(3):
100
101                 # Check if cell is empty
102                 if (board[i][j] != 'X' and board[i][j] != 'O'):
103
104                     # Make the move
105                     temp = board[i][j]
106                     board[i][j] = player
107
108                     # Call minimax recursively and choose
109                     # the maximum value
110                     val = minimax(board, depth+1, False, alpha, beta)
111                     best = max(best, val)
112                     alpha = max(alpha, best)
113
114                     # Undo the move
115                     board[i][j] = temp
116
117                     # Alpha Beta Pruning
118                     if beta <= alpha:
119                         break
120
121     return best

```

Minimizer 時，若是 $\beta \leq \alpha$ 就退出

```

123     # If this minimizer's move
124     else:
125         best = MAX
126
127         # Traverse all cells
128         for i in range(3):
129             for j in range(3):
130
131                 # Check if cell is empty
132                 if (board[i][j] != 'X' and board[i][j] != 'O'):
133
134                     # Make the move
135                     temp = board[i][j]
136                     board[i][j] = opponent
137
138                     # Call minimax recursively and choose
139                     # the minimum value
140                     val = minimax(board, depth+1, True, alpha, beta)
141                     best = min(best, val)
142                     beta = min(beta, best)
143
144             # Undo the move
145             board[i][j] = temp
146
147             # Alpha Beta Pruning
148             if beta <= alpha:
149                 break
150

```

5. 找出 ai player 最適合的一步

```

156 def findBestMove(board):
157     bestVal = MIN
158     bestMove = (-1, -1)
159
160     for i in range(3):
161         for j in range(3):
162
163             # Check if cell is empty
164             if (board[i][j] != 'X' and board[i][j] != 'O'):
165
166                 # Make the move
167                 temp = board[i][j]
168                 board[i][j] = player
169
170                 # compute evaluation function
171                 moveVal = minimax(board, 0, False, MIN, MAX)
172
173                 # Undo the move
174                 board[i][j] = temp
175
176                 if (moveVal > bestVal):
177                     bestMove = (i, j)
178                     bestVal = moveVal
179
180     return bestMove

```

6.印棋盤

```

183 # show board
184 def showBoard(board):
185     for i in range(3):
186         j = 0
187         print("| %c | %c | %c |" % (board[i][j], board[i][j+1], board[i][j+2]))
188

```

7.main

Real player，下 O 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

```

200         # real Player's turn
201         if (isMovesLeft(board) == False):
202             print("moves is left")
203             break
204
205         print("\nround %d :" % i)
206         i += 1
207         showBoard(board)
208         enter = input("Your move : ")
209         realPlayer = int(enter)
210         realPlayer -= 1
211         row = realPlayer // 3
212         col = realPlayer % 3
213
214         if (realPlayer == -1):
215             print("enter 0 -> exit the game")
216             break
217         elif (board[row][col] == 'X' or board[row][col] == 'O'):
218             print("can't enter this, exit the game, too")
219             break
220         else:
221             board[row][col] = 'X'
222
223         if (ifWin(board) == True):
224             print("\nreal Player win!! yayaya")
225             showBoard(board)
226             print("\n")
227             break
228

```

Ai player，贏的話會幫玩家惋惜

```

229         # ai Player's turn
230         print("\nround %d :" % i)
231         i += 1
232         showBoard(board)
233         if (isMovesLeft(board) == False):
234             print("The game is a tie!")
235             break
236         bestMove = findBestMove(board)
237         aiPlayer = 3 * bestMove[0] + bestMove[1] + 1
238         print("\nAi's move : %d" % aiPlayer)
239         board[bestMove[0]][bestMove[1]] = 'O'
240
241         if (ifWin(board) == True):
242             print("ai Player win!! ohno")
243             showBoard(board)
244             print("\n")
245             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 :  
| 0 | 2 | 3 |  
| 4 | X | 6 |  
| 7 | 8 | 9 |  
Your move : 7  
  
round 3 :  
| 0 | 2 | 3 |  
| 4 | X | 6 |  
| X | 8 | 9 |  
Ai's move : 3  
  
round 4 :  
| 0 | 2 | 0 |  
| 4 | X | 6 |  
| X | 8 | 9 |  
Your move : 2  
  
round 5 :  
| 0 | X | 0 |  
| 4 | X | 6 |  
| X | 8 | 9 |  
Ai's move : 8  
  
round 6 :  
| 0 | X | 0 |  
| 4 | X | 6 |  
| X | 0 | 9 |  
Your move : 6  
  
round 7 :  
| 0 | X | 0 |  
| 4 | X | X |  
| X | 0 | 9 |  
Ai's move : 4  
  
round 8 :  
| 0 | X | 0 |  
| 0 | X | X |  
| X | 0 | 9 |  
Your move : 9  
  
round 9 :  
| 0 | X | 0 |  
| 0 | X | X |  
| X | 0 | 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 | 0 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |
Your move : 8

round 3 :
| X | 0 | 3 |
| 4 | 5 | 6 |
| 7 | X | 9 |

Ai's move : 7

round 4 :
| X | 0 | 3 |
| 4 | 5 | 6 |
| 0 | X | 9 |
Your move : 4

round 5 :
| X | 0 | 3 |
| X | 5 | 6 |
| 0 | X | 9 |

Ai's move : 5

round 6 :
| X | 0 | 3 |
| X | 0 | 6 |
| 0 | X | 9 |
Your move : 9

round 7 :
| X | 0 | 3 |
| X | 0 | 6 |
| 0 | X | X |

Ai's move : 3
ai Player win!! ohno
| X | 0 | 0 |
| X | 0 | 6 |
| 0 | 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 :

	0		2		3	
	4		X		6	
	7		8		9	

Your move : 8

round 3 :

	0		2		3	
	4		X		6	
	7		X		9	

AI's move : 2

round 4 :

	0		0		3	
	4		X		6	
	7		X		9	

Your move : 0

enter 0 -> exit the game