

| 자유 게시판 | >

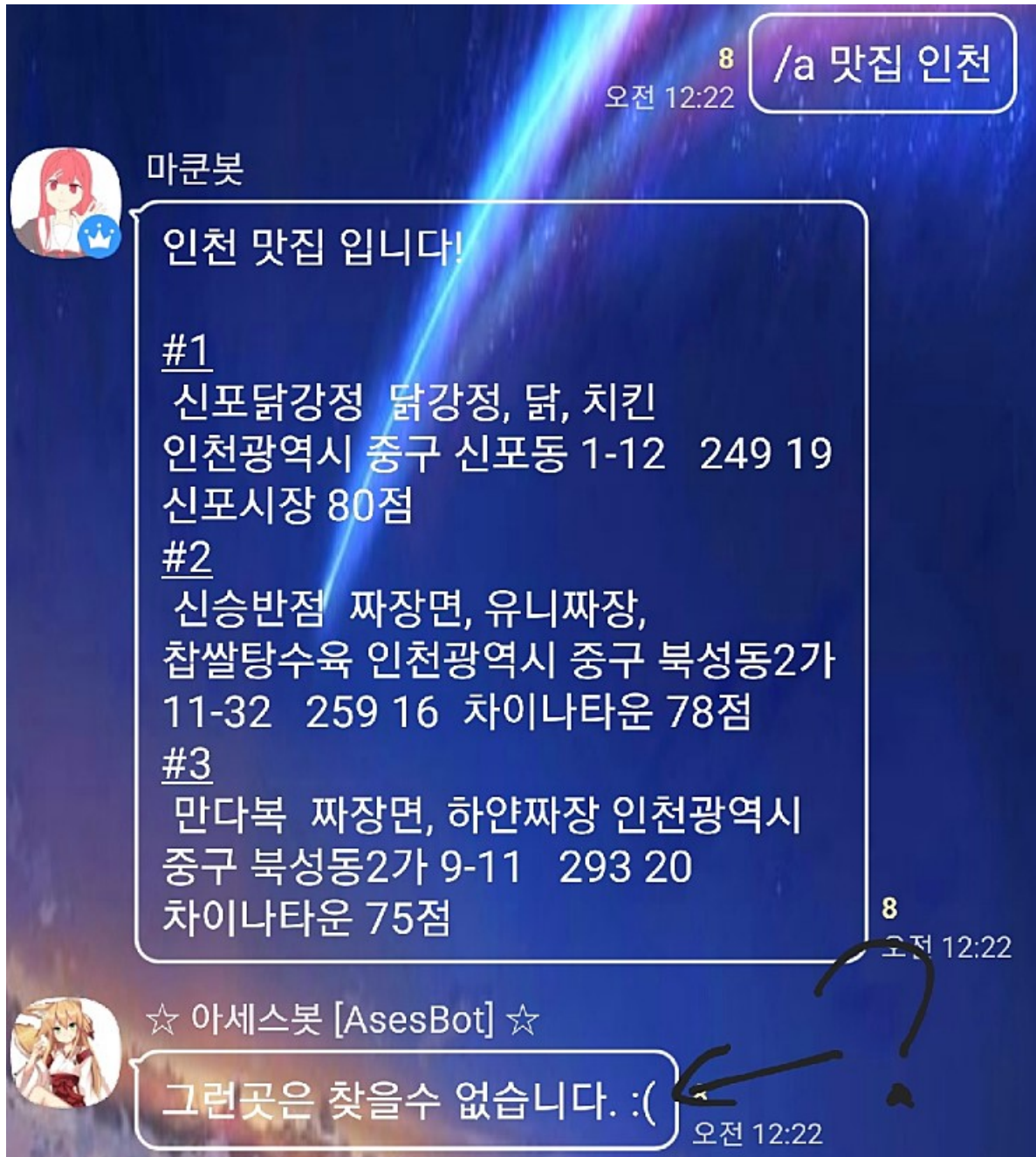
## 이해 할수 없는 봇의 세계



클칠 챗봇 마스터 1:1 채팅

2019.02.18. 00:26 조회 103

댓글 4 URL 복사



분명히 같은 소스인데 내것만 오류남;;



클칠님의 게시글 더보기 >

❤ 좋아요 1 💬 댓글 4

공유 | 신고

댓글 등록순 최신순 C

댓글알림 ☐



7L

⋮



ㄷㄷ

2019.02.18. 00:32 답글 쓰기



Changa A

ㄴ0o0ㄱ

2019.02.18. 02:15 답글 쓰기



P1911

응와 저도 그런적 있었쥬

2019.02.18. 11:38 답글 쓰기



ImHm

ㅋㅋㅋㅋ

2019.02.18. 23:38 답글 쓰기

Hibot

댓글을 남겨보세요



등록

글쓰기

답글

목록

▲ TOP

## '자유 게시판' 게시판 글

이 게시판 새글 구독하기 ☐

알아두면 유용한 단축키 **[1]**

Starpia

2019.02.18.

카페등급 올리기 ( 2 ) ( 가지 3 >> 4 ) **[5]**

Starpia

2019.02.18.

이해 할수 없는 붓의 세계 🐼 **[4]**

쿨칠

2019.02.18.

오우 벌써 가지 3단계네요 🐼 **[3]**

—M—

2019.02.17.

새카봇 2 공개합니다 🐼 **[6]**

SBYT

2019.02.17.

1 2 3

전체보기

## 이 카페 인기글

JPG 개발 일지 #7

재승

♡0 💬4

[붓게임] 세븐 다이스 v1.0.0

ZUMP

♡1 💬8

복불

하하하하

♡0 💬5

```
1 def is_valid(board):
2     for row in board:
3         if row.count('O') > 3:
4             return False
5     for col in range(3):
6         if board[0][col].count('O') > 3:
7             return False
8     return True
9
10 def get_next_move(board):
11     for row in range(3):
12         for col in range(3):
13             if board[row][col] == ' ':
14                 return (row, col)
15
16 def minimax(board, depth, is_maximizing):
17     if depth == 0:
18         return evaluate(board)
19     if is_maximizing:
20         best_score = -float('inf')
21         for row, col in get_next_move(board):
22             board[row][col] = 'O'
23             score = minimax(board, depth - 1, False)
24             board[row][col] = ' '
25             best_score = max(best_score, score)
26     else:
27         best_score = float('inf')
28         for row, col in get_next_move(board):
29             board[row][col] = 'X'
30             score = minimax(board, depth - 1, True)
31             board[row][col] = ' '
32             best_score = min(best_score, score)
33     return best_score
34
35 def evaluate(board):
36     if board[0][0] == 'O' and board[0][1] == 'O' and board[0][2] == 'O':
37         return 1
38     if board[1][0] == 'O' and board[1][1] == 'O' and board[1][2] == 'O':
39         return 1
40     if board[2][0] == 'O' and board[2][1] == 'O' and board[2][2] == 'O':
41         return 1
42     if board[0][0] == 'X' and board[0][1] == 'X' and board[0][2] == 'X':
43         return -1
44     if board[1][0] == 'X' and board[1][1] == 'X' and board[1][2] == 'X':
45         return -1
46     if board[2][0] == 'X' and board[2][1] == 'X' and board[2][2] == 'X':
47         return -1
48     return 0
```

틱택토 (Player vs Bot)

```
1 def is_valid(board):
2     for row in board:
3         if row.count('O') > 3:
4             return False
5     for col in range(3):
6         if board[0][col].count('O') > 3:
7             return False
8     return True
9
10 def get_next_move(board):
11     for row in range(3):
12         for col in range(3):
13             if board[row][col] == ' ':
14                 return (row, col)
15
16 def minimax(board, depth, is_maximizing):
17     if depth == 0:
18         return evaluate(board)
19     if is_maximizing:
20         best_score = -float('inf')
21         for row, col in get_next_move(board):
22             board[row][col] = 'O'
23             score = minimax(board, depth - 1, False)
24             board[row][col] = ' '
25             best_score = max(best_score, score)
26     else:
27         best_score = float('inf')
28         for row, col in get_next_move(board):
29             board[row][col] = 'X'
30             score = minimax(board, depth - 1, True)
31             board[row][col] = ' '
32             best_score = min(best_score, score)
33     return best_score
34
35 def evaluate(board):
36     if board[0][0] == 'O' and board[0][1] == 'O' and board[0][2] == 'O':
37         return 1
38     if board[1][0] == 'O' and board[1][1] == 'O' and board[1][2] == 'O':
39         return 1
40     if board[2][0] == 'O' and board[2][1] == 'O' and board[2][2] == 'O':
41         return 1
42     if board[0][0] == 'X' and board[0][1] == 'X' and board[0][2] == 'X':
43         return -1
44     if board[1][0] == 'X' and board[1][1] == 'X' and board[1][2] == 'X':
45         return -1
46     if board[2][0] == 'X' and board[2][1] == 'X' and board[2][2] == 'X':
47         return -1
48     return 0
```

네이버에서 지역별 날씨 정보 크롤링하기 /

```
1 def is_valid(board):
2     for row in board:
3         if row.count('O') > 3:
4             return False
5     for col in range(3):
6         if board[0][col].count('O') > 3:
7             return False
8     return True
9
10 def get_next_move(board):
11     for row in range(3):
12         for col in range(3):
13             if board[row][col] == ' ':
14                 return (row, col)
15
16 def minimax(board, depth, is_maximizing):
17     if depth == 0:
18         return evaluate(board)
19     if is_maximizing:
20         best_score = -float('inf')
21         for row, col in get_next_move(board):
22             board[row][col] = 'O'
23             score = minimax(board, depth - 1, False)
24             board[row][col] = ' '
25             best_score = max(best_score, score)
26     else:
27         best_score = float('inf')
28         for row, col in get_next_move(board):
29             board[row][col] = 'X'
30             score = minimax(board, depth - 1, True)
31             board[row][col] = ' '
32             best_score = min(best_score, score)
33     return best_score
34
35 def evaluate(board):
36     if board[0][0] == 'O' and board[0][1] == 'O' and board[0][2] == 'O':
37         return 1
38     if board[1][0] == 'O' and board[1][1] == 'O' and board[1][2] == 'O':
39         return 1
40     if board[2][0] == 'O' and board[2][1] == 'O' and board[2][2] == 'O':
41         return 1
42     if board[0][0] == 'X' and board[0][1] == 'X' and board[0][2] == 'X':
43         return -1
44     if board[1][0] == 'X' and board[1][1] == 'X' and board[1][2] == 'X':
45         return -1
46     if board[2][0] == 'X' and board[2][1] == 'X' and board[2][2] == 'X':
47         return -1
48     return 0
```

오류뜨는데 해결법 알려주세요ㅠㅠ

에러를 알람으로 남게 해보자!

Lunar  
♡2 ...4

```
1 def is_valid(board):
2     for row in board:
3         if row.count('O') > 3:
4             return False
5     for col in range(3):
6         if board[0][col].count('O') > 3:
7             return False
8     return True
9
10 def get_next_move(board):
11     for row in range(3):
12         for col in range(3):
13             if board[row][col] == ' ':
14                 return (row, col)
15
16 def minimax(board, depth, is_maximizing):
17     if depth == 0:
18         return evaluate(board)
19     if is_maximizing:
20         best_score = -float('inf')
21         for row, col in get_next_move(board):
22             board[row][col] = 'O'
23             score = minimax(board, depth - 1, False)
24             board[row][col] = ' '
25             best_score = max(best_score, score)
26     else:
27         best_score = float('inf')
28         for row, col in get_next_move(board):
29             board[row][col] = 'X'
30             score = minimax(board, depth - 1, True)
31             board[row][col] = ' '
32             best_score = min(best_score, score)
33     return best_score
34
35 def evaluate(board):
36     if board[0][0] == 'O' and board[0][1] == 'O' and board[0][2] == 'O':
37         return 1
38     if board[1][0] == 'O' and board[1][1] == 'O' and board[1][2] == 'O':
39         return 1
40     if board[2][0] == 'O' and board[2][1] == 'O' and board[2][2] == 'O':
41         return 1
42     if board[0][0] == 'X' and board[0][1] == 'X' and board[0][2] == 'X':
43         return -1
44     if board[1][0] == 'X' and board[1][1] == 'X' and board[1][2] == 'X':
45         return -1
46     if board[2][0] == 'X' and board[2][1] == 'X' and board[2][2] == 'X':
47         return -1
48     return 0
```

틱택토 (Player vs Player)

자동학습 봇 특정 말만 무시하기?

헤히  
♡0 ...4