در این کد ما یک تابع play_game() پیاده کردیم که در آن هر یک از بازیکن ها (به ترتیب انسان و کامپیوتر) بازی میکنند.

اگر بخواهیم مطابق آنچه در صورت پروژه گفته شده، هر بازیکن در هر نوبت بتواند دو مهره را در صفحه بگذارد کافیست این تابع را به صورت زیر تغییر دهیم:

```
def play_game(BOARD):
    turn_count = 0
    while True:
        # Player move
        get_move(BOARD)
        get_move(BOARD)
        turn_count += 1
        if is_winning(BOARD, 1):
            print("Player wins!")
            break
        if is_full(BOARD):
            print("It's a draw!")
            break

        # Computer move
        if turn_count <= 3:
            move = random_move(BOARD)
        else:
            move = best_move(BOARD)

        if move:
            BOARD[move[0]][move[1]] = 2
            if turn_count<=3:
                move=random_move(BOARD)
            else:
                move=best_move(BOARD)
            if move:
                print("Computer moved:")
                print(BOARD)
                sub_index = np.random.randint(1, 5)
                direction = np.random.choice([-1, 1, 0])
                BOARD = rotate_subsquare(BOARD, sub_index, direction)
                print("Computer rotated:")
                print(BOARD)
                if is_winning(BOARD, 2):
                    print("Computer wins!")
                    break
                if is_full(BOARD):
                    print("It's a draw!")
                    break
```

در هر نوبت بازیکن میتواند ورودی خود را به صورت (I,j) که 0<i,j<5 وارد میکند. سپس برنامه میپرسد میخواهی کدام یک از چهار قسمت را بچرخانی و در این صورت بین گزینه های 0,-1,1 برای چرخش 90 درجه در جهت های مختلف را وارد میکند.

سپس چک میشود که آیا بازی تمام شده یا خیر. و همینطور اگر برد پر شده باشد، بازی با نتیجه مساوی به اتمام میرسد.

اما زمانی که نوبت کامپیوتر است:

به علت زیاد بودن پیچیدگی زمانی این مرحله، وقتی turn_count<=3 برقرار باشد، حرکت ماشین برای گذاشتن مهره به صورت تصادفی رخ میدهد. سپس به صورت رندوم انتخاب میشود که کدام زیر مربع و به چه جهتی چرخش داشته باشد. اما حرکت best_move برای کامپیوتر به چه صورت انتخاب میشود؟ در اینجا از الگوریتم minimax استفاده میشود و به دلیل پیچیدگی زمانی زیاد، ما depth را 3 در نظر گرفتیم.

نتیجه یک دور اجرای برنامه: ورودی های کاربر در این خروجی به رنگ آبی نمایش داشده شده:

[0 0 0 0 0 0]]

[0 0 0 0 0 0]

[0 0 0 0 0 0]

[0 0 0 0 0 0]

[0 0 0 0 0 0]

[[0 0 0 0 0 0]

Enter move (row, col): (1,1)

[0 0 0 0 0 0]]

[0 0 0 0 1 0]

[0 0 0 0 0 0]

[0 0 0 0 0 0]

[0 0 0 0 0 0]

[[0 0 0 0 0 0]

Enter the subsquare to rotate (1-4): 1

Enter the direction (-1, 0, 1): -1

[0 0 0 0 0 0]]

[0 0 0 0 1 0]

[0 0 0 0 0 0]

[0 0 0 0 0 0]

[0 0 0 0 0 0]

[[0 0 0 0 0 0]

Computer moved:

[0 0 0 0 0 0]]

[0 0 0 0 1 0]

[0 0 0 0 0 0]

[0 0 2 0 0 0]

[0 0 0 0 0 0]

[[0 0 0 0 0 0]

:Computer rotated

[0 0 0 0 0 0]]

[0 0 0 0 1 0]

[0 0 0 0 0 0]

[0 0 2 0 0 0]

[0 0 0 0 0 0]

[[0 0 0 0 0 0]

:Enter move (row, col) (2,2)

[0 0 0 0 0 0]]

[0 0 0 0 1 0]

[0 0 0 1 0 0]

[0 0 2 0 0 0]

[0 0 0 0 0 0]

[[0 0 0 0 0 0]

Enter the subsquare to rotate (1-4): 3

Enter the direction (-1, 0, 1): 0

[0 0 0 0 0 0]]

[0 0 0 0 1 0]

[0 0 0 1 0 0]

[0 0 2 0 0 0]

[0 0 0 0 0 0]

[[0 0 0 0 0 0]

:Computer moved

[0 0 0 0 0 0]]

[0 0 2 0 1 0]

[0 0 0 1 0 0]

[0 0 2 0 0 0]

[0 0 0 0 0 0]

[[0 0 0 0 0 0]

:Computer rotated

[0 0 0 0 0 0]]

[0 0 2 0 1 0]

[0 0 0 1 0 0]

[2 0 0 0 0 0]

[0 0 0 0 0 0]

[[0 0 0 0 0 0]

Enter move (row, col): (2,3)

[0 0 0 0 0 0]]

[0 0 2 0 1 0]

[0 0 1 1 0 0]

[2 0 0 0 0 0]

[0 0 0 0 0 0]

[[0 0 0 0 0 0]

Enter the subsquare to rotate (1-4): 1

Enter the direction (-1, 0, 1): 1

[0 0 0 0 0 0]]

[0 0 2 0 1 0]

[0 0 1 0 0 1]

[2 0 0 0 0 0]

[0 0 0 0 0 0]

[[0 0 0 0 0 0]

Computer moved:

[0 0 2 0 0 0]]

[0 0 2 0 1 0]

[0 0 1 0 0 1]

[2 0 0 0 0 0]

[0 0 0 0 0 0]

[[0 0 0 0 0 0]

Computer rotated:

[0 0 2 0 0 0]]

[0 0 2 0 1 0]

[0 0 1 0 0 1]

[0 0 2 0 0 0]

[0 0 0 0 0 0]

[[0 0 0 0 0 0]

Enter move (row, col): (2,1)

[0 0 2 0 0 0]]

[0 0 2 0 1 0]

[0 0 1 0 1 1]

[0 0 2 0 0 0]

[0 0 0 0 0 0]

[[0 0 0 0 0 0]

Enter the subsquare to rotate (1-4): 1

Enter the direction (-1, 0, 1): 1

[0 0 2 0 0 1]]

[0 0 2 0 1 1]

[0 0 1 0 0 0]

[0 0 2 0 0 0]

[0 0 0 0 0 0]

[[0 0 0 0 0 0]

:Computer moved

[0 0 2 0 2 1]]

[0 0 2 0 1 1]

[0 0 1 0 0 0]

[0 0 2 0 0 0]

[0 0 0 0 0 0]

[[0 0 0 0 0 0]

:Computer rotated

[0 0 2 0 2 1]]

[0 0 2 0 1 1]

[0 0 1 0 0 0]

[0 0 0 0 0 0]

[0 0 0 0 0 0]

[[0 0 2 0 0 0]

Enter move (row, col): (2,2)

[0 0 2 0 2 1]]

[0 0 2 0 1 1]

[0 0 1 1 0 0]

[0 0 0 0 0 0]

[0 0 0 0 0 0]

[[0 0 2 0 0 0]

Enter the subsquare to rotate (1-4): 2

Enter the direction (-1, 0, 1): 1

[2 2 1 0 2 1]]

[0 0 0 0 1 1]

[0 0 0 1 0 0]

[0 0 0 0 0 0]

[0 0 0 0 0 0]

[[0 0 2 0 0 0]

Computer moved:

[2 2 1 2 2 1]]

[0 0 0 0 1 1]

[0 0 0 1 0 0]

[0 0 0 0 0 0]

[0 0 0 0 0 0]

[[0 0 2 0 0 0]

Computer rotated:

[2 2 1 2 2 1]]

[0 0 0 0 1 1]

[0 0 0 1 0 0]

[0 0 0 0 0 0]

[0 0 0 0 0 0]

[[0 0 2 0 0 0]

Enter move (row, col): (3,3)

[2 2 1 2 2 1]]

[0 0 0 0 1 1]

[0 0 0 1 0 0]

[0 0 1 0 0 0]

[0 0 0 0 0 0]

[[0 0 2 0 0 0]

Enter the subsquare to rotate (1-4): 2

Enter the direction (-1, 0, 1): -1

[0 0 2 2 2 1]]

[0 0 2 0 1 1]

[0 0 1 1 0 0]

[0 0 1 0 0 0]

[0 0 0 0 0 0]

[[0 0 2 0 0 0]

:Computer moved

[0 0 2 2 2 1]]

[0 0 2 0 1 1]

[0 0 1 1 0 0]

[0 0 1 0 0 0]

[0 2 0 0 0 0]

[[0 0 2 0 0 0]

:Computer rotated

[0 0 2 2 2 1]]

[0 0 2 0 1 1]

[0 0 1 1 0 0]

[0 0 0 0 0 0]

[0 2 0 0 0 0]

[[2 0 1 0 0 0]

:Enter move (row, col) (3,5)

[0 0 2 2 2 1]]

[0 0 2 0 1 1]

[0 0 1 1 0 0]

[1 0 0 0 0 0]

[0 2 0 0 0 0]

[[2 0 1 0 0 0]

Enter the subsquare to rotate (1-4): 4

Enter the direction (-1, 0, 1): 1

[0 0 2 2 2 1]]

[0 0 2 0 1 1]

[0 0 1 1 0 0]

[0 0 1 0 0 0]

[0 2 0 0 0 0]

[[1 0 2 0 0 0]

:Computer moved

[0 2 2 2 2 1]]

[0 0 2 0 1 1]

[0 0 1 1 0 0]

[0 0 1 0 0 0]

[0 2 0 0 0 0]

[[1 0 2 0 0 0]

:Computer rotated

[0 2 2 2 2 1]]

[0 0 2 0 1 1]

[0 0 1 1 0 0]

[0 0 1 0 0 0]

[0 2 0 0 0 0]

[[1 0 2 0 0 0]

(3,5) :Enter move (row, col)

[0 2 2 2 2 1]]

[0 0 2 0 1 1]

[0 0 1 1 0 0]

[1 0 1 0 0 0]

[0 2 0 0 0 0]

[[1 0 2 0 0 0]

Enter the subsquare to rotate (1-4): 4

Enter the direction (-1, 0, 1): -1

[[0 2 2 2 2 1]

[0 0 2 0 1 1]

[0 0 1 1 0 0]

[1 0 1 0 0 0]

[0 2 0 0 0 0]

[[2 0 1 0 0 0]

:Computer moved

[[2 2 2 2 2 1]

[0 0 2 0 1 1]

[0 0 1 1 0 0]

[1 0 1 0 0 0]

[0 2 0 0 0 0]

[[2 0 1 0 0 0]

:Computer rotated

[[2 2 2 2 2 1]

[0 0 2 0 1 1]

[0 0 1 1 0 0]

[2 0 1 0 0 0]

[0 2 0 0 0 0]

[[1 0 1 0 0 0]

!Computer wins