

CS 160

CS Orientation

More Lists in Python...

Odds and Ends

- Assignment #9???

Finish Tic-tac-toe...

```
1 def print_board(b):
2     print(b[0][0]+'|'+b[0][1]+'|'+b[0][2])
3     print('-----')
4     print(b[1][0]+'|'+b[1][1]+'|'+b[1][2])
5     print('-----')
6     print(b[2][0]+'|'+b[2][1]+'|'+b[2][2])
7
8 #####
9 # Description: gets the location from the player and places the players
10 #              piece at that location
11 # Parameters: game board with spaces, 'x's, or 'o's, the player's piece
12 # Pre-conditions: The player's piece contains either a x or o character
13 # Post-condition: The board contains a space replaced with the player's
14 #                 piece. (This means this function must check that row/col
15 #                 contains a space before placing the piece on the board!)
16 # Returns: none
17 #####
18
19 def get_location(b,p):
20     #this should be in a loop while b[row][col] is not a space to satisfy
21     #the postcondition
22     row=int(input("Enter the row, 0-2: "))
23     col=int(input("Enter the col, 0-2: "))
24     b[row][col]=p
25
```

-- INSERT --

1,1

Top

```
25
26 def check_winner(b,p):
27     #check horizontal
28     for x in range(3):
29         if(b[x][0]==p and b[x][1]==p and b[x][2]==p):
30             return True
31
32     #check vertical
33     for x in range(3):
34         if(b[0][x]==p and b[1][x]==p and b[2][x]==p):
35             return True
36
37     #check diagonal
38     if(b[0][0]==p and b[1][1]==p and b[2][2]==p):
39         return True
40     if(b[0][2]==p and b[1][1]==p and b[2][0]==p):
41         return True
42
43     #return false if the horizontal, vertical, or diagonal direction do
44     #not have 3 in a row.
45     return False
46
47 def main():
48     board=[[' ']*3, [' ']*3, [' ']*3]
49     print_board(board)
-- INSERT --
```

```
46
47 def main():
48     board=[[' ']*3, [' ']*3, [' ']*3]
49     print_board(board)
50     player1='x'
51     player2='o'
52     winner=False
53     turn=1
54     player_piece=player2
55
56     while((not winner) and turn<=9):
57         #change the player
58         if(player_piece==player1):
59             player_piece=player2
60         else:
61             player_piece=player1
62         get_location(board,player_piece)
63         print_board(board)
64         winner=check_winner(board, player_piece)
65
66         turn+=1
67
68     print(player_piece+" wins!")
69
70 main()
-- INSERT --
```

How do we change the travel program to use a 2-d array?

	speed	hours
trip1		
trip2		

	trip1	trip2	...	
speed				
hours				

this becomes harder
and requires
append
to create
because
we don't
know # of rows

how many trips?

travel_info = [[] * trips, [] * trips]

row1 row2

how many cols