CS 160
CS Orientation

More Lists in Python...
Odds and Ends

• Assignment #9???
Finish Tic-tac-toe...
```python
def print_board(b):
    print(b[0][0]+'|'+b[0][1]+'|'+b[0][2])
    print('-----')
    print(b[1][0]+'|'+b[1][1]+'|'+b[1][2])
    print('-----')
    print(b[2][0]+'|'+b[2][1]+'|'+b[2][2])

################################################################

# Description: gets the location from the player and places the players
# piece at that location
# Parameters: game board with spaces, 'x's, or 'o's, the player's piece
# Pre-conditions: The player's piece contains either a x or o character
# Post-condition: The board contains a space replaced with the player's
#                piece. (This means this function must check that row/col
#                contains a space before placing the piece on the board!)
# Returns: none
################################################################

def get_location(b,p):
    # This should be in a loop while b[row][col] is not a space to satisfy
    # the postcondition
    row=int(input("Enter the row, 0-2: "))
    col=int(input("Enter the col, 0-2: "))
    b[row][col]=p
```

25 def check_winner(b, p):
26     # check horizontal
27     for x in range(3):
28         if (b[x][0] == p and b[x][1] == p and b[x][2] == p):
29             return True
30
31     # check vertical
32     for x in range(3):
33         if (b[0][x] == p and b[1][x] == p and b[2][x] == p):
34             return True
35
36     # check diagonal
37     if (b[0][0] == p and b[1][1] == p and b[2][2] == p):
38         return True
39     if (b[0][2] == p and b[1][1] == p and b[2][0] == p):
40         return True
41
42     # return false if the horizontal, vertical, or diagonal directions do not have 3 in a row.
43     return False
44
45 def main():
46     board=[[' ']*3, [' ']*3, [' ']*3]
47     print_board(board)
def main():
    board=[[' ']*3, [' ']*3, [' ']*3]
    print_board(board)
    player1='x'
    player2='o'
    winner=False
    turn=1
    player_piece=player2

    while((not winner) and turn<=9):
        # change the player
        if(player_piece==player1):
            player_piece=player2
        else:
            player_piece=player1
        get_location(board, player_piece)
        print_board(board)
        winner=check_winner(board, player_piece)
        turn+=1

    print(player_piece+' wins!')

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

<table>
<thead>
<tr>
<th>trip1</th>
<th>trip2</th>
</tr>
</thead>
<tbody>
<tr>
<td>speed</td>
<td>hours</td>
</tr>
</tbody>
</table>

- This is harder and requires append to create because we don't know # of rows.

```
how many trips?
travel_info = [[J*trips, [J*trips]
row1
row2
```

- How many rows?