ECS 32A - Nested Collections/Loops

Aaron Kaloti

UC Davis - Summer Session #1 2020

Review: Lists Can Store Any Type

- Let's get right into it. Recall that a list can store values of any/different types.

```
>>> l = [2,3,8]
>>> l = ["abc","def"]
>>> l = [2,"2","abc",5.3]
>>> l
[2, '2', 'abc', 5.3]
```

Nested Lists

- This implies that lists can store other lists as well.

```
>>> l = [[1,8],[2,5]]
>>> l
[[1, 8], [2, 5]]
>>> l[0]
[1, 8]
>>> l[1]
[2, 5]
>>> l[2]
...
IndexError: list index out of range
>>> len(l)
2
>>> l[-1]
[2, 5]
```

Example: Seating Chart

```
>>> row1 = ["Zack","Aakash","Ethan","Emily","Frank"]
>>> row2 = ["Bob","Harold","Kumar","Drake","Josh"]
>>> row3 = ["Homer","Marge","Dave","Buster","Lindsay"]
```

```
>>> seating_chart = [row1,row2,row3]
>>> seating_chart
[['Zack', 'Aakash', 'Ethan', 'Emily', 'Frank'], ['Bob', 'Harold', 'Kumar', 'Drake',
'Josh'], ['Homer', 'Marge', 'Dave', 'Buster', 'Lindsay']]
```

- What is the output of each of the following?

```
>>> seating_chart[1][2]
...
>>> seating_chart[0][3]
...
>>> seating_chart[2][4]
...
>>> seating_chart[3][4]
...
```

Example: Seating Chart

```
>>> row1 = ["Zack","Aakash","Ethan","Emily","Frank"]
>>> row2 = ["Bob","Harold","Kumar","Drake","Josh"]
>>> row3 = ["Homer","Marge","Dave","Buster","Lindsay"]
```

```
>>> seating_chart = [row1,row2,row3]
>>> seating_chart
[['Zack', 'Aakash', 'Ethan', 'Emily', 'Frank'], ['Bob', 'Harold', 'Kumar', 'Drake',
'Josh'], ['Homer', 'Marge', 'Dave', 'Buster', 'Lindsay']]
```

- What is the output of each of the following?

```
>>> seating_chart[1][2]
'Kumar'
>>> seating_chart[0][3]
...
>>> seating_chart[2][4]
...
>>> seating_chart[3][4]
...
```

Example: Seating Chart

```
>>> row1 = ["Zack","Aakash","Ethan","Emily","Frank"]
>>> row2 = ["Bob","Harold","Kumar","Drake","Josh"]
>>> row3 = ["Homer","Marge","Dave","Buster","Lindsay"]
```

```
>>> seating_chart = [row1,row2,row3]
>>> seating_chart
[['Zack', 'Aakash', 'Ethan', 'Emily', 'Frank'], ['Bob', 'Harold', 'Kumar', 'Drake',
'Josh'], ['Homer', 'Marge', 'Dave', 'Buster', 'Lindsay']]
```

- What is the output of each of the following?

```
>>> seating_chart[1][2]
'Kumar'
>>> seating_chart[0][3]
'Emily'
>>> seating_chart[2][4]
...
>>> seating_chart[3][4]
...
```

Example: Seating Chart

```
>>> row1 = ["Zack","Aakash","Ethan","Emily","Frank"]
>>> row2 = ["Bob","Harold","Kumar","Drake","Josh"]
>>> row3 = ["Homer","Marge","Dave","Buster","Lindsay"]
```

```
>>> seating_chart = [row1,row2,row3]
>>> seating_chart
[['Zack', 'Aakash', 'Ethan', 'Emily', 'Frank'], ['Bob', 'Harold', 'Kumar', 'Drake',
'Josh'], ['Homer', 'Marge', 'Dave', 'Buster', 'Lindsay']]
```

- What is the output of each of the following?

```
>>> seating_chart[1][2]
'Kumar'
>>> seating_chart[0][3]
'Emily'
>>> seating_chart[2][4]
'Lindsay'
>>> seating_chart[3][4]
...
```

Example: Seating Chart

```
>>> row1 = ["Zack","Aakash","Ethan","Emily","Frank"]
>>> row2 = ["Bob","Harold","Kumar","Drake","Josh"]
>>> row3 = ["Homer","Marge","Dave","Buster","Lindsay"]
```

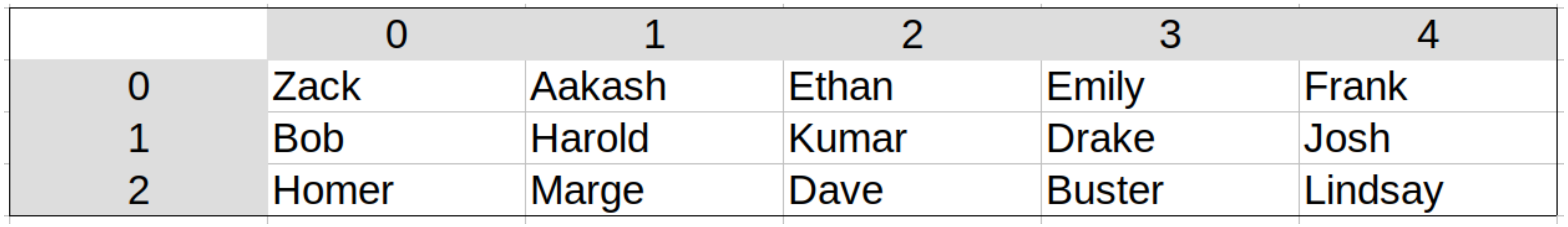
```
>>> seating_chart = [row1,row2,row3]
>>> seating_chart
[['Zack', 'Aakash', 'Ethan', 'Emily', 'Frank'], ['Bob', 'Harold', 'Kumar', 'Drake',
'Josh'], ['Homer', 'Marge', 'Dave', 'Buster', 'Lindsay']]
```

- What is the output of each of the following?

```
>>> seating_chart[1][2]
'Kumar'
>>> seating_chart[0][3]
'Emily'
>>> seating_chart[2][4]
'Lindsay'
>>> seating_chart[3][4]
...
IndexError: list index out of range
```

Example: Seating Chart

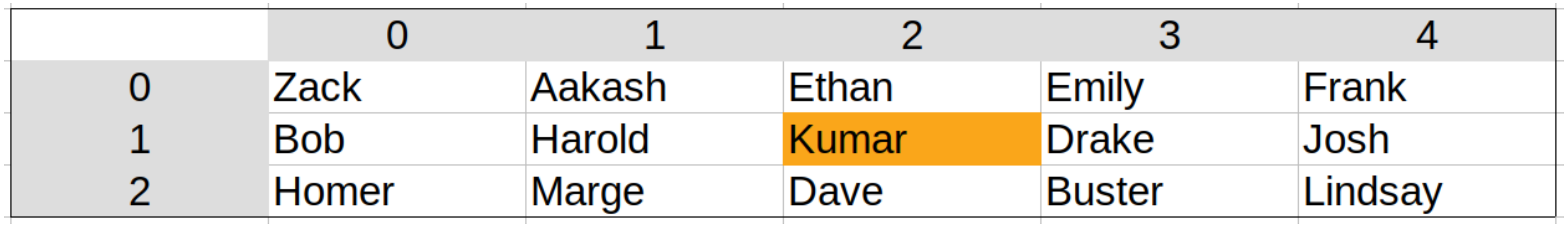
```
>>> row1 = ["Zack","Aakash","Ethan","Emily","Frank"]
>>> row2 = ["Bob","Harold","Kumar","Drake","Josh"]
>>> row3 = ["Homer","Marge","Dave","Buster","Lindsay"]
>>> seating_chart = [row1,row2,row3]
```



	0	1	2	3	4
0	Zack	Aakash	Ethan	Emily	Frank
1	Bob	Harold	Kumar	Drake	Josh
2	Homer	Marge	Dave	Buster	Lindsay

Example: Seating Chart

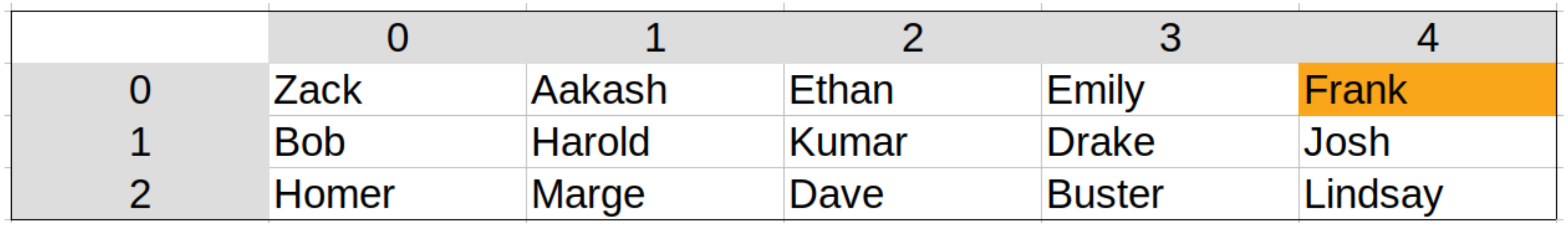
```
>>> row1 = ["Zack","Aakash","Ethan","Emily","Frank"]
>>> row2 = ["Bob","Harold","Kumar","Drake","Josh"]
>>> row3 = ["Homer","Marge","Dave","Buster","Lindsay"]
>>> seating_chart = [row1,row2,row3]
>>> seating_chart[1][2]
'Kumar'
```



	0	1	2	3	4
0	Zack	Aakash	Ethan	Emily	Frank
1	Bob	Harold	Kumar	Drake	Josh
2	Homer	Marge	Dave	Buster	Lindsay

Example: Seating Chart

```
>>> row1 = ["Zack","Aakash","Ethan","Emily","Frank"]
>>> row2 = ["Bob","Harold","Kumar","Drake","Josh"]
>>> row3 = ["Homer","Marge","Dave","Buster","Lindsay"]
>>> seating_chart = [row1,row2,row3]
>>> seating_chart[0][4]
'Frank'
```



	0	1	2	3	4
0	Zack	Aakash	Ethan	Emily	Frank
1	Bob	Harold	Kumar	Drake	Josh
2	Homer	Marge	Dave	Buster	Lindsay

Example: Seating Chart

```
>>> row1 = ["Zack","Aakash","Ethan","Emily","Frank"]
>>> row2 = ["Bob","Harold","Kumar","Drake","Josh"]
>>> row3 = ["Homer","Marge","Dave","Buster","Lindsay"]
>>> seating_chart = [row1,row2,row3]
>>> seating_chart[2][3]
'Buster'
```

	0	1	2	3	4
0	Zack	Aakash	Ethan	Emily	Frank
1	Bob	Harold	Kumar	Drake	Josh
2	Homer	Marge	Dave	Buster	Lindsay

Printing a Nested List

```python
row1 = ["Zack","Aakash","Ethan","Emily","Frank"]
row2 = ["Bob","Harold","Kumar","Drake","Josh"]
row3 = ["Homer","Marge","Dave","Buster","Lindsay"]
seating_chart = [row1,row2,row3]

for row in seating_chart:
    for name in row:
        print(name,end=' ')
    print()
```

seating_chart.py

```
Zack Aakash Ethan Emily Frank
Bob Harold Kumar Drake Josh
Homer Marge Dave Buster Lindsay
```

Printing a Nested List

```python
row1 = ["Zack","Aakash","Ethan","Emily","Frank"]
row2 = ["Bob","Harold","Kumar","Drake","Josh"]
row3 = ["Homer","Marge","Dave","Buster","Lindsay"]
seating_chart = [row1,row2,row3]

for row in seating_chart:
    for name in row:
        print(name,end=' ')
    print()
```

seating_chart.py

```
Zack Aakash Ethan Emily Frank
Bob Harold Kumar Drake Josh
Homer Marge Dave Buster Lindsay
```

- During the first iteration of the "outer loop", `row` will equal the list given by `row1`, and `name` will take on each value in that list (i.e. "Zack", "Aakash", etc.).
- During the second iteration of the "outer loop", `row` will equal the list given by `row2`, and `name` will take on each value in that list (i.e. "Bob", "Harold", etc.).
- During the third iteration of the "outer loop", `row` will equal the list given by `row3`, and `name` will take on each value in that list (i.e. "Homer", "Marge", etc.).

Printing a Nested List

```python
row1 = ["Zack","Aakash","Ethan","Emily","Frank"]
row2 = ["Bob","Harold","Kumar","Drake","Josh"]
row3 = ["Homer","Marge","Dave","Buster","Lindsay"]
seating_chart = [row1,row2,row3]

for row in seating_chart:
    for name in row:
        print(name,end='  ')
    print()
```

seating_chart.py

- Why do we need `end='  '`?

Printing a Nested List

```python
row1 = ["Zack","Aakash","Ethan","Emily","Frank"]
row2 = ["Bob","Harold","Kumar","Drake","Josh"]
row3 = ["Homer","Marge","Dave","Buster","Lindsay"]
seating_chart = [row1,row2,row3]

for row in seating_chart:
    for name in row:
        print(name)
```

seating_chart_bad.py

```
Zack
Aakash
Ethan
Emily
Frank
Bob
Harold
Kumar
Drake
Josh
Homer
Marge
Dave
Buster
Lindsay
```

HW #6

- A nested list may also be referred to as a 2-dimensional list or 2D list.
- Your HW #6 will involve 5-dimensional lists.
- Ask any questions you have!

HW #6... Gottem

- A nested list may also be referred to as a 2-dimensional list or 2D list.
- Your HW #6 will involve 5-dimensional lists. Just kidding.
- Ask any questions you have!

Traversing a 2D List with `range()`

- You can also use a `range()`-based `for` loop.

```
row1 = ["Zack","Aakash","Ethan","Emily","Frank"]
row2 = ["Bob","Harold","Kumar","Drake","Josh"]
row3 = ["Homer","Marge","Dave","Buster","Lindsay"]
seating_chart = [row1,row2,row3]

for i in range(len(seating_chart)):
    row = seating_chart[i]
    for j in range(len(row)):
        name = row[j]
        print(name,end=' ')
    print()
```

seating_chart_other.py

```
Zack Aakash Ethan Emily Frank
Bob Harold Kumar Drake Josh
Homer Marge Dave Buster Lindsay
```

- Unlike the other kind of for loop, this `range()`-based `for` loop has the advantage of tracking which position we are currently at in the 2D list.
    - For example, if we are at "Drake", `i` will be 1, and `j` will be 3.

Traversing a 2D List with `range()`

- You can also use a `range()`-based `for` loop.

```python
row1 = ["Zack","Aakash","Ethan","Emily","Frank"]
row2 = ["Bob","Harold","Kumar","Drake","Josh"]
row3 = ["Homer","Marge","Dave","Buster","Lindsay"]
seating_chart = [row1,row2,row3]

for i in range(len(seating_chart)):
    row = seating_chart[i]
    for j in range(len(row)):
        name = row[j]
        print(name,end=' ')
    print()
```

seating_chart_other.py

```
Zack Aakash Ethan Emily Frank
Bob Harold Kumar Drake Josh
Homer Marge Dave Buster Lindsay
```

- I made an extra `row` variable merely for clarity.

Traversing a 2D List with `range()`

- You can also use a `range()`-based `for` loop.

```python
row1 = ["Zack","Aakash","Ethan","Emily","Frank"]
row2 = ["Bob","Harold","Kumar","Drake","Josh"]
row3 = ["Homer","Marge","Dave","Buster","Lindsay"]
seating_chart = [row1,row2,row3]

for i in range(len(seating_chart)):
    for j in range(len(seating_chart[i])):
        name = seating_chart[i][j]
        print(name,end=' ')
    print()
```

```
Zack Aakash Ethan Emily Frank
Bob Harold Kumar Drake Josh
Homer Marge Dave Buster Lindsay
```

- You could have done this as well.

Traversing a 2D List with `range()`

- You can also use a `range()`-based `for` loop.

```python
row1 = ["Zack","Aakash","Ethan","Emily","Frank"]
row2 = ["Bob","Harold","Kumar","Drake","Josh"]
row3 = ["Homer","Marge","Dave","Buster","Lindsay"]
seating_chart = [row1,row2,row3]

for i in range(len(seating_chart)):
    for j in range(len(seating_chart[i])):
        print(seating_chart[i][j],end=' ')
    print()
```

```
Zack Aakash Ethan Emily Frank
Bob Harold Kumar Drake Josh
Homer Marge Dave Buster Lindsay
```

- Or this.

Traversing a 2D List with `range()`

- You can also use a `range()`-based `for` loop.

```
row1 = ["Zack","Aakash","Ethan","Emily","Frank"]
row2 = ["Bob","Harold","Kumar","Drake","Josh"]
row3 = ["Homer","Marge","Dave","Buster","Lindsay"]
seating_chart = [row1,row2,row3]

for i in range(len(seating_chart)):
    for j in range(len(seating_chart)):
        print(seating_chart[i][j],end=' ')
    print()
```

```
Zack Aakash Ethan
Bob Harold Kumar
Homer Marge Dave
```

- But just make sure to avoid a mistake in the inner `for` loop. The above needs to be `len(seating_chart[i])`. Using `len(seating_chart)` would only work if the `seating_chart` were always a square.

Traversing a 2D List with `while`

- You can also use a `range()`-based `for` loop.

```
row1 = ["Zack","Aakash","Ethan","Emily","Frank"]
row2 = ["Bob","Harold","Kumar","Drake","Josh"]
row3 = ["Homer","Marge","Dave","Buster","Lindsay"]
seating_chart = [row1,row2,row3]

i = 0
while i < len(seating_chart):
    row = seating_chart[i]
    j = 0
    while j < len(row):
        name = row[j]
        print(name,end=' ')
        j += 1
    i += 1
    print()
```

seating_chart_while.py

```
Zack Aakash Ethan Emily Frank
Bob Harold Kumar Drake Josh
Homer Marge Dave Buster Lindsay
```

- As with a `range()`-based `for` loop, using a `while` loop has the advantage of tracking which position we are currently at in the 2D list.

Traversing with One Loop

- If you know that each row in the list is of the same *fixed* length, you can traverse using one loop[1].

```python
row1 = ["Zack","Aakash","Ethan","Emily","Frank"]
row2 = ["Bob","Harold","Kumar","Drake","Josh"]
row3 = ["Homer","Marge","Dave","Buster","Lindsay"]
seating_chart = [row1,row2,row3]

for [a,b,c,d,e] in seating_chart:
    print(a,end=' ')
    print(b,end=' ')
    print(c,end=' ')
    print(d,end=' ')
    print(e,end=' ')
    print()
```

seating_chart_one_loop.py

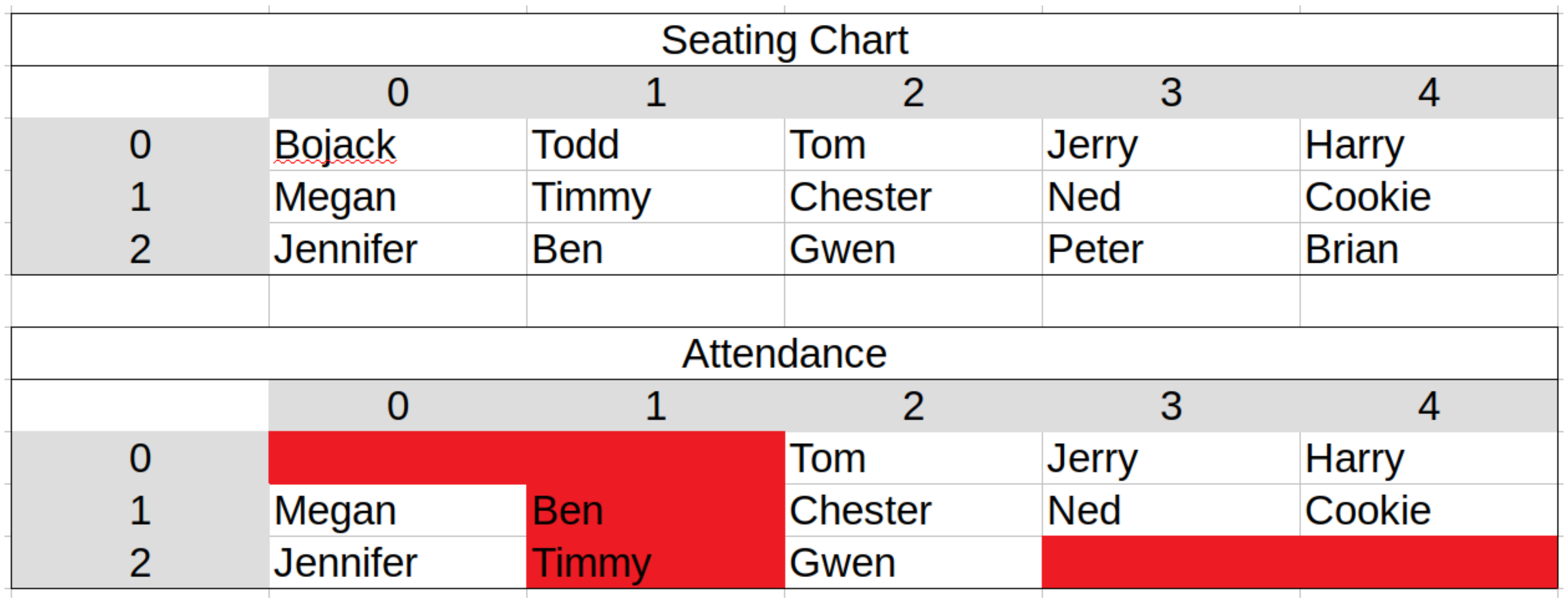
```
Zack Aakash Ethan Emily Frank
Bob Harold Kumar Drake Josh
Homer Marge Dave Buster Lindsay
```

- If the rows could be any length (while still being the same length as *each other*), then you have to use two loops.

1. The square brackets around `a,b,c,d,e` are optional.

Notes on HW #5

- On some part of HW #5, you will write a function that takes a seating chart (a 2D list) and the attendance that day (another 2D list), and you will need to count the number of absences, including students who are in the wrong seats.



Seating Chart

	0	1	2	3	4
0	Bojack	Todd	Tom	Jerry	Harry
1	Megan	Timmy	Chester	Ned	Cookie
2	Jennifer	Ben	Gwen	Peter	Brian

Attendance

	0	1	2	3	4
0	🟥	🟥	Tom	Jerry	Harry
1	Megan	Ben	Chester	Ned	Cookie
2	Jennifer	Timmy	Gwen	🟥	🟥

- In the assignment, an empty seat will be denoted by an empty string.

Nested Collections

- 2D lists are not the only kinds of nested collections that we can have.
- Here are examples of other kinds:
    - list of lists of lists (i.e. a 3D list)
    - a dictionary that maps tuples to lists
        - example: `d = {(5,3):[8,1,2], (1,4,9):["abc"]}`
    - a tuple of lists of dictionaries that map strings to integers
        - example: `d = ([{"key1":1,"key2":2},{"key3":3}],[{"key4":4,"key5":5,"key6":6}])`

The `in` Operator

- As with "one-dimensional" collections like lists, proper use of the `in` operator can save you from using one more loop than necessary in certain cases. However, it works differently than you might expect for 2D collections.

```
row1 = ["Zack","Aakash","Ethan","Emily","Frank"]
row2 = ["Bob","Harold","Kumar","Drake","Josh"]
row3 = ["Homer","Marge","Dave","Buster","Lindsay"]
seating_chart = [row1,row2,row3]

print("1:", row1 in seating_chart)
print("2:", ["Bob","Harold","Kumar","Drake","Josh"] in seating_chart)
print("3:", "Harold" in seating_chart)
print("4:", ["Harold"] in seating_chart)
print("5:", "Harold" in seating_chart[1])
print("6:", "Harold" in seating_chart[2])
```

seating_chart_in.py

```
1: True
2: True
3: False
4: False
5: True
6: False
```

Example: Belongs to Both

- Write a function called `belongs_to_both` that takes two lists and a 2-tuple (i.e. a tuple of length 2). The function should return `True` if the first element of the given tuple is in the first list and the second element is in the second list; otherwise, it should return `False`.
- Here are examples of how the function should behave:

```
>>> belongs_to_both([5,8,3],[2,1],(5,2))
True
>>> belongs_to_both([5,8,3],[2,1],(3,2))
True
>>> belongs_to_both([5,8,3],[2,1],(8,5))
False
```

To Be Continued...