

Lecture 11: Nested Lists

CS 51P

October 12, 2022



Tom Yeh
he/him/his

Class News

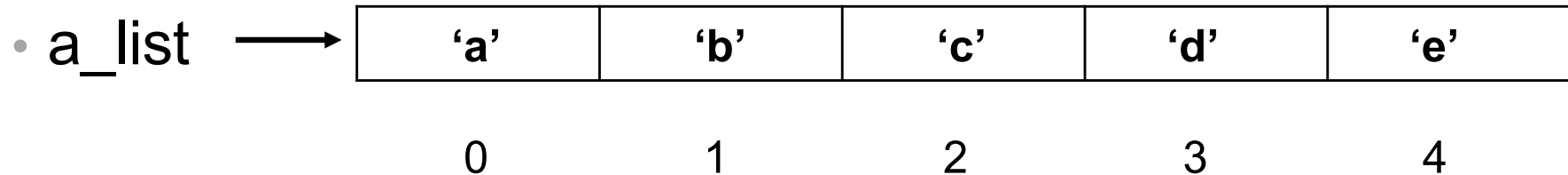
- How was the checkpoint?
- Assignment 5 – Image Manipulation
 - Due date postponed by 2 days to **Thursday** for Fall Break

Learning Goals

- Nested Lists
- Images

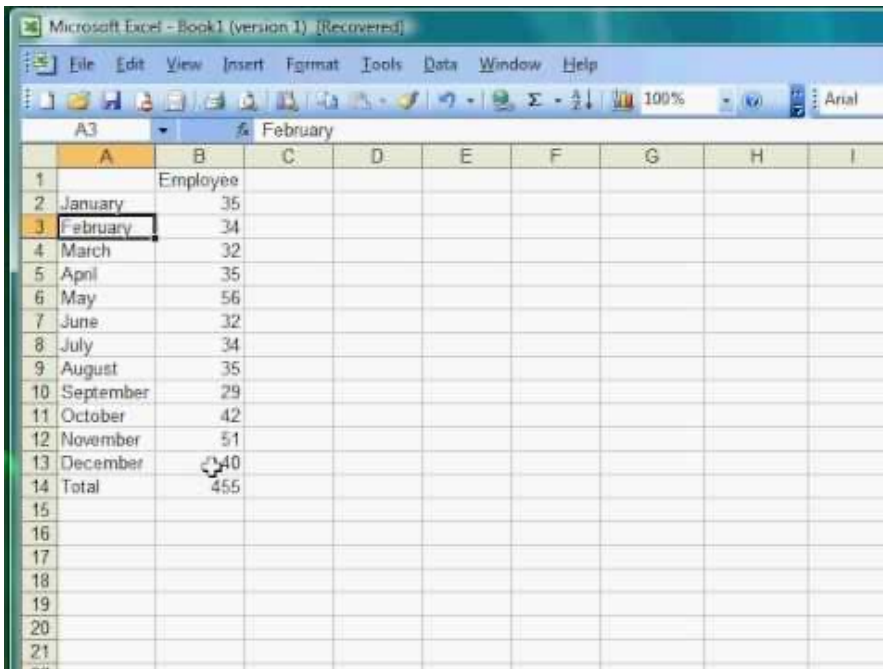
Previously...

- A list is an ordered collection of elements
- `a_list = ['a', 'b', 'c', 'd', 'e']`



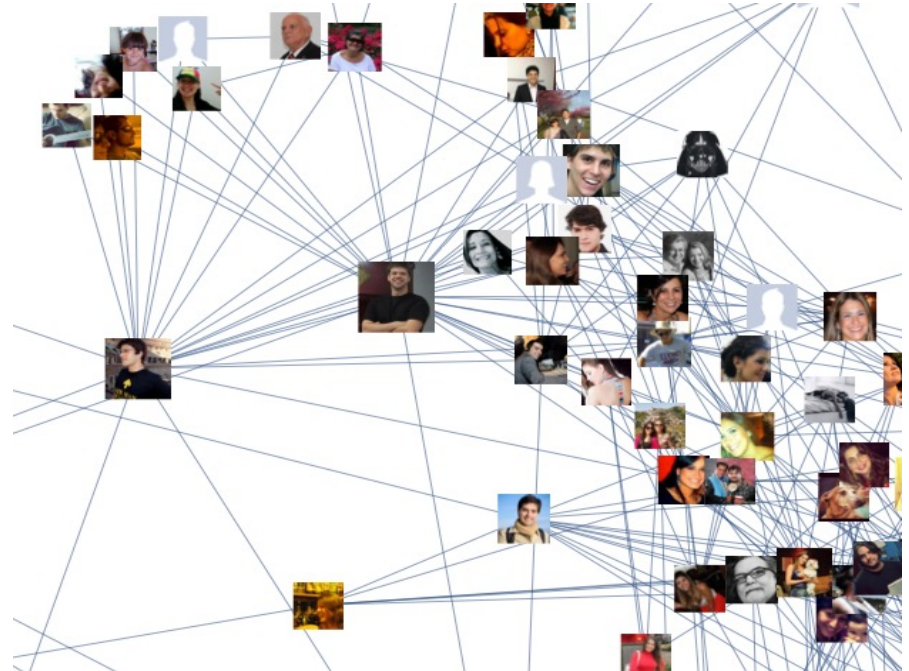
Matrices

- Can think of lists as a one-dimensional matrix
- What if you want to use a 2-dimensional matrix?
- Can create a **list of lists** aka a **nested list**!



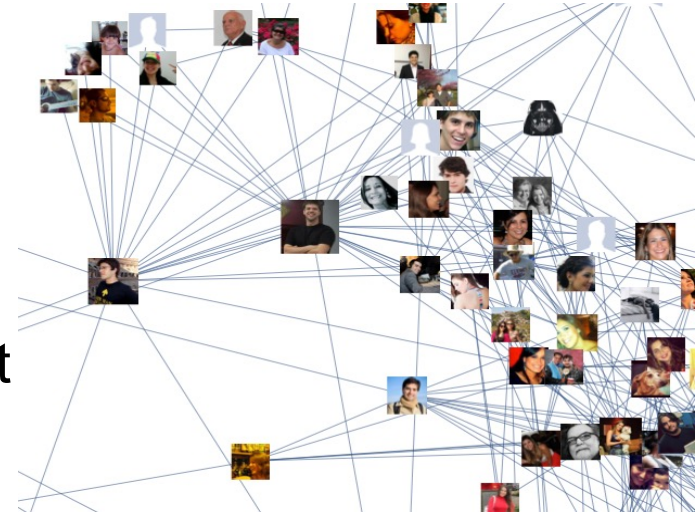
Microsoft Excel - Book1 (version 1) [Recovered]

	A	B	C	D	E	F	G	H	I
1		Employee							
2	January	35							
3	February	34							
4	March	32							
5	April	35							
6	May	56							
7	June	32							
8	July	34							
9	August	35							
10	September	29							
11	October	42							
12	November	51							
13	December	40							
14	Total	455							
15									
16									
17									
18									
19									
20									
21									



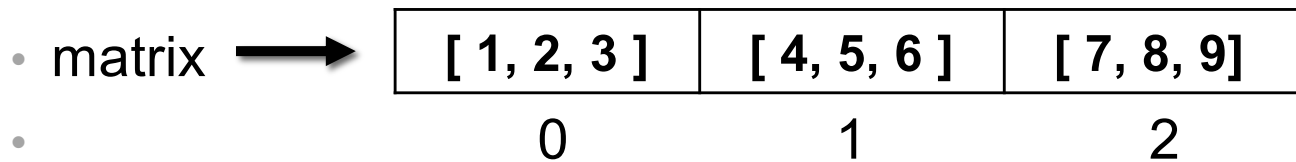
2-Dimensional List

- 2-D list is a list of lists
 - Each element of “outer” list is just another list
 - Can think of this as a grid or matrix
- Example:
 - 2-D list of users’ friends or contacts
 - Each element of outer list is a person’s friends list
 - matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]



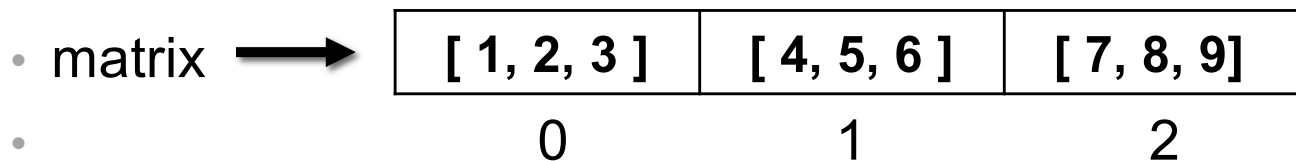
2-Dimensional List

- 2-D list is a list of lists
 - Each element of “outer” list is just another list
 - Can think of this as a grid or matrix
- Example:
 - matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]

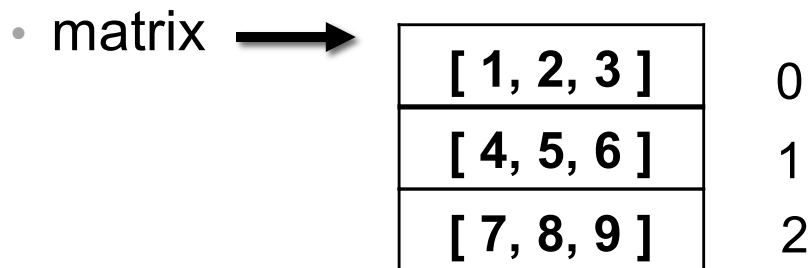


2-Dimensional List

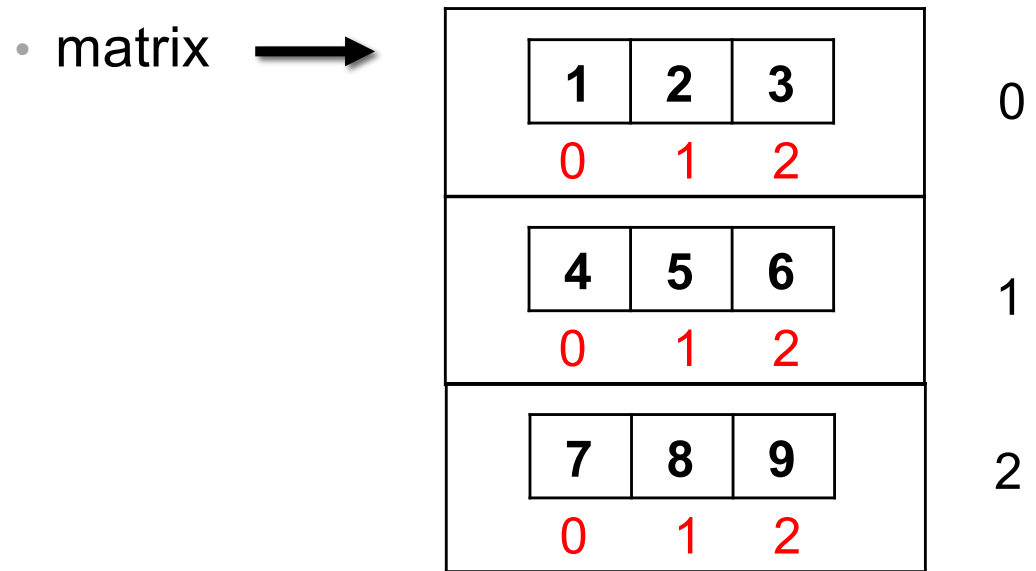
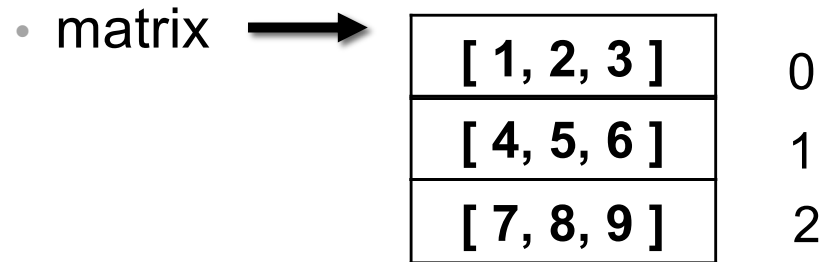
- 2-D list is a list of lists
 - Each element of “outer” list is just another list
 - Can think of this as a grid or matrix
- Example:
 - matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]



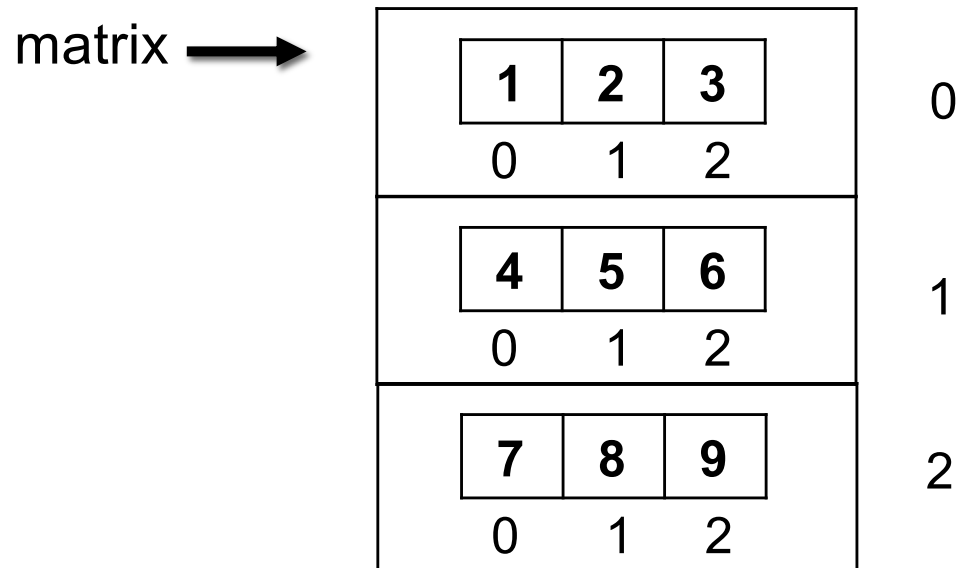
- May be easier to visualize like this:



2-Dimensional List

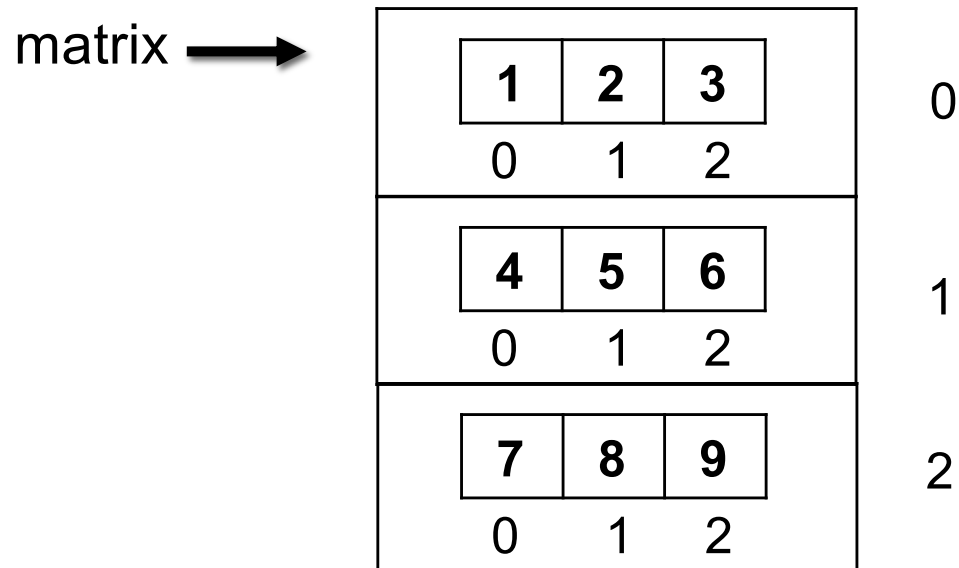


2-Dimensional List



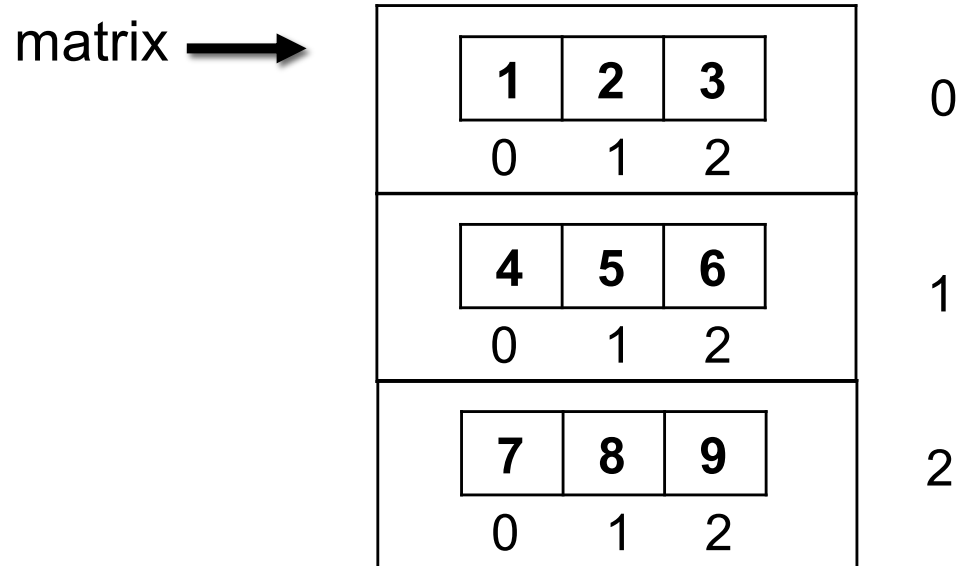
- To access elements, specify index in “outer” list first (row)
- Then index in “inner” list (column)
- `matrix[0][0]` → 1
- `matrix[1][0]` → 4
- `matrix[2][2]` → 9

2-Dimensional List



- To access elements, specify index in “outer” list first, then index in “inner” list
- `matrix[1][2]` → ?
- `matrix[2][1]` → ?
- `matrix[0][2]` → ?

2-Dimensional List



- What if we only specify one index?
- matrix[0] → ?
- matrix[1] → ?
- matrix[2] → ?

More Fun with Lists

- Do the inner lists all have to be the same size?
 - No! Be careful if sizes are not the same.
 - ragged = [[1, 2, 3], [4], [5, 6]]
 - ragged[0] [1, 2, 3]
 - ragged[1] [4]
 - ragged[2] [5, 6]

Example

- Define a function `nested_total` that takes a list of lists of ints and returns the sum of all the values.

```
list = [[1,2], [3], [4,5,6]]  
sum = nested_total(list)  
print(sum)
```

Exercise

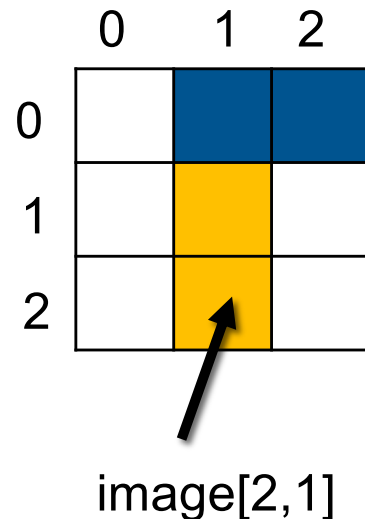
- Define a function `nested_avg` that takes a list of lists of ints and returns a list with each sublist averaged

```
list = [[1,2], [3], [4,5,6]]  
list_avg = nested_avg(list)  
print(list_avg)
```

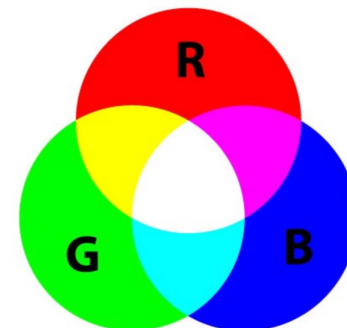
```
[1.5, 3.0, 5.0]
```

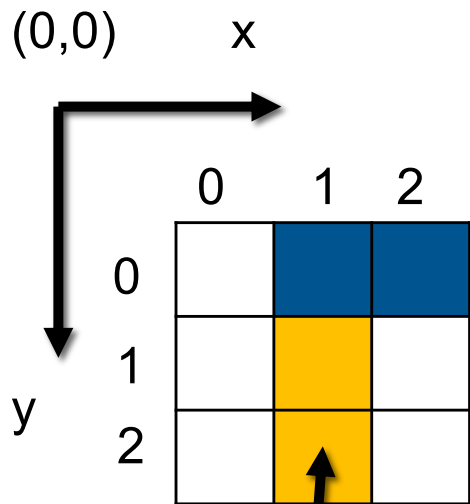
Images

Image →

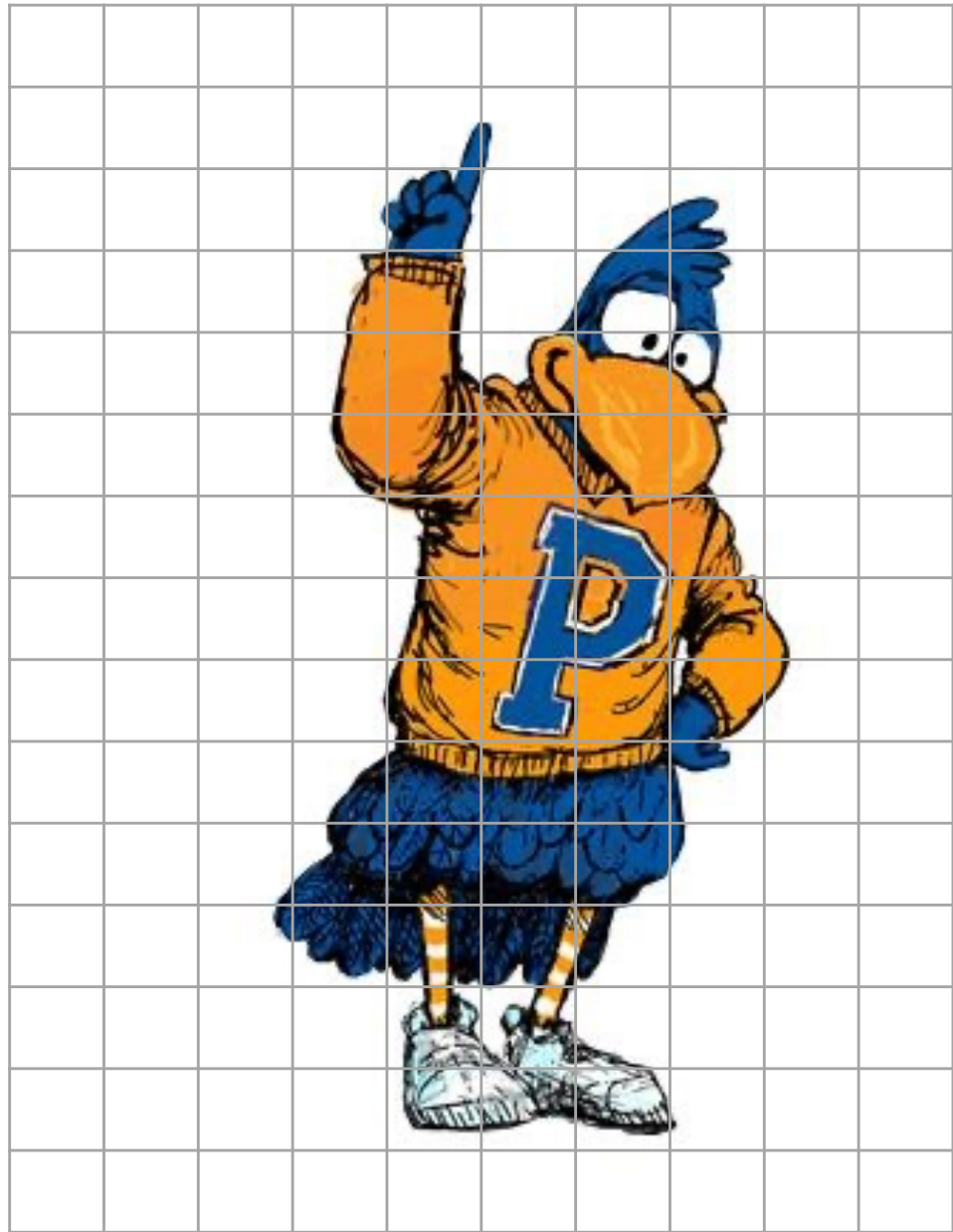


- Images are 2D list of tiny squares called pixels
- Each pixel holds RGB values
 - Red, Green, and Blue
 - Each value is the brightness for the color
 - Can make any color from RGB
 - Additive vs subtractive (RYB)





Pixels [2,1]
red: 255
green: 165
blue: 23



Multi-dimensional Lists

- Can we have more than 2 dimensions?

Multi-dimensional Lists

- Can we have more than 2 dimensions?

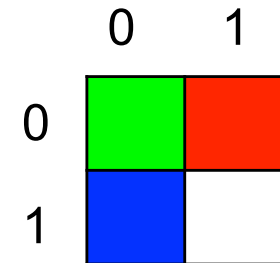
- Yes! As many as you would like (within reason).

- `image = [[[0, 255, 0], [255, 0, 0]], [[0, 0, 255], [255, 255, 255]]]`

- `image[0]` \longrightarrow ?

- `image[0][1]` \longrightarrow ?

- `image[0][1][0]` \longrightarrow ?





Example - Sudoku

LEVEL: Beginner

		9	6		7	4	3	1
8				5	3			9
	6		2			5		
		8	9					6
		2		4		7		5
					1			
			5	9	4	3		2
	2	7		3				1
4			1		2	6	5	

www.dctech.com/sudoku/

```
board = [[0,0,9,6,0,7,4,3,1],  
         [8,0,0,0,5,3,0,0,9],  
         [0,6,0,2,0,0,5,0,0],  
         ...  
         [4,0,0,1,0,2,6,5,0]]
```

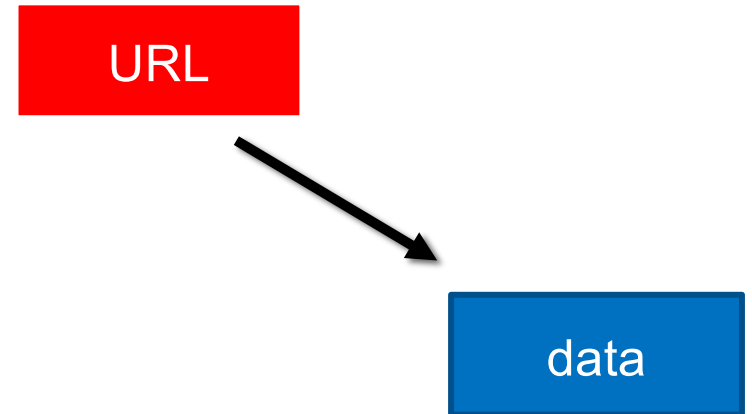
- Rules of the game:
 - Grid of 9x9 spaces
 - Each row, column, and 3x3 square needs to have the numbers 1-9, without repeating any numbers within row, column or square
- write a function `set_value` that takes a nested list `board` and ints `i`, `j`, `n` and updates the `(i,j)`th entry of `board` to be the value `n`

When lists are passed as parameters

- Variables that act like they are **copied**.
- integer variable
- float data
- boolean
- string
- These types are immutable. You copy the values for parameters.



- Variables that act like their **URL** is copied.
- list variable
- URL
- data
- These types are mutable. You get reference (URL) for parameters. Changes are in place when you assign.



Exercise - Sudoku

LEVEL: Beginner

		9	6		7	4	3	1
8				5	3			9
	6		2			5		
		8	9					6
		2		4		7		5
					1			
			5	9	4	3		2
	2	7		3				1
4			1		2	6	5	

www.dctech.com/sudoku/

```
board = [[0,0,9,6,0,7,4,3,1],  
         [8,0,0,0,5,3,0,0,9],  
         [0,6,0,2,0,0,5,0,0],  
         ...  
         [4,0,0,1,0,2,6,5,0]]
```

- write a function `check_row_i` that takes an int `i` and a nested list `board`. The function should return `True` if and only if row `i` contains each integer from 1 through 9 exactly once.
- write a function `check_column_i` that takes an int `i` and a nested list `board`. The function should return `True` if and only if column `i` contains each integer from 1 through 9 exactly once.

Additional Exercises - Sudoku

LEVEL: Beginner

		9	6		7	4	3	1
8				5	3			9
	6		2			5		
		8	9					6
		2		4		7		5
					1			
			5	9	4	3		2
	2	7		3				1
4			1		2	6	5	

www.dctech.com/sudoku/

```
board = [[0,0,9,6,0,7,4,3,1],  
         [8,0,0,0,5,3,0,0,9],  
         [0,6,0,2,0,0,5,0,0],  
         ...  
         [4,0,0,1,0,2,6,5,0]]
```

- write a function `check_block_ij` that takes ints `i` and `j` and a nested list `board`. The function should return `True` if and only if the 3x3 block starting at row `i`, column `j` contains each integer from 1 through 9 exactly once
- write a function `check_solution` that takes a nested list `board` and returns `True` if and only if `board` represents a correctly solved puzzle.

Recap

- Nested lists – multi-dimensional lists
- Image – 2D matrix of pixels