Algorithms and Data Structures for Data Science

Multi-dimensional Lists

CS 277

Brad Solomon

January 30, 2023

UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

Department of Computer Science

Learning Objectives

Introduce multi-dimensional lists

Discuss other list data types in Python (what are they good at?)

Go over mini-project expectations and instructions

Introduce the two fundamental list implementations

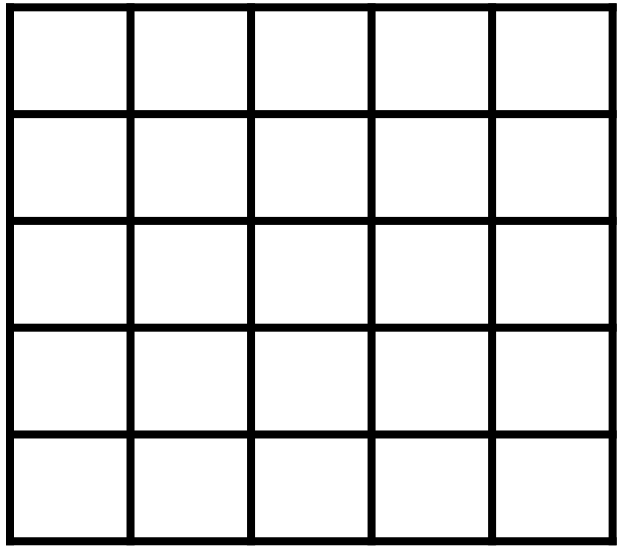
List Abstract Data Type

A minimally functional list must have the following functions:

Constructor:	`__init__()`	
Insert:	`append(x)`	`insert(i, x)`
Delete:	`remove(x)`	`pop()`
Index	`__getitem__() []`	`index(x)`
Size()**	`len(list)`	

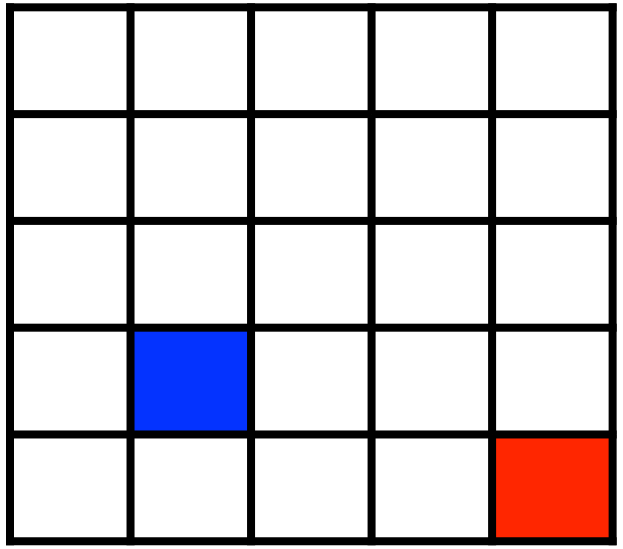
Multi-dimensional lists

How can we make a matrix in Python?



Multi-dimensional lists

How is a matrix in Python indexed?



Multi-dimensional lists

7	8	9
4	5	6
1	2	3

Lists in Python store objects. Lists in Python are objects.

```
def makeMatrix():
```

```
M[2] =
```

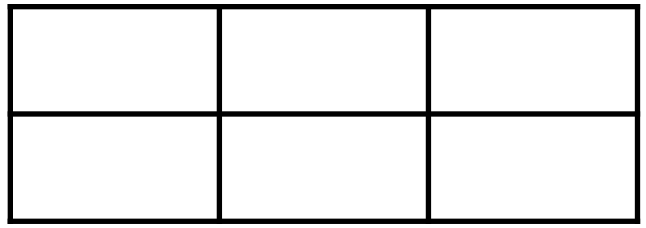
```
M[1][0] =
```

```
M[0][2] =
```

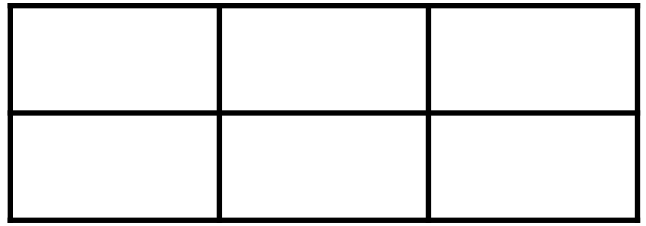
Multi-dimensional lists

```
1  import copy
2
3  orig = [ [1,2,3], [4, 5, 6]]
4  not_copy = orig
5  s_copy = orig.copy()
6  d_copy = copy.deepcopy(orig)
7
8  orig[1][1]=3
9  not_copy[0][0]=2
10 s_copy[0][2]=10
11 d_copy[1][2]=15
12
13 print(orig)
14 print(not_copy)
15 print(s_copy)
16 print(d_copy)
17
18
19
20
21
22
```

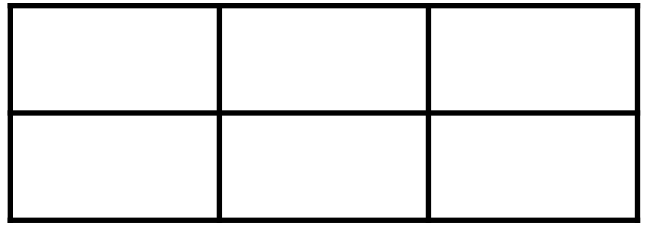
Orig



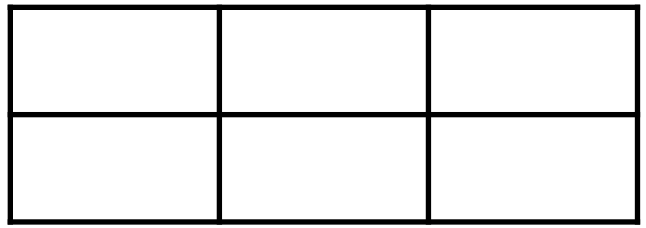
NotCopy



Shallow

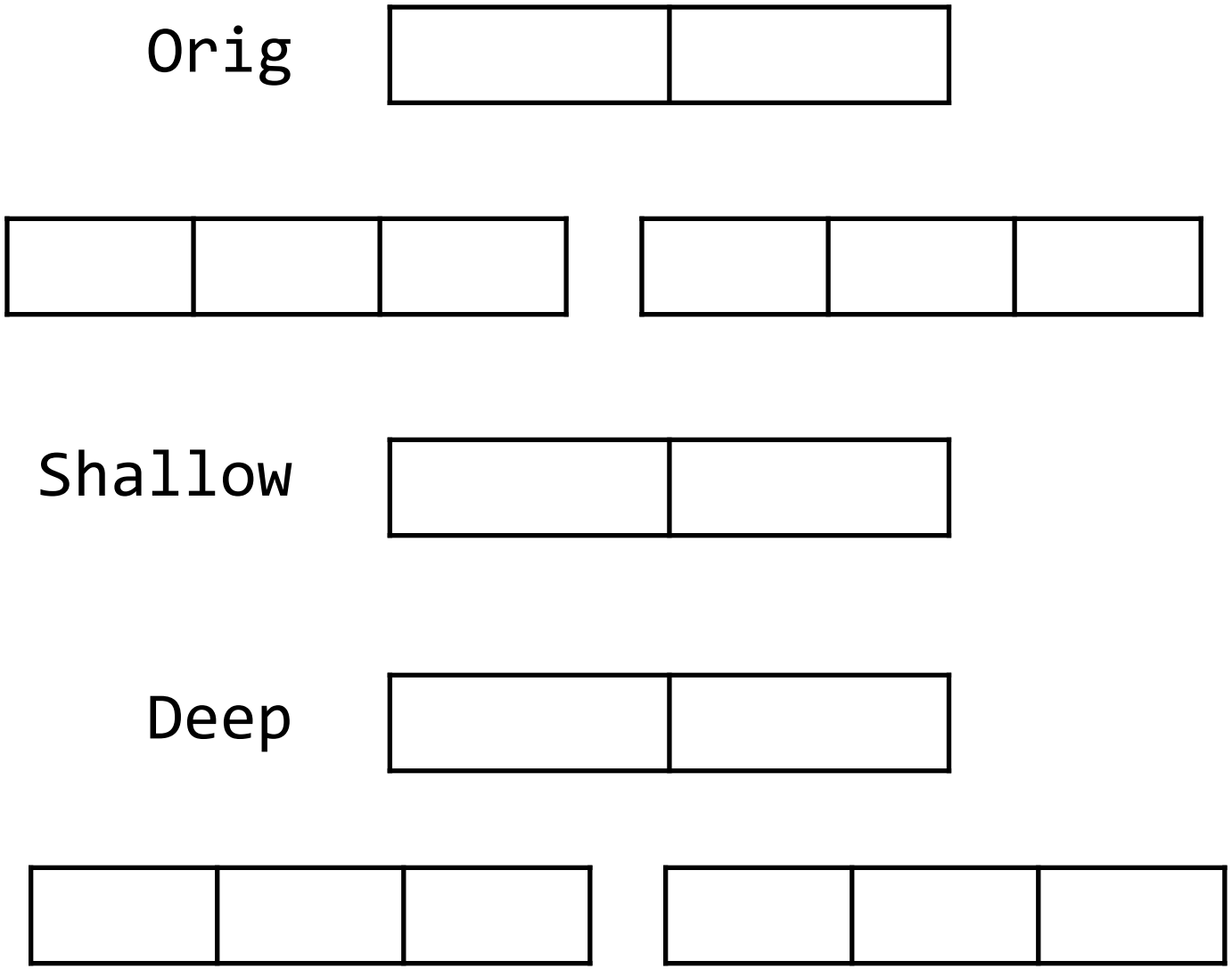


Deep



Multi-dimensional lists

```
1  import copy
2
3  orig = [ [1,2,3], [4, 5, 6]]
4  not_copy = orig
5  s_copy = orig.copy()
6  d_copy = copy.deepcopy(orig)
7
8  orig[1][1]=3
9  not_copy[0][0]=2
10 s_copy[0][2]=10
11 d_copy[1][2]=15
12
13 print(orig)
14 print(not_copy)
15 print(s_copy)
16 print(d_copy)
17
18
19
20
21
22
```



Deep vs Shallow Copying

A **shallow copy** creates a new object but when possible reuses references

A **deep copy** creates a new object for every object (in the object)

```
1  myList = [1, 2, 3, 4, 5]
2
3  print(myList)
4
5  print(len(myList))
6
7  print(myList[2])
8
9  clist = myList.copy()
```

Deep vs Shallow Copying

A **shallow copy** creates a new object but when possible reuses references

A **deep copy** creates a new object for every object (in the object)

```
1  myList = [1, 2, 3, 4, 5]
2
3  print(myList)
4
5  print(len(myList))
6
7  print(myList[2])
8
9  clist = myList.copy()
```

```
1  myTuple = (1, 2, 3, 4, 5)
2
3  print(myTuple)
4
5  print(len(myTuple))
6
7  print(myTuple[2])
8
9  ctuple = myTuple
```

```
1  myNP = np.array([1,2,3,4,5])
2
3  print(myNP)
4
5  print(len(myNP))
6
7  print(myNP[2])
8
9  cnp=myNP.copy()
```

Python Tuple

In Python, a tuple is an immutable list.

```
1  myTuple = ([1,2,3], "ABC")
2
3  print(myTuple[1])
4
5  myTuple[0] = [3, 4, 5]
6
7  myTuple[0].append("oops")
8
9  print(myTuple)
10
11
12
13
14
15
16
17
18
```

Python Numpy List

A numpy list in Python is designed for scientific computing

```
 1 import numpy as np
 2
 3 r1 = [7, 8, 9]
 4 r2 = [4, 5, 6]
 5 r3 = [1, 2, 3]
 6 plist = [r1, r2, r3]
 7
 8 print(plist)
 9
10 myNP = np.array(plist)
11
12 print(myNP)
13
14 myNP = myNP.reshape(1, 9)
15
16 print(myNP)
17
18 odd = myNP % 2
19
20 print(odd)
```

Honorable Mention: pandas

```
1  import pandas
2
3  pd.read_table('myFile.csv')
4
5
6
7  pd.read_table('myFile.csv',delimiter=',')
8
9
10
11
12 pd.read_table('myFile.csv',delimiter=',',
13 usecols = ['Netid','Grade'])
14
15
16
17
18
19
```

Mini-Projects in CS 277

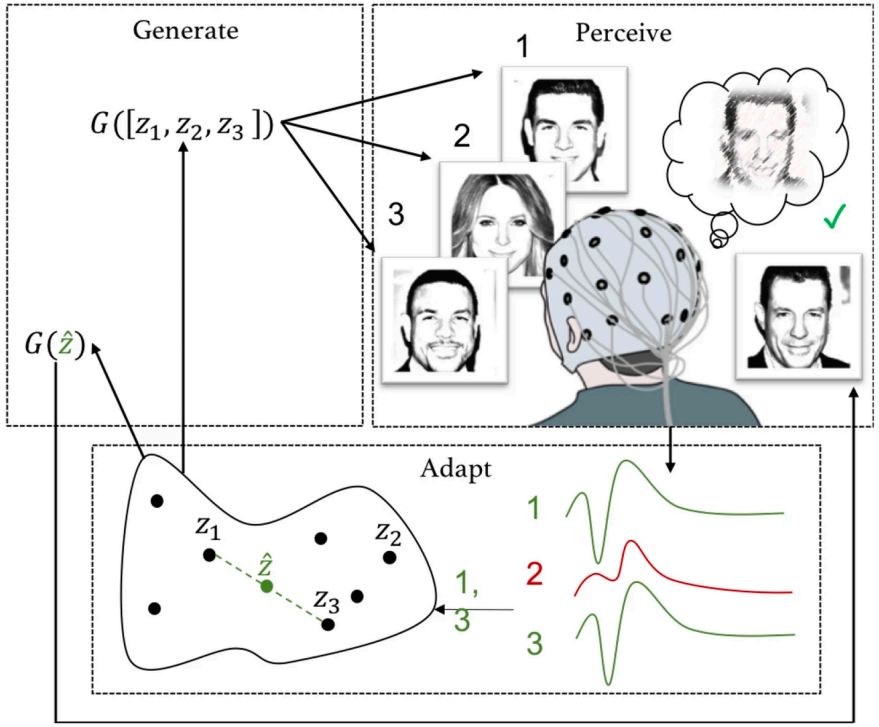
Goal: Teach and practice coding skills alongside data science

Two parts:

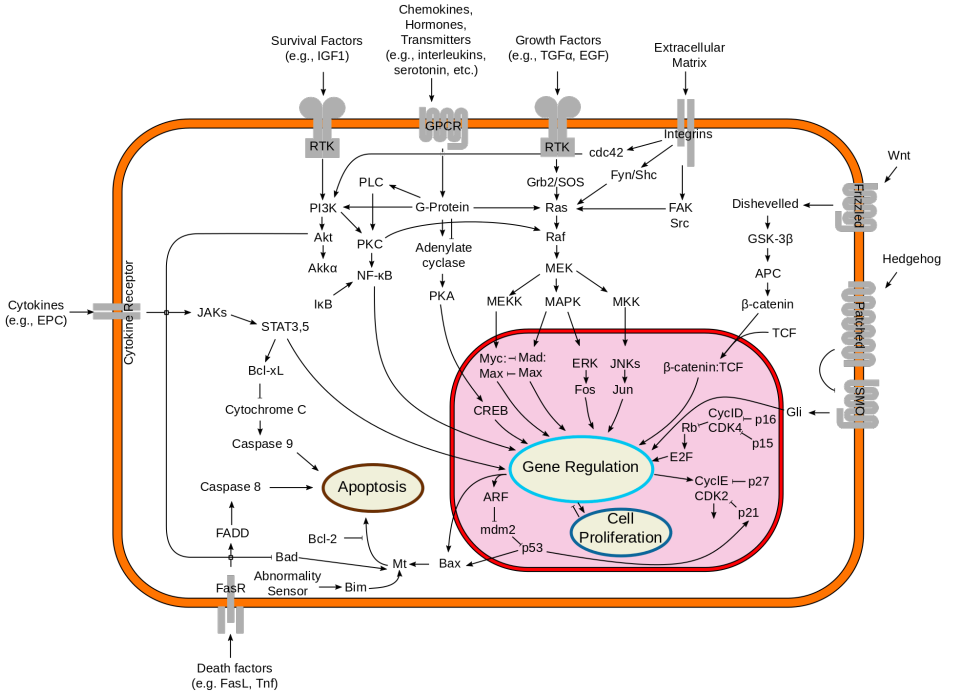
A coding assignment

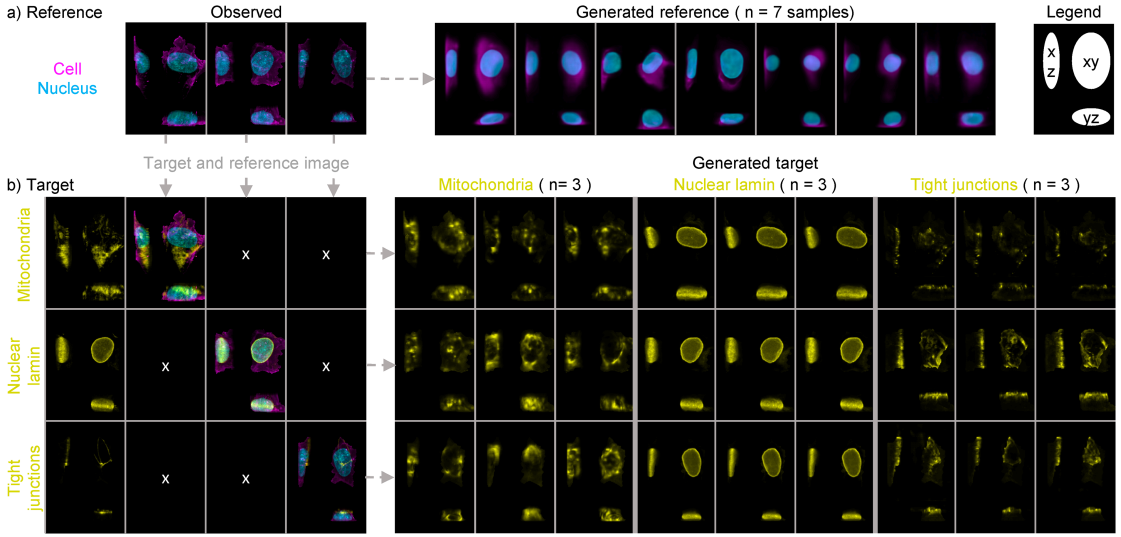
A data visualization

Computational Modeling



Kangassalo, L. et al (2020)
https://doi.org/10.1038/s41598-020-71287-1





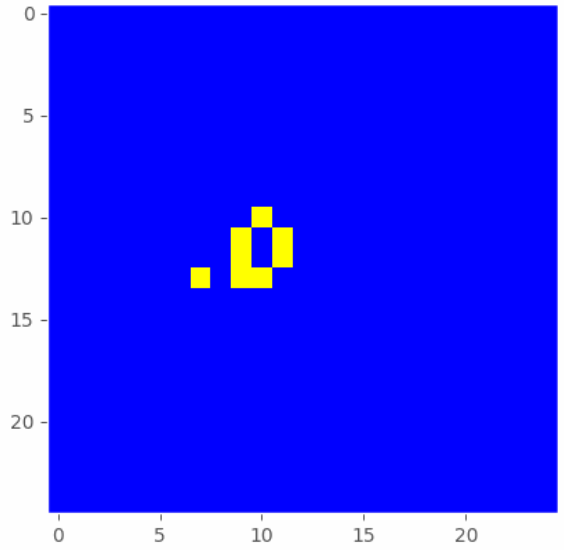
Donovan-Maiye RM et al. (2022) A deep generative model of 3D single-cell organization. PLOS Computational Biology 18(1): e1009155. https://doi.org/10.1371/journal.pcbi.1009155

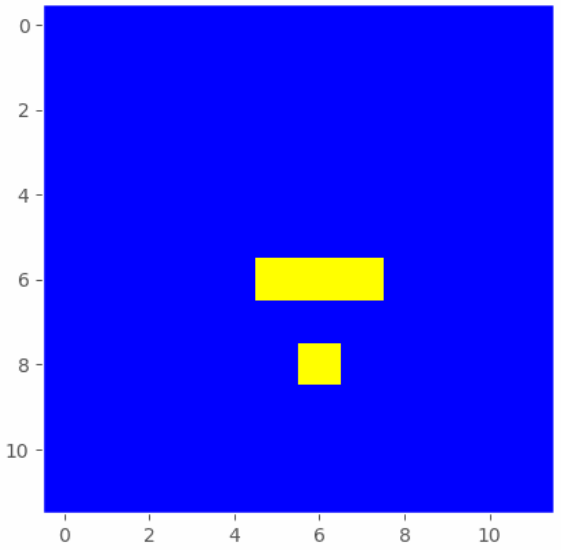
Cellular Automata

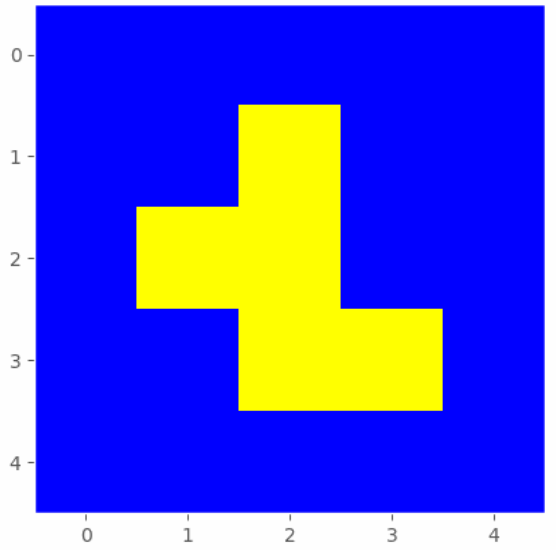
A computational model consisting of a **matrix** and a **set of rules**

Each iteration, the matrix changes based on its current state

There are a number of emergent behaviors that can be discovered!







Part 1: Coding the Cellular Automata

CellAutomata

Learning Objectives:

1. Practice data parsing and multi-dimensional lists in Python
2. Introduce data visualization using matplotlib
3. Explore a time-sensitive computational model (Cellular Automata)
4. Modify and use an existing code base to perform data science

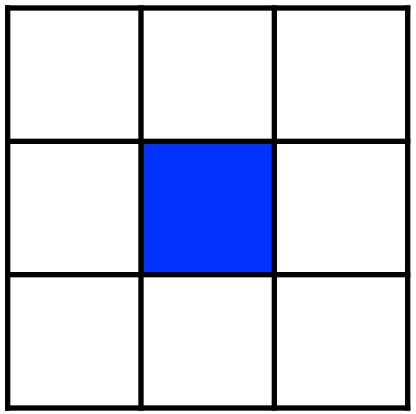
Name	Type	Description
`emptyMatrix`	function	Given two integers, build a row x col matrix.
`getSize`	function	Given a matrix, return a list containing [row, col]
`importValues`	function	Given a matrix and a filename, insert the values into matrix and return it.
`update`	function	Given a matrix, return the corresponding matrix for the next timestep

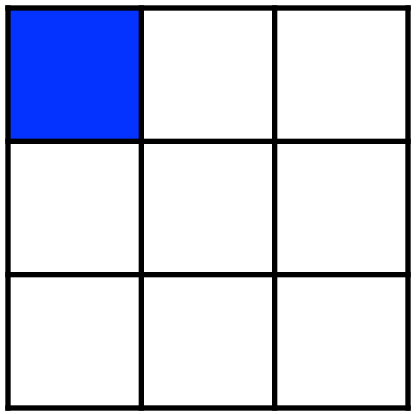
Open workspace

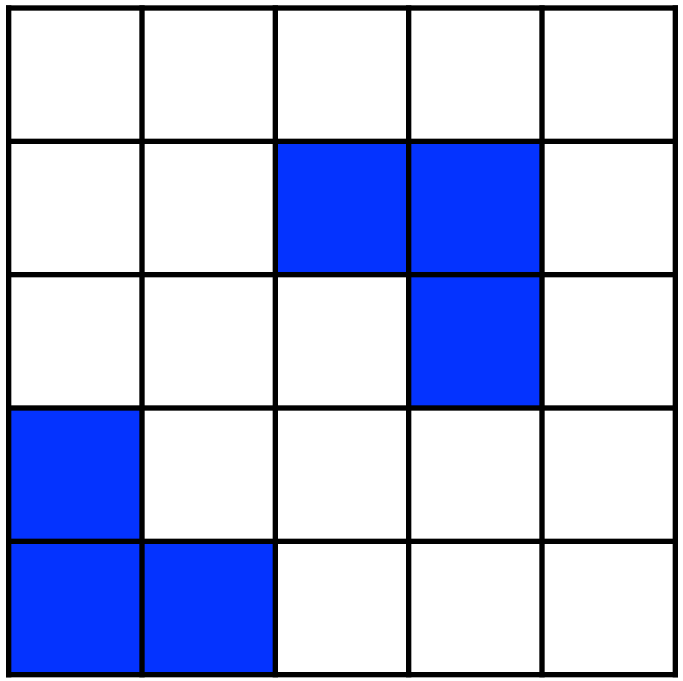
Save & Grade | Save only | New variant

Conway's Game of Life Rules

A 'cell' is either alive or dead and has at most 8 neighbors around it

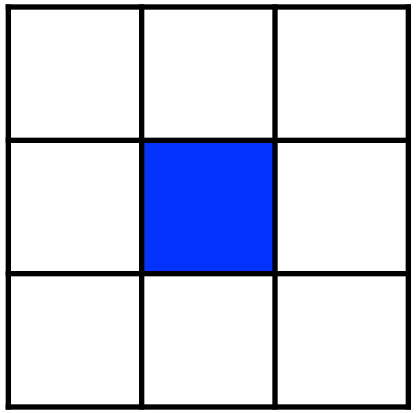


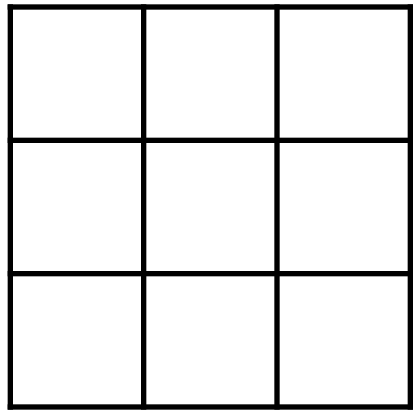


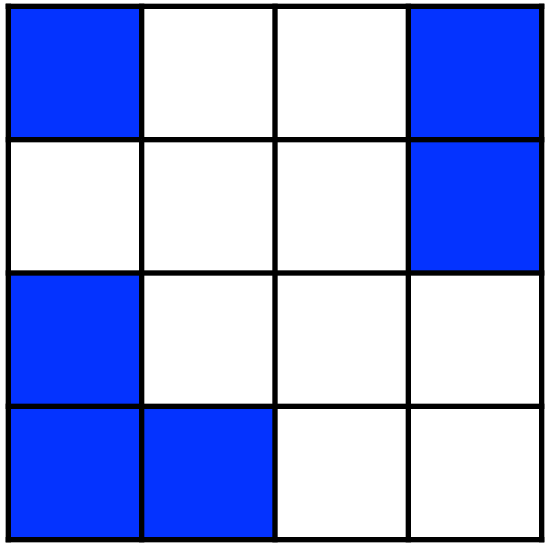


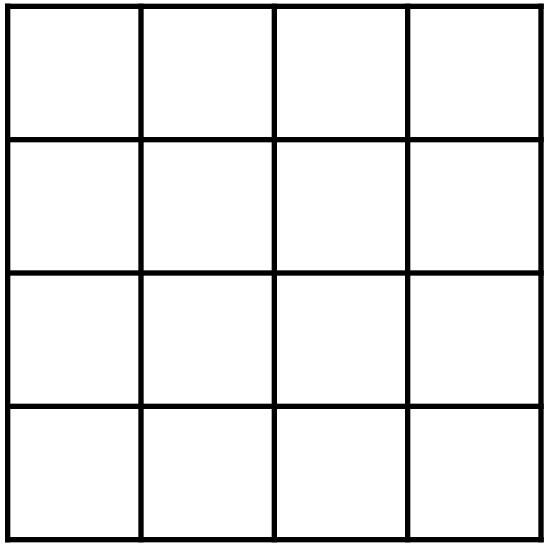
Conway's Game of Life Rules

1. Any live cell with fewer than two live neighbours dies.



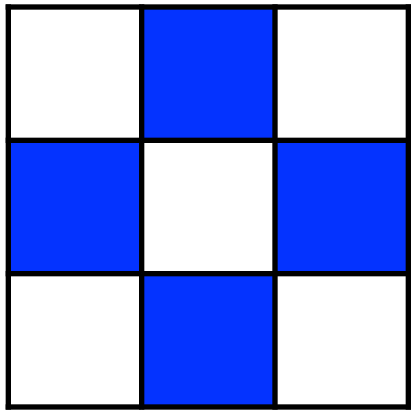


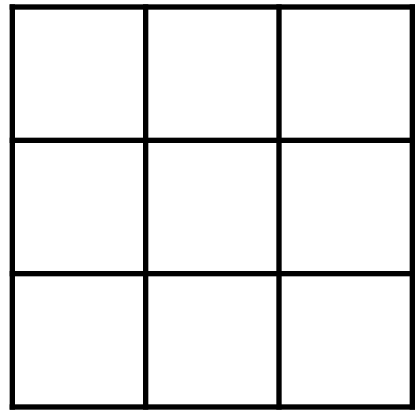


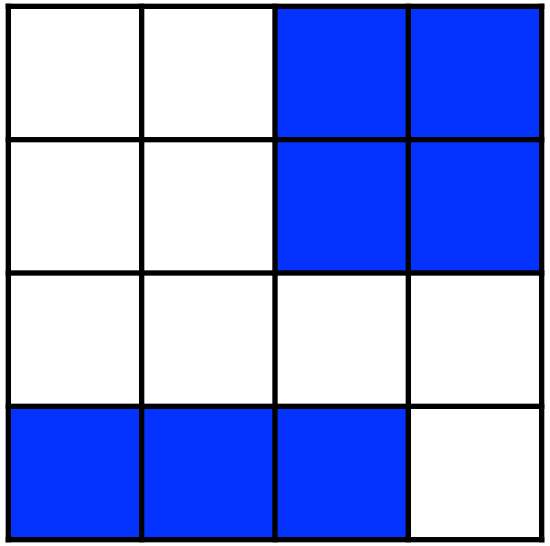


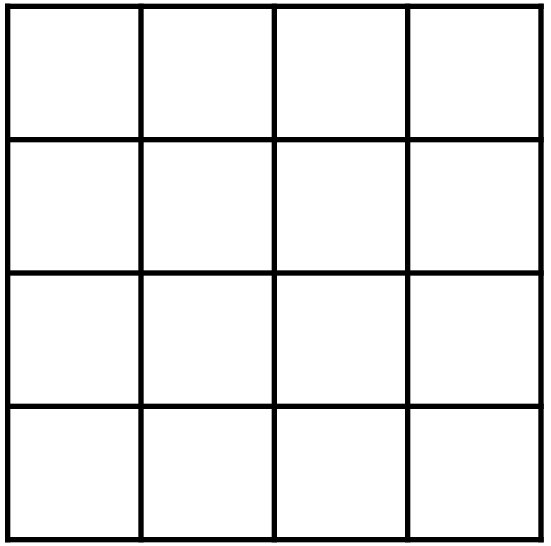
Conway's Game of Life Rules

2. Any live cell with two or three live neighbors lives.



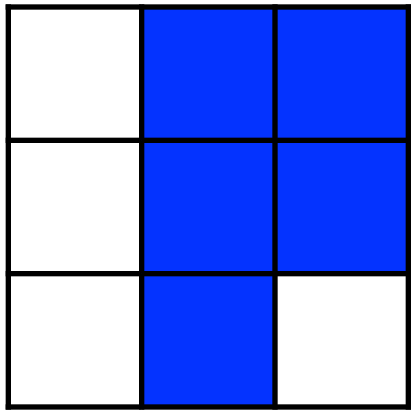


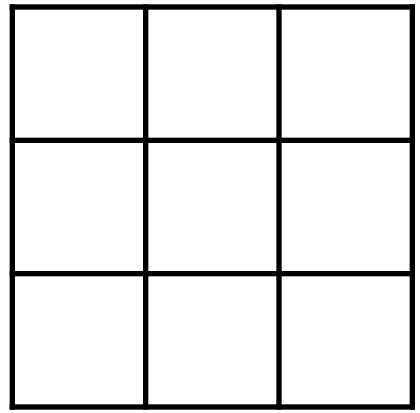


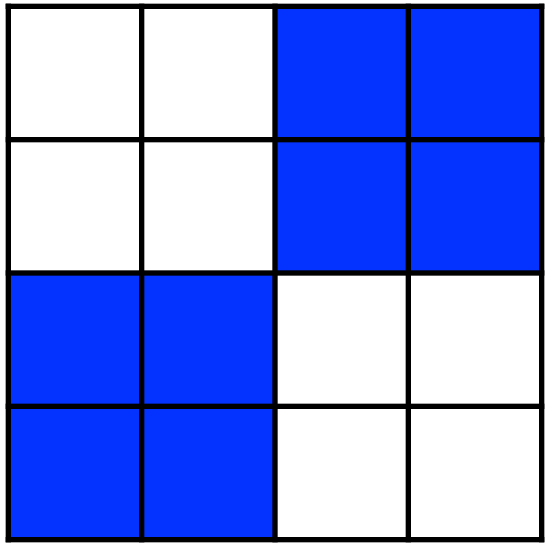


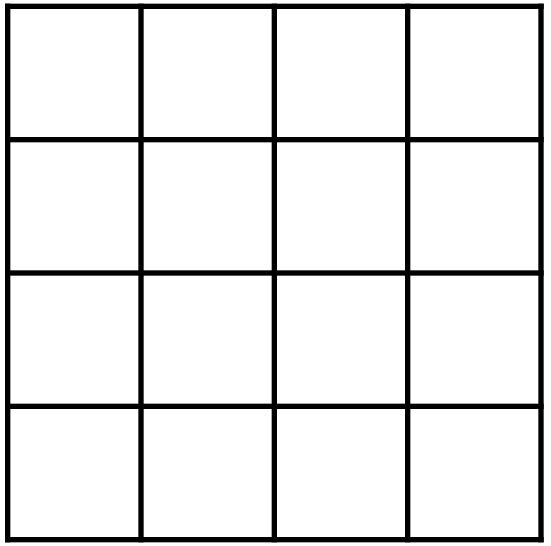
Conway's Game of Life Rules

3. Any live cell with more than three live neighbours dies



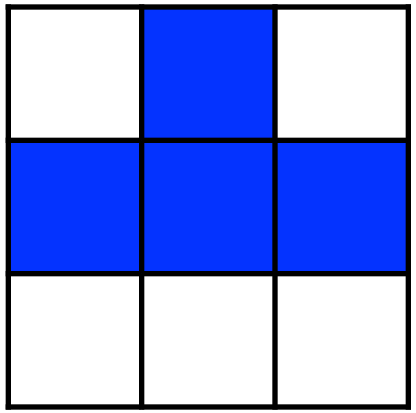


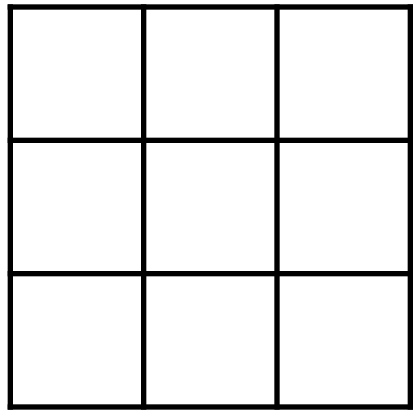


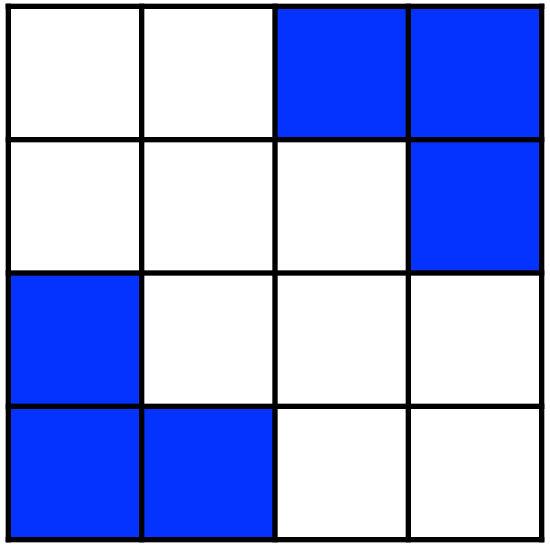


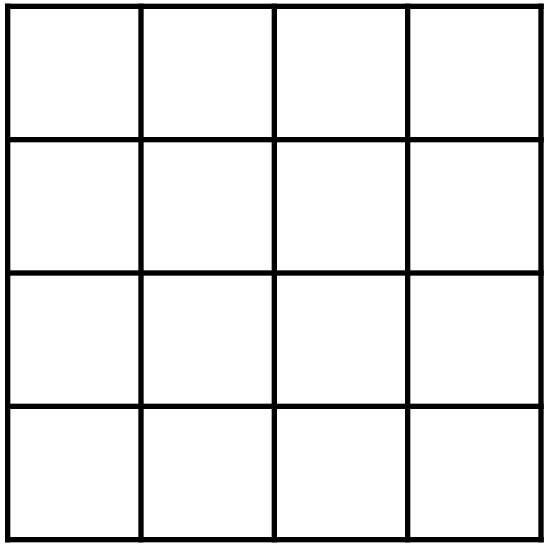
Conway's Game of Life Rules

4. Any dead cells with exactly three live neighbours becomes a live cell.





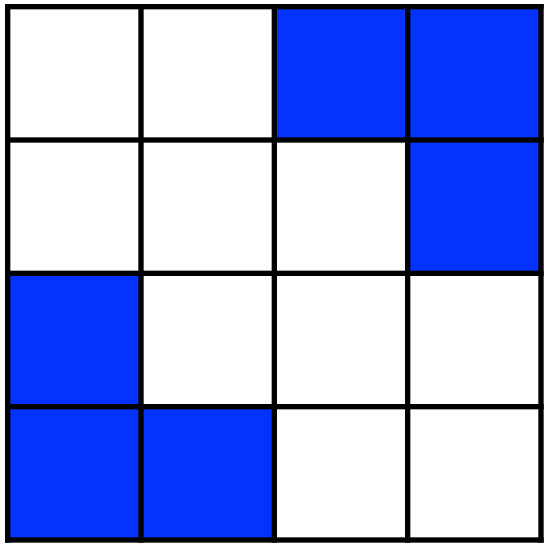


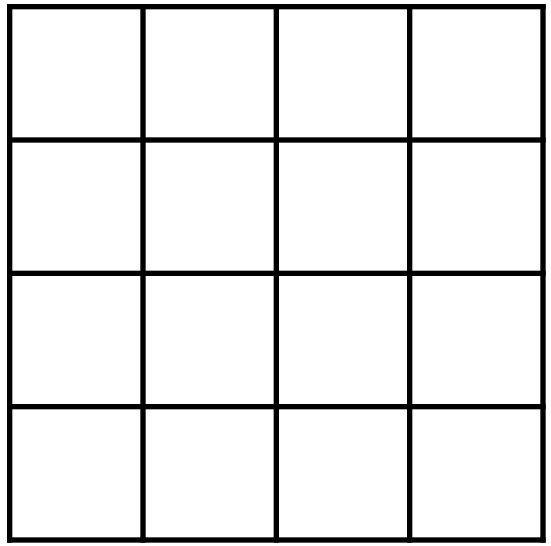


Conway's Game of Life

A fun demo version of the game: https://playgameoflife.com/

Note: Every cell is updated **at the same time**





Part 2: Data Science!

1. **Dataset Link:** Where did you find your dataset? Provide a direct link to its source but do not give the download link directly. (The website you link should ideally include some information about the dataset you have selected).

Github Code Link

2. **Dataset Format:** What is the format of your input file? At minimum, you *must* give the total number of rows in your dataset and what each column means even if it is as trivial as an (x,y) coordinate pair. If you have multiple input files (such as different states on a chess game) you should describe this too.

Normal Normal

Your answer here

3. **Dataset Visualization:** What will your output look like? At minimum, this should describe what a row / column means and what each color in your visualization means.

Normal Normal

Your answer here

Github Link

Please provide a direct link to the code you used to produce your visualization and the visualization itself.

Github Code Link

Github Visualization Link

Part 2: Modifying code

```
def updateFig(frameNum, img, matrix):
    if (frameNum == 0):
        return img

    nm = update(matrix)

    matrix[:] = nm[:]

    img.set_data(nm)

    return img
```

```
def animate(matrix, steps, outname):
    matrix = np.array(matrix)

    cmap = colors.ListedColormap(['blue', 'yellow', 'red'])
    bounds = [0, 1, 2, 3]
    norm = colors.BoundaryNorm(bounds, cmap.N)

    fig, ax = plt.subplots()
    plt.grid(False)
    img = ax.imshow(matrix, cmap=cmap, norm=norm)
    ani = animation.FuncAnimation(fig, updateFig, fargs=(img, matrix), frames=steps)


    ani.save(outname, fps=2)
```

Github Tutorial

Github is a version control and collaboration platform

For our purposes: it's a way to save to the cloud in a secure way

```
git clone <REPO-LINK>
```

```
git add <FILENAME>
```

```
git commit -m <MESSAGE>
```

```
git push
```

List Implementations

1. Linked List

2. Array list

List Implementations

1. Linked List

2. Array list