Algorithms and Data Structures for Data Science

Multi-dimensional Lists

CS 277
Brad Solomon

January 30, 2023
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

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

def makeMatrix():

    M[2] =

    M[1][0] =

    M[0][2] =
import copy
orig = [[1,2,3], [4, 5, 6]]
not_copy = orig
s_copy = orig.copy()
d_copy = copy.deepcopy(orig)
orig[1][1]=3
not_copy[0][0]=2
s_copy[0][2]=10
d_copy[1][2]=15
print(orig)
print(not_copy)
print(s_copy)
print(d_copy)
Multi-dimensional lists

```python
import copy
orig = [[1, 2, 3], [4, 5, 6]]
not_copy = orig
s_copy = orig.copy()
d_copy = copy.deepcopy(orig)
orig[1][1] = 3
not_copy[0][0] = 2
s_copy[0][2] = 10
d_copy[1][2] = 15
print(orig)
print(not_copy)
print(s_copy)
print(d_copy)
```

Orig

Shallow

Deep
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).

```python
myList = [1, 2, 3, 4, 5]
print(myList)
print(len(myList))
print(myList[2])
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).

```python
myList = [1, 2, 3, 4, 5]
print(myList)
print(len(myList))
print(myList[2])
clist = myList.copy()

tuple = (1, 2, 3, 4, 5)
print(tuple)
print(len(tuple))
print(tuple[2])
ctuple = tuple

np = np.array([1, 2, 3, 4, 5])
print(np)
print(len(np))
print(np[2])
cnp=np.copy()
```
Python Tuple

In Python, a tuple is an immutable list.

```python
myTuple = ([1,2,3], "ABC")
print(myTuple[1])
myTuple[0] = [3, 4, 5]
myTuple[0].append("oops")
print(myTuple)
```
Python Numpy List

A numpy list in Python is designed for scientific computing

```python
import numpy as np

r1 = [7, 8, 9]
r2 = [4, 5, 6]
r3 = [1, 2, 3]
plist = [r1, r2, r3]

print(plist)

myNP = np.array(plist)

print(myNP)

myNP = myNP.reshape(1, 9)

print(myNP)

odd = myNP % 2

print(odd)
```
import pandas

pd.read_table('myFile.csv')

pd.read_table('myFile.csv', delimiter=',')

pd.read_table('myFile.csv', delimiter=',', usecols=['Netid', 'Grade'])
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

### 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

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>emptyMatrix</td>
<td>function</td>
<td>Given two integers, build a row x col matrix.</td>
</tr>
<tr>
<td>getSize</td>
<td>function</td>
<td>Given a matrix, return a list containing [row, col]</td>
</tr>
<tr>
<td>importValues</td>
<td>function</td>
<td>Given a matrix and a filename, insert the values into matrix and return it.</td>
</tr>
<tr>
<td>update</td>
<td>function</td>
<td>Given a matrix, return the corresponding matrix for the next timestep</td>
</tr>
</tbody>
</table>

[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).

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.

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.
Part 2: Modifying code

```python
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