DS 2001: Introduction to Programming with Data / Lab Practicum
Prof. Rachlin / Northeastern University

Complex Systems

Overview

Today we explore the world of complex systems. In nature and in society, interesting patterns emerge from the repeated application of even the simplest of rules. Such patterns are known as *emergent phenomena.* Let's play a little simulation game using *lists within lists* that we learned about in class. Recall that to fetch or update a value inside a list of lists, we use *double indexing*. For example, the following list, L, has three sub-lists that I have arrayed as a 3x4 grid, but it is just a list with three sub-lists each of length 4.

```
L = [[1, 2, 3, 4],
     [5, 6, 7, 8],
     [9, 10, 11, 12]]
```

If I wanted to change the "7" to a 99, I would say:

```
L[1][2] = 99
```

because I am updating the 2nd list (at index 1) and the 3rd element in that list at index 2.

In this exercise we are going to carry out a little simulation. Imagine 201 students all sitting together in one long row day after day. On day 0, only one student sitting in the very middle of the row is wearing a mask. We'll use the value **1** to represent wearing a mask while **0** represents not wearing a mask. Our array is thus initialized as follows:

```
L[0][100] = 1.
```

We are assuming that our 201 students always sit in the same seat each day, and *of course* they always come to class! The NEXT day students come to class either wearing a mask or not wearing a mask according to the following rule: *If yesterday a student was sitting next to exactly ONE student wearing a mask, then they will wear a mask today. Otherwise, they will NOT wear a mask today.* We know who is wearing a mask on day 0. Calculate who wears a mask on EACH day for the next year.

Step by Step Instructions

Step 1. Write a function called **initialize_array(m, n, init_value=0)** that creates an initial m x n array of values, with all values initialized to 0 by default. Here, m is the number of rows (sub-lists) and n is the number of columns (elements in each sub-list). In this lab, m represents the number of days, and n represents the number of students. We'll run this simulation for m=365 days and n=201 students. (Download lol.py from the website schedule if you're stuck. I showed how to do this in my section.)

Step 2. Each student decides to wear a mask on the *next* day of class according to the rule described above: We know what happened on day 0 – only one student in the middle was wearing a mask. Now determine who wears a mask on days 1…364. Take special care to correctly handle the students at the ends of the row who only ever sit next to one person. Implement a function called **apply_rule(L)** that determines which students are wearing a mask on each subsequent day. (Why don't we need to pass the number of rows and columns of L to the function as parameters?)

Step 3. Plot your results by converting your two-dimensional array (list-of-lists) into an image. A dedicated function called **array_to_image(L)** would be appropriate here. First decide on the dimensions and resolution of your image. You'll want to experiment with different image dimensions and resolutions. I found that these settings worked well:

```
plt.figure(figsize=(4, 5), dpi = 200)
```

The figsize parameter defines the width x height dimensions of the image in inches, and dpi is the number of dots (pixels) per inch. The higher the dpi the better the resolution.

Next, convert your list-of-lists into an image:

```
plt.imshow(L)
```
or
```
plt.imshow(L, cmap='binary')
```

To save your image as a .png you can use the command:

```
plt.savefig('masks.png')
```

You may want to add function parameters to control how your image is rendered.

What to submit

Submit your code (.py) your plot (.png) and a comment in your header about how amazingly cool your result is – were you surprised by the emergent pattern?