**ENGR 103 Studio 7: Lists**

**Goals:**

In this studio, we will be finding specific values within a large nested list. It will be an exercise in **indexing**.

Python can store a bunch of entities in a list. They can be numbers, strings, objects, functions, or even more lists! That’s right – you can have a list where each entry is another list, and even each of those lists may be made of further lists. The practice is called **nesting** lists. It sounds overcomplicated, but it’s a structure that shows up often in real life. Imagine that a professor teaches several classes, each with several students, each with several assignment grades. They may store all of the grades in one big nested list and access information in a manner like this: “In my 3rd course, for the 10th student, tell me the 5th assignment grade.” The list, called “Grades”, could be indexed as Grades[Course][Student][Assignment], and this specific grade would be indexed as Grades[3][10][5].

In this studio, you’ll be provided with a three-level list (called the Grand List) and your task will be to access its many elements using GrandList[X][Y][Frame]. Upon opening and running the code, you should see an animation containing the frames shown below in Figure 1.

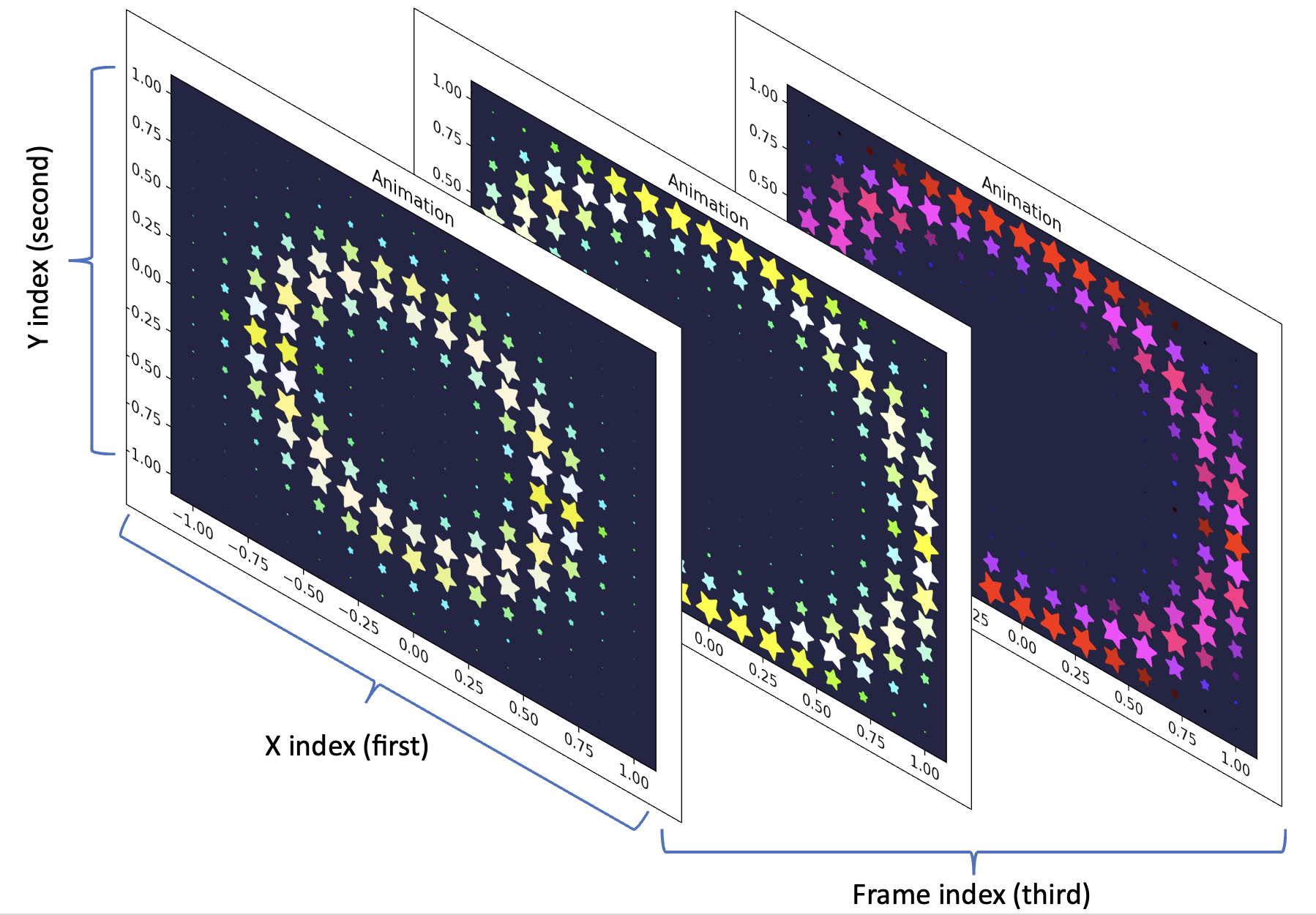


Figure 1. Stills from animation of 3-level list. An individual Node exists at every X index and Y index, for every Frame of the animation. Nodes are indexed as GrandList[X][Y][Frame].

**Part 0: Getting Familiar with the Code**

**Make it look nice:**

Download the Studio 6 code from Canvas and open it in PyCharm. As usual, your first step should be to make the code look readable – press the arrows next to each function definition line to collapse each function into one line. When you are done, the entire code should fit on your screen without scrolling.

Later on, as you expand the functions to observe or edit them, remember to re-collapse them when you are finished. It is easy to get lost in long pieces of code, and the seconds of scrolling add up.

**Make sure it’s working:**

Run the code once to see a demonstration and make sure that everything can run on your machine. You will have to download the packages numpy and matplotlib. Make sure that the animation plays at reasonable speed. If it takes too long (~ > 30s) find the variables in the main function called Num\_Picels\_X and Num\_Pixels\_Y and lower them until the animation plays smoothly.

When the animation finishes, click on the plot window. It should advance through a few more plots that show results from the later parts of the studio, but we haven’t yet written the code for those parts – they will show better results soon.

**Read through and understand:**

Go through each function and try to understand what it does. It is not necessary to understand each line – just read the function definitions and get a general idea of what is accomplished in each.

Try to get an idea of what happens in the function Build\_Lists, and what the finally returned List looks like. In this studio, you will only need to edit the functions in the class called Studio\_Functions.

**Deliverable:**

1. Describe what is stored in the **lowest** level of the List built in Build\_Lists(). For instance, if I set Entry = GrandList[X][Y][Frame], is Entry a number, string, object, function, or something else? (hint: it is the **highest** level of the list which contains more lists – the **lowest** level contains single entities, like assignment grades in our earlier example)
2. Describe what the class Node represents. What information does it store?
3. If we set Num\_Pixels\_X = Nx, Num\_Pixels\_Y=Ny, and Num\_Frames=F, then how many Nodes are stored in the GrandList?
4. How would you find the Color of the node at X\_Index=2, Y\_Index=3, Frame\_Index=4 ?

**Part 1: Find the frame at which a node has its maximum value**

**Find the class “Studio\_Functions” and its function “Find\_Frame\_With\_MaxVal”**

Throughout this studio, the only sections you need to edit are within the Studio\_Functions class. The function Find\_Frame\_With\_MaxVal is the first within this class.

The function looks at a specific location, defined by inputs X\_Index and Y\_Index, and finds the frame at which the node at that location shows a maximum value.

**Edit “Find\_Frame\_With\_MaxVal”**

Find the **one** line to edit in this function. It is embedded within a loop that steps through every frame. The purpose of the command is to identify the value of the node at the location defined by the inputs and the frame defined by the loop. When you finish editing the line, it should read something like:

Current\_Val = GrandList[Index][Index][Index].Value

**Use “Find\_Frame\_With\_MaxVal”**

Make sure that, in the main function, variables are set as follows:

Num\_Pixels\_X = 15

Num\_Pixels\_Y = 15

Num\_Frames = 80

X\_Index\_P1P2 = 5

Y\_Index\_P1P2 = 12

Control\_Func = H.Control\_Function\_1

Run the code again and, once the animation finishes, advance to the next plot, titled “Part 1…”. Make sure (qualitatively) that the frame does appear to capture the maximum value at that location.

**Note** – the size of each star is proportional to the Value of the corresponding node.

**Deliverable:**

1. A screenshot of your plot titled “Part 1…”
2. At which frame is this node’s value maximum? (Hint: check the plot’s title)

**Part 2: Plot the Value at a given location over time.**

**Find the class “Studio\_Functions” and its function “Find\_Value\_vs\_Time”**

The function looks at a specific location, defined by inputs X\_Index and Y\_Index, and returns the value at this point, at every frame.

**Edit “Find\_Value\_vs\_Time”**

Find the **two** lines to edit in this function, and read the surrounding comments to get an idea of what the lines should do. They should obtain the Time and Value at the node defined by X\_Index, Y\_Index, and Frame. The indexing should be similar to what you wrote in Part 1.

**Use “Find\_Value\_vs\_Time”**

Make sure that, in the main function, variables are set as follows:

Num\_Pixels\_X = 15

Num\_Pixels\_Y = 15

Num\_Frames = 80

X\_Index\_P1P2 = 5

Y\_Index\_P1P2 = 12

Control\_Func = H.Control\_Function\_1

Run the code again and, once the animation finishes, advance to the next plot, titled “Part 2…”. Make sure (qualitatively) that the plot appears to track the value of our node of interest.

**Deliverable:**

1. A screenshot of your plot titled “Part 2…” after you’ve edited Find\_Value\_vs\_Time.
2. At what time is this node’s value maximum? (Hint: check the x axis).

**Part 3: Plot the total of all Values in a frame over time.**

**Find the class “Studio\_Functions” and its function “Find\_Total\_vs\_Time”**

The function looks at every location at every frame, and returns the total of all node Values at each frame.

**Edit “Find\_Total\_vs\_Time”**

Find the **one** line to edit in this function, and read the surrounding comments to get an idea of what the lines should do. It should obtain the Value at the node defined by X, Y, and Frame, and **add it to a running total**. The indexing should be similar to what you wrote in Part 1 and 2, and the total should increase with every iteration through the loop.

**Use “Find\_Total\_vs\_Time”**

Make sure that, in the main function, variables are set as follows:

Num\_Pixels\_X = 15

Num\_Pixels\_Y = 15

Num\_Frames = 80

X\_Index\_P1P2 = 5

Y\_Index\_P1P2 = 12

Control\_Func = H.Control\_Function\_1

Run the code again and, once the animation finishes, advance to the next plot, titled “Part 3…”. Make sure (qualitatively) that the plot appears to track the total value over time.

**Deliverable:**

1. A screenshot of your plot titled “Part 3…”
2. At what time is the total value of all nodes maximum? (Hint: check the x axis).

**Part 4: Find the brightest star.**

**Find the class “Studio\_Functions” and its function “Find\_Brightest”**

The function looks at every location at every frame, and returns the node with the brightest color of them all. In this case, “Brightest” means the same thing as “Whitest.”

**Edit “Find\_Brightest”**

Find the **two** lines to edit in this function, and read the surrounding comments to get an idea of what the lines should do. The first should obtain the brightness value of the start at the given X, Y, and Frame. You can access the node’s Color the same way you access its Value, but keep in mind that the Color is a 3-element list of RGB values. The brightness is the average of those three values. The second line to edit should assign the node at this X, Y, and Frame to the variable Brightest.

**Use “Find\_Brightest”**

Make sure that, in the main function, variables are set as follows:

Num\_Pixels\_X = 15

Num\_Pixels\_Y = 15

Num\_Frames = 80

X\_Index\_P1P2 = 5

Y\_Index\_P1P2 = 12

Control\_Func = H.Control\_Function\_1

Run the code again and, once the animation finishes, advance to the next plot, titled “Part 4…”. Make sure (qualitatively) that the plot appears to highlight a very bright/white star.

**Deliverable:**

1. A screenshot of your plot titled “Part 4…”
2. What is the brightness of the brightest star? (Hint: check plot title)

**Part 5: Make fun art!**

**Find the line in the main function that defines which control function to use**

The default is function #1, but you can change it to #2 or #2 and see what happens.

**Find the class “Helpers” and its functions “Control\_Function\_x”**

These functions control the value of the nodes at each location and time. Write your own! They should yield values that change from 0 to 1 (**strictly** from 0 to 1).

**Inspiration:**

