Tutorial04-nestedloops+nestedlists

June 6, 2022

Due Saturday, June 11 at 11:59 pm.

1 Make a new notebook

Create a new jupyter notebook and edit its name from "`Untitled`" to "`tutorial04_{identikey}`", replacing "`{identikey}`" with your identikey (get rid of the squiggly brackets).

In this tutorial, you're going to calculate the positions of the four Galilean moons of Jupiter at different times, and visualize how these moons move. Elements of this tutorial (and previous ones) may be helpful for Homework D.

2 think+type: moons of Jupiter (2 pts)

In this ***think+type****, you're defining four lists that you're going to need for the rest of the tutorial. Please check that you've defined your variables accurately.*

```
[ ]: # the names of the four Galilean moons
     moons = ['Io', 'Europa', 'Ganymede', 'Callisto']

     # the orbital periods of those moons, in days
     periods = [1.77, 3.55, 7.15, 16.69]

     # the rough size of the orbits, in arcseconds
     orbitsizes = [111, 178, 282, 499]

     # the initial orbital phases of the moons, in radians
     initialphases = [2.726, 3.685, 5.971, 1.669]
```

```
[ ]: print(moons)
     print(periods)
     print(orbitsizes)
     print(initialphases)
```

Look carefully at the lists you've created, and make sure they're accurate.

3 think+type: printing lists (1 pt)

*In this **think+type**, you will use a loop to print these lists in a somewhat more readable fashion.*

```
[ ]: for i in range(4):
         print(f'{moons[i]:10} {periods[i]:5.2f} {orbitsizes[i]:4.0f}␣
      ↪{initialphases[i]:6.3f}')
```

4 think+code: being flexible (8 pts)

*In this **think+code** you will think about how to generalize the loop you just wrote to lists of an arbitrary size.*

All of the lists we've defined so far have a length of 4. We might later change the lists `moons`, `periods`, `orbitsizes`, and `initialphases` so that they contain more than 4 values; for example, they might contain all of Jupiter's 79 known moons. Let's look at the block of code you just typed for printing the values in the lists. It would be annoying to have to change this simple block of code every time I changed the lengths of my lists.

Copy and paste the above code (from *think+type: printing lists*) into a new code cell. **Modify it so that it would print all entries in these lists, no matter how many moons we had defined.** Run it to make sure it workds! It should still print just the four entries we've defined, but the number 4 should not appear anywhere in your code. *(Hint: Is there a way to determine the length of a list?)*

5 think+type: `zip()` (1 pt)

*This **think+type**, you'll see a quick example of how to "zip" through multiple lists in a single for loop.*

The `zip` function is a shortcut that allows your for loops to access the corresponding elements from parallel lists, one by one.

```
[ ]: for m, p, o, i in zip(moons, periods, orbitsizes, initialphases):
         print(f'{m:10} {p:5.2f} {o:4.0f} {i:6.3f}')
```

6 think+pair+code: where are the moons? (8 pts)

*In this **think+pair+code** section, please work together with your neighbor(s). It can sometimes be way easier to solve coding problems if you talk through them with someone else!*

With the lists of moon data you have already defined, we have enough information to figure out approximately where the moons will be, relative to Jupiter, at any particular time. The moons revolve around Jupiter in almost circular orbits. If we assume the orbital plane of these moons is almost edge on as seen from Earth (it is), then the moons will appear to us to move back and forth along a line. The equation to calculate the projected position $x_i(t)$ of moon i at time t can be written as

$$x_i(t) = A_i \sin\left(\frac{2\pi t}{P_i} + \theta_i\right)$$

where A_i is the orbit size for moon i (`orbitsizes`, in arcseconds), P_i is the orbital period of moon i (`periods`, in days), θ_i is the initial phase of the moon in its orbit (`initialphases`, in radians).

Please write code that uses a `for` loop to build up a **list** of the apparent positions of the four moons at the time $t = 2.3$ days, or using our equation above, $x_i(t = 2.3)$. Here are the first few lines of what your code might look like...

```
[ ]: # we need to import the variable pi and function sin
     from math import pi, sin

     # let's define the time at which we want to know the moons' positions
     t = 2.3
```

Write the rest of your code so it makes use of the variable called `t`, because we're going to rely on changing that variable in a later part of the tutorial. You'll want to write this code in a separate `Code` cell from the one above so that you can cut-and-paste it into a later part of the tutorial.

```
[ ]: # create an empty list of moon positions for this time
     positionsAtThisTime = []

     # loop through all the moons
     for # ... (you can take it from here!)
```

Please check your code to make sure it's working by printing out your `positionsAtThisTime` list *outside* of your loop. Your calculated list of positions at $t = 2.3$ should be approximately `[-110.4, 177.1, 279.3, 284.5]` but likely more precise than that.

```
[ ]: print(positionsAtThisTime)
```

7 think+type: define a helpful function (4 pts)

*This **think+type** gives you access to a tool that will be useful for the rest of your tutorial.*

In *Siderius Nuncius*, Galileo made lots of drawings of the positions of the moons of Jupiter at different times. In the rest of this tutorial, you're going to use code to make your own versions of his moon drawings. To help with that, Prof. Berta-Thompson wrote a cool function to help visualize your calculations of the positions of the Galilean moons of Jupiter.

For this tutorial, you will need to get that code into your notebook so you can use it.

If you are on `JupyterHub` (i.e. accessing Jupyter through https://scorpius.colorado.edu), the module is in one of my directories on Scorpius and you need to set up your path (more on paths later) to get to it. If you are on Anaconda (i.e. installed Python on your own computer) you need to download the file `galileo.py` (available under this tutorial on Canvas) and place it in an appropriate place on your computer. Follow the correct sequence below depending on your setup:

```
[ ]: #JupyterHub users:
     import sys
     sys.path.append('/home/hama2717/astr2600/shared/')

     #Anaconda users:
     #    Copy galileo.py from Canvas into
     #    the same directory this tutorial is located
```

```
[ ]: # Everyone
     from galileo import drawMoons
```

If you are working from anaconda on your own computer (or the above command produces an error), download the file galileo.py from from `Canvas` (under Modules|Tutorials) and open it using a text editor. Select all the text in that file, and copy it into a new Code cell in your jupyter notebook. Run that cell; this will define the function, so that you can use it in later cells.

```
[ ]: # read the documentation for this function
     help(drawMoons)
```

We'll talk about how to write your own functions in the next tutorial. For now, let's use that `drawMoons` function to show your calculated positions of the moons. Let's first test it out to see what it does:

```
[ ]: # try it with some made-up positions
     drawMoons(moons, [50, 100, 200, 400], labels=True)
```

The big dot is Jupiter!

```
[ ]: # try it with our calculated positions
     drawMoons(moons, positionsAtThisTime)
```

8 think+type+code: calculate the moon positions over time (8 pts)

We'd like to estimate the moons' positions at multiple times. Galileo observed whenever he could (so sometimes clouds got in his way), but we'll create an evenly-spaced grid of times (days) at which we'd like to calculate the positions. Then we'll visualize the moons' positions, just like Galileo did in his drawings!

We need to calculate the positions of ***each*** of the moons at ***each*** of the times we specified in our `times` list. To do this, we need to loop over both the time points and the elements of the moon lists. The following block of code will do this. Please note, there is a big chunk of commands from the "**think+pair+code: where are the moons?**" section that you can simply copy and paste your results from above. You will need to indent that pasted block of code so that it runs inside the outer loop (*hint: highlight the entire code block and hit the Tab key*).

Since you have nested loops, indentation is **VERY** important for defining which loop you are in.

Notice the last two commands are indented only once so they belong *only* to the first `for` loop.

Take a moment to admire your results; imagine how Galileo must have felt when he first started tracking Jupiter's moons!

[ ]:
```python
# create evenly spaced observations from 0 to 49 days
times = range(50)

# create an empty list to store all moon positions
positions = []

# loop over all times
for t in times:

    # *** COPY+PASTE YOUR where are the moons loop FROM ABOVE ***
    # create an empty list of moon positions for this time
    positionsAtThisTime = []

    # loop over all the moons
    for ... #insert your for loop code here
        # Be sure to indent your code inside this second for loop
        # You can easily indent and entire block of code by highlighting
        # all of it and hitting tab.

    # *** END COPY+PASTE FROM ABOVE ***

    # Return to only one level of indentation
    # append the positions for this time point to our big list
    positions.append(positionsAtThisTime)

    # make a drawing of where the moons are located
    drawMoons(moons, positionsAtThisTime)
```

9 think+write: lists within a list (8 pts)

*In this **think+write**, you need to think about the lists you created in the previous block of code. Please create a "Markdown/Raw NBConvert" cell, add the short title "lists within a list:", and fill in your answer there.*

1. At the end of running that last block of code, what is the length of the list positions?
2. Does this length correspond to the number of time points or the number of moons?
3. The expression `positions[0]` is itself a list. What does this list represent?
4. The expression `positions[1]` is also a list. How is this different from `positions[0]`?

10 (bonus) think+fun: Tricks Galileo wished he had

You can skip this final section if you're out of time.

If you have extra time (or later) try rerunning drawMoons with the labels and/or color options turned on. This will label the respective moons and/or print each moon in a different color. To turn these options on, run the drawMoons command with `labels=True` and/or `color=True` keywords in the function call. *(We'll learn about optional keywords later this semester. Since they are optional, you can include one or the other or both. Try it out!)*

e.g. `drawMoons(moons, positionsAtThisTime, labels=True, color=True)`

11 save+upload

Be sure to save the final version of your notebook. To submit, click "File|Download As >|HTML (html)" inside jupyter notebook to convert your notebook into an HTML web page file. It should have a name like `tutorial04_{identikey}.html`. Please upload this HTML file as your submission to "Tutorial 05" on Canvas.