L1

August 31, 2023

1 ASTR 1040 Recitation 1: Introduction to Jupyter notebooks

1. What is a Jupyter notebook? Jupyter notebooks are a teaching tool that allow you to interface typing, pretty mathematical expressions, and coding all in one place. You will be using Jupyter notebooks to do your homeworks in this class, writing answers in with a combination of text + using Python as a fancy calculator.

1. What is Python?

Python is a *high level interpreted* programming language. What does this mean? Python's higher level syntax means that Python code is highly readable and much easier to intuitively understand than many lower-level langauges–this makes it a great language to learn and start your programming journey in! In Python you don't have to use excessive amounts of brackets & or deal with many of the other syntax frustrations lower level langauges often introduce. Because Python is an interpreted langauge, this means that it's easy to modify programs and run them in real time, but the downside is this makes it significantly slower than a lower level compiled langauge like C or Fortran.

The best of both worlds may lie in up and coming languages like Julia, which boast C-like performance but with higher-level Python syntax. Why learn Python then instead of something like Julia? It's still a little easier than Julia, but more importantly it is stabler and used more broadly across fields – making Python perhaps ***the*** most useful language you can learn today, especially for research in astronomy. That being said I personally prefer Julia to Python and am happy to teach you that as well. :)

2. What can I use Python for?

Nearly anything! It's used for scientific research at universities and national labs (including NASA!) but it's also used extensively by web developers, game designers, and industry.

3. What will we be doing?

We are largely going to use Python as a fancy calculator in this class, mostly to prevent pesky unit mistakes. I promise we are doing this to actually try to make your lives *easier*, not harder. No prior understanding of coding/science required–we will learn together as we go!

1.1 Lesson 1: Starting out

You are running Python out of a Jupyter Notebook right now. I like them because they are easy to mesh text (like this) and code (like you will see below). Let's start by making sure you know how to open Jupyter notebook and saving your work to your personal folder. You can make Python programs in any text editor and run them from the command line, something we might explore later, but for now we will work mostly out of notebooks for their ease of use.

Jupyter notebooks have two kinds of cells–code and markdown. This is a markdown cell (which makes text formatting easy). You can change the cell type with the dropdown menu at the top of the page. Markdown cells are useful as notes to accompany your code/explain things.

1.1.1 Exercise 1: make your first markdown note

Change the cell type below from code to markdown, and write "Hello world!". Run the cell by clicking the run button or by pushing Shift+Enter. Next, modify the cell to make italicize or bold the text (hint: look in one of the cells I've written above to find an example of how to do this – you can see the "raw" markdown by double clicking into any cell).

Hello World!

1.1.2 Exercise 2: say hello with code

It's important to be able to interact with our code to understand what it's doing and effectively debug it, and the easiest way to do this is with the print() function. Here's an example:

[2]:
```
#This is a comment: anything after a hashtag/pound sign will be ignored by the␣
 ↪program,
#so we can write whatever we want in plain english without causing an error!
print("my name is Kirk")
```

```
my name is Kirk
```

Your turn: write your own print command in the cell below to get the computer to say hello world!

[1]:
```
#your code here
print("my name is Avery")
```

```
my name is Avery
```

1.1.3 Exercise 3: variables

Variables are an incredibly important part of any programming language. They allow us to store information for later use by the program, and we create them with the = sign (also known as the assignment operator). Here's an example:

[4]:
```
x = 5 #here I have made a variable called x that has a value of 5
print(x) #let's see what x is!
```

```
5
```

[5]:
```
string = "my name is kirk" #here I have made a variable called string that holds␣
 ↪the phrase I wrote above
print(string) #notice the output is the same as it was before
```

```
my name is kirk
```

Your turn: write your own code below that will save "hello world" to a variable, then print that variable to the console

[2]:
```
#your code here
string = "hello world"
print(string)
```

```
hello world
```

Variable rules cheat sheet:

1. Variable names must be unique. If you name two things x it will only remember the second one.
2. Variable names cannot start with a number. For example, `variable1 = x` is fine but `1variable = x` is not.
3. No spaces! Variable names must be connected. You can combine multiple words with underscores (ie `my_variable = x`) or using camelCase (ie `myVariable = x`)
4. Variables can be added/subtracted/etc together (as long as they're the same type) to create a new result (this is the most common way we use them).

1.1.4 Exercise 4: manipulating variables

We can use the standard mathematical operators on variables to do math (or other things) with them, assuming that they are the same type (we can't add 5 to "hello" for example). Consider the following simple word problem:

John has 12 cookies and Sally has 3. John eats half his cookies and Sally eats 1, but John is feeling kind of sick of cookies now so he gives half of his remaining cookies to Sally, who then eats another one. How many cookies do John and Sally have now?

[1]:
```
johnCookies = 12 #john starts with 12
sallyCookies = 3 #sally starts with 3

print("John starts with",johnCookies,"cookies and Sally starts␣
 ↪with",sallyCookies,"cookies.")

johnCookies = johnCookies/2 #john eats half his cookies
sallyCookies = sallyCookies - 1 #sally eats 1

print("After eating half John now has",johnCookies,"cookies and after eating one␣
 ↪Sally now has",sallyCookies,"cookies.")

sallyCookies = sallyCookies + johnCookies/2 -1 #sally gets half of john's␣
 ↪remaining cookies but eats 1
johnCookies = johnCookies/2 #john gives away half of his remaining cookies

print("John gave half of his reamining cookies to Sally, who ate another one.")
print("At the end John has",johnCookies,"cookies and Sally␣
 ↪has",sallyCookies,"cookies.")
```

```
John starts with 12 cookies and Sally starts with 3 cookies.
After eating half John now has 6.0 cookies and after eating one Sally now has 2
cookies.
John gave half of his reamining cookies to Sally, who ate another one.
At the end John has 3.0 cookies and Sally has 4.0 cookies.
```

Your turn: Solve this word problem with code!

Captain Kirk has 200 photon torpedoes available on the Enterprise, with 12 Klingon birds of prey closing in fast. Each bird of prey can take 8 torpedo hits before being destroyed. Assuming Sulu can outmaneuver the Klingons and the Scotty can keep the ship's integrity intact during the battle, how many torpedoes will the Enterprise be left with?

Challenge: Chekov is a good shot, but nobody's perfect. If 25% of the shots miss, how many torpedoes are left?

[4]:
```
#your code here

torpedoes = 200
birds = 12
torpsLeft = 200 - (12 * 8)
print("Torpedoes left: ", torpsLeft)
```

```
Torpedoes left:  104
```

1.1.5 Exercise 5: using modules

By default Python only comes with a few parts "turned on." This is to save memory/other computing resources for things you may not need. There are many extra packages that we can use in Python to do tasks that might otherwise take a long time, and to access these packages we import them. One useful additional package that is prevalent across astronomy is the astropy package, which we will be using throughout the course as one of them any things it can help us do is keep track of our units! Here is an example:

Note: If the cell below gives you an error, you may need to install astropy (click the link for instructions / ask me for help)

[7]:
```
from astropy import units #notice how the from/import statement is colored␣
↪differently--this means it's an important Python keyword

earthSunDistance = 1*units.AU #here we have created a variable called␣
↪earthSunDistance that has a value of 1 AU (because that's the distance to the␣
↪Sun!)
print("The Earth is ",earthSunDistance,"from the Sun.")
print("In SI units (meters), the Earth is ",earthSunDistance.to(units.m),"from␣
↪the Sun.") #notice how we can use the .to() method to convert units
```

```
The Earth is  1.0 AU from the Sun.
In SI units (meters), the Earth is  149597870700.0 m from the Sun.
```

[7]:
wk

Your turn: What percent are you done with the semester? Following the example above knowing that there are **16 weeks** in a semester and you are nearly done with your **second day**, use the `units` functionality from `astropy` to set up a ratio and convert that to a percentage!

Hint: You should create two variables, one with the number of weeks in the semester (i.e. `semesterLength = 16*units.week`) and the other with the number of days completed thus far (i.e. `daysCompleted = 2*units.day`). Then create a new variable that divides the days completed by the total time in the semester, and finally convert that to a percentage. There are a few ways to do this, but pay close attention to the output and make sure your units match up with the units of a percentage! If you get $\frac{d}{wk}$ you may need to convert units, or check out the `dimensionless_unscaled` method here

[17]:
```
#your code here

from astropy import units
semesterLength = 16 * units.week
daysCompleted = 2 * units.day
x = (daysCompleted / semesterLength).decompose()
percent = x * 100
print(f"Percentage of semester done: {percent}%")
```

```
Percentage of semester done: 1.785714285714286%
```

1.1.6 Exercise 6: 1030 Review / practice homework type problem

Here are some equations that you may find helpful for the following problems:

Kepler's Third law:

$$a^3 = cP^2$$

(where c is a constant)

Newton's law of gravitation:

$$F = ma = \frac{Gm_1m_2}{r^2}$$

Equations of circular motion:

$$a_c = \frac{v^2}{r} = \omega^2 r$$

$$\omega = 2\pi f = \frac{2\pi}{P}$$

`astropy` conveniently has constants baked in for us to use, so we don't have to look up the value of G every time we want to use it, but instead we can just do:

```
from astropy.constants import G
print(G)
  Name   = Gravitational constant
  Value  = 6.6743e-11
  Uncertainty  = 1.5e-15
```

```
Unit  = m3 / (kg s2)
Reference = CODATA 2018
```

and now we can use `G` like any other code variable!

Now on to your practice problems...

1. The International Space Station (ISS) complets an orbit of the Earth once every ~ 90 minutes. Use the mass of the Earth ($\sim 6 \times 10^{24}$kg) and Newton's law of gravitation to calculate the height of the ISS's orbit above the Earth's surface.

[43]:
```python
from astropy.constants import G # you'll need this, so I imported it for you
import numpy as np

# angular velocity => omega = 2*pi / p
# centripetal acceleration => ac == v**2 / r == omega**2 * r
# >> ac = omega**2 * r
# centripetal force = Fc = m * ac
# F = m * a = G*m1*m2 / r**2

'''
KNOW:
- p = 90 minutes
- m1 (Earth) = 6e24 kg

WANT:
- r (height of ISS)

m*ac == G*m1*m2 / r**2
m * (omega**2 * r) == G*m1*m2 / r**2
r**3 == G*m1*m2 / m1 * omega**2

>> r = cbrt(G*m1*m2 / m2 * omega**2)
>> r = cbrt(G * m1 / omega**2)
'''

m1 = 6e24 * units.kilogram
p = 90 * 60 * units.second
omega = 2 * np.pi / p
r = np.cbrt((G * m1) / (omega**2))
r
```

[43]:
6662873.8 m

2. The asteroid Eros has crashed into Venus, nudging its orbit inward by roughly 5%.

First think conceptually — should Venus now orbit faster or slower? Answer this conceptual question by editing the markdown below:

Your conceptual answer and reasoning: Venus should now be orbiting faster than before since the size of its orbit has shrunken, which means its orbital speed must increase.

Then use **Kepler's third law** to calculate the percent change in the orbital period in the code cell below.

[24]:
```
from astropy import units
import numpy as np

# kepler's 3rd law: a^3 = cP^2 >> Use proportions

# p1**2 / a1**3 == p2**2 / a2**3
# p1**2 / (0.95 * a1)**3 == p2**2 / a2**3 => p1**2 / (0.95**3 * a1**3) == p2**2 /
 ↪ a2**3
# p2**2 / p1**2 == 0.95**3 * a1**3 / a2**3 => p2**2 / p1**2 == 0.95**3
# p2 / p1 == sqrt(0.95**3)

x = 0.95**3
change = 100 - np.sqrt(x) * 100
change
```

[24]:
```
7.405453724314853
```

1.1.7 Wrapping up

We will have you submit your completed homeworks as PDFs to Canvas. To convert your Jupyter notebook to a PDF, you can use the built-in convert to PDF functionality (at the top of the page there should be an option for it) or alternatively you can just print to PDF like you would any other website! You'll submit a PDF of your completed version of this assignment as your first homework.