Tutorial24-Astropy

July 31, 2022

Due on Tuesday, August 2nd at 11:59pm.

1 Make a new notebook

Create a new jupyter notebook and edit its name from “Untitled” to ”tutorial##_{identikey}”.

```
[ ]: import numpy as np
     import matplotlib.pyplot as plt
     %matplotlib inline
```

2 think+type: using astropy for calculations

The ionization energy of hydrogen is 13.6 eV. The unit “eV” stands for “electron-volts”, the amount of energy gained by one electron moving across 1 volt of electric potential. The ionization energy is the same as the binding energy of an electron to a proton in the ground quantum state ($n = 1$). If a neutral hydrogen atom is hit by a photon of this energy, it will be ionized.

We can calculate the ground-state energy of hydrogen (sometimes called the Rydberg constant) $E_{ionization}$ from fundamental quantities as

$$E_{ionization} = \frac{1}{2} m_e c^2 \alpha^2$$

where m_e is the mass of the electron, c is the speed of light, and $\alpha \approx 1/137$ is the “fine structure constant”, a unitless measure of the strength of electromagnetism.

Let’s use the above formula to re-calculate the ground state energy of hydrogen using `astropy`:

```
[ ]: import astropy.constants as co
     import astropy.units as u
```

```
[ ]: energy = 0.5 * co.m_e * co.c**2 * co.alpha**2
     energy
```

Is this right? It’s hard to tell with those units. Let’s convert it to more familiar units:

```
[ ]: energy.to(u.eV)
```

Nice! You can see how straightforward it is to do calculations with units using `astropy`. Thanks `astropy`!

What wavelength would a photon with that energy have? Remember that photon wavelength λ and energy E are related via Planck's constant h as $E = hc/\lambda$.

```
[ ]: wavelength = co.h * co.c / energy
     wavelength
```

Those units are nasty. Let's simiplfy them:

```
[ ]: wavelength.decompose()
```

3 think+type: What is the distance to Neptune in light-minutes *right now*?

```
[ ]: from astropy.coordinates import get_body
     from astropy.time import Time
     nep = get_body('neptune',Time.now())
     print(nep)
```

This is the location of Neptune **right now** (which is pretty neat!) Now take the distance and divide by the speed of light:

```
[ ]: time =  nep.distance / co.c
     print(time)
```

Finally, convert to minutes of time:

```
[ ]: print(time.to(u.min))
```

We can check this by knowing that the light travel time from the Sun to the Earth (1 AU) is about 8.3 minutes:

```
[ ]: print(8.3*u.min*nep.distance.value)
```

You may notice that `astropy Quantity` objects have values and units. We can access each separately:

```
[ ]: print(f'This Quantity has value "{nep.distance.value}" in "{nep.distance.unit}"␣
      ↪units.')
```

4 think+code: how old is the Universe?

*In this **think+pair+code**, please discuss with your zoommates how to accomplish the requested task.*

The universe is expanding, and this expansion is characterized by the Hubble Constant, H_0. If we run time backwards, then the universe would be collapsing, and eventually everything will wind up in one place at one time, marking the time of the Big Bang. If the Universe were to expand at the same rate over its entire history (constant expansion), then its age would be $1/H_0$.

Calculate the age of the Universe ***in Gigayears*** assuming that $H_0 = 70$ km/s/Mpc. Start by defining the Hubble constant (H_0) using appropriate astropy units. (You'll need to make an educated guess what the appropriate units are named but they are very likely as you would expect.) **Use only astropy units and unit conversions to do this calculation.**

Check: Does your answer make sense?

5 think+type: more fun with astropy

astropy has snazzy tools for handling times.

```
[ ]: from astropy.time import Time

     # when did Apollo 11 land on the moon?
     apollo = Time('1969-7-20 20:17')
```

```
[ ]: apollo
```

```
[ ]: # what's the Julian Date of the Apollo landing
     apollo.jd
```

```
[ ]: # how many years ago was the landing, in units of years?
     dt = Time.now() - apollo
     dt.to('year')
```

astropy can also determine coordinates for many common objects.

```
[ ]: # find the sky coordinates of a star
     from astropy.coordinates import SkyCoord
     star = SkyCoord.from_name('Betelgeuse')
     star
```

```
[ ]: # figure out the galactic coordinates of the star
     star.galactic
```

```
[ ]: # figure out what constellation the star is in
     # based on its coordinates
     from astropy.coordinates import get_constellation
     get_constellation(star)
```

6 think+pair+code: talk about frequent flier miles!

*In this **think+pair+code**, use some of the tools you just learned about to code the following problem.*

Calculate how far you've travelled around the Sun (on Planet Earth Airlines) from the day of your birth (use your birth time if you know it!) to this exact moment. To do this:

- You'll likely need to use `astropy.time`, `astropy.units`, and `astropy.constants` in various arrangements.
  - All of which you've imported previously in this tutorial so there is no need to import them again.
- You can treat the Earth's movement as a circular orbit centered on the Sun.
  - Ignore the distance you travel during the Earth's daily rotation (although you can add that in later if you like!)
- Set up the variables you need as `Quantity` objects that include units.
- Do the calculation.
- Simplify (`decompose`) as needed, print the answer, and change the units (`.to`) of the result to another unit. (Sadly sheppeys are not available units in `astropy`.)

There are *oodles* of amazing things `astropy` tools can do. Go check them out!

7 Save and 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 `tutorial##_{identikey}.html`. Please upload this HTML file as your submission to "Tutorial ##" on `Canvas`.