Tutorial25 - Astropy

Open a new jupyter notebook and call it "`Tutorial_25_{your identikey}`". In the first cell of your notebook, please type your name and the name of this tutorial.

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

1 think+code: using `astropy` for calculations

You are probably familiar with the ionization energy of hydrogen, 13.6 eV. This is the same thing as the binding energy of an electron to a proton in the ground quantum state ($n = 1$).

You probably learned something about the Rydberg constant for this energy, but you may not have seen the ground state energy of hydrogen expressed in terms of the electron rest mass energy scaled by the strength of electromagnetic force:

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

Here, α is the fine structure constant, a unitless measure of the strength of electromagnetism. It is roughly equal to $1/137$.

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

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

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

Is this right? Let's convert it to more familiar units:

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

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

Let's try another: What is the distance to Neptune in light-minutes *right now*?

```
In [ ]: 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 **now** (which is pretty neat!). Now take the distance and divide by the speed of light:

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

Finally, convert to minutes of time:

```
In [ ]: 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:

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

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

```
In [ ]: print 'This Quantity has value {} and {} units.'.\
            format(nep.distance.value, nep.distance.unit)
```

2 think+pair+code: how old is the Universe?

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 $1/H_0$.

Please caluculate the age of the Universe ***in Gigayears*** assuming that $H_0 = 70\,\mathrm{km/s/Mpc}$. **Use only `astropy` units and unit conversions to do this calculation.** Start by defining the Hubble constant using appropriate units.

Does your answer make sense?

3 think+code: high-redshift galaxies

Let's return to the Hubble Extemely Deep Field:

```
In [ ]: import astropy.io.fits
        path = '/home/jdarling/astr2600/'
        hdus = astropy.io.fits.open(path + 'HubbleXDF.fits')
```

Let's remind ourselves what is in this image file:

```
In [ ]: hdus.info()
```

There are two images called F435W (blue, centered around 435 nm) and F775W (red, centered around 775 nm). Let's confirm that those faint, distant galaxies we identified in lecture are invisible in blue light:

```
In [ ]: # Pull image arrays
        F435 = hdus['F435W'].data
        F775 = hdus['F775W'].data

In [ ]: # Read distant galaxies file for overlay
        import astropy.io.ascii
        distantgalaxies = astropy.io.ascii.read(path + 'XDF_idropouts.txt')
        ra = distantgalaxies['RA']
        dec = distantgalaxies['DEC']

        # Convert world coordinate system to pixels
        from astropy import wcs
        wcs775 = wcs.WCS(hdus['F775W'].header)
        xgal, ygal = wcs775.wcs_world2pix(ra, dec, 0)
```

Now make side-by-side images of a patch of sky for each filter:

```
In [ ]: fig,axes = plt.subplots(1,2,sharex=True,sharey=True,figsize=(12,6))

        # Set up plot parameters
        plt.xlim(3600,3700); plt.ylim(2300,2200)
        n, x = np.percentile(F775.flatten(), [1, 99])
        imargs = dict(vmin=n, vmax=x, cmap='gray')
        sargs = dict(facecolor='none', edgecolor='orange',s=1000)

        # Blue
        plt.sca(axes[0])
        plt.imshow(F435, **imargs)
        plt.scatter(xgal, ygal, **sargs)
        plt.title('Blue')

        # Red
        plt.sca(axes[1])
        plt.imshow(F775, **imargs)
        plt.scatter(xgal, ygal, **sargs)
        plt.title('Red')
```

This is a nice example of how *not* detecting something is meaningful.

4 think+pair+code: your own calculation

Working with your neighbor(s), do the following:

- come up with a calculation you want to do that involves units
- set up the variables you need as `Quantities` that include units
  - For example: `distance = 10 * u.apc`
- do the calculation

- simplify (`decompose`) and change the units (`.to`) of the result as needed

When finished, save your notebook and upload it to D2L in the "Tutorial 25" folder.