Working with units in code

How we used to keep track of units in code:

```
speed = 100. # km/s
speed_kms = 100.
```

- implicit
- doesn’t scale well
- operations don’t carry units

astropy.units

Store, convert, display units in code

Key concepts:

Unit objects

Quantity objects

Physical types

Constants

Equivalencies

astropy.units

Unit objects

Quantity objects

Physical types

Constants

Equivalencies

- objects that represent units
- includes base units (e.g., meter)
- also all SI prefixes (e.g., kilometer, …)
- mostly used through the **Quantity** object

```
>>> import astropy.units as u
>>> u.meter
>>> u.m
>>> u.kilometer
>>> u.km
```

astropy.units

Unit objects

Quantity objects

Physical types

Constants

Equivalencies

- number or array * unit = Quantity
- represents a value or array of values with units
- can convert to other *equivalent* units

```
>>> q1 = 15 * u.meter
>>> q2 = [4., 8., 15., 16.] * u.km
>>> q2.to(u.femtometer)
```

astropy.units

Unit objects

Quantity objects

Physical types

Constants

Equivalencies

- most units have a physical type
  - e.g., meter is a ‘length’, a Newton is a ‘force’
- quantities can always be converted to a unit with the same physical type
- Physical types can be used for dimensional analysis (v4.3+)

```
>>> from astropy.units import physical
>>> physical.length / physical.time**2
PhysicalType('acceleration')
```

astropy.constants

Unit objects

Quantity objects

Physical types

Constants

Equivalencies

- constants are special quantity-like objects
  - e.g., gravitational G, speed of light c
- some constants are also implemented as units, e.g., solar mass

```
>>> from astropy.constants import G, c
>>> Rsch = 2 * G * (1*u.Msun) / c**2
>>> Rsch.to(u.earthRad)
<Quantity 0.00046302975433126663 earthRad>
```

astropy.units

Unit objects

Quantity objects

Physical types

Constants

Equivalencies

- unit conversions also support *equivalency* conversions, e.g., from wavelength to frequency

fails: length → frequency

```
>>> (5577.*u.Angstrom).to(u.THz)
```

```
>>> (5577.*u.Angstrom).to(u.THz, u.spectral())
<Quantity 537.5514757037834 THz>
```