Integration

• Numeric integration much studied
  – But hard to make robust for all functions
  – Problem is finding where function changes rapidly
• Use wavesets customized to each component
  – Indicates where function has rapidly changing detail
• Integration machinery is generic
Integration (cont.)

```python
>>> source = GaussianAbsorption(4000, 20, 0.8) * \\
    ... (Gaussian(3000, 100) + 1.e10*BlackBody(6000))
>>> source.integrate()  # returns integrated flux
```

```python
>>> source = BlackBody(6000).normalize(VMag(23.3))
>>> source = (Gaussian(3000, 50) + 
    ... 1.e10*BlackBody(6000)).normalize(Jy(3.1), 
    ... trans=JohnsonV)
```
Empirical Spectra

- Can be treated exactly like analytical functions
  - Just has a lot of parameters!
  - Can use customized interpolation

```python
>>> data = pyfits.getdata('fuse.fits')
>>> source = TableSource(flux=data.field('flux'),
...                        wavelength=data.field('wave'))
```
Valid Range

• Need limits to integration
• Info on valid range of combined components
  – Intersection of component ranges

```python
>>> source.waverange()
[985., 1085.]
>>> BlackBody(6000).waverange()
[None, None]
```
Normalization

- Need to scale components to various measures of intensity
- Optional bandpass for normalization
- Generic machinery (part of abstract class)

```python
>>> source = BlackBody(6000).normalize(VMag(23.3))
>>> source = (Gaussian(3000,50) + ... 1.e10*Blackbody(6000)).normalize(Jy(3.1), ... trans=JohnsonV)
```
class SourceFunction:

    def __call__(self, wavelengths):
        return self.resteval(invZ(wavelengths, self.z))/(1.+self.z)

    def waveset(self):
        return Z(self.restwaveset(), self.z)

The following shows how redshift would be used.

>>> source = BlackBody(6000,z=1.3) # or
>>> source = BlackBody(6000).z(1.3)
>>> source = (Gaussian(3000,20)+Blackbody(6000)).z(1.3) # or
>>> source = Z(Gaussian(3000,20)+Blackbody(6000), 1.3)
>>> source = (GaussianAbsorption(3000,20,0.8,z=.3)* \ 
...           BlackBody(6000,.5)).z(.8)
Units

• Complication of transformation between $\nu$ and $\lambda$

```python
>>> asJy(BlackBody(6000)(NM(300))) # evaluate at 300 nm,
... # return in Janskys
>>> source = GaussianSource(EV(0.5),EV(0.1)) # centered at
... # .5 electron volts
>>> source(NM(10), funits=Fnu) # or
>>> asFnu(source(NM(10), NM(10)))
```
Advantages

- Polymorphism
- Data and Information Hiding
- Decoupling/minimal interfaces
- Flexibility
- Takes advantage of language syntax
Disadvantages

• Harder to follow program flow as compared to procedural flow
• Harder to get mindset if experienced in procedural programming
• Not appropriate for all problems