basics of numeric representations

Ben Bolker

21:00 05 March 2017

Integers

- stored as binary digits
- `n` digits gives room to store *signed* values between min of $-2^(n-1)$ and max of $2^(n-1)-1$
- base Python automatically switches from 64-bit to arbitrary-length integers as necessary
- in `numpy` can specify precision of integers, or *unsigned* integers (e.g. `uint8`)
- you should rarely do this
- and be careful if you do

information about types and ranges

Floats

- default 64-bit floats: 1 sign bit, 11 bits for exponent (`x`), 52 bits for mantissa (`m`)
- can store up to *approximately* $2^(2^(x-1))$; numbers less than 10^{-324} *underflow* to zero; numbers greater than 10^{308} give `OverflowError`
- only 52 bits of precision in mantissa; for $x \leq -16$, `1+x` *underflows* to `1.0`.
- similar issues occur as long as addends are far apart, e.g. `10**9+10**-8`
- what can you do?
  - more *stable* algorithm (e.g. add items in increasing order)
  - work on the log scale (i.e. add log values rather than multiplying values)
  - extended/arbitrary precision floats: `decimal` module (built in), `mpmath` - always be careful comparing floating point:

```
import math
a = math.sqrt(2)
```

Use something like this:

```
def approx_equal(a, b, tol=1e-8):
```

Lots more detail here