Numerical Python

modeling, numerics

CS101 Lecture #17

2016-11-28

Administrivia

Administrivia

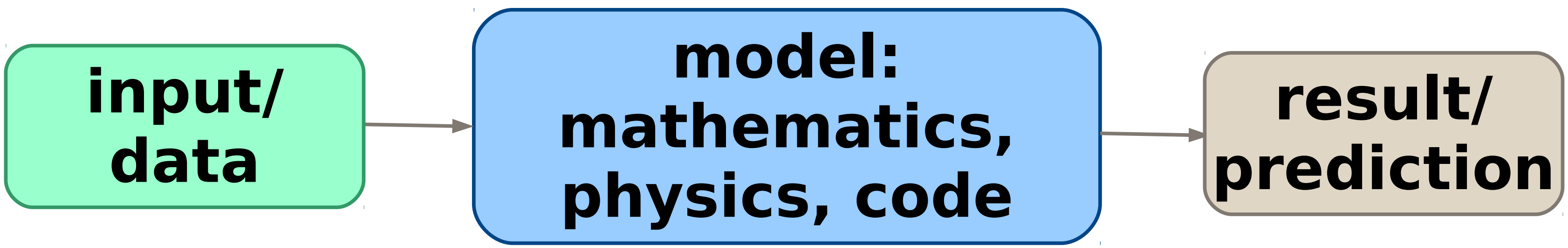
- Homework #8 is due Friday, Dec. 2.
- Homework #9 is due Friday, Dec. 9.
- Midterm #2 is Monday, Dec. 19 from 7–10 p.m.

Modeling

all models are wrong
(but some are useful)

—George Box, statistician

elements of modeling
the same story



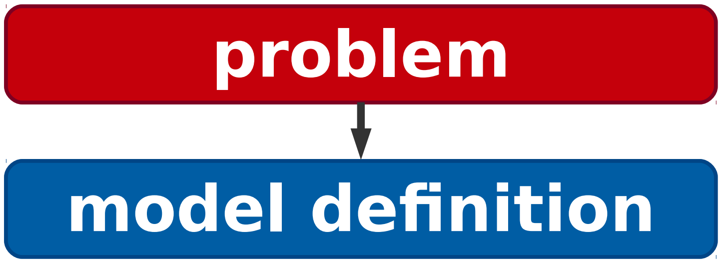
elements of modeling

the model lifecycle

problem

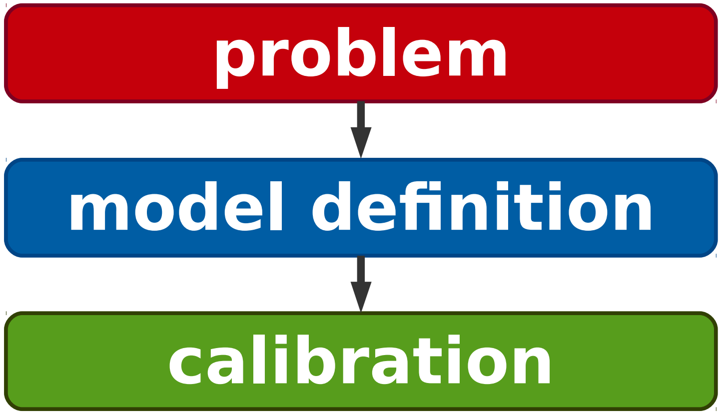
elements of modeling

the model lifecycle



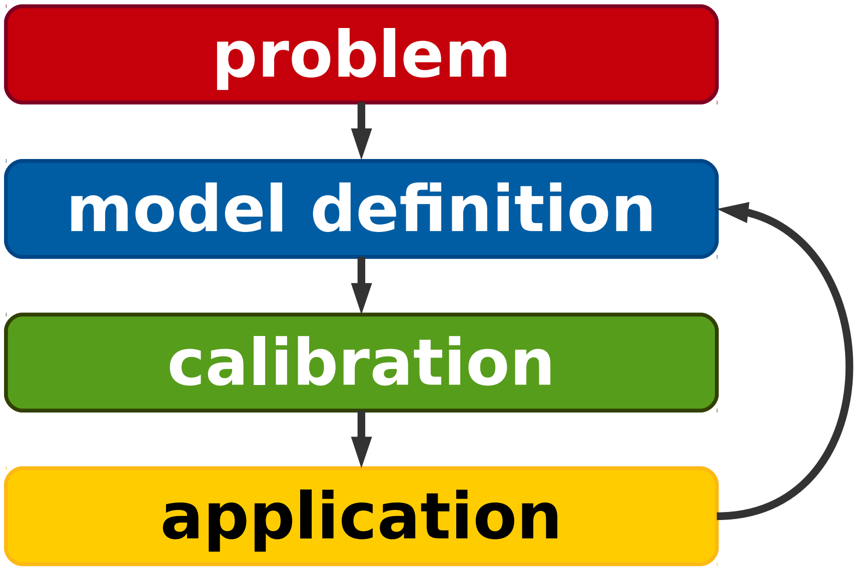
elements of modeling

the model lifecycle

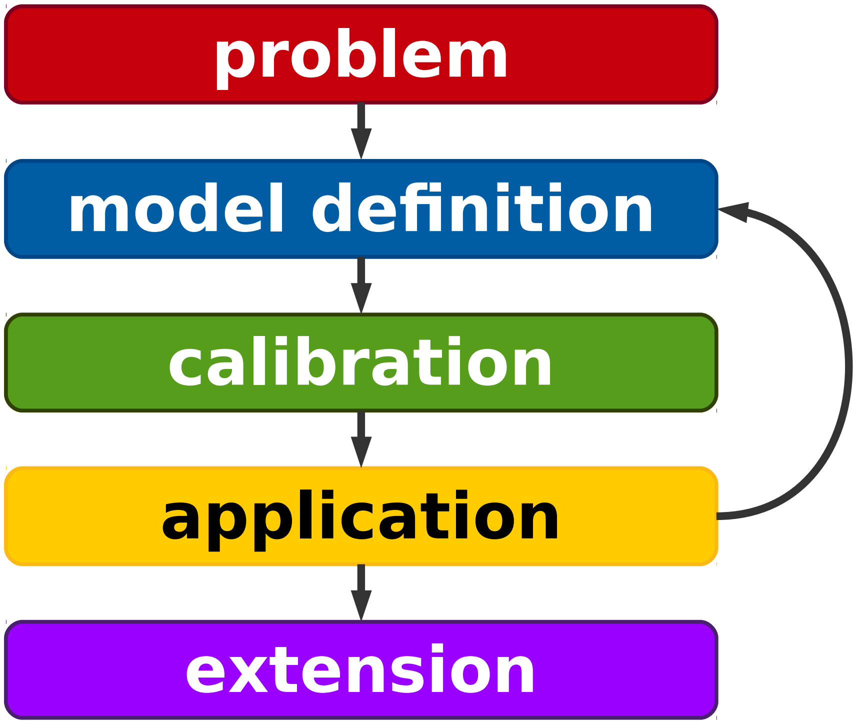


elements of modeling

the model lifecycle

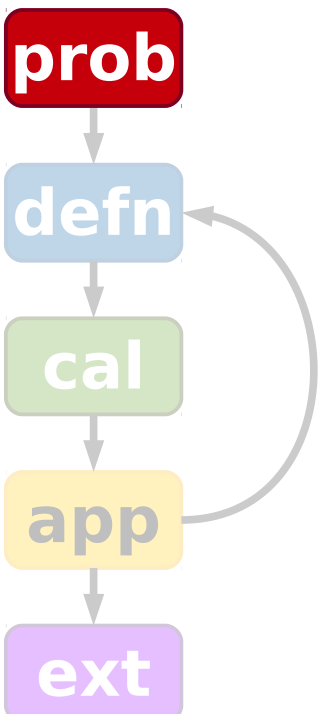


elements of modeling
the model lifecycle

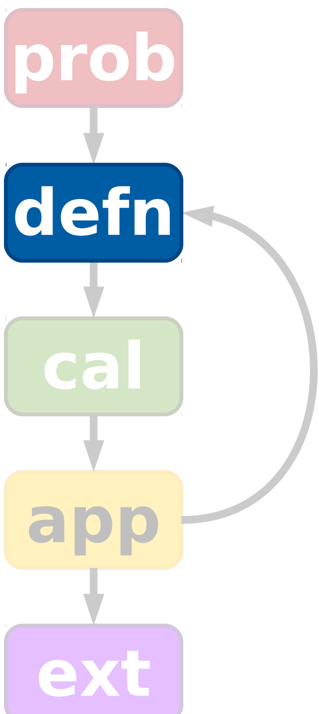


elements of modeling

problem statement



elements of modeling
model definition



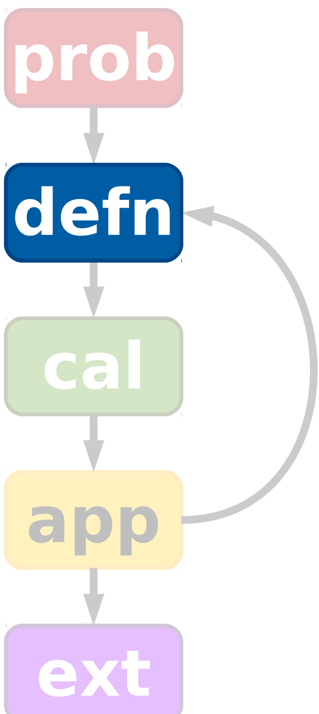
physics & math

implementation

$$\Delta L = \alpha \left(T - T_0 \right)$$

elements of modeling

model definition



physics & math

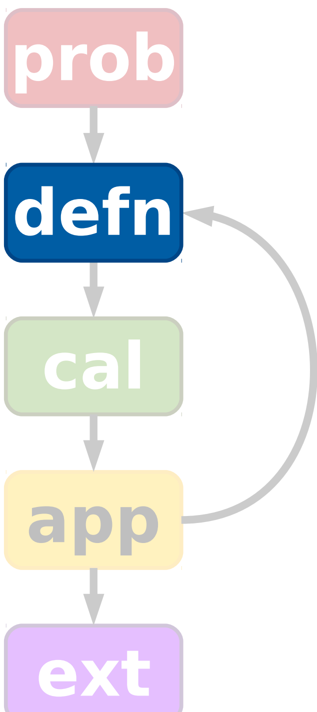
implementation

$$\Delta L = \alpha \left(T - T_0 \right)$$

$$y = mx + b$$

elements of modeling

model definition



physics & math

implementation

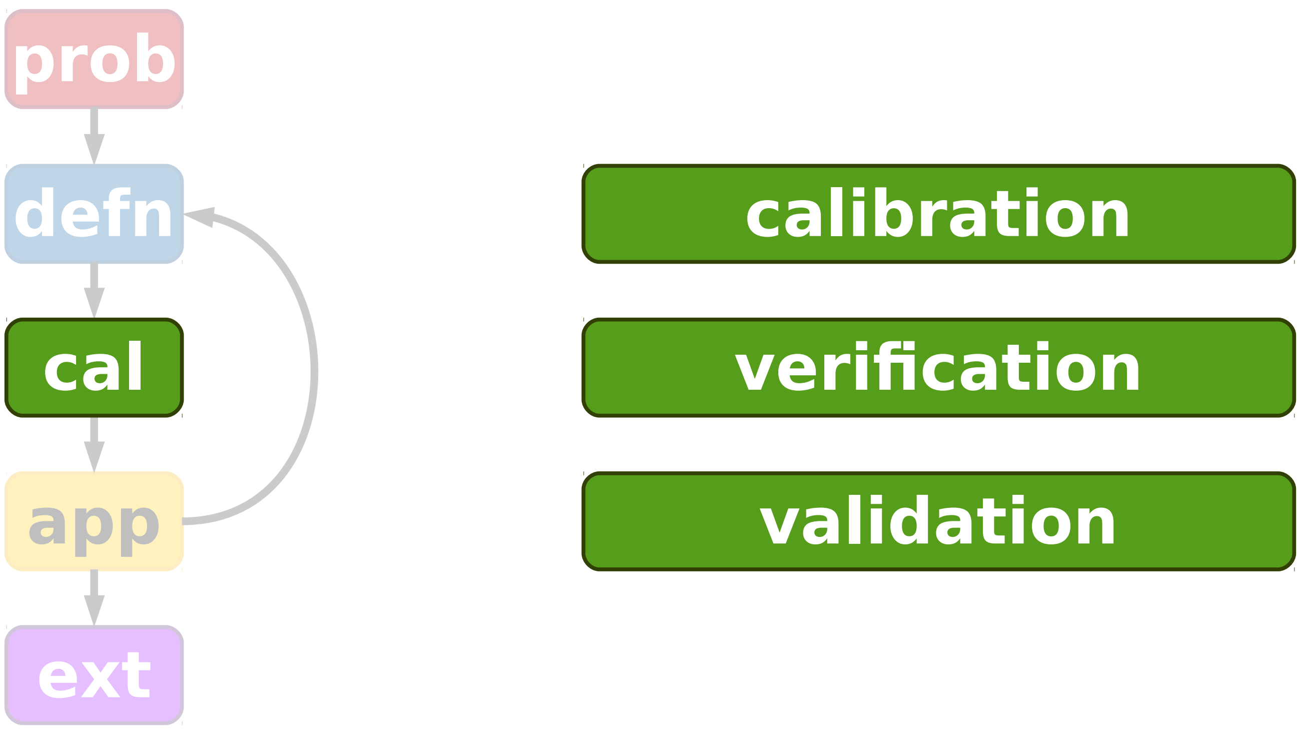
$$\Delta L = \alpha \left(T - T_0 \right)$$

$$y = mx + b$$

$$\Delta L = \alpha T + \left(-\alpha T_0 \right)$$

elements of modeling

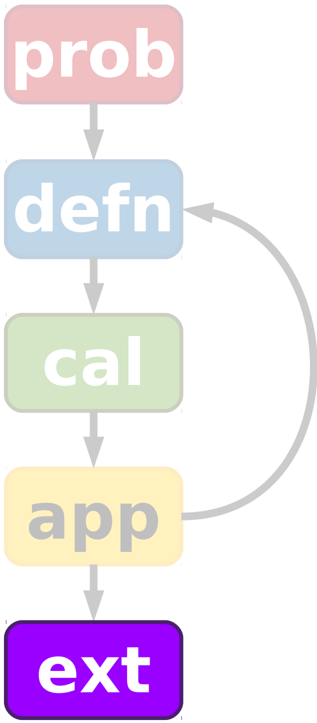
calibration



elements of modeling
application

elements of modeling

extension



shortcomings

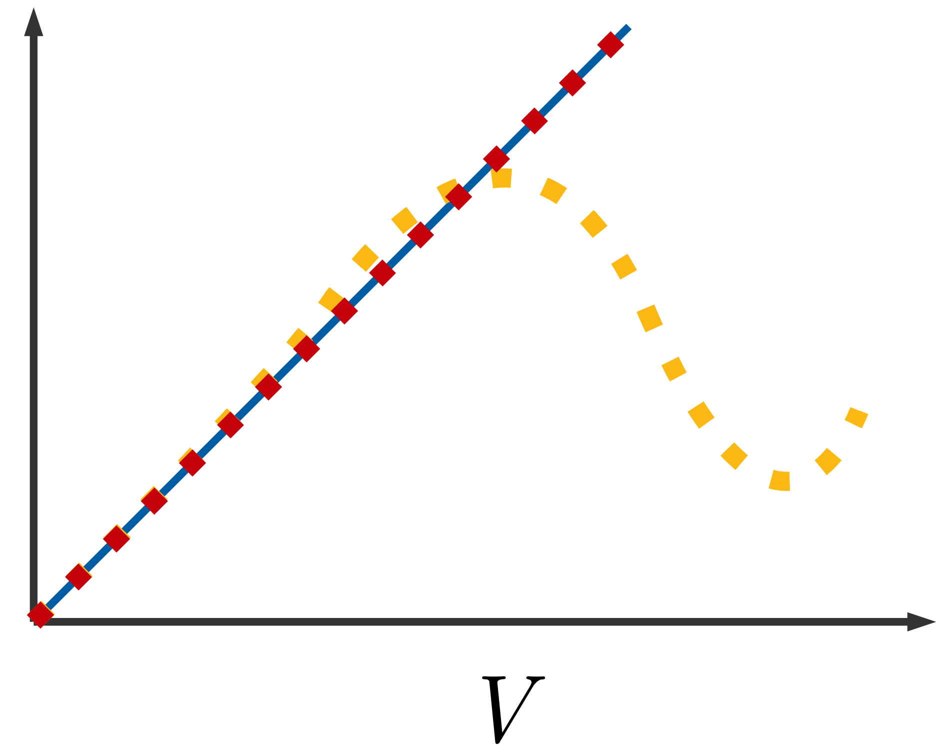
surprises

scientific programming

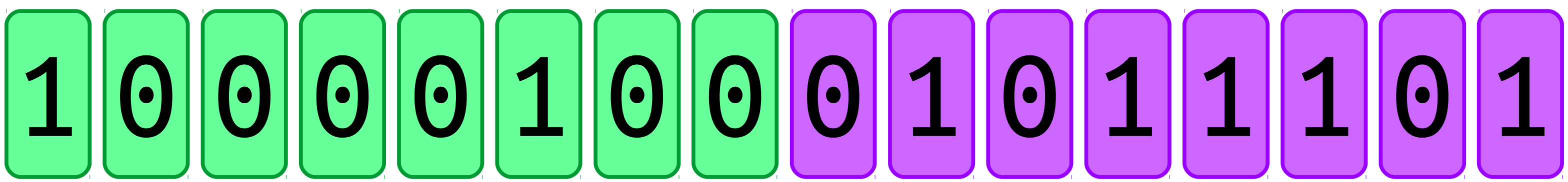
model failure: ohm's law

$$V = IR$$

$$y = mx + b$$



1	0	0	0	0	1	0	0	0	1	0	1	1	1	0	1


1 0 0 0 0 1 0 0
0 1 0 1 1 1 0 1

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}
1	0	0	0	0	1	0	0	0	1	0	1	1	1	0	1

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}
1	0	0	0	0	1	0	0	0	1	0	1	1	1	0	1

$$2^7 + 2^2 + 2^{-2} + 2^{-4} + 2^{-5} + 2^{-6} + 2^{-8}$$

$$= 128 + 4 + ½ + 1/32 + 1/64 + 1/256$$

$$= 132.55078125$$

1.1_{10}	`0001100110011001100110011001100110011001100110011010`
0.8_{10}	`1001100110011001100110011001100110011001100110011010`
$1.1-0.8_{10}$	`0011001100110011001100110011001100110011001100110100`

1.1_{10}	`0001100110011001100110011001100110011001100110011010`
0.8_{10}	`1001100110011001100110011001100110011001100110011010`
$1.1-0.8_{10}$	`0011001100110011001100110011001100110011001100110100`
0.3_{10}	`0011001100110011001100110011001100110011001100110011`

1.1_{10}	0001100110011001100110011001100110011001100110011010
0.8_{10}	1001100110011001100110011001100110011001100110011010
$1.1-0.8_{10}$	0011001100110011001100110011001100110011001100110100
0.3_{10}	0011001100110011001100110011001100110011001100110011

1.1_{10}	0001100110011001100110011001100110011001100110011010
0.8_{10}	1001100110011001100110011001100110011001100110011010
$1.1-0.8_{10}$	0011001100110011001100110011001100110011001100110100
0.3_{10}	0011001100110011001100110011001100110011001100110011
Δ_{10}	0000000000000000000000000000000000000000000000000001

Don't compare directly:

- `a == b   # never do this for floats!`

- `np.isclose( a, b, rtol=1e-05, atol=1e-08)`
- `np.allclose(a, b, rtol=1e-05, atol=1e-08)`

Parameters:

- `rtol  # relative tolerance (w/i percent)`
- `atol  # absolute tolerance`

The number 0.1_{10} is written in binary as

The number 0.1_{10} is written in binary as

$$0.0001100110011001100110011001100..._{2},$$

The number 0.1_{10} is written in binary as

$$0.0001100110011001100110011001100\ldots_2,$$

which the machine represents as

$$0.00011001100110011001100_2.$$

The number 0.1_{10} is written in binary as

$$0.0001100110011001100110011001100..._2,$$

which the machine represents as

$$0.00011001100110011001100_2.$$

The difference of these numbers is

$$0.0000000000000000000000011001100..._2,$$

The number 0.1_{10} is written in binary as

$0.0001100110011001100110011001100\ldots_2$,

which the machine represents as

$0.00011001100110011001100_2$.

The difference of these numbers is

$0.00000000000000000000000011001100\ldots_2$,

rendered in decimal as about $0.000\,000\,095_{10}$.

The number 0.1_{10} is written in binary as

$0.0001100110011001100110011001100..._2$,

which the machine represents as

$0.00011001100110011001100_2$.

The difference of these numbers is

$0.0000000000000000000000011001100..._2$,

rendered in decimal as about $0.000\,000\,095_{10}$.

$$100\ \text{hr} \times 60\,\frac{\text{min}}{\text{hr}} \times 60\,\frac{\text{s}}{\text{min}} \times (10 \times 0.000\,000\,095_{10}) = 0.34\ \text{s}$$

The number 0.1_{10} is written in binary as

$0.0001100110011001100110011001100..._{2}$,

which the machine represents as

$0.00011001100110011001100_{2}$.

The difference of these numbers is

$0.0000000000000000000000011001100..._{2}$,

rendered in decimal as about $0.000\,000\,095_{10}$.

$$100\text{ hr} \times 60\,\frac{\text{min}}{\text{hr}} \times 60\,\frac{\text{s}}{\text{min}} \times (10 \times 0.000\,000\,095_{10}) = 0.34\text{ s}$$

Which of the following expressions is liable to experience problems with numerical error? Assume all variables are defined and have appropriate type.

A. `( a / 1e5 < 0 )`
B. `( b <= 1.0 )`
C. `( c ** 0.5 ) / 2`
D. `( d == 0.4 )`

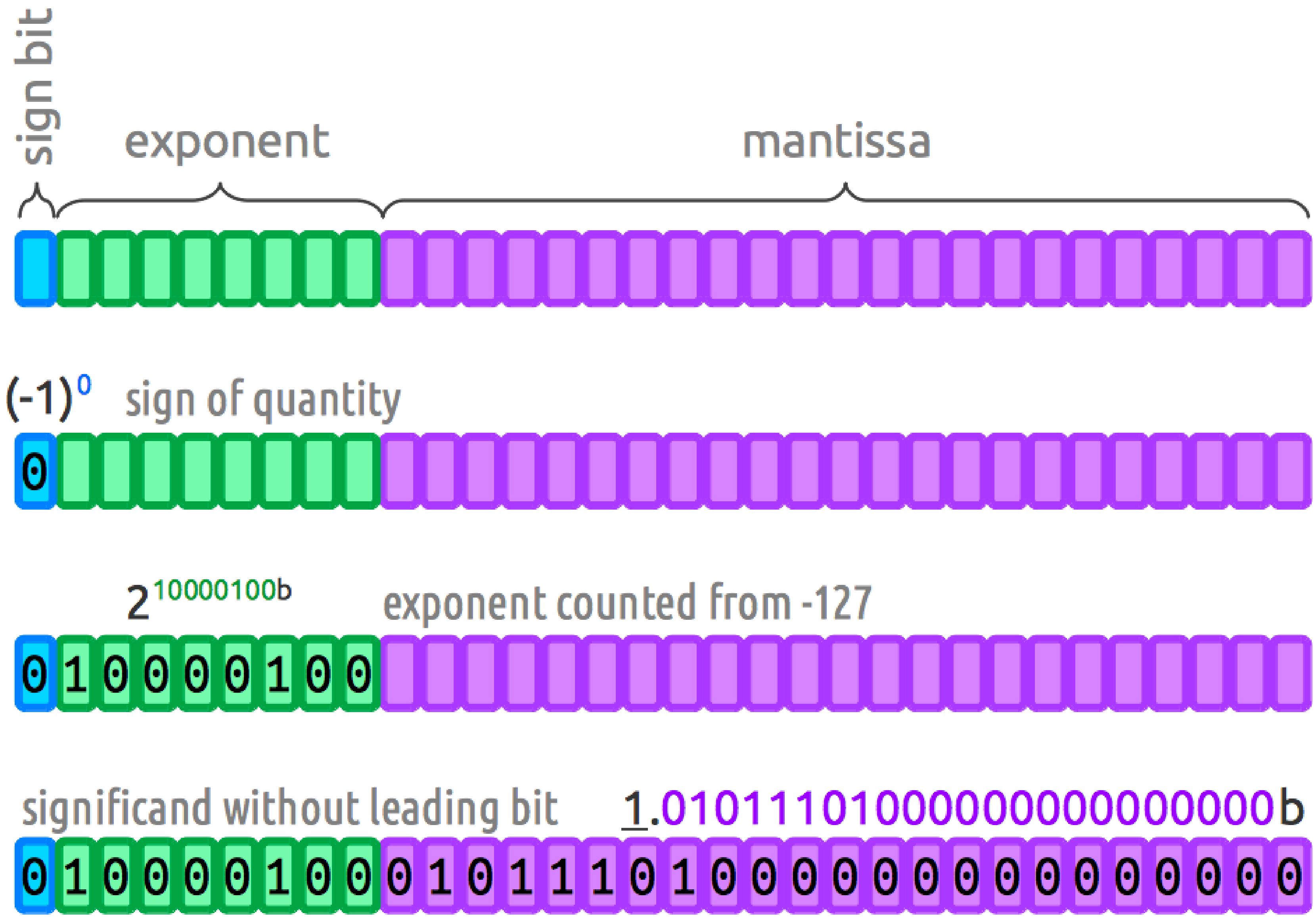
What does this mean?

```
seatingAvail = guests < 150
```

What does this mean?

```
seatingAvail = guests < 150
seatingAvail = \
  guests < MaximumOccupancy
```

don’t use magic numbers!



$$= (-1)^{0} \times 2^{10000100b} \times 01011101000000000000000b$$
$$= (-1)^{0} \times 2^{132-127} \times 1.01011101b$$
$$= (-1)^{0} \times 2^{5} \times 1.36328125$$
$$= 43.625$$