Lecture 3 - Units, Data Types, Error

* Prayer / AMA

I. Units

* All physical equations have units.

  Units are a standard reference for a physical dimension like length, time, mass, temperature, etc.

* There are two main standard system of units:
  - SI (système Internationale)
  - English.

* See the handout for the "Big 13" units. I expect you to be able to convert between these without looking them up (i.e. Need to memorize).

  example.

  $7\ N = ?\ lbf$

  $$7\,\text{N} \times \frac{1\,lbf}{4.448\,\text{N}} = 1.574\ lbf$$

* An equation must have consistent units on both sides to be physically consistent. This principle is called "dimensional homogeneity"

  e.g. $F = ma$ in SI

  $$N = kg\,\frac{m}{s^2} \Rightarrow N = N \quad \checkmark$$

* Being systematic helps one to keep track of units in computational tools.

Tips:

- Pick a single unit system
- Convert at beginning and end only (if possible)
- Use either: kg, m, s or slug, ft, s
- Use an absolute temp scale

Activity Unit conversions in Excel

II. Data Types

* Variables must be stored in memory. Memory on a computer is a sequence of discrete 1's and 0's. But we want different kinds of variables:

- Base 10 integers
- Characters / Strings
- Boolean variables
- Decimals, aka Floating point numbers.

* Memory is measured by the number of 1's and 0's.

1 bit = a single 1/0

1 byte = 8 bits

kilo byte = 1000 bytes

mega byte = 10^6 bytes

Giga = 10^9

Tera = 10^{12}

Peta = 10^{15}

* Boolean variables are the most straightforward.

0 = False
1 = True
} In principle, only 1 bit!

* Integers are the next most simple.

1 0 1 1 ← bits, binary

2^3 2^2 2^1 2^0 = 11 (decimal, base 10)

- The size of the memory dictates how large of an integer you can represent
- Need another bit for +/- sign
- A 2-byte integer will represent numbers from −32,768 to 32,767

* Characters use a table to convert a binary number to a letter. The original table, the ASCII table, was 7-bits big (0-127) and was later expanded to 8-bits (1 byte).

e.g. 65 = 01000001 = A

109 = 0110 1101 = m

Most languages now use a bigger table called "unicode," as does the web.

* Decimals are the most complicated. They are called "floating point" variables or "floats" for short. Floats are essentially "scientific notation". They store a number-part (mantissa) and an exponent-part (exponent).

e.g $4038.2 \rightarrow 0.40382 \times 10^{4}$ (4 ↑ exponent; 0.40382 ← mantissa)

- The data type allocates some bits for the mantissa, some for the exponent and some for +/- signs for both.
- A typical float of 4 bytes can represent numbers with 8 digits of precision in the decimal part & max exponent is +/ 37

  For 16 bytes we get about 16 digits of precision and +/- 308 is the max exponent.

III. Round-off Error

* Because numbers on a computer have finite precision, we need to pay attention to error due to the imprecision

5 examples of error

- Truncation error: $2/3 = 0.\overline{6}$

  $= 0.6666 \times 10^{0}$ w/ 4 digits

  error of 6.6×10^{-5}
- under flow error: $5.4 \times 10^{-400} = 0$

  when using 16 byte floats
- over flow error: $38 \times 10^{400} = \infty$

  when using 16 byte floats

- Round-off error from addition

ex. 4 digits of precision

$$0.4300 \times 10^4 + 0.1 \times 10^0 \Rightarrow 0.4300 \times 10^4 + 0.0000 * \times 10^4 = 0.4300 \times 10^4$$

↳ not enough to add!

- Round-off error from subtraction

$$0.4300 \times 10^4 - 0.4290 \times 10^4 = 0.0010 \times 10^4 \rightarrow 0.10?? \times 10^2$$

garbage numbers

Activity — Data Types and Finite Precision Error in Python