COURSE WEBSITE

http://physics177-2018.github.io
↳ not the old one on my account

TODAY: START w/ PRACTICUM
↳ HUB & GitHub

then: ERROR

QUESTIONS?

I AM A GUIDE, NOT NEC. A GURU

this class is about YOUR path
→ YOUR PORTFOLIO
→ YOUR SOLUTIONS
(NOT "the" solution)
(Namaste)

~~[illegible]~~ A BIT ABOUT PYTHON
↑
Relatively NEW language

PRO: READABLE
compared to older languages } vs. eg FORTRAN

"FULLY POWERED" ← vs. eg script languages
- OBJECT ORIENTED
- MANY LIBRARIES

QUICK (to write) vs. C++
- scriptable
- interpreted

CON: SLOW → we won't be too worried
RECENT TREND: HARDWARE WILL CATCH UP, BETTER TO CODE UNDERSTANDABLY

BUG HUNTING

MODIFYING CODE → eg in a software company: executing Python may be slower... but you gain time on human end: bug hunt, update,

two big themes in this class

Practicum

Python interpreter

> Python3

EXAMPLES FROM :

An Informal Intro to Python
docs.python.org/3/tutorial

Jupyter

↳ same! but with cells

MarkDown Code

cells have to be executed (SHIFT + ENTER)

MarkDown → typeset

Code → evaluate
AS IF YOU JUST PUT IT INTO INTERPRETER

GitHub

↳ your first assignment
homework 1b.ipynb
walkthrough

ROUNDING ERROR

numbers in Python — CH. 4 Newman (rev. CH. 2 for BASICS)

integers
float ← decimals
complex ← $a + ib$ → $a + jb$

REMEMBER: RULES FOR MIXING TYPES!

all this is ultimately <u>binary</u>

<u>REMINDER</u>

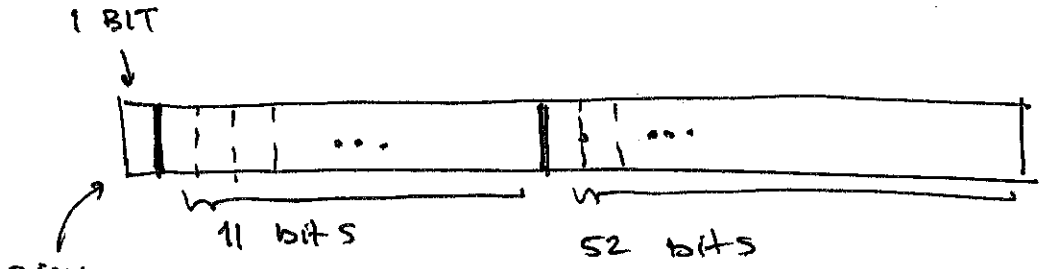
0	1	2	3	4	5	...
0	1	10	11	100	101	...

<u>QUESTION</u>: What is the best way to count on your hands? ← MAX # you can count?
↑ finger binary

CLAIM: 1 hand: $(2^5 - 1) = 31$
LESSON: EXPONENTS

<u>Python</u>: 64 bit ⟷ float

encoding: "IEEE 754" ← some standard
↑ name doesn't matter, but I'm really curious <u>why</u> it has this name



"fraction" / significand / mantissa

$$\boxed{\|\,-\,|\,\cdots} = (-)^{\text{SIGN}} \times \left[1.\underbrace{\sqcup\sqcup\cdots\sqcup}_{\text{FRACTION (52 bits)}}\right]_2 \times 2^{\text{exp}-1023}$$

1 bit

base 2

11 bit

$2^{10} = 1024$

BIAS: allows big #s & small #s

claim [check this]:

the largest number this can encode is

$1.797\cdots \times 10^{308}$ (base 10)

so avoid numbers this big

TRY: $x = 1.1 \times 10^{308}$, $y = 10 * x$ } 1.1E308 * 10 = inf

not a mathematical truth!

smallest # : $2.2\cdots \times 10^{-308}$

2 kinds of Error

① numerical precision / rounding error
↑ limits imposed by Python

② approximation
↑ <u>our</u> choice
eg ε of Taylor expansion
or size of Riemann sum boxes

- you cannot beat num. precision (IRREDUCIBLE)
  ↳ can work around w/ diff. data types

- so no point in improving <u>approx</u> beyond rounding error.
  ↑ <u>IN FACT</u>: usually you will be <u>time</u> limited.

<u>RULE OF THUMB</u> : float precision is 16 digits (in DECIMAL)

SIX FIGS

$$2^{52} \approx 4.5 \times 10^{15}$$

$$2^{-52} \approx 2.2 \times 10^{-16}$$