03 Dec' 24

Floating point numbers =

$0.3\overline{9} = 0.3999\text{-----}$

$x = 0.3999\text{---}$

$10x = 3.9999\text{---}$

$x = 0.3999\text{---}$

$9x = 3.6000\text{----}$

$x = \frac{3.6}{9} = 0.4$; $0.3\overline{9} = 0.4$

Repeting decimals (with infinite number of digits after decimal point) cannot be

shown in python. Python can only show a finite (not infinite) number of digits after decimal point.
That's why when we do floating point arithmetic, sometimes numbers are slightly incorrect.

0.4 can be 0.3999999998 in python, while actually it is $0.3\overline{9}$.

$\frac{10}{7} = 1.\overline{428571}$, but python shows it as 1.4285714285714286
This is called finite precision (accuracy), which

tells the difference between actual value and the value shown in python.

For example, when you do

2.3 * 4.6 in python, it shows

result = 10.57999999....98

actual value = 10.5800000...00

python result = 10.5799.....98

actual - python values = 00.000000000002

This is a very small difference that doesn't matter when we work with large numbers.

→ For using very large or very small numbers, we use exponential notation in python.

$2e10 = 2 \times 10^{10} = 20,000,000,000$.

This is in the form $ae b$ where $a \in \mathbb{Q}, b \in \mathbb{Z}$

$3e-4 = 3 \times 10^{-4} = 3 \times 0.0001 = 0.0003$

$1.2e^{-10} = 1.2 \times 10^{-10} = 0.000000000012$

↓ Python shows this

↓ not this since it is very small.

$1.2e^{-4} = 1.2 \times 10^{-4} = 0.00012$

↓ Python shows this as it's not very small

HW: Do these manually and in Python and when results are different, explain why.

(1) 2e3 + 5e-1

(2) 1.6e2 * 4e-3

(3) 1e-4 / 2e-2

(4) (3e3) **2

(5) 1.2345e6

(6) 1.2345e-12

(7) 0.00008789e3

(8) 4e-1 + 2e1

(9) 0.1 + 0.2 == 0.3

(10) 1000000000.0 + 1 == 1000000001.0

(11) 1000000000.1 + 1.2 == 1000000001.3

→ HW: Write these numbers in exponential notation and also check in python (by subtracting)

(1) 123400 == 1.234e5

(2) 0.000567

(3) 8000000000

(4) 149,600,000

→ $0.1 + 0.2 == 0.3$

May return 'false' in python because

$0.1 + 0.2$ can be 0.3 (or) 0.30000001

(or) 0.29999999998 in python

as Python only has finite precision.

→ <u>HW:</u> Write a function that takes three floating point numbers a, b, c as parameters and returns $[a + (b * c)] - [a / (b + c)]$.

Use this function 10 times with different parameters.