# Floats vs the Decimal package: What to do when $1/5 \neq 0.2$

Elizabeth Goodman

Hackbright Academy

*elizabethsqg@gmail.com*

January 23, 2017

## Overview

## Checking things are equal

Computation with integers? Great!

Python's even pretty good at really large integers.

Computation with floats? Python will deceive you!

(But really it's a hardware problem.)

## Checking things are equal

Computation with integers? Great!

Python's even pretty good at really large integers.

Computation with floats? Python will deceive you!

(But really it's a hardware problem.)

```
>>> 1.1+2.2
3.3000000000000003
```

Why does that matter?

$1.1+2.2 == 3.3$ returns False. So do a lot of other things.

## We can't see the rounding errors, but they're happening

I wrote a function to read amounts on a receipt from a file, convert to floats, add them up, then check if they equal the subtotal written at the bottom.

### Good

```
>>>print subtotal
123.12
>>>print written_subtotal
123.12
```

## We can't see the rounding errors, but they're happening

I wrote a function to read amounts on a receipt from a file, convert to floats, add them up, then check if they equal the subtotal written at the bottom.

### Good

```
>>>print subtotal
123.12
>>>print written_subtotal
123.12
```

### Bad

```
>>>subtotal == written_subtotal
False
>>>subtotal
123.11999999999999
```

## Binary expansion of fractions

### 1/3 is still a problem

| Number | Decimal | Binary | Float (52 places) |
|---|---|---|---|
| $1/3$ | 0.333... | 0.0101010101... | $1.0101 \ldots 11 \times 2^{-2}$ |

## Binary expansion of fractions

### 1/3 is still a problem

| Number | Decimal | Binary | Float (52 places) |
|---|---|---|---|
| $1/3$ | 0.333... | 0.0101010101... | $1.0101\ldots 11 \times 2^{-2}$ |

### But 1/10 is also a problem

| Number | Decimal | Binary | Float (52 places) |
|---|---|---|---|
| $1/10$ | 0.1 | $0.000110011\ldots$ | $1.100...11010 \times 2^{-3}$ |

## Binary expansion of fractions

### 1/3 is still a problem

| Number | Decimal | Binary | Float (52 places) |
|---|---|---|---|
| $1/3$ | 0.333... | 0.0101010101... | $1.0101\ldots11 \times 2^{-2}$ |

### But 1/10 is also a problem

| Number | Decimal | Binary | Float (52 places) |
|---|---|---|---|
| $1/10$ | 0.1 | $0.000110011\ldots$ | $1.100...11010 \times 2^{-3}$ |

### Actual value of 0.1 stored as a float (on my computer):

```
.1000000000000000055511151231257827021181583404541015625
```

## from decimal import Decimal

### From the Python docs:

"...computers must provide an arithmetic that works in the same way as the arithmetic that people learn at school."

## from decimal import Decimal

### From the Python docs:

"...computers must provide an arithmetic that works in the same way as the arithmetic that people learn at school."

### A Decimal object looks like

`Decimal('2.20')` prints as `2.20` (no formatting needed!)

## from decimal import Decimal

### From the Python docs:

"...computers must provide an arithmetic that works in the same way as the arithmetic that people learn at school."

### A Decimal object looks like

`Decimal('2.20')` prints as `2.20` (no formatting needed!)

### Things that work well

- Trailing 0s and displaying them
- String formatting
- Sort works numerically (10.00 >2.00 unlike strings)
- Accurate comparisons
- Customize rounding and precision
- `Decimal ('2.0') == Decimal ('2.00') == 2`
- My receipt function!

## Decimal Do's and Don'ts

| Do: | Don't: |
|---|---|
| Convert from string<br>`Decimal('1.10')` | Convert from float<br>`Decimal(1.10)` |

## Decimal Do's and Don'ts

| Do: | Don't: |
|---|---|
| Convert from string `Decimal('1.10')` | Convert from float `Decimal(1.10)` |
| Mix with ints `Decimal('1.10')+2` | Mix with floats `Decimal('1.10')+2.0` |

## Decimal Do's and Don'ts

| Do: | Don't: |
|---|---|
| Convert from string `Decimal('1.10')` | Convert from float `Decimal(1.10)` |
| Mix with ints `Decimal('1.10')+2` | Mix with floats `Decimal('1.10')+2.0` |
| Round using the .quantize method `Decimal('.126').quantize(Decimal('.01'))` | Round function `round(Decimal('.126'), 2)` |

## Questions?

Things I can answer:

- Why 52 places??
- What about speed?
- Binary expansions
- Offline: other stuff about floating point computations
- Maybe some other questions