Fraction

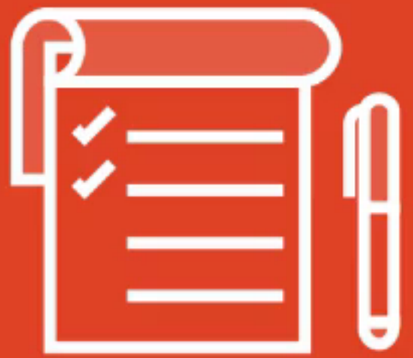
Austin Bingham
COFOUNDER - SIXTY NORTH

@austin_bingham

Robert Smallshire
COFOUNDER - SIXTY NORTH

@robsmallshire

Overview



`fractions.Fraction`

Store integral numerator and denominator to represent rational numbers exactly

Construction from `float` and `str`

Be aware of pitfalls

Arithmetic with `Fraction`

Using the standard `math` module

fractions.Fraction

$$\frac{2}{3}$$

2 ← numerator → 4

3 ← demoninator → 5

Must not be zero!

fractions.Fraction

```
>>> from fractions import Fraction
>>> two_thirds = Fraction(2, 3)
>>> two_thirds
Fraction(2, 3)
>>> four_fifths = Fraction(4, 5)
>>> four_fifths
Fraction(4, 5)
>>> Fraction(5, 0)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/Users/sixty-north/.pyenv/versions/3.8.0/lib/python3.8/fractions.py", li
ne 178, in __new__
    raise ZeroDivisionError('Fraction(%s, 0)' % numerator)
ZeroDivisionError: Fraction(5, 0)
>>> Fraction(933262154439441526816992388562)
Fraction(933262154439441526816992388562, 1)
>>>
```

fractions.Fraction

```
>>> Fraction(0.5)
Fraction(1, 2)
>>> Fraction(0.1)
Fraction(3602879701896397, 36028797018963968)
>>> Fraction(Decimal('0.1'))
Fraction(1, 10)
>>> Fraction('22/7')
Fraction(22, 7)
>>> Fraction(2, 3) + Fraction(4, 5)
Fraction(22, 15)
>>> Fraction(2, 3) - Fraction(4, 5)
Fraction(-2, 15)
>>> Fraction(2, 3) * Fraction(4, 5)
Fraction(8, 15)
>>> Fraction(2, 3) / Fraction(4, 5)
Fraction(5, 6)
>>> Fraction(2, 3) // Fraction(4, 5)
0
>>> Fraction(2, 3) % Fraction(4, 5)
Fraction(2, 3)
>>>
```

`Fraction` does not support `sqrt()`

The result of `sqrt()` could be irrational

This is not representable with `Fraction`

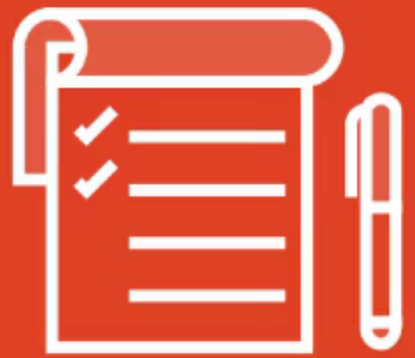
fractions.Fraction

```
>>> from math import floor
>>> floor(Fraction('4/3'))
1
>>>
```

Trade-offs with Numeric Types

Numeric Types	Qualities
`int`	precision
`float`	exactness
`Decimal`	convenience
`Fraction`	performance

Summary



`Fraction` represents rational numbers

Can be constructed from 1 or 2 integers

Raise `ZeroDivisionError` for zero denominators

Can be constructed from `floats`

Subject to inexact `float` representations

Can construct from `Decimal` without surprises

Supports standard arithmetic

Does not support square root

Can be passed to `math.floor()` and `math.ceil()`