RATIONAL NUMBERS

Rational numbers are fractions of integer numbers

Ex: $\frac{1}{2}$ $-\frac{22}{7}$

Any real number with a finite number of digits after the decimal point is also a rational number

$0.45 \rightarrow \frac{45}{100}$ $0.123456789 \rightarrow \frac{123456789}{10^9}$

So $\frac{8.3}{4}$ is also rational $\frac{8.3}{4} = \frac{83/10}{4} = \frac{83}{10} \times \frac{1}{4} = \frac{83}{40}$

as is $\frac{8.3}{1.4}$ since $\frac{8.3}{1.4} = \frac{83/10}{14/10} = \frac{83}{10} \times \frac{10}{14} = \frac{83}{14}$

The Fraction Class

Rational numbers can be represented in Python using the Fraction class in the fractions module

```
from fractions import Fraction

x = Fraction(3, 4)
y = Fraction(22, 7)
z = Fraction(6, 10)
```

Fractions are automatically reduced:

```
Fraction(6, 10) → Fraction(3, 5)
```

Negative sign, if any, is always attached to the numerator:

```
Fraction(1, -4) → Fraction(-1, 4)
```

Constructors

```
Fraction(numerator=0, denominator=1)

Fraction(other_fraction)

Fraction(float)

Fraction(decimal)

Fraction(string)
```

```
    Fraction('10')     → Fraction(10, 1)

    Fraction('0.125')  → Fraction(1, 8)

    Fraction('22/7')   → Fraction(22, 7)
```

Standard arithmetic operators are supported: `+, -, *, /`

and result in `Fraction` objects as well

$$\frac{2}{3} \times \frac{1}{2} = \frac{2}{6} = \frac{1}{3}$$

```
Fraction(2, 3) * Fraction(1, 2) → Fraction(1, 3)
```

$$\frac{2}{3} + \frac{1}{2} = \frac{4}{6} + \frac{3}{6} = \frac{7}{6}$$

```
Fraction(2, 3) + Fraction(1, 2) → Fraction(7, 6)
```

getting the numerator and denominator of `Fraction` objects:

```
x = Fraction(22, 7)
x.numerator          → 22
x.denominator        → 7
```

`float` objects have finite precision ⇒ any `float` object can be written as a fraction!

```
Fraction(0.75)     → Fraction(3, 4)
Fraction(1.375)    → Fraction(11, 8)
```

```
import math

x = Fraction(math.pi)          → Fraction(884279719003555, 281474976710656)
y = Fraction(math.sqrt(2))     → Fraction(6369051672525773, 4503599627370496)
```

Even though π and $\sqrt{2}$ are both irrational

internally represented as floats

⇒ finite precision real number

⇒ expressible as a rational number

but it is an approximation

Converting a `float` to a `Fraction` has an important caveat

We'll examine this in detail in a later video on floats

$\frac{1}{8}$ has an exact float representation

`Fraction(0.125)` → `Fraction(1, 8)`

$\frac{3}{10}$ does not have an exact float representation

```
Fraction(0.3)          → Fraction(5404319552844595, 18014398509481984)
format(0.3, '.5f')     → 0.30000
format(0.3, '.25f')    → 0.2999999999999999888977698
```

Constraining the denominator

Given a `Fraction` object, we can find an approximate equivalent fraction with a constrained denominator

using the `limit_denominator(max_denominator=1000000)` instance method

i.e. finds the closest rational (which could be precisely equal) with a denominator that does not exceed `max_denominator`

```
x = Fraction(math.pi)             → Fraction(884279719003555, 281474976710656)
                                    3.141592653589793

x.limit_denominator(10)           → Fraction(22, 7)
                                    3.142857142857143

x.limit_denominator(100)          → Fraction(311, 99)
                                    3.141414141414141

x.limit_denominator(500)          → Fraction(355, 113)
                                    3.141592920353983
```

Code