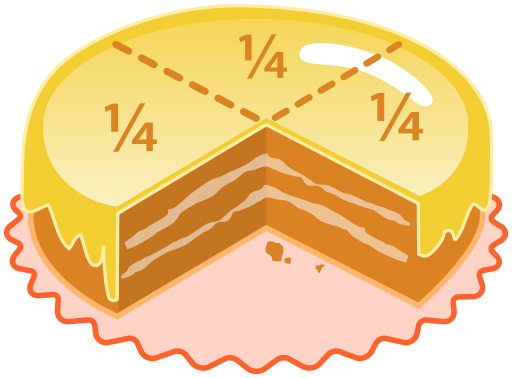
Mathematics with Python and Ruby/Fractions in Ruby



A fraction is nothing more than the exact quotient of an integer by another one. As the result is not necessarily a decimal number, fractions have been used way before the decimal numbers: They are known at least since the Egyptians and Babylonians. And they still have many uses nowadays, even when they are more or less hidden like in these examples:

1. If a man is said to be 5'7" high, this means that his height, in feet, is $5 + \frac{7}{12} = \frac{67}{12}$.
2. Hearing that it is 8 hours 13, one can infer that, past midday, $8 + \frac{13}{60} = \frac{493}{60}$ hours have passed.
3. If a price is announced as *three quarters* it means that, in dollars, it is $\frac{3}{4}$: Once again, a fraction!
4. Probabilities are often given as fraction (mostly of the egyptian type). Like in "There is one chance over 10 millions that a meteor fall on my head" or "Stallion is favorite at five against one".
5. Statistics too like fractions: "5 people over 7 think there are too many surveys".

The equality *0.2+0.5=0.7* can be written as $\frac{2}{10}+\frac{5}{10} = \frac{7}{10}$ but conversely, $\frac{1}{2}+\frac{1}{3} = \frac{5}{6}$ cannot be written as a decimal equality because such an equality would not be exact.

To enter a fraction, the *Rational* object is used:

```
a=Rational(24,10) puts(a)
```

The simplification is automatic. An other way is to use *mathn*, which changes the behavior of the *slash* operator:

```
require 'mathn' a=24/10 puts(a)
```

It is also possible to get a fraction from a real number with its *to_r* method. Yet the fraction is ensured to be correct only if its denominator is a power of 2[1]:

```
a=1.2 b=a.to_r puts(b)
```

In this case, *to_r* from String is more exact:

```
puts "1.2".to_r #=> (6/5) puts "12/10".to_r #=> (6/5)
```

1 Numerator

To get the numerator of a fraction f, one enters *f.numerator*:

```
a=Rational(24,10) puts(a.numerator)
```

Notice, the result is **not** 24, because the fraction will be reduced to 12/5.

2 Denominator

To get the denominator of a fraction f, one enters *f.denominator*:

```
a=Rational(24,10) puts(a.denominator)
```

3 Value

An approximate value of a fraction is obtained by a conversion to a *float*:

```
a=Rational(24,10) puts(a.to_f)
```

4 Unary operations

4.1 Negation

Like any number, the negation of a fraction is obtained while preceding its name by the minus sign "-":

a=Rational(2,−3) puts(-a)

4.2 inverse

To invert a fraction, one divides 1 by this fraction:

a=Rational(5,4) puts(1/a)

5 Binary operations

5.1 Addition

To add two fractions, one uses the "+" symbol, and the result will be a fraction.

a=Rational(34,21) b=Rational(21,13) puts(a+b)

If a non-fraction is added to a fraction, the resulting type will vary. Adding a fraction and an integer will result in a fraction. But, adding a fraction and a float results in a float.

5.2 Subtraction

Likewise, to subtract two fractions, one writes the *minus* sign between them:

a=Rational(34,21) b=Rational(21,13) puts(a-b)

5.3 Multiplication

The product of two fractions will ever be a fraction either:

a=Rational(34,21) b=Rational(21,13) puts(a*b)

5.4 Division

The integer quotient and remainder are still defined for fractions:

a=Rational(34,21) b=Rational(21,13) puts(a/b) puts(a%b)

5.5 Exponentiation

If the exponent is an integer, the power of a fraction will still be a fraction:

a=Rational(3,2) puts(a**12) puts(a**(−2))

But if the exponent is a float, even if the power is actually a fraction, *Ruby* will give it as a float:

a=Rational(9,4) b=a**0.5 puts(b) puts(b.to_r)

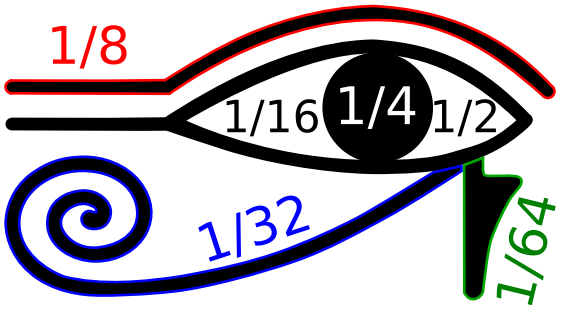
6 Farey mediant

Ruby has no method to compute the Farey mediant of two fractions, but it is easy to create it with a *definition*:

def Farey(a,b) n=a.numerator+b.numerator d=a.denominator+b.denominator return Rational(n,d) end a=Rational(3,4) b=Rational(1,13) puts(Farey(a,b))

7 Egyptian fractions

To write a fraction like the Egyptians did, one can use Fibonacci's algorithm:



def egypt(f) e=f.to_i f-=e list=[e] begin e=Rational(1,(1/f).to_i+1) f-=e list.push(e) end while f.numerator>1 list.push(f) return list end require 'mathn' a=21/13 puts(egypt(a))

The algorithm can be summed up like this:

1. One extracts the integer part of the fraction (with *to_i*) and stores it in a list;
2. One subtracts to f (the remaining fraction) the largest integer inverse possible;
3. And so on while the numerator of f is larger than one.
4. Finally one adds the last fraction to the list.

8 Notes

[1] Well, it is true that $\frac{5404319552844595}{4503599627370496} \simeq \frac{6}{5}$ but anyway...

9 Text and image sources, contributors, and licenses

9.1 Text

- **Mathematics with Python and Ruby/Fractions in Ruby** *Source:* https://en.wikibooks.org/wiki/Mathematics_with_Python_and_Ruby/Fractions_in_Ruby?oldid=3121715 *Contributors:* Recent Runes, Whym, Alain Busser and Anonymous: 5

9.2 Images

- **File:Cake_quarters.svg** *Source:* https://upload.wikimedia.org/wikipedia/commons/b/b9/Cake_quarters.svg *License:* Public domain *Contributors:* Own work *Original artist:* Acdx, R. S. Shaw
- **File:Oudjat.SVG** *Source:* https://upload.wikimedia.org/wikipedia/commons/f/f5/Oudjat.SVG *License:* CC-BY-SA-3.0 *Contributors:* ? *Original artist:* ?

9.3 Content license

- Creative Commons Attribution-Share Alike 3.0