Class **Rational** < Numeric

Rational numbers are expressed as the ratio of two integers. When the denominator exactly divides the numerator, a rational number is effectively an integer. Rationals allow exact representation of fractional numbers, but some real values cannot be expressed exactly and
1.9 so cannot be represented as rationals.

Class Rational is normally relatively independent of the other numeric classes, in that the result of dividing two integers with the / operator will normally be a (truncated) integer (the quo method will always return a rational result). However, if the mathn library is loaded into a program, integer division may generate a Rational result. Also see the rational library on page 796 for additional methods on rational numbers.

```
r1 = Rational("1/2")    # =>   1/2
r2 = 4.quo(5)           # =>   4/5
r1 * r2                 # =>   2/5
```

Instance methods

Arithmetic operations

Performs various arithmetic operations on self.

self	+	*numeric*	Addition
self	–	*numeric*	Subtraction
self	*	*numeric*	Multiplication
self	/	*numeric*	Division
self	%	*numeric*	Modulo
self	**	*numeric*	Exponentiation
self	-@		Unary minus

Comparisons

Compares self to other numbers.

<, <=, ==, >=, and >.

<=>

self <=> *numeric* → −1, 0, +1

Comparison—Returns −1, 0, or +1 depending on whether self is less than, equal to, or greater than *numeric*. Although Rational's grandparent, mixes in Comparable, Rational does not use that module for performing comparisons, instead implementing the comparison operators explicitly.

```
Rational("4/2") <=> Rational("98/49")   # =>   0
Rational("3/4") <=> 41                  # =>   -1
Rational("0") <=> 0.0                   # =>   0
```

==

self == *numeric*

Returns true is self has the same value as *numeric*. Comparisons against integers and rational numbers are exact; comparisons against floats first convert self to a float.

ceil

self.ceil → *numeric*

Returns the smallest integer greater than or equal to self.

```
Rational("22/7").ceil    # =>   4
Rational("-22/7").ceil   # =>   -3
```

denominator

self.denominator → *a_number*

Returns the denominator of self.

```
Rational("2/3").denominator   # =>   3
```

div

self.div(*numeric*) → *integer*

Returns the integral result of dividing self by *numeric*.

```
Rational("11/2") / 2     # =>   11/4
Rational("11/2").div 2   # =>   2
```

fdiv

self.fdiv(*numeric*) → *float*

Returns the floating-point result of dividing self by *numeric*.

```
Rational("11/2") / 2      # =>   11/4
Rational("11/2").fdiv 2   # =>   2.75
```

floor

self.floor → *numeric*

Returns the largest integer less than or equal to self.

```
Rational("22/7").floor    # =>   3
Rational("-22/7").floor   # =>   -4
```

numerator

self.numerator → *a_number*

Returns the numerator of self.

```
Rational("2/3").numerator   # =>   2
```

quo

self.quo(*numeric*) → *numeric*

1.9

Synonym for Rational#/.

round

self.round → *numeric*

Rounds self to the nearest integer.

```
Rational("22/7").round    # =>   3
Rational("-22/7").round   # =>   -3
```

to_f

self.to_f → *float*

Returns the floating-point representation of self.

```
Rational("37/4").to_f   # =>   9.25
```

to_i

self.to_i → *integer*

Returns the truncated integer value of self.

```
Rational("19/10").to_i    # =>   1
Rational("-19/10").to_i   # =>   -1
```

to_r

self.to_r → self

Returns self.

truncate

self.truncate → *numeric*

Returns self truncated to an integer.

```
Rational("22/7").truncate    # =>   3
Rational("-22/7").truncate   # =>   -3
```