Class

Complex < Numeric

1.9

Represents complex numbers, represented internally as numbers with a real and imaginary part, both of which can be any scalar number. Note that scalar comparison operations (<=>, <, and so on) are not defined on complex numbers (which would argue that Complex should not be a subclass of Numeric, but that ship has sailed). Also see the standard library, somewhat confusingly named complex, on page 737, for a way add complex number support to standard math functions, as well as the mathn library on page 767 for a way of integrating complex numbers into regular arithmetic (so that the square root of −1 returns Complex::I).

```
v1 = Complex(2,3)                                  # =>   (2+3i)
v2 = Complex("0+2i")   # Alternative constructor   # =>   (0+2i)
v1 + v2                                            # =>   (2+5i)
v1 * v2                                            # =>   (-6+4i)
v2**2                                              # =>   (-4+0i)
v2**2 == -4                                        # =>   true

# Euler's theorem
include Math
E**(PI*Complex::I)   # =>   (-1.0+1.22464679914735e-16i)
```

Class constants

I The imaginary unit.

Class methods

polar

Complex.polar(*magnitude*, *angle*) → *complex*

Returns the complex number represented by the given polar coordinates.

```
Complex.polar(1.23, 0.5)       # =>   1.07942655112516+0.589693412483l7i
Complex.polar(1, Math::PI/2)   # =>   6.12323399573677e-17+1.0i
```

rect

Complex.rect(*read*, *imag*) → *complex*

Returns the complex number represented by the given real and imaginary parts.

```
Complex.rect(1.23, 0.5)   # =>   1.23+0.5i
```

rectangular

Complex.rectangular(*read*, *imag*) → *complex*

Synonym for Complex.rect.

Instance methods

Arithmetic operations

Performs various arithmetic operations on *complex*.

complex	+	*numeric*	Addition
complex	–	*numeric*	Subtraction
complex	*	*numeric*	Multiplication
complex	/	*numeric*	Division
complex	**	*numeric*	Exponentiation
complex	-@		Unary minus
complex	-+		Unary plus

==

complex == *other* → true or false

Returns true if *complex* does equals *other*, converting *other* to a complex number if necessary.

```
Complex::I == Complex(0,1)   # =>   true
Complex::I == Complex(1,0)   # =>   false
Complex(1,0) == 1            # =>   true
Complex(1,0) == "1"          # =>   false
```

abs

complex.abs → *number*

Returns the absolute value (magnitude) of *complex*.

```
Complex::I.abs     # =>   1.0
Complex(1,1).abs   # =>   1.4142135623731
```

abs2

complex.abs2 → *number*

Returns the square of the absolute value (magnitude) of *complex*.

```
Complex::I.abs2     # =>   1
Complex(1,1).abs2   # =>   2
```

angle

complex.angle → *number*

Returns the angle between the x-axis and a line from the origin to *complex*. By convention, Complex(0,0).angle is 0.

```
Complex(1, 0).angle   # =>   0.0
Complex(1, 1).angle   # =>   0.785398163397448
Complex(0, 1).angle   # =>   1.5707963267949
```

arg

complex.arg → *number*

Synonym for Complex#angle.

conj

complex.conj → *a_complex*

Synonym for Complex#conjugate.

conjugate

complex.conjugate → *a_complex*

Returns the conjugate of *complex* (the reflection of *complex* around the x-axis).

```
Complex::I.conjugate     # =>   (0-1i)
Complex(1,1).conjugate   # =>   (1-1i)
```

denominator

complex.denominator → *number*

Returns the lowest common multiple of the denominators of the real and imaginary parts of *complex*.

```
Complex("1/3+1/4i").denominator   # =>   12
Complex(-2, 4).denominator        # =>   1
```

eql?

complex.eql(*other*) → true or false

Returns true only if *other* is a complex number with real and imaginary parts eql? to *complex*'s.

```
Complex(1, 0).eql?(Complex(1,0))      # =>   true
Complex(1, 0).eql?(Complex(1.0, 0))   # =>   false
Complex(1, 0).eql?(1)                 # =>   false
Complex(1, 0) == Complex(1,0)         # =>   true
Complex(1, 0) == Complex(1.0, 0)      # =>   true
Complex(1, 0) == 1                    # =>   true
```

fdiv

complex.fdiv(*other*) → *a_complex*

Returns *complex* / *other* after converting the real and imaginary parts of *complex* to floats. (Contrast with Complex#quo.)

```
c1 = Complex(1, 2)
c2 = Complex(2, 2)
c1 /c2        # =>   ((3/4)+(1/4)*i)
c1.fdiv(c2)   # =>   (0.75+0.25i)
```

imag

complex.imag → *number*

Returns the imaginary part of *complex*.

```
Complex(2, -3).imag   # =>   -3
```

imaginary

complex.imaginary → *number*

Synonym for Complex#imag.

magnitude

complex.magnitude → *int* or *float*

Returns the magnitude of *complex*(the distance of *complex* from the origin of the number line. The positive square root of $real^2 + imag^2$.

```
Complex(3, 4).magnitude   # =>   5.0
Complex::I.magnitude      # =>   1.0
```

numerator

complex.numerator → *a_complex*

If cd is *complex*.denominator and re and im are the real and imaginary parts of *complex*, *complex*.numerator is as follows:

$$re.numerator \times \frac{cd}{re.denominator} + im.numerator \times \frac{cd}{im.denominator} i$$

phase

complex.phase → [*magnitude*, *angle*]

Returns the phase angle of *complex* (the angle between the positive x-axis and the line from the origin to $(real, imag)$), measured in radians.

```
Complex(3, 4).phase    # =>  0.927295218001612
Complex(-3, 4).phase   # =>  2.21429743558818
```

polar

complex.polar → [*magnitude*, *angle*]

Returns *complex* as polar coordinates.

```
Complex(1,1).polar     # =>  [1.4142135623731, 0.785398163397448]
Complex(-2,-3).polar   # =>  [3.60555127546399, -2.15879893034246]
```

quo

complex.quo(*other*) → *a_complex*

Returns *complex* / *other* after converting the real and imaginary parts of *complex* to rational numbers. (Contrast with Complex#fdiv.)

```
c1 = Complex(1, 2)
c2 = Complex(2, 2)
c1 /c2       # =>  ((3/4)+(1/4)*i)
c1.quo(c2)   # =>  ((3/4)+(1/4)*i)
```

rect

complex.rect → [*complex*.real, *complex*.imag]

Returns an array containing the real and imaginary components of *complex*.

```
Complex::I.rect   # =>  [0, 1]
```

rectangular

complex.rectangular → [*complex*.real, *complex*.imag]

Synonym for Complex#rect.

real

complex.real → *number*

Returns the real part of *complex*.

```
Complex(2, 3).real   # =>  2
```

real?

complex.real? → *false*

Complex numbers are never real numbers (even if their imaginary part is zero).

```
Complex(1, 1).real?   # =>  false
Complex(1, 0).real?   # =>  false
```

to_f

complex.to_f → *float*

Returns the real part of *complex* as a float, raising an exception if the imaginary part is not zero.

```
Complex(2, 0).to_f   # =>   2.0
```

to_i

complex.to_i → *float*

Returns the real part of *complex* as an integer, raising an exception if the imaginary part is not zero.

```
Complex(2.2, 0).to_i   # =>   2
```

to_r

complex.to_r → *float*

Returns the real part of *complex* as a rational number, raising an exception if the imaginary part is not zero.

```
Complex(2.5, 0).to_r   # =>   (5/2)
```