Class Fixnum < Integer

A Fixnum holds Integer values that can be represented in a native machine word (minus 1 bit). If any operation on a Fixnum exceeds this range, the value is automatically converted to a Bignum.

Fixnum objects have immediate value. This means that when they are assigned or passed as parameters, the actual object is passed, rather than a reference to that object. Assignment does not alias Fixnum objects. Because there is effectively only one Fixnum object instance for any given integer value, you cannot, for example, add a singleton method to a Fixnum.

Instance methods

Arithmetic operations

Performs various arithmetic operations on *fix*.

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

Bit operations

Performs various operations on the binary representations of the Fixnum.

~ *fix*			Invert bits
fix	|	*numeric*	Bitwise OR
fix	&	*numeric*	Bitwise AND
fix	^	*numeric*	Bitwise EXCLUSIVE OR
fix	<<	*numeric*	Left-shift *numeric* bits
fix	>>	*numeric*	Right-shift *numeric* bits (with sign extension)

Comparisons

Compares *fix* to other numbers. Fixnum.

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

<=>

fix <=> *numeric* → $-1, 0, +1$

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

```
42 <=> 13   # =>   1
13 <=> 42   # =>   -1
-1 <=> -1   # =>   0
```

[ ] *fix*[*n*] → 0, 1

Bit Reference—Returns the *n*th bit in the binary representation of *fix*, where *fix*[0] is the least significant bit.

```
a = 0b11001100101010
30.downto(0) {|n| print a[n] }
```

produces:

```
0000000000000000011001100101010
```

abs *fix*.abs → *int*

Returns the absolute value of *fix*.

```
-12345.abs    # =>   12345
12345.abs     # =>   12345
```

div *fix*.div(*numeric*) → *integer*

1.9

Division that always produces an integral result. Not affected by the mathn library (unlike Fixnum#/).

```
654321.div(13731)      # =>   47
654321.div(13731.34)   # =>   47
```

even? *fix*.even? → true or false

1.9

Returns true is *fix* is even.

```
1.even?   # =>   false
2.even?   # =>   true
```

divmod *fix*.divmod(*numeric*) → *array*

See Numeric#divmod on page 617.

fdiv *fix*.fdiv(*numeric*) → *float*

1.9

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

```
63.fdiv(9)              # =>   7.0
654321.fdiv(13731)      # =>   47.6528293642124
654321.fdiv(13731.24)   # =>   47.6519964693647
```

magnitude *fix*.magnitude → *int*

1.9

Returns the magnitude of *fix* (the distance of *fix* from the origin of the number line). Synonym for Fixnum#abs. See also Complex#magnitude.

modulo *fix*.modulo(*numeric*) → *numeric*

Synonym for Fixnum#%.

```
654321.modulo(13731)      # =>   8964
654321.modulo(13731.24)   # =>   8952.72000000001
```

odd?

1.9

fix.odd? → true or false

Returns true if *fix* is odd.

```
1.odd?   # =>  true
2.odd?   # =>  false
```

size

fix.size → *int*

Returns the number of *bytes* in the machine representation of a Fixnum.

```
1.size           # =>  4
-1.size          # =>  4
2147483647.size  # =>  4
```

succ

1.9

fix.succ → *int*

Returns *fix* + 1.

```
1.succ    # =>  2
-1.succ   # =>  0
```

to_f

fix.to_f → *float*

Converts *fix* to a Float.

to_s

fix.to_s(*base*=10) → *string*

Returns a string containing the representation of *fix* radix *base* (2 to 36).

```
12345.to_s                             # =>  "12345"
12345.to_s(2)                          # =>  "11000000111001"
12345.to_s(8)                          # =>  "30071"
12345.to_s(10)                         # =>  "12345"
12345.to_s(16)                         # =>  "3039"
12345.to_s(36)                         # =>  "9ix"
84823723233035811745497171.to_s(36)  # =>  "anotherrubyhacker"
```

zero?

fix.zero? → true or false

Returns true if *fix* is zero.

```
42.zero?   # =>  false
0.zero?    # =>  true
```