Class
Bignum < Integer

Bignum objects hold integers outside the range of Fixnum. Bignum objects are created automatically when integer calculations would otherwise overflow a Fixnum. When a calculation involving Bignum objects returns a result that will fit in a Fixnum, the result is automatically converted.

For the purposes of the bitwise operations and [], a Bignum is treated as if it were an infinite-length bitstring with 2's complement representation.

While Fixnum values are immediate, Bignum objects are not—assignment and parameter passing work with references to objects, not the objects themselves.

Instance methods

Arithmetic operations

Performs various arithmetic operations on *big*.

big	+	*number*	Addition
big	–	*number*	Subtraction
big	*	*number*	Multiplication
big	/	*number*	Division
big	%	*number*	Modulo
big	**	*number*	Exponentiation
big	-@		Unary minus

Bit operations

Performs various operations on the binary representations of the Bignum.

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

<=>

big <=> *number* → −1, 0, +1

Comparison—Returns −1, 0, or +1 depending on whether *big* is less than, equal to, or greater than *number*. This is the basis for the tests in Comparable.

==

big == *obj* → true or false

Returns true only if *obj* has the same value as *big*. Contrast this with Bignum#eql?, which requires *obj* to be a Bignum.

```
68719476736 == 68719476736.0   # =>   true
```

[ ]

big[*n*] → 0, 1

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

```
a = 9**15
50.downto(0) do |n|
  print a[n]
end
```

produces:

```
00010111011010000011100001111001010011110001011100 1
```

abs

big.abs → *bignum*

Returns the absolute value of *big*.

```
1234567890987654321.abs    # =>   1234567890987654321
-1234567890987654321.abs   # =>   1234567890987654321
```

div

big.div(*number*) → *other_number*

Synonym for Bignum#/.

```
-1234567890987654321.div(13731)        # =>   -89910996357706
-1234567890987654321.div(13731.0)      # =>   -89910996357705
-1234567890987654321.div(-987654321)   # =>   1249999989
```

divmod

big.divmod(*number*) → *array*

See Numeric#divmod on page 617.

eql?

big.eql?(*obj*) → true or false

Returns true only if *obj* is a Bignum with the same value as *big*. Contrast this with Bignum#==, which performs type conversions.

```
68719476736.eql? 68719476736     # =>   true
68719476736  ==  68719476736     # =>   true
68719476736.eql? 68719476736.0   # =>   false
68719476736  ==  68719476736.0   # =>   true
```

fdiv

big.fdiv(*number*) → *float*

1.9

Returns the floating-point result of dividing *big* by *number*. Alias for Bignum#quo.

```
-1234567890987654321.fdiv(13731)        # =>   -89910996357705.5
-1234567890987654321.fdiv(13731.0)      # =>   -89910996357705.5
-1234567890987654321.fdiv(-987654321)   # =>   1249999989.60938
```

magnitude

big.magnitude → *bignum*

1.9

Returns the magnitude of *big*(the distance of *big* from the origin of the number line. Synonym for Bignum#abs. See also Complex#magnitude.

modulo

big.modulo(*number*) → *number*

Synonym for Bignum#%.

remainder

big.remainder(*number*) → *other_number*

Returns the remainder after dividing *big* by *number*.

```
-1234567890987654321.remainder(13731)      # =>  -6966
-1234567890987654321.remainder(13731.24)   # =>  -9906.22531493148
```

size

big.size → *integer*

Returns the number of bytes in the machine representation of *big*.

```
(256**10 - 1).size   # =>  12
(256**20 - 1).size   # =>  20
(256**40 - 1).size   # =>  40
```

to_f

big.to_f → *float*

Converts *big* to a Float. If *big* doesn't fit in a Float, the result is infinity.

to_s

big.to_s(*base*=10) → *str*

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

```
12345654321.to_s          # =>  "12345654321"
12345654321.to_s(2)       # =>  "1011011111110110111011110000110001"
12345654321.to_s(8)       # =>  "133766736061"
12345654321.to_s(16)      # =>  "2dfdbbc31"
12345654321.to_s(26)      # =>  "1dp1pc6d"
78546939656932.to_s(36)   # =>  "rubyrules"
```