Class

# Integer < Numeric

Subclasses: Bignum, Fixnum

Integer is the basis for the two concrete classes that hold whole numbers, Bignum and Fixnum. (If you've come here looking for the iterator step, it's on page 621.)

## Instance methods

### ceil

*int*.ceil → *integer*

Synonym for Integer#to_i.

### chr

*int*.chr → *string*

Returns a string containing the ASCII character represented by the receiver's value.

```
65.chr    # =>  "A"
?a.chr    # =>  "a"
230.chr   # =>  "\xE6"
```

### denominator

*int*.denominator → *integer*

1.9

Converts the denominator of the rational representation of *int*.

```
1.denominator     # =>  1
1.5.denominator   # =>  2
num = 1.0/3
num.to_r          # =>  (6004799503160661/18014398509481984)
num.denominator   # =>  18014398509481984
```

### downto

*int*.downto( *integer* ) {| *i* | *block* } → *int*

Iterates *block*, passing decreasing values from *int* down to and including *integer*.

```
5.downto(1) {|n| print n, ".. " }
print "  Liftoff!\n"
```

*produces:*

```
5.. 4.. 3.. 2.. 1..   Liftoff!
```

### even?

*int*.even? → true or false

1.9

Returns true if *int* is even.

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

### floor

*int*.floor → *integer*

Returns the largest integer less than or equal to *int*. Equivalent to Integer#to_i.

```
1.floor      # =>  1
(-1).floor   # =>  -1
```

### gcd

*int*.gcd( *other_integer* ) → *integer*

1.9

Returns the greatest common denominator of *int* and *other_integer*.

```
10.gcd(15)   # =>   5
10.gcd(16)   # =>   2
10.gcd(17)   # =>   1
```

### gcdlcm

*int*.gcdlcm( *other_integer* ) → [ *gcd*, *lcm* ]

1.9

Returns both the GCD and LCM of *int* and *other_integer*.

```
10.gcdlcm(15)   # =>   [5, 30]
10.gcdlcm(16)   # =>   [2, 80]
10.gcdlcm(17)   # =>   [1, 170]
```

### integer?

*int*.integer? → true

Always returns true.

### lcm

*int*.lcm( *other_integer* ) → *integer*

1.9

Returns the lowest common multiple of *int* and *other_integer*.

```
10.lcm(15)   # =>   30
10.lcm(20)   # =>   20
10.lcm(-2)   # =>   10
```

### next

*int*.next → *integer*

Returns the Integer equal to *int* + 1.

```
1.next      # =>   2
(-1).next   # =>   0
```

### numerator

*int*.numerator → *integer*

1.9

Converts the numerator of the rational representation of *int*.

```
1.numerator     # =>   1
1.5.numerator   # =>   3
num = 1.0/3
num.to_r        # =>   (6004799503160661/18014398509481984)
num.numerator   # =>   6004799503160661
```

### odd?

*int*.odd? → true or false

1.9

Returns true is *int* is odd.

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

### ord

*int*.ord → *int*

1.9

The ord method was added to assist in the migration from Ruby 1.8 to 1.9. It allows ?A.ord to return 65. If ?A returns a string, ord will be called on that string and return 65; if ?A returns an integer, then Numeric#ord is called, which is basically a no-op.

**pred** *int*.pred → *integer*

1.9 Returns *int* – 1.

**round** *int*.round → *integer*

Synonym for Integer#to_i.

**succ** *int*.succ → *integer*

Synonym for Integer#next.

**times** *int*.times {| *i* | *block* } → *int*

Iterates block *int* times, passing in values from zero to *int* −1.

```
5.times do |i|
  print i, " "
end
```

*produces:*

```
0 1 2 3 4
```

**to_i** *int*.to_i → *int*

Returns *int*.

**to_int** *int*.to_int → *integer*

Synonym for Integer#to_i.

**to_r** *int*.to_r → *number*

1.9 Converts *int* to a rational number.

```
1.to_r    # =>  1/1
-1.to_r   # =>  -1/1
```

**truncate** *int*.truncate → *integer*

Synonym for Integer#to_i.

**upto** *int*.upto( *integer* ) {| *i* | *block* } → *int*

Iterates *block*, passing in integer values from *int* up to and including *integer*.

```
5.upto(10) {|i| print i, " " }
```

*produces:*

```
5 6 7 8 9 10
```