Ruby master - Feature #4574

Numeric#within

04/13/2011 09:05 PM - mame (Yusuke Endoh)

Status:	Rejected	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	
Target version:	2.0.0	

Description

=begin
Hello,

Many people have written programs that limits an integer/float
within a range like:

v = [[v, min].max, max].min

or

v = v < min ? min : (v < max ? v : max)

or

if v < min
v = min
elsif max < v
v = max
end

. But these are all too complex in spite of frequent idiom.
"[min, v, max].sort[1]" is cool, but slightly cryptic.

So I propose Numeric#within.

v = v.within(min, max)

Examples:

p 0.within(2, 5) #=> 2
p 2.within(2, 5) #=> 2
p 3.within(2, 5) #=> 3
p 5.within(2, 5) #=> 5
p 6.within(2, 5) #=> 6

What do you think?

Some Japanese committers agree with this idea, and proposed
other candidates of method name:

- Numeric#bound
- Numeric#limit
- Numeric#clip
- Numeric#into
- Numeric#crop
- Numeric#trim
- Range#bound # usage: (2..5).bound(0)

Anyway, I think that the length should be less than 9, because
it should be shorter than the sort idiom:

[min, v, max].sort[1]
v.xxxxxxxxx(min, max)

Here is a patch including both Numeric#within and Range#bound.

diff --git a/numeric.c b/numeric.c
index 34c378b..dad485f 100644
--- a/numeric.c
+++ b/numeric.c
@@ -1600,6 +1600,43 @@ num_round(int argc, VALUE* argv, VALUE num)
return flo_round(argc, argv, rb_Float(num));
}

+static int
+r_le(VALUE a, VALUE b)
+{

- int c;
- VALUE r = rb_funcall(a, rb_intern("<=>"), 1, b); +
- if (NIL_P(r))
- return (int)Qfalse;
- c = rb_cmpint(r, a, b);
- if (c == 0)
- return (int)INT2FIX(0);
- if (c < 0)
- return (int)Qtrue;
- return (int)Qfalse; +} + +/*
- * call-seq:
- *   num.within(min, max) -> new_num
- *
- * Bounds num so that min <= new_num <= max.
- * Returns min when num < min, max when num > end, otherwise
- * num itself.
- */ + +static VALUE +num_within(VALUE num, VALUE min, VALUE max) +{
- if (r_le(min, num)) {
- if (r_le(max, num)) {
- return max;
- }
- return num;
- }
- return min; +} + /*
    - call-seq:
    - num.truncate -> integer @@ -3424,6 +3461,7 @@ Init_Numeric(void) rb_define_method(rb_cNumeric, "round", num_round, -1); rb_define_method(rb_cNumeric, "truncate", num_truncate, 0); rb_define_method(rb_cNumeric, "step", num_step, -1);

- rb_define_method(rb_cNumeric, "within", num_within, 1);

  rb_cInteger = rb_define_class("Integer", rb_cNumeric);
  rb_undef_alloc_func(rb_cInteger);
  diff --git a/range.c b/range.c
  index 1866df1..2dd99f5 100644
  --- a/range.c
  +++ b/range.c
  @@ -923,6 +923,41 @@ range_cover(VALUE range, VALUE val)
  return Qfalse;
  }

+/*

- * call-seq:
- *   rng.bound(val) -> new_val
- *
- * Bounds val so that beg <= new_val <= end. This method returns
- * beg when val < ben, end when val > end, otherwise, val itself.
- * Raises an exception if val >= end and the range is exclusive.
- *
- *   (2..5).bound(0) #=> 2

- *   (2..5).bound(2) #=> 2
- *   (2..5).bound(3) #=> 3
- *   (2..5).bound(5) #=> 5
- *   (2..5).bound(6) #=> 5
- *   (2...5).bound(6) #=> ArgumentError
- */ + +static VALUE +range_bound(VALUE range, VALUE val) +{
- VALUE beg, end; +
- beg = RANGE_BEG(range);
- end = RANGE_END(range);
- if (r_le(beg, val)) {
- if (r_le(end, val)) {
- if (EXCL(range)) {
- rb_raise(rb_eArgError, "more than or equal to the excluded range");
- }
- return end;
- }
- return val;
- }
- return beg; +} + static VALUE range_dumper(VALUE range) { @@ -1051,4 +1086,5 @@ Init_Range(void) rb_define_method(rb_cRange, "member?", range_include, 1); rb_define_method(rb_cRange, "include?", range_include, 1); rb_define_method(rb_cRange, "cover?", range_cover, 1);
- rb_define_method(rb_cRange, "bound", range_bound, 1); }

--
Yusuke Endoh mame@tsg.ne.jp
=end

Related issues:

Related to Ruby master - Feature #5549: Comparable#min, Comparable#max	**Rejected**	**11/03/2011**

History

#1 - 04/13/2011 09:42 PM - mrkn (Kenta Murata)

=begin
I like Numeric#limit and/or Numeric#clip. I think "limit" is more mathematical, and "clip" is more easy to understand its functionality from its name. So I believe it is acceptable that the both names are provided.

In contrast, I feel that Numeric#within returns boolean so I don't want to accept it.

=end

#2 - 04/13/2011 09:49 PM - sakuro (Sakuro OZAWA)

=begin
I prefer this sort of method be an instance method of Range.

You propse Range#bound, but 'bound' as a verb is similar to jump/leap (and its intransitive), isn't it? Do you mean past particle form of 'bind' ?

Anyway I propose Range#clamp as its name.

Examples:

- (())
- (())

=end

#3 - 04/13/2011 09:52 PM - sakuro (Sakuro OZAWA)

=begin
For Japanese: clamp □□□□□□□□□□□□/□□□□□□□□□:)

=end

#4 - 04/13/2011 10:23 PM - mame (Yusuke Endoh)

=begin
Hello,

2011/4/13 redmine@ruby-lang.org:

> I prefer this sort of method be an instance method of Range.

I can accept Range method, but why?
I think that the subject of this feature is the limited Numeric.

> You propse Range#bound, Â but 'bound' as a verb is similar to jump/leap (and its intransitive), isn't it? Do you mean past particle form of 'bind' ?

It resembles:

http://en.wikipedia.org/wiki/Upper_and_lower_bounds

But I did not think the name so deeply.
Another name is welcome.

> Anyway I propose Range#clamp as its name.
>
> Examples:
>
> - (())
> - (())

Thanks for the information.
The name would be good if it is really conventional.

--
Yusuke Endoh mame@tsg.ne.jp
=end

#5 - 04/13/2011 11:31 PM - Eregon (Benoit Daloze)

=begin
Hello,
Kenta Murata wrote:

> I like Numeric#limit and/or Numeric#clip. I think "limit" is more mathematical, and "clip" is more easy to understand its functionality from its name. So I believe it is acceptable that the both names are provided.

I agree, I find #limit and #clip more explicit than the others of the list (and #clamp) ...

> In contrast, I feel that Numeric#within returns boolean so I don't want to accept it.

... but I prefer #within, the fact it does not have a final '?' is enough for me to not expect a boolean.

And that is the only name which do not make me think twice to know what it does.

Yusuke Endoh wrote:

> Some Japanese committers agree with this idea

It's nice to also see this discussion on ruby-core.

> I can accept Range method, but why?
> I think that the subject of this feature is the limited Numeric.

A Range method does not make much sense to me either (and it creates a likely useless Range).

So I prefer Numeric#within(lower_bound, upper_bound).
=end

#6 - 04/14/2011 12:23 AM - mame (Yusuke Endoh)

=begin
Hello,

2011/4/13 redmine@ruby-lang.org:

> > I like Numeric#limit and/or Numeric#clip. Â I think "limit" is more mathematical, and "clip" is more easy to understand its functionality from its name. Â So I believe it is acceptable that the both names are provided.

> I agree, I find #limit and #clip more explicit than the others of the list (and #clamp) ...

Thank you, but I'm going to like clamp :-)
"clamp" seems not only the term of computer graphics, but also
the term of mathematics. Kenta found the following article:

http://en.wikipedia.org/wiki/Saturation_arithmetic

> If the result of an operation is greater than the maximum it is set ("clamped") to the maximum, while if it is below the minimum it is clamped to the minimum.
>
> Yusuke Endoh wrote:
>
>> Some Japanese committers agree with this idea
>
> It's nice to also see this discussion on ruby-core.

Note that I have not gotten matz's approval yet :-)

>> I can accept Range method, but why?
>> I think that the subject of this feature is the limited Numeric.

I forgot to tell another issue of Range method.
It is difficult to define the behavior of "(2...5).bound(6)".
My previous patch lets the code raise an ArgumentError, but
I'm not sure if it is right.

--
Yusuke Endoh mame@tsg.ne.jp
=end

#7 - 04/14/2011 12:19 PM - kosaki (Motohiro KOSAKI)

=begin
Gack!
simple paste made RD format error. ;-)
Let's retry.

Btw, Linux kernel has following macro.

```
/**
 * clamp - return a value clamped to a given range with strict typechecking
 * @val: current value
 * @min: minimum allowable value
 * @max: maximum allowable value
 *
 * This macro does strict typechecking of min/max to make sure they are of the
 * same type as val.  See the unnecessary pointer comparisons.
 */
#define clamp(val, min, max) ({                 \
        typeof(val) __val = (val);              \
        typeof(min) __min = (min);              \
        typeof(max) __max = (max);              \
        (void) (&__val == &__min);              \
        (void) (&__val == &__max);              \
        __val = __val < __min ? __min: __val;   \
        __val > __max ? __max: __val; })
```

=end

#8 - 04/14/2011 12:30 PM - kosaki (Motohiro KOSAKI)

=begin
More off topic. Here is very similar discussion by phthonia.

http://stackoverflow.com/questions/4092528/how-to-clamp-an-integer-to-some-range-in-python
=end

#9 - 04/14/2011 07:49 PM - aprescott (Adam Prescott)

=begin
I think Numeric#within is perhaps less intuitive than something like Range#bound or Range#trim. It feels more natural to have it be a method on Range. I like Range#clamp but it might not be so obvious that restricts the argument to within the range; then again, (2..5).restrict(6) might lack some aesthetic appeal.
=end

#10 - 04/15/2011 07:23 AM - cjheath (Clifford Heath)

=begin
On 14/04/2011, at 9:40 PM, Nikolai Weibull wrote:

> On Thu, Apr 14, 2011 at 12:49, redmine@ruby-lang.org wrote:
>
>> I think Numeric#within is perhaps less intuitive than something
>>
>> like Range#bound or Range#trim. It feels more natural to have it be
>>
>> a method on Range. I like Range#clamp but it might not be so
>>
>> obvious that restricts the argument to within the range; then
>>
>> again, (2..5).restrict(6) might lack some aesthetic appeal.
>
> In my mind, Numeric#clamp is the right choice:

clamp is also the term used in electronics for this function.

Clifford Heath.
=end

#11 - 07/10/2011 03:15 PM - kosaki (Motohiro KOSAKI)

- Category set to core

- Status changed from Open to Assigned

- Target version changed from 1.9.3 to 2.0.0

It's too late for 1.9.3.

#12 - 04/10/2012 06:43 PM - matz (Yukihiro Matsumoto)

- Status changed from Assigned to Rejected

I reject this propsal, since the name discussion diverged.
If someone really wants this feature (and believes #clamp is the right name), resubmit the feature request.

Matz.

#13 - 12/15/2014 05:39 AM - findchris (Chris Johnson)

Resubmitted as a new feature request here:
https://bugs.ruby-lang.org/issues/10594