Ruby trunk - Feature #2152

Split functionality of Float#inspect and Float#to_s

09/28/2009 06:45 PM - rogerdpack (Roger Pack)

Status:	Closed	
Priority:	Normal	
Assignee:	matz (Yukihiro Matsumoto)	
Target version:	2.6	

Description

=begin
Reposting here as a feature request.

As a side note, I think I figured out why the confusion occured in posting the original--if one navigates to "feature requests" then click "new issue" it defaults to a Bug report, even though you were browsing feature requests.

==========Proposal==========
Currently the Float#to_s and Float#inspect return an "accurate" representation of the internal float, to help alert users of floating point inconsistencies.

This is better than the old way. Some have recently pointed out that the old functionality has been lost [the old "pretty float" style]. This is surprising to them.

In order to have 1.9.2 add functionality and not also lose the old, I propose to split the function of Float#inspect and Float#to_s thus:

String#to_s will be the pretty way
and
String#inspect will be the more accurate way.

With this in place, in irb users are notified of inaccuracies, and it is less surprising.

I'm not totally set on using those method names--the proposal is that both be available somewhere.

Thanks!
-r
=end

History

#1 - 10/10/2009 01:06 AM - rogerdpack (Roger Pack)

=begin
re: Float#to_s and Float#inspect being split to do different functionality.

Would a patch be accepted?
Thanks.
-r
=end

#2 - 10/10/2009 07:24 AM - marcandre (Marc-Andre Lafortune)

- Category set to core

- Assignee set to matz (Yukihiro Matsumoto)

=begin

=end

#3 - 11/11/2009 12:10 AM - rogerdpack (Roger Pack)

=begin
If no feedback on this then I suppose I shall go ahead and write up a patch for it.
Thanks!
-r
=end

#4 - 01/20/2010 06:11 AM - Eregon (Benoit Daloze)

=begin
+1
(
see discussions
here: http://redmine.ruby-lang.org/issues/show/1841
and here: http://blade.nagaokaut.ac.jp/cgi-bin/vframe.rb/ruby/ruby-core/22325?22249-23532+split-mode-vertical
)
=end

#5 - 01/20/2010 02:01 PM - matz (Yukihiro Matsumoto)

=begin
Hi,

In message "Re: [ruby-core:25820] [Feature #2152] Split functionality of Float#inspect and Float#to_s"
on Mon, 28 Sep 2009 18:45:15 +0900, Roger Pack redmine@ruby-lang.org writes:

|I'm not totally set on using those method names--the proposal is that both be available somewhere.

I am not against provide both ways to represent float numbers, but I am
not sure to assign which behavior to which name, or to add a new
method. We need to investigate more on names.

```
                          matz.
```

=end

#6 - 01/20/2010 07:26 PM - Eregon (Benoit Daloze)

=begin
Hi dear Ruby creator !
|Hi,
|I am not against provide both ways to represent float numbers, but I am
|not sure to assign which behavior to which name, or to add a new
|method. We need to investigate more on names.
| matz.
I don't know exactly why, but it seems very clear to me to have the pretty style at #to_s and the complete, full representation with #inspect.

The fact is I really don't like to write things like '%.2f' % 2.34 (it sounds so old), I would love to simply do "My Float: #{f}".
About inspect, it's the representation used by default in irb, and then is very useful to see the full representation when playing with Float. It's also what Kernel#p uses.

Well, many classes that implements both are more complete in #inspect, and shorter in #to_s. #to_s is, as I think, for printing, while #inspect is more for debugging, or printing a complete representation. (Rational, Complex, String, Array(in 1.8), Hash, Encoding, the default inspect(Kernel#inspect,undocumented))
=end

#7 - 01/22/2010 12:22 PM - RickDeNatale (Rick DeNatale)

=begin
While I can appreciate the desire to make floats a little clearer to beginners. I have to reluctantly pitch in with a -1 on this.

The reason is that such seemingly simple and cosmetic changes have had negative effects on the existing code base far out of proportion to the change.

A case in point is the change to the output of Array#to_s in Ruby 1.9 which has, surprisingly, been one of the most frequent source of bugs I've found in 'porting' code from 1.8 to 1.9, sometimes causing subtle bugs which took a bit more time than I would have liked to track down.

And just the other night, at our local Ruby user's group, one of the more experienced guys mentioned the same thing to me without prompting.

While on one hand I appreciated the billable hours such things have generated, I'd rather have provided a bit more value to the client.
=end

#8 - 02/25/2010 09:33 PM - Eregon (Benoit Daloze)

=begin
Sorry to answer so late.

I think the change of Array#to_s is not comparable here. Many people did use the fact that Array#to_s did what Array#join does now.

But here it's very different: Float are almost not used for their real representation like it is now default with #to_s (it is not intended for them to get a lot of decimals I mean). I think in any app you want to code, you will want to use a fixed representation ('%.2f'), or a nice one (without ...9999999). So, I think not a lot of people use Float#to_s in 1.9.2, because it's not appropriate to show the data, except if you absolutely want to show the errors in Float. I think changing #to_s to the old behavior will break very few code. And would even improve compatibility with earlier versions!

Here is a simple example:
ruby -e 'p 729.0/10'

ruby-1.8.7-p249: 72.9
ruby-1.9.2-head: 72.90000000000001

"While I can appreciate the desire to make floats a little clearer to beginners."
Float#to_s has changed, and we propose to make #to_s like before, while keeping #inspect for the full representation.

This change would improve compatibility and tracking Float errors for beginners (and make you still wise of what is a Float when you are 'inspecting' it).

It would also, according to me, be more consistent with Ruby standards of #to_s and #inspect (I said upper why).
=end

#9 - 03/26/2010 05:25 AM - rogerdpack (Roger Pack)

- File add_float_inspect.diff added

=begin
Here's a patch which restores Float#to_s to its "human readable" form and introduces Float#inspect for the "more verbose" form.

Hopefully it will find approval (another name than inspect would be ok, too, though I prefer inspect so that in irb I can know what is going on).

I'd be happy to add tests or what not, once a decision is made (or even before, if requested).

Thanks!
-rp
=end

#10 - 04/01/2010 02:21 AM - rogerdpack (Roger Pack)

=begin
Matz any feedback on this patch, when you get a chance?
Thanks.
-rp
=end

#11 - 04/01/2010 03:08 AM - marcandre (Marc-Andre Lafortune)

=begin
There are two different issues here.

1) First, there is currently a bug in trunk:

72.9.to_s # ==> "72.90000000000001"

This is plainly wrong. A float is an approximation and it is well-defined defined by how much it is an approximation.

Mathematically speaking, a float can be seen as a tiny range of real numbers; any of these numbers will be approximated to the same float (for example 72.9, 72.90000000000001 and 72.900000000000000000042)

#to_s (and #inspect) should choose the simplest string representation that is included in the approximate range a float represents.

In other words, if string_rep_1.to_f == string_rep_2.to_f, then the simplest of string_rep_1 and string_rep_2 is to be preferred.

72.9.to_s must return "72.9", the simplest representation.

2) Calculations between floats introduce errors due to these approximations. For example:

(1-0.9) == 0.1 # ==> false

This feature request asks that (1-0.9).to_s # => "0.1"

I object to this, because 1-0.9 is not == to 0.1 and thus should not have the same string representation.

I would recommend that instead we consider adding (probably in 1.9.3) an optional argument to Float#to_s that would limit the number of decimal showing, for instance.
=end

#12 - 04/01/2010 03:38 AM - mame (Yusuke Endoh)

=begin
Hi,

2010/4/1 Marc-Andre Lafortune redmine@ruby-lang.org:

> 1) First, there is currently a bug in trunk:
>
> 72.9.to_s # ==> "72.90000000000001"
>
> This is plainly wrong.

I guess that your story is not logical.

Consider 72.90000000000001.to_s. With your suggestion, it will return
"72.9". Don't you complain the behavior?

I think Float#to_s can select any representation. Any representation
is not "wrong", and

> #to_s (and #inspect) should choose the simplest string representation that is included in the approximate range a float represents.

it is the best representation, I agree.

I wonder whether or not the suggestion is included in 1.9.2...

> 2) Calculations between floats introduce errors due to these approximations. For example:
>
> (1-0.9) == 0.1 # ==> false
>
> This feature request asks that (1-0.9).to_s ?# => "0.1"
>
> I object to this, because 1-0.9 is not == to 0.1 and thus should not have the same string representation.

Agreed.

--
Yusuke ENDOH mame@tsg.ne.jp

=end

#13 - 04/01/2010 04:04 AM - Eregon (Benoit Daloze)

=begin

> Issue #2152 has been updated by Marc-Andre Lafortune.
>
> There are two different issues here.
>
> 1) First, there is currently a bug in trunk:
>
> 72.9.to_s # ==> "72.90000000000001"
>
> #to_s (and #inspect) should choose the simplest string representation that is included in the approximate range a float represents.

That's what we ask for #to_s
We'd like to keep #inspect to find easily why a calculation is wrong
(and even if you know how Float are, this security is really not too
much, so much "bugs" can happen)

> In other words, if string_rep_1.to_f == string_rep_2.to_f, then the simplest of string_rep_1 and string_rep_2 is to be preferred.

Agreed for to_s, inspect is here irrelevant.

> 2) Calculations between floats introduce errors due to these approximations. For example:
>
> (1-0.9) == 0.1 # ==> false
>
> This feature request asks that (1-0.9).to_s # => "0.1"

No, but it's one consequence. 1.8 behave like this. Nobody would ever
want to compare the String representations to compare the value...

> I object to this, because 1-0.9 is not == to 0.1 and thus should not have the same string representation.

I think it should. It's in fact inconsistent now to read from a String
"72.9".to_f.to_s
Again, inspect has no matter with it. It's intended for showing full
representation as I think.

> I would recommend that instead we consider adding (probably in 1.9.3) an optional argument to Float#to_s that would limit the number of decimal showing, for instance.

That's a good idea, better than old way "%.20f" % 0.1

I think we really need the old #to_s for 1.9.2, it's definitely
inconsistent the current representation.
Do you agree?

=end

#14 - 04/01/2010 04:39 AM - mame (Yusuke Endoh)

=begin
Hi,

2010/4/1 Benoit Daloze eregontp@gmail.com:

> 1) First, there is currently a bug in trunk:
>
> 72.9.to_s # ==> "72.90000000000001"
>
> #to_s (and #inspect) should choose the simplest string representation that is included in the approximate range a float represents.

> That's what we ask for #to_s

No. You are asking the different behavior from Marc-Andre.

Marc-Andre is insisting it choose the simplest representation
without inaccuracy. But the old behavior you want chooses
a simple one *that may be even inaccurate*.

> 2) Calculations between floats introduce errors due to these approximations. For example:
>
> (1-0.9) == 0.1 # ==> false
>
> This feature request asks that (1-0.9).to_s ?# => "0.1"

> No, but it's one consequence. 1.8 behave like this. Nobody would ever
> want to compare the String representations to compare the value...

Even in 1.8, the following returns the result including error:

p 1.4 - 0.1 - 1.2 #=> 0.0999999999999999

It may cause a bug that is hard to test and reproduce.

Honestly, I used to like the old behavior. But I knew that the
implementation of the old behavior was very uncool and dirty.
It is almost like:

("%.15f" % float).sub(/0+$/, "")

Now I dislike the old behavior.

--
Yusuke ENDOH mame@tsg.ne.jp

=end

#15 - 04/01/2010 06:01 AM - Eregon (Benoit Daloze)

=begin
Hi,
On 31 March 2010 21:39, Yusuke ENDOH mame@tsg.ne.jp wrote:

> Hi,
> No. You are asking the different behavior from Marc-Andre.
>
> Marc-Andre is insisting it choose the simplest representation
> *without inaccuracy*. But the old behavior you want chooses
> a simple one *that may be even inaccurate*.

You're right, I considered simplest as *not so accurate but concise*.

> Even in 1.8, the following returns the result including error:
>
> p 1.4 - 0.1 - 1.2 #=> 0.0999999999999999
>
> It may cause a bug that is hard to test and reproduce.

I never noticed that in 1.8, so I suppose is quite rare.
Maybe we should accept an epsilon, appropriate for the double type,
that depends of the range between 2 double
and then "round" the Float when we want to print.
Or maybe decide of a fixed number of decimals is easier ?

I think nobody likes ...9999999999 when the exact result would be
....1(0000), even if the internal value could never be that number.

Let's make Float beautiful with default printing ;)

> Honestly, I used to like the old behavior. But I knew that the
> implementation of the old behavior was very uncool and dirty.
> It is almost like:
>
> ("%.15f" % float).sub(/0+$/, "")
>
> Now I dislike the old behavior.

For what I have read in the C code, it's in fact very different
between versions.
I think a good idea would to allow that optional parameter that would
basically act as:
def to_s(n = DEFAULT_PRECISION)
"%.#{n}f" % float # Please write this in C as beautiful as you can :)
end
and the default value be according to have a similar result to 1.8.

Well, and if both implementation have pro/cons, then isn't the first
proposition a solution ?
Let people happy with beautiful output when result is (almost) exact,
and let them *inspect* the object to see what's the real value (and
have a better implementation there)

We are most of us agreeing to a cleaner syntax, while we don't want to
cheat and remove too much accuracy

So, double have a range of (2**-52) ~ 2e-16. We then should round at
15~16 digits:

> n = 1.4 - 0.1 - 1.2
> => 0.09999999999999987
> "%.15f" % n
> => "0.100000000000000"
> "%.16f" % n
> => "0.0999999999999999"

15 looks fine to me, because we all know Float aren't accurate with
some operations.

Current implementation go until the 17th digit, which is non-sense for a double.

=end

#16 - 04/04/2010 01:21 AM - znz (Kazuhiro NISHIYAMA)

- Status changed from Open to Assigned

- Target version changed from 1.9.2 to 2.0.0

=begin

=end

#17 - 04/08/2010 02:02 AM - matz (Yukihiro Matsumoto)

=begin
Hi,

In message "Re: [ruby-core:29169] [Feature #2152] Split functionality of Float#inspect and Float#to_s"
on Thu, 1 Apr 2010 02:21:55 +0900, Roger Pack redmine@ruby-lang.org writes:

|Matz any feedback on this patch, when you get a chance?

As I stated in [ruby-core:27631], I don't object about having two
string representation, one "human readable" and the other "more
verbose". Even though I am not fully against, it doesn't mean, I
enjoy having two representation. Float is float is float. We should
know about error when you treat with float, shouldn't we?

Anyway, I feel like we still have two points:

(1) We've got consensus which method to behave which so far. Since
#inspect is a method for "human readable" string representation,
I think it is more suitable to get "human readable" float
representation. Others want to #to_s to be "human readable" to
ease "Float #{f}".

(2) Besides that, I am not sure whether the old behavior is the best
way to generate "human readable" representation.

```
                          matz.
```

=end

#18 - 04/08/2010 02:57 AM - Eregon (Benoit Daloze)

=begin
Hi,
On 7 April 2010 19:02, Yukihiro Matsumoto matz@ruby-lang.org wrote:

> Hi,
>
> As I stated in [ruby-core:27631], I don't object about having two
> string representation, one "human readable" and the other "more
> verbose". Even though I am not fully against, it doesn't mean, I
> enjoy having two representation. Float is float is float. We should
> know about error when you treat with float, shouldn't we?
>
> Anyway, I feel like we still have two points:
>
> (1) We've got consensus which method to behave which so far. Since
> #inspect is a method for "human readable" string representation,
> I think it is more suitable to get "human readable" float
> representation. Others want to #to_s to be "human readable" to
> ease "Float #{f}".
>
> (2) Besides that, I am not sure whether the old behavior is the best
> way to generate "human readable" representation.
>
> ```
>                                                   matz.
> ```
>
> Then let's choose which representation is "human readable", maybe a third
> one?

I spoke a bit about that in my last post, while I fear I'm more of a

beginner about technique:

So, double have a range (between two of them) of 2e-52 ~ 10e-16. Should we
then round at
15~16 digits ?

> n = 1.4 - 0.1 - 1.2
> => 0.09999999999999987
> "%.15f" % n
> => "0.100000000000000"
> "%.16f" % n
> => "0.0999999999999999"

So I think most people would agree with the "%.15f" result (without the
zeros of course).
The question is then: How to do it properly ?

P.S.: To my opinion, #inspect is usually more verbose than #to_s, so I
continue staying straight in my idea.
The bad thing with Float is you currently still need to "my Float: #{"%.2f"
% f}"
Inspect is, to my opinion, used mostly when debugging, almost never to print
result.
So if there is a verbose representation, I think it belongs to #inspect.
(I expect to have precise, even too verbose output when I 'p f')

#19 - 04/08/2010 08:30 PM - mame (Yusuke Endoh)

=begin
Hi,

2010/4/8 Benoit Daloze eregontp@gmail.com:

> So, double have a range (between two of them) of 2e-52 ~ 10e-16. Should we then round at
> 15~16 digits ?

There is no silver bullet against floating point error :-)

"%.15f" % (1.9 + 0.7 - 0.2) #=> "2.399999999999999"

--
Yusuke ENDOH mame@tsg.ne.jp

=end

#20 - 04/15/2010 04:10 AM - Eregon (Benoit Daloze)

=begin

> The option previously proposed is
>
> Float#to_s : human
> Float#inspect : verbose
>
> Another option:
>
> Float#to_s : verbose
> Float#inspect : verbose
> Float#to_hs : human (or to_ps or what not, for pretty string--not sure
> if there is a convention for that name or not)
>
> Any feedbacks or preferences there? I'd probably be ok either way.
> -rp
>
> I think a new name would not be so welcomed (and confusing).

Why to have two methods (to_s and inspect) doing the same thing?
They are not supposed to be aliased,
this is the case in many objects (except the ones having a unique perfect
representation).
So let's change one !

I think it's time to decide.
And I think the initial proposition is the relevant one (name convention,
backward compatibility, useful, ...).

Who (else) is disagreeing to this feature ?
... and who is agreeing ?

#21 - 04/19/2010 07:45 AM - Eregon (Benoit Daloze)

=begin
Sorry for my noisy messages with html on redmine, all my apologies.

I just send here back the last message from ruby-core (which seems to not be on redmine):

On 15 April 2010 22:38, Roger Pack wrote:

> The option previously proposed is
>
> Float#to_s : human
> Float#inspect : verbose
>
> Another option:
>
> Float#to_s : verbose
> Float#inspect : verbose
> Float#to_hs : human (or to_ps or what not, for pretty string--not sure

> Why to have two methods (to_s and inspect) doing the same thing?
> They are not supposed to be aliased,
> this is the case in many objects (except the ones having a unique perfect
> representation).

Good point.
Because of this +1 for first option mentioned, from me.
It would seem in this instance that we do actually prefer two methods,
and two already exist for us to use.

Thanks!
-rp

Can we still decide for this or is it too late ?
=end

#22 - 04/21/2010 11:54 PM - rogerdpack (Roger Pack)

=begin
It appears that some others agree, as well:

http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-core/23123
http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-core/23126
http://blade.nagaokaut.ac.jp/cgi-bin/scat.rb/ruby/ruby-core/22710

And probably a few others I didn't dig up.
-rp
=end

#23 - 05/11/2010 02:18 AM - Eregon (Benoit Daloze)

=begin
On 10 May 2010 18:59, Roger Pack rogerdpack2@gmail.com wrote:

> I recently discovered Float#round the other day, if that's of any use.
>
> 1.1-0.9
> => 0.20000000000000007
>
> (1.1-0.9).round(2)
> => 0.2

Sure, I did know about it, just forgot in this conversion.

So that makes pointless the idea of the supplementary argument to #to_s

So the real deal is still there (and for *many* messages):

Should we accept #to_s returns a pretty-human-like representation of Float,
or do we want to keep current-more-realistic representation for both
#inspect and #to_s?

(But then having in code such horrible "%.2f" % f or "f: #{f.round(2)}")

I already gave my opinion and arguments
(compatibility, consistency with core/stdlib, usefulness,
user-friendly, handy, not useless double method, ...)

Please, those who are not according, be explicit ;)

Best Regards,
B.D.

=end

#24 - 10/27/2012 04:56 AM - ko1 (Koichi Sasada)

- Description updated

Who's ball?

#25 - 11/24/2012 09:26 AM - mame (Yusuke Endoh)

- Target version changed from 2.0.0 to 2.6

#26 - 10/22/2017 01:38 AM - mame (Yusuke Endoh)

- Status changed from Assigned to Closed

The discussion has been stalled for seven years, and there are no strong complaint. I think we can close this ticket.

Files

add_float_inspect.diff	2.17 KB	03/26/2010	rogerdpack (Roger Pack)