Ruby trunk - Feature #9528

mathn.rb library

02/18/2014 01:28 PM - umair.amjad (Umair Amjad)

Status:	Rejected	
Priority:	Normal	
Assignee:	hsbt (Hiroshi SHIBATA)	
Target version:		

Description

I want to add factorial method mathn.rb file as feature of Math module.

History

#1 - 02/19/2014 11:11 AM - umair.amjad (Umair Amjad)

Please guide how can I contribute, I have code written on my local.

#2 - 02/22/2014 05:20 AM - charliesome (Charlie Somerville)

- Priority changed from 5 to Normal

Hi Umair,

You should attach a .patch file and wait for feedback.

#3 - 02/23/2014 08:55 AM - zzak (Zachary Scott)

- Category set to lib

- Status changed from Open to Feedback

- Assignee set to zzak (Zachary Scott)

- Target version set to 2.2.0

If you're using subversion you can use the svn diff or svn di command to output a patch, and then just upload the file. For example:

svn diff lib/mathn.rb > my-patch-to-mathn.diff

You can read more about svn diff in the svnbook

Bonus points, when requesting a feature please try to give as many details as possible about your feature:

1) What is your proposed change?
2) Why would people use it? (use cases)
3) Why should this be added to ruby?

Are just a few of the questions you could try to answer. We have some more detailed documentation on contributing.rdoc

#4 - 07/27/2014 09:28 PM - zzak (Zachary Scott)

- Status changed from Feedback to Closed

Its been 5 months without any feedback, so I'm closing this.

If you have any specific questions about how to contribute, please feel free to reply or email the list ruby-core@ruby-lang.org or email me personally at zzak@ruby-lang.org

May the Ruby be with you..

#5 - 11/09/2014 06:34 AM - martin_vahi (Martin Vahi)

- File the_code.rb added

- File run_demo.bash added

Well, I have the same wish, except that I also have a demo code available.

A speed optimized demo can be downloaded from

```
http://longterm.softf1.com/2014/demos/2014_11_08_ruby_factorial_proposal/the_code.rb
```

and run by

```
http://longterm.softf1.com/2014/demos/2014_11_08_ruby_factorial_proposal/run_demo.bash
```

For 100000.factorial the speed difference is literally roughly 20-fold (not just 20%).

#6 - 11/09/2014 07:54 AM - hsbt (Hiroshi SHIBATA)

- Status changed from Closed to Open

#7 - 11/10/2014 06:38 PM - gogotanaka (Kazuki Tanaka)

I like your propose. But I'd be glad if Math.gamma(x) could make sense for you :)

http://www.ruby-doc.org/core-2.1.4/Math.html#method-c-gamma

Even if we'er gonna add new method, except mathn might be better.
Because mathn became deprecated. #10169

Thanks, gogo.

#8 - 11/13/2014 07:01 AM - martin_vahi (Martin Vahi)

I wasn't aware of the existence of the gamma function before reading Your comment. I guess I got a bit smarter due to Your comment. Thank You for that. :-)

According to some sources, including the

```
http://mathworld.wolfram.com/GammaFunction.html
```

it seems to me that the gamma function is an approximation. I think that a clean solution for functions that are based on approximations should always have a maximum error size as a second argument. For example,

```
sin(x)
```

is actually calculated through series and is never absolutely correct. Therefore the

```
sin(x)
cos(x)
gamma(x)
etc...
```

should be

```
sin(x,absolute_value_of_max_error=<some default value>)
cos(x,absolute_value_of_max_error=<some default value>)
gamma(x,absolute_value_of_max_error=<some default value>)
etc...
```

The IEEE_754

```
https://en.wikipedia.org/wiki/IEEE_floating_point
```

determines some "default" error "size" through its rounding. Due to the exponent mechanism of the IEEE_754, the same property that gives

```
fd_big=(9**99).to_f
puts "No difference detected." if fd_big == (fd_big+1.to_f)
```

there is no single minimum approximation-result-changing value for the error size. Therefore, to find a clean solution for the proper implementation of the gamma/sin/cos/etc. function(s), further work has to be done and that's probably going to be pretty complex and time consuming. However, it is a fact that the current Math.gamma(x) implementation is flawed, because it gives IEEE_754 "infinity" for Math.gamma(10000). That probably limits cryptography related experiments.

The good news is that it seems (at least to me) that dependency wise factorial of integers is very general. Even some forms of the gamma(x) formulae depend on factorials of integers. That's why it seems to me that the proposed

```
Fixnum.factorial
Bignum.factorial
```

do not clutter the stdlib. That is to say, as of my current comment, I stick with my initial proposal.

Well, one way or the other, I still thank You all for Your answers and efforts. :-)

#9 - 01/05/2018 09:00 PM - naruse (Yui NARUSE)

- Target version deleted (2.2.0)

#10 - 01/31/2018 07:18 AM - hsbt (Hiroshi SHIBATA)

- Assignee changed from zzak (Zachary Scott) to hsbt (Hiroshi SHIBATA)

- Status changed from Open to Rejected

mathn.rb has been removed at Ruby 2.5.0 [Feature #10169]

#11 - 01/31/2018 01:43 PM - martin_vahi (Martin Vahi)

Given that my proposal is more general than the obsoleted mathn.rb,
would it be helpful, if I filed a new Feature Request that would
suggest an addition of a new class to the Ruby stdlib?

For example, the function apply_binary_operator might be
wrapped to a class named "Math_optimizations" and
the watershed concatenation algorithm would be the default
heuristic for the

```
    Math_optimizations.apply_binary_operator
```

The heuristic might be later changed by adding
an optional 4. argument to the function call.
The signature would look like:

```
    def Math_optimizations.apply_binary_operator(
        x_identity_element,array_in,
        func_operator_that_might_be_noncommutative,
        heuristic="watershed_concatenation")
```

Files

the_code.rb	7.49 KB	11/09/2014	martin_vahi (Martin Vahi)
run_demo.bash	591 Bytes	11/09/2014	martin_vahi (Martin Vahi)