Ruby master - Feature #4897

Define Math::TAU and BigMath.TAU. The "true" circle constant, Tau=2*Pi. See http://tauday.com/

06/18/2011 07:46 AM - sbaird (Simon Baird)

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

Description

Firstly please read the Tau Manifesto at http://tauday.com/ . It's quite long but essential to understanding why this is a good idea.

Here is a patch on trunk that implements this:
http://simonbaird.blogspot.com/2011/06/tau-in-ruby.html

Allow me to anticipate and respond in advance to some common objections:

1. It's easy to define it yourself so why put this in core.
   Possibly correct, but I think this is the right thing to do. Tau is important. And it's a pretty small patch.

2. If this constant goes in then pretty soon someone will want every other math constant and there are hundreds of them. (Slippery slope argument).
   The circle constant is one of the two most important numbers is mathematics. It's not just another math constant. We already define Pi.

Related issues:

Related to Ruby master - Feature #13694: Add Tau constant to Math	**Rejected**
Has duplicate Ruby master - Feature #17496: Add constant Math::TAU	**Open**

History

#1 - 06/18/2011 08:26 AM - sbaird (Simon Baird)

Sorry for the too long title. Don't know how to edit. Suggested title:

Define Math::TAU and BigMath.TAU. The "true" circle constant, Tau=2*Pi

#2 - 06/18/2011 08:28 AM - sbaird (Simon Baird)

Direct link to the gist containing my patch:
https://gist.github.com/1029552

Edit: mention this is a patch (pluid61's patch is better though)

#3 - 06/18/2011 08:33 AM - sbaird (Simon Baird)

See http://tauday.com/

Edit: particularly http://tauday.com/tau-manifesto

#4 - 03/25/2012 04:23 PM - mame (Yusuke Endoh)

- Category changed from core to Joke

- Status changed from Open to Feedback

#5 - 03/25/2012 04:29 PM - duerst (Martin Dürst)

We have discussed this at today's developers' meeting in Akihabara.

We highly doubt that there are many mathematicians, physicists, engineers, and so on, who use τ. Once τ is widely accepted in these communities, we might add it.

Just as a personal comment, I prefer π to τ because I can eat a pie, but not a tau :-)

#6 - 03/25/2012 11:57 PM - trans (Thomas Sawyer)

I'm trying to get used to the idea of eating a pizza taue, myself. :-)

This is the classic chicken and egg situation -- "We'll do it if it's popular", but "It won't get popular unless people do it".

I think Tau=2*Pi is a good idea. So if it were just up to me, I'd add it just to show support. It's a rather tiny and harmless addition.

#7 - 10/27/2012 06:13 AM - ko1 (Koichi Sasada)

- Target version changed from 2.0.0 to 3.0

#8 - 02/24/2013 07:17 PM - trans (Thomas Sawyer)

Could this patch be applied now? As previously said, it's a good thing to show support for and it's a rather tiny and otherwise harmless addition.

#9 - 02/25/2013 10:34 PM - sbaird (Simon Baird)

https://github.com/search?q=%22TAU+%3D+2+%2A+Math.PI%22&type=Code

https://github.com/search?q=%22TAU+%3D+PI+*+2%22&type=Code

#10 - 02/26/2013 06:20 PM - nobu (Nobuyoshi Nakada)

Why is it called as τ, half of π?

#11 - 02/26/2013 07:12 PM - phluid61 (Matthew Kerwin)

nobu (Nobuyoshi Nakada) wrote:

> Why is it called as τ, half of π?

It's actually two of π. The reason for the name is justified here: http://tauday.com/tau-manifesto#sec:one_turn

In summary: tau is the first letter of the Greek word "tornos" (lathe), which is the root of the English word "turn;" and the tau constant (2*PI) is the ratio of a circle's radius to its circumference (i.e. one turn.) Also "the horizontal line in each letter suggests that we interpret the “legs” as denominators, so that π has two legs in its denominator, while τ has only one. Seen this way, the relationship τ=2π is perfectly natural."

#12 - 02/26/2013 07:23 PM - phluid61 (Matthew Kerwin)

On 26 February 2013 19:20, nobu (Nobuyoshi Nakada) nobu@ruby-lang.orgwrote:

> Why is it called as τ, half of π?

It's actually two of π. The reason for the name is justified here:
http://tauday.com/tau-manifesto#sec:one_turn

In summary: tau is the first letter of the Greek word "tornos" (lathe),
which is the root of the English word "turn;" and the tau constant (2*PI)
is the ratio of a circle's radius to its circumference (i.e. one turn.)
Also "the horizontal line in each letter suggests that we interpret the
"legs" as *denominators*, so that π has two legs in its denominator,
while τ has only one. Seen this way, the relationship τ=2π is perfectly
natural."

EDIT: apologies for the double-posting. I'm not quite sure how I managed it.

#13 - 02/27/2013 09:26 AM - Student (Nathan Zook)

Please, just say no. This garbage is one and only thing that has made me really glad that I decided to leave academia. Having to dissuade every crank from this idea would ruin my mood for weeks.

There a large number of excellent candidate constants to be included. There is absolutely no cause to include constants that are power-of-two multiples of each other. You want tau in math? Fine. module Math ; TAU = 2 * PI ; end. Done. Put that in all your files. Move it to your initialization code. Publish a gem. I don't care. But don't waste the time of people who understand that shifting a binary point by 1 really isn't a big deal.

-Math::PI

#14 - 02/27/2013 01:41 PM - bug (Harrison Reiser)

Wow, such vitriol ! I could point out a correlation between it and your stance on academia, but I would be flying in the face of not only courtesy but also the whole point of all this. I assure you that tau is not a joke, and has everything to do with mere common sense, both in pedagogy and practice (which, as it were, seems to be what many have come to know Ruby for).

While, as you say, not having the full circle constant is "not a big deal" given a simple means to derive it, I would be shocked to learn that any

practiced programmer makes use of pi alone more often than 2*pi. For this very reason many libraries for other languages include both constants, not just for coding efficiency but for runtime efficiency. So far, this constant has almost ubiquitously been named TWO_PI (or some variation thereof) simply by convention, but the fact of its prevalence should be enough indication of its independent usefulness (an important distinction, one which I have no intention of debating further, having already been, repeatedly and at length -- see https://en.wikipedia.org/wiki/User:Waldir/Tau for examples).

Now, dynamic languages such as Ruby and Python stand in a unique position. As intuitiveness and coder efficiency are among their primary goals, they indeed have no reason to include a constant named TWO_PI, which would overall detract from these goals. However, a constant named TAU would serve as an indication that this number has some distinction of its own, namely the aforementioned independent usefulness. Therefore if it is to be included at all it should be with conviction rather than the attitude of mere quiet tolerance (as of a vocal minority), as it would indeed stand as a statement to the programming community.

The decision to postpone its addition until recognition spreads further is reasonable, but as trans mentioned it perpetuates the chicken and egg problem. Either way, I hope I have raised sufficient objection to categorizing this issue as a "Joke".

-bug

#15 - 02/27/2013 02:08 PM - drbrain (Eric Hodel)

Despite all the argument for this constant, nobody has provided a patch, so it seems like a joke.

Runtime performance of a constant lookup is 3% +/- 1.2% faster than multiplying a float by 2 on my machine over 20 million multiplications. 3% for a built-in constant doesn't seem like a big deal.

I don't trust a wikipedia user page as a reference, it doesn't go through the same vetting process as a regular wikipedia page.

Benchmarks:

```
$ cat bm.rb
require 'benchmark'

module Math
  TAU = 2*Math::PI
end

range = Float::EPSILON..10.0

N = Integer ARGV.shift

case ARGV.shift
when 'tau' then
  puts Benchmark.measure {
    N.times do
      2 * Math::PI * rand(range)
    end
  }.real
when '2pi' then
  puts Benchmark.measure {
    N.times do
      2 * Math::TAU * rand(range)
    end
  }.real
else
  abort "#{$0} N tau|2pi"
end
$ for i in `jot 20`; do ruby bm.rb 1_000_000 2pi; done > 2pi.txt
$ for i in `jot 20`; do ruby bm.rb 1_000_000 tau; done > tau.txt
$ ministat tau.txt 2pi.txt
x tau.txt
+ 2pi.txt
+--------------------------------------------------------------------------+
|x    xx    x    +   +     +    +      +                                   |
|x x x*xxx+ x * x+   * ++x +++  +     ++        +                          +|
| |____M_A___|_|_____________MA_________________|                           |
+--------------------------------------------------------------------------+
    N           Min           Max        Median           Avg        Stddev
x  20      0.486589      0.504393      0.491397    0.49242015  0.0049904288
+  20      0.490699      0.544828      0.506571     0.5073239   0.011828075
Difference at 95.0% confidence
  0.0149037 +/- 0.00581011
  3.02663% +/- 1.17991%
  (Student's t, pooled s = 0.00907766)
```

#16 - 02/27/2013 02:13 PM - trans (Thomas Sawyer)

> Despite all the argument for this constant, nobody has provided a patch, so it seems like a joke.

"Here is a patch on trunk that implements this: http://simonbaird.blogspot.com/2011/06/tau-in-ruby.html"

#17 - 02/27/2013 02:25 PM - phluid61 (Matthew Kerwin)

- File tau.patch added

drbrain (Eric Hodel) wrote:

> Despite all the argument for this constant, nobody has provided a patch, so it seems like a joke.

The gist linked in Comment 2 was a diff/patch, althought it's outdated and no longer applies cleanly. I've attached a new patch (generated using git diff). Please let me know if I'm supposed to generate it with --no-prefix

#18 - 02/27/2013 02:52 PM - drbrain (Eric Hodel)

- Status changed from Feedback to Assigned

- Assignee set to matz (Yukihiro Matsumoto)

The patch is fine.

#19 - 02/27/2013 05:23 PM - duerst (Martin Dürst)

Hello Eric,

I'm confused by the code below. First, it uses Math::PI with the 'tau' option, and Math::TAU with the '2pi' option. Second, even when using Math::TAU, it includes a multiplication by 2.

Regards, Martin.

On 2013/02/27 14:08, drbrain (Eric Hodel) wrote:

> Benchmarks:

```
$ cat bm.rb
require 'benchmark'

module Math
  TAU = 2*Math::PI
end

range = Float::EPSILON..10.0

N = Integer ARGV.shift

case ARGV.shift
when 'tau' then
  puts Benchmark.measure {
    N.times do
      2 * Math::PI * rand(range)
    end
  }.real
when '2pi' then
  puts Benchmark.measure {
    N.times do
      2 * Math::TAU * rand(range)
    end
  }.real
else
  abort "#{$0} N tau|2pi"
end
```

#20 - 02/28/2013 01:13 AM - marcandre (Marc-Andre Lafortune)

- Category changed from Joke to core

duerst (Martin Dürst) wrote:

> Hello Eric,
>
> Second, even when using
> Math::TAU, it includes a multiplication by 2.

Right, I'd say the test doesn't say much... Actually, any difference apparently measured has to be meaningless, there's no intrinsic difference in either cases.

For what it's worth:

```
require 'fruity'

module Math
  TAU = 2 * PI
end
compare do
  pi  { Math.cos(2 * Math::PI) }
  tau { Math.cos(Math::TAU) }
end
# => tau is faster than pi by 50.0% ± 10.0%
```

I had to call Math.cos, because otherwise just accessing Math::TAU is too difficult to time.

Math currently has 2 constants (PI and E)
I don't see a huge downside in increasing this. On the other hand, TAU is so easy to define, I would trust that anyone knowing of it's existence would simply define it.

#21 - 02/28/2013 02:23 AM - david_macmahon (David MacMahon)

I vote -1 on this idea because the name "TAU" is used in a number of fields to represent a wide variety of things:

http://en.wikipedia.org/wiki/Greek_letters_used_in_mathematics,_science,_and_engineering#.CE.A4.CF.84_.28tau.29

For example, tau as also used as a constant representing the golden ratio (1.618...).

The names "PI" and "E" are used far more consistently across fields. Giving special preference to one (proposed!) use of the name "TAU" seems unfair to the other (established!) uses.

How about making a "twopi" gem that defines Math::TAU and BigMath.TAU? If it is as useful as its proponents claim, it will undoubtedly become a very popular gem and the public outcry to add it to the core of the language will be deafening.

Dave

#22 - 02/28/2013 07:10 AM - phluid61 (Matthew Kerwin)

- File tau.patch added

I just noticed a stupid typo in the patch I submitted. Sorry.

#23 - 02/28/2013 07:10 AM - JosephLindenberg (Joseph Lindenberg)

The Processing programming language recently added TAU:
http://code.google.com/p/processing/source/browse/trunk/processing/core/src/processing/core/PConstants.java

It does seem to be showing up more and more frequently. Here's a recent SAMS book where the author uses it in his examples:
http://books.google.com/books?id=BFda3Z71Y5YC&printsec=frontcover

I'm in favor of adding TAU. It's definitely not the same as having a bunch of multiples and submultiples of a constant. In fact, you really could say that right now, we have the submultiple-of-a-constant HALF_TAU, just under a different name.

#24 - 02/28/2013 07:38 AM - JosephLindenberg (Joseph Lindenberg)

david_macmahon (David MacMahon) wrote:

> For example, tau is also used as a constant representing the golden ratio (1.618...).

No, not anymore. φ (phi) is used for that now.

#25 - 03/01/2013 05:50 AM - drbrain (Eric Hodel)

Martin, you are right. With a corrected benchmark there is an 8.5% +/- 1.2% improvement:

```
  x tau.txt
```

```
+ 2pi.txt
+--------------------------------------------------------------------------+
|xx      x x                                + +      +     +               |
|xx x    xxxx x x           x    +  + + + + +++ +  + +  ++   + +          +|
| |_____A_____|                      |_________MA_________|                |
+--------------------------------------------------------------------------+
    N           Min           Max        Median           Avg        Stddev
x  20      0.526745      0.557717      0.535967     0.5353094  0.0073255609
+  20      0.562776       0.60921      0.579785     0.5810366   0.011508619
Difference at 95.0% confidence
  0.0457272 +/- 0.00617423
  8.5422% +/- 1.15339%
  (Student's t, pooled s = 0.00964656)
```

#26 - 03/01/2013 06:15 AM - alexeymuranov (Alexey Muranov)

+1

#27 - 03/01/2013 06:34 AM - marcandre (Marc-Andre Lafortune)

drbrain (Eric Hodel) wrote:

> Martin, you are right. With a corrected benchmark there is an 8.5% +/- 1.2% improvement:

To be more accurate, you'd have to subtract the time of an empty loop. If you compare 42 * pi with 42 *pi; 42 * pi using your technique, you won't get the expected answer (100%). I get ~50% with your technique.

I should introduce some of the techniques in ministat in my fruity gem.

#28 - 03/01/2013 09:37 AM - sbaird (Simon Baird)

david_macmahon (David MacMahon) wrote:

> I vote -1 on this idea because the name "TAU" is used in a number of fields to represent a wide variety of things:
> For example, tau as also used as a constant representing the golden ratio (1.618...).

mhartl addresses this here:
http://tauday.com/tau-manifesto#sec:ambiguous_notation

#29 - 03/01/2013 09:46 AM - sbaird (Simon Baird)

I don't think the benchmarks are particularly relevant here. We should define Tau because it's important, not for any performance benefit.

#30 - 03/01/2013 08:05 PM - Eregon (Benoit Daloze)

sbaird (Simon Baird) wrote:

> I don't think the benchmarks are particularly relevant here. We should define Tau because it's important, not for any performance benefit.

Indeed.

Although I am not sure of the importance of having TAU. One PI is fine for unit conversions as well as computing the area of a circle. But there are also many cases of 2*PI, which sounds less-than-ideal.

#31 - 04/14/2013 02:57 AM - bug (Harrison Reiser)

Eregon (Benoit Daloze) wrote:

> Although I am not sure of the importance of having TAU. One PI is fine for unit conversions as well as computing the area of a circle. But there are also many cases of 2*PI, which sounds less-than-ideal.

They are more common than you would think. Finding the area of circles is actually relatively specialized, in comparison with, for example, circular sectors (1/2 * angle * $r^{2)}$, for which the circle formula is a special case: 1/2 * tau * r^2 = pi * $r^{2.}$ The use of pi is an aesthetic optimization, not representative of any underlying geometric relationship. (However, the use of pi in runtime optimization is still applicable for such special cases.)

This mere coincidence becomes even more obvious when looking at the derivation of the above formulae, and as with everything discussed heretofore, is already discussed in the Tau Manifesto. Incidentally, I recommend anyone still wavering to read it for the benefit of making more informed opinions.

#32 - 06/28/2014 08:49 AM - sbaird (Simon Baird)

Since today is 6/28, here's a pull request on trunk:

- https://github.com/ruby/ruby/pull/644

See also:

- http://tauday.com/state-of-the-tau

Best wishes and Happy Tau Day... :)

#33 - 07/02/2015 07:36 AM - sbaird (Simon Baird)

I'm a bit late, but here it is rebased for Tau Day 2015:

- https://github.com/ruby/ruby/pull/644

#34 - 07/20/2015 11:05 PM - mrkn (Kenta Murata)

As I wrote in https://github.com/ruby/ruby/pull/644#issuecomment-123082639, I closed the pull-request for the present time.

#35 - 08/11/2016 01:39 AM - sbaird (Simon Baird)

This may be of interest: Guido van Rossum recently reopened the Python version of this request and is going to add tau to Python. See https://bugs.python.org/issue12345 .

#36 - 10/14/2016 03:12 AM - nobu (Nobuyoshi Nakada)

At the developers' meeting this week, mrkn suggested TWO_PI instead.

#37 - 10/14/2016 05:41 AM - duerst (Martin Dürst)

On 2016/10/14 13:35, RRRoy BBBean wrote:

> Would such an interconnected system be packaged as a bunch of individual gems, or is there some higher-level packaging concept than the Ruby gem?
>
> I personally think gems are best when kept as small as possible, highly cohesive and highly coherent. But that creates a secondary problem? How to organize larger units of reusable code? I don't know.

It's easy. Gems can depend on other gems. So we can have some highly cohesive and highly coherent gems at the bottom, and then other, more application-oriented gems that pull the necessary basic gems together. This can be done on more and more levels if necessary.

Composition is a great concept, and applying it recursively is even better.

Regards, Martin.

#38 - 10/14/2016 06:42 AM - sbaird (Simon Baird)

Nobuyoshi Nakada wrote:

> At the developers' meeting this week, mrkn suggested TWO_PI instead.

I don't think TWO_PI would be very useful since it misses the main point of tau, which is that tau is the correct circle constant, (and pi is a 2000 year old blunder)[1]. If you consider this diagram [2] with TWO_PI in place of tau, then it loses its meaning.

Defining HALF_TAU could be useful though. ;)

[1] Of course not everyone agrees on this, but http://tauday.com/tau-manifesto is quite convincing IMO

[2] http://tauday.com/assets/figures/tau-angles.png

#39 - 10/14/2016 08:38 AM - shyouhei (Shyouhei Urabe)

Just to add my POV:

I'm a complete amateur in pure Mathematics. To me who is an outsider of academia, I can't say we should accept or reject this request right now. All I can say is it seems not widely used in papers for some reason I don't know. It's not clear to me wether the constant τ is just yet to receive wider acceptance because it's relatively young, or is actively receiving low marks from academic community. I'm just not sure.

This constant is a constant (by nature), so once introduced in ruby it's highly expected to stay defined forever. Because of this property I'm afraid of

wrong decision. Is it really safe we add this? In order to make this sure I think we need either more time, or an expert's counsel, or maybe both.

#40 - 10/14/2016 09:40 AM - sbaird (Simon Baird)

See also https://github.com/jneen/math-tau .

#41 - 11/25/2016 06:49 AM - matz (Yukihiro Matsumoto)

- Status changed from Assigned to Rejected

For the time being, use math-tau gem.
I will add it after (and only after) it became time-proven major, probably due to Python's tau.

Matz.

#42 - 06/30/2017 05:49 AM - naruse (Yui NARUSE)

- Related to Feature #13694: Add Tau constant to Math added

#43 - 01/01/2021 02:13 AM - nobu (Nobuyoshi Nakada)

- Has duplicate Feature #17496: Add constant Math::TAU added

Files

tau.patch	1.2 KB	02/27/2013	phluid61 (Matthew Kerwin)
tau.patch	1.2 KB	02/28/2013	phluid61 (Matthew Kerwin)