Ruby master - Feature #14697

Introducing Range#% as an alias to Range#step

04/19/2018 06:31 AM - mrkn (Kenta Murata)

Status:	Closed	
Priority:	Normal	
Assignee:	mrkn (Kenta Murata)	
Target version:	2.6	

Description

In #13904, Enumerator::ArithmeticSequence has been accepted for the representation of a range with step value.
And in #12912, a new syntax of endless range (1..) has been accepted.

Combining these new features, we can write an endless step range like (1..).step(2) in Ruby 2.6.
It can be used for array slicing like python's 1::2.

If Range#% is introduced as an alias to Range#step, we can write a step range like (1..)%2.
This notation is already introduced numo-narray.

Related issues:

Related to Ruby master - Feature #12912: An endless range `(1..)`	**Closed**

Associated revisions

Revision 85f192b0 - 09/28/2018 02:18 AM - mrkn (Kenta Murata)

range.c: Add Range#%

[Feature #14697] [ruby-core:86588]

git-svn-id: svn+ssh://ci.ruby-lang.org/ruby/trunk@64869 b2dd03c8-39d4-4d8f-98ff-823fe69b080e

Revision 64869 - 09/28/2018 02:18 AM - mrkn (Kenta Murata)

range.c: Add Range#%

[Feature #14697] [ruby-core:86588]

Revision 64869 - 09/28/2018 02:18 AM - mrkn (Kenta Murata)

range.c: Add Range#%

[Feature #14697] [ruby-core:86588]

History

#1 - 04/19/2018 06:49 AM - matz (Yukihiro Matsumoto)

- Related to Feature #12912: An endless range `(1..)` added

#2 - 04/19/2018 06:54 AM - matz (Yukihiro Matsumoto)

Looks good to me. Any opinion?

Matz.

#3 - 04/19/2018 02:18 PM - marcandre (Marc-Andre Lafortune)

I am not convinced that step is used enough to justify this. I know I basically never use it. Here's the number of uses for some projects:
rails: 3 uses
bundler: 0 uses
sinatra: 0 uses
WikiEduDashboard: 0 uses (a typical Rails app: https://github.com/WikiEducationFoundation/WikiEduDashboard)

When thinking about this, I am able to see the relation between "modulo" and "step", but it wasn't immediately obvious at all.

In summary: my opinion is that it is not worth the cognitive load.

#4 - 04/19/2018 10:20 PM - baweaver (Brandon Weaver)

Have we considered a name like every?

```
(1..).every(2) # => 2, 4, 6, 8
```

I did not know that step could do this until I read this. The name does not clearly indicate that it would do that to me, but that may also be my lack of knowledge of it. % feels like moving in the opposite direction of clarity. It'd be great for terseness and golf but may be overkill for general usage.

#5 - 04/23/2018 01:56 PM - mrkn (Kenta Murata)

I'm supposing that this new notation of Range#step is mostly used for slicing numerical arrays like Numo::NArray.
This usecase is very similar to Python's slice notation used for slicing numpy's arrays.

With this new notation, ary[::2] in python can be written as ary[(0...)%2] in Ruby 2.6.
Slicing an array with a stepped range is often used in data analysis.
For example, if daru supports this new notation, we can pick up rows of even index in a dataframe df by df.row[(0..)%2].

#6 - 05/01/2018 11:00 PM - marcandre (Marc-Andre Lafortune)

I understand.
I guess it's a way to make Enumerator::ArithmeticSequence even more "core", even if it's rarely used.
Has there been discussion of:

- defining Enumerator::ArithmeticSequence#===?
- supporting Array#[] with Enumerator::ArithmeticSequence argument?

#7 - 05/16/2018 05:30 PM - mrkn (Kenta Murata)

- Project changed from CommonRuby to Ruby master

#8 - 05/17/2018 05:36 AM - matz (Yukihiro Matsumoto)

It seems no one has a strong objection against Range#%. Accepted.

Matz.

#9 - 05/17/2018 10:08 AM - shevegen (Robert A. Heiler)

I personally am not hugely comfortable with endless Range, but I understand
the reasoning given by mame for it. There is nothing shorter than omission
of characters. :D

Since endless Range was accepted, I think using % on Range, as explained by
mrkn, makes sense too. I personally like step more because it tells me
more (somewhat similar reason for as to why I prefer to not omit an end
range in the ruby code that I write), but I think since endless Range was
accepted, accepting the issue request here makes sense too. (I am not sure
if anyone understood what I was trying to say, but in short, +1 to the
issue request.)

Ruby allows for different paradigms and writing styles and people can
always decide for their own how (and what) to write/code anyway. It's a
similar situation with @@ class variables. One can decide to use them or
not use them. I realized that I don't really need them, so I don't use
them in my own code.

#10 - 07/05/2018 07:36 AM - mrkn (Kenta Murata)

- Target version set to 2.6

- Assignee changed from matz (Yukihiro Matsumoto) to mrkn (Kenta Murata)

- Status changed from Open to Assigned

#11 - 09/28/2018 02:19 AM - mrkn (Kenta Murata)

- Status changed from Assigned to Closed

Applied in changeset trunk|r64869.

---

range.c: Add Range#%

[Feature #14697] [ruby-core:86588]