Ruby master - Feature #15950
Allow negative length in `Array[]`, `Array[]=`, `String[]`, `String[]=`, `String#[]`, `String#[]=`, `String#slice`, `String#slice!

06/22/2019 03:09 AM - sawa (Tsuyoshi Sawada)

Status: Rejected
Priority: Normal
Assignee:
Target version:

Description
To take the first n characters of a string, using [] is straightforward:

"abcdefg"[0, 3] #=> "abc"

But to take the last n characters, we need to use n in two arguments: in the index (in negative form) in addition to the length:

"abcdefg"[-3, 3] #=> "fgh"

This is cumbersome.

I wish negative length to be allowed, and be interpreted as measuring leftward (while cycling the receiver if necessary).

"abcdefg"[0, -3] #=> "fgh"
"abcdefg"[5, -3] #=> "cde"

If there is not enough characters or elements, it should stop at the boundary.

"abcdefg"[1, -3] #=> "a"

History
#1 - 06/22/2019 03:11 AM - sawa (Tsuyoshi Sawada)
- Subject changed from Allow negative length in `Array[]`, `Array[]=`, `String[]`, `String[]=` to Allow negative length in `Array[]`, `Array[]=`, `String[]`, `String[]=`

#2 - 06/22/2019 03:22 AM - sawa (Tsuyoshi Sawada)
- Description updated

#3 - 06/22/2019 06:03 AM - nobu (Nobuyoshi Nakada)
Negative offset is allowed, but negative length has never been allowed, IIRC.

#4 - 06/22/2019 06:58 AM - shevegen (Robert A. Heiler)
Hmm. I do not necessarily doubt that it may be useful (at the least for some users), but that also actually surprised me. For example:

"abcdefg"[0, -3] #=> "fgh"

I guess my brain got confused here, since the 0 would put my brain to assume to "start at the most left position". So that code confused me.

I am mostly neutral on this though, but have a slight preference to keep the status quo without adding negative length. Either way, it may be a good idea to ask matz because this may have been a old design decision (at the least it seems plausible to me that this may have been in ruby for a long time).

#5 - 06/22/2019 07:04 AM - Eregon (Benoit Daloze)
- Status changed from Open to Rejected

sawa (Tsuyoshi Sawada) wrote:
But to take the last n characters, we need to use n in two arguments: in the index (in negative form) in addition to the length:

```
"abcdefg"[-3, 3] # => "fgh"
```

Why not:

```
"abcdefg"[-3..1] # => "fgh"
```

I would think the Range form is more idiomatic for this usage.

Also, with endless ranges, we can just do:

```
"abcdefg"[-3..] # => "fgh"
```

I find the negative length very confusing (I can't even understand the example above, rotation should not be part of indexing IMHO), and I'm negative on making indexing semantics a lot more complicated.

So I think there are already good ways to achieve this, and I don't see a rationale to include this complicated feature, so I would like to reject this ticket.

Feel free to motivate why it's better than the existing 2 alternate syntax, though.

#6 - 06/22/2019 12:12 PM - sawa (Tsuyoshi Sawada)

shevegen (Robert A. Heiler) wrote:

```
"abcdefg"[0, -3] # => "fgh"
```

I guess my brain got confused here, since the 0 would put my brain to assume to "start at the most left position". So that code confused me.

It is the same idea as negative index. If you did not have the concept of cycling, then you would be surprised with negative index too. You would need to say "0 lets me assume to start at the left most position. If I had a negative index instead, then what can be even more left of that?" The key to this is cycling: to assume that the left end is attached to the right end so that you can continuously move beyond this border. I suggested to apply that notion to length as well. To give a metaphor, there is a video game called Pac Man, in which if you keep moving the character leftward past the left edge of the screen, you appear from the right edge, still moving leftward.

And needless to say, this proposal would be irrelevant to Ruby beginners who insist on needlessly creating a range object as in "abcdefg"[0...3] to take the first three characters of the string, when you can, and should, do "abcdefg"[0, 3], which is more straightforward and efficient.

#7 - 06/22/2019 02:29 PM - Eregon (Benoit Daloze)

sawa (Tsuyoshi Sawada) wrote:

And needless to say, this proposal would be irrelevant to Ruby beginners who insist on needlessly creating a range object as in "abcdefg"[0...3] to take the first three characters of the string, when you can, and should, do "abcdefg"[0, 3], which is more straightforward and efficient.

I am not a Ruby beginner and I would still recommend the Range notation, which I think is more idiomatic and easier to read. So whether people should use the Range notation or not is obviously subjective.

Moreover, you mention performance but it's not so relevant here. For instance, TruffleRuby can escape analyze the Range, so it has no cost in this case. And even on MRI in such a micro-benchmark there is only 10% of difference, which seems negligible.
Warming up --------------------------------------

<table>
<thead>
<tr>
<th></th>
<th>s[0, 3]</th>
<th>411.458k i/100ms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>s[0...3]</td>
<td>369.962k i/100ms</td>
</tr>
</tbody>
</table>

Calculating -------------------------------------

<table>
<thead>
<tr>
<th></th>
<th>s[0, 3]</th>
<th>9.298M (± 2.3%) i/s - 46.495M in 5.003131s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>s[0...3]</td>
<td>8.471M (± 1.8%) i/s - 42.546M in 5.024158s</td>
</tr>
</tbody>
</table>

Comparison:

<table>
<thead>
<tr>
<th></th>
<th>s[0, 3]: 9298307.0 i/s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>s[0...3]: 8470923.4 i/s - 1.10x slower</td>
</tr>
</tbody>
</table>

ruby -v bench.rb
truffleruby 19.0.0, like ruby 2.6.2, GraalVM CE Native [x86_64-linux]

Warming up --------------------------------------

<table>
<thead>
<tr>
<th></th>
<th>s[0, 3]</th>
<th>1.201M i/100ms</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>s[0...3]</td>
<td>1.204M i/100ms</td>
</tr>
</tbody>
</table>

Calculating -------------------------------------

<table>
<thead>
<tr>
<th></th>
<th>s[0, 3]</th>
<th>27.141M (±31.9%) i/s - 114.054M in 5.016116s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>s[0...3]</td>
<td>26.553M (±29.1%) i/s - 115.561M in 5.044515s</td>
</tr>
</tbody>
</table>

Comparison:

<table>
<thead>
<tr>
<th></th>
<th>s[0, 3]: 27140752.2 i/s</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>s[0...3]: 26553278.2 i/s - same-ish: difference falls within error</td>
</tr>
</tbody>
</table>

#8 - 06/26/2019 05:29 AM - sawa (Tsuyoshi Sawada)

- Subject changed from Allow negative length in `Array[]`, `Array[]=`, `String[]`, `String[]=` to Allow negative length in `Array[]`, `Array#slice`, `Array[]=`, `String[]`, `String[]=`, `String#slice`

#9 - 06/26/2019 05:45 AM - sawa (Tsuyoshi Sawada)

- Subject changed from Allow negative length in `Array[]`, `Array#slice`, `Array[]=`, `String[]`, `String[]=`, `String#slice` to Allow negative length in `Array[]`, `Array[]=`, `Array#slice`, `Array#slice!`, `String[]`, `String[]=`, `String#slice`, `String#slice!`

#10 - 06/28/2019 09:15 AM - Eregon (Benoit Daloze)

I'm negative on making indexing semantics a lot more complicated.

To clarify, I think adding more logic to indexing would slow down indexing in general, which is undesirable and goes against your goal.

Besides, this is really hard to read and understand, and is not intuitive, contrary to existing indexing ways.

#11 - 07/11/2019 05:35 AM - matz (Yukihiro Matsumoto)

I don't see any rational use of this proposal. Rejection confirmed.

Matz.