Subject: Re: Ruby range operators? Re: IDL 8.0 compile_opt changes
Posted by Maarten[1] on Mon, 11 Jan 2010 08:41:10 GMT
View Forum Message <> Reply to Message

```
On Jan 8, 10:16 pm, Paul van Delst <paul.vande...@noaa.gov> wrote:
>  Maarten wrote:
>>  On Jan 7, 6:56 pm, mgalloy <mgal...@gmail.com> wrote:
>>>  I think we are agreeing here, but just to be sure: Python and IDL would
>>>  be specifying the endpoints of the range in the same way, it's just that
>>>  Python always includes the start index and excludes the end index (even
>>>  if not using negative indices):
>
>>>> >> a = [1, 2, 3, 4]
>>>> >> a[1:3]
>>>      [2, 3]
>
>>  Yes. Although this is a fundamental difference that is the result of a
>>  choice both language developers made. Thinking about it a bit longer,
>>  I don't think the two can be made to act the same: IDL always includes
>>  the end index of the range, while Python always excludes it. Some
>>  emphasis on this in the documentation may be needed, as Python
>>  probably is the most widespread programming language that offers the
>>  facility of negative indices.
>
>  Well, since they're mucking about with operators in general, maybe ITTVIS could go the
>  ruby route and introduce the ".." and "..." range operators. The former is an inclusive
>  range (same functionality as ":") and the latter is a range that excludes the higher
>  value. So,
>
>  $ irb
>  irb(main)> a = [1,2,3,4,5,6]
>  => [1, 2, 3, 4, 5, 6]
>
>  irb(main)> a[1..3]
>  => [2, 3, 4]
>
>  irb(main)> a[1...3]
>  => [2, 3]
>
>  irb(main)> a[1..-1]
>  => [2, 3, 4, 5, 6]
>
>  irb(main)> a[1...-1]
>  => [2, 3, 4, 5]
```

That is one option. Of course, python doesn't stop at a[1:-1], it can
also do a[-1:1:-1], resulting in [6, 5, 4, 3] (with a as above). That
is, it includes a stride (including negative stride) in its array

indexing.

> BTW, if IDL 8.0 will allow operator overloading, will it also allow for operator
> definition? The overloading should allow for ".." having the same result as ":", but will
> we be able to define functions/procedures that can be overloaded with "..." ?

Are you sure you want to open that can of worms? Adding this once will
preclude _any_ future syntax changes or additions, as _someone_ will
have implemented a conflicting operator.

Maarten