Subject: Re: Ruby range operators? Re: IDL 8.0 compile_opt changes
Posted by Kenneth P. Bowman on Fri, 08 Jan 2010 22:08:31 GMT
View Forum Message <> Reply to Message

The discussion of negative indices has my head hurting, but I admit that
I have always wanted to be able to subscript like this

```
a = b[3:*-1]
```

instead of

```
n = N_ELEMENTS(b)
a = b[3:n-2]
```

Cheers, Ken

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

Subject: Re: Ruby range operators? Re: IDL 8.0 compile_opt changes
Posted by Maarten[1] on Mon, 11 Jan 2010 16:20:30 GMT
View Forum Message <> Reply to Message

On Jan 11, 9:41 am, Maarten <maarten.sn...@knmi.nl> wrote:
> 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.

Oh, to add to the fun: python uses ... for a different operation:
specifying all elements for all non-explicitly mentioned dimensions.
This allows you to write code the can handle an arbitrary number of
dimensions of your array.

```
import numpy as np
a = np.arange(0,3*4*5*6,1)
a = a.reshape((3,4,5,6))
b = a[...,2,:]
c = a[1,...,0]
print(c.shape)
(4, 5)
```

```
print(c)
array([[120, 126, 132, 138, 144],
       [150, 156, 162, 168, 174],
       [180, 186, 192, 198, 204],
       [210, 216, 222, 228, 234]])
print(b.shape)
(3, 4, 6)
```

(won't print here, too large).

Best,

Maarten - Each language will reinvent similar things in different ways
- Sneep

Subject: Re: Ruby range operators? Re: IDL 8.0 compile_opt changes
Posted by Paul Van Delst[1] on Mon, 11 Jan 2010 16:33:13 GMT

View Forum Message <> Reply to Message

Maarten wrote:
> On Jan 8, 10:16 pm, Paul van Delst <paul.vande...@noaa.gov> wrote:
>> 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.

Well, that can of worms has been opened many times previously in other languages (even Fortran since 1990!). I would think the rules for compiler/interpreter writers are well established.

cheers,

paulv

Subject: Re: Ruby range operators? Re: IDL 8.0 compile_opt changes
Posted by R.Bauer on Wed, 13 Jan 2010 09:41:25 GMT

View Forum Message <> Reply to Message

Maarten schrieb:

> 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

Hi

as I have written before, I would wish a complete redesign of idl features without any compatibility flag.

You can follow this process currently by python 2 to python 3.

At some point it makes more sense to refactor your application and remove all of the deprecated functions instead of adding more and more compatibility crap.

cheers
Reimar
