
Subject: Loop Arrays

Posted by [Ken Mankoff](#) on Tue, 09 Oct 2001 19:33:52 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi,

I am interested in creating circular arrays, where subscripts that would be out-of-bounds on a regular array just start indexing on the other side of the array.

ex:

```
a = circleIndgen( 10 )
```

```
print, a[ -1 ]
```

```
9
```

```
print, a[ 11 ]
```

```
1
```

```
print, a[ 0,10,20,100 ]
```

```
0, 0, 0, 0
```

```
print, a[ 8:11 ]
```

```
8, 9, 0, 1
```

```
;;; not sure if this makes sense, but i think it can easily be
```

```
;;; done if the rest is possible...
```

```
print, a[ 8:2 ]
```

```
8, 9, 0, 1
```

I think that overloading the [] operators is not an option from my understanding of IDL. Does anyone know if this is possible? Desirable? Dumb?

Thanks,

-k.

--

Ken Mankoff

LASP://303.492.3264

<http://lasp.colorado.edu/~mankoff/>

Subject: Re: Loop Arrays

Posted by [K. Bowman](#) on Tue, 09 Oct 2001 20:50:23 GMT

[View Forum Message](#) <> [Reply to Message](#)

In article

<Pine.LNX.4.33.0110091318470.28938-100000@snoe.colorado.edu>, Ken Mankoff <mankoff@lasp.colorado.edu> wrote:

Try the MOD function.

Ken Bowman

Subject: Re: Loop Arrays

Posted by [David Fanning](#) on Tue, 09 Oct 2001 21:09:41 GMT

[View Forum Message](#) <> [Reply to Message](#)

K. Bowman (k-bowman@null.tamu.edu) writes:

> Try the MOD function.

Oh, man. You're no fun, Ken. :-(

David

--

David W. Fanning, Ph.D.

Fanning Software Consulting

Phone: 970-221-0438, E-mail: david@dfanning.com

Coyote's Guide to IDL Programming: <http://www.dfanning.com/>

Toll-Free IDL Book Orders: 1-888-461-0155

Subject: Re: Loop Arrays

Posted by [Craig Markwardt](#) on Wed, 10 Oct 2001 05:46:14 GMT

[View Forum Message](#) <> [Reply to Message](#)

Ken Mankoff <mankoff@I.HATE.SPAM.cs.colorado.edu> writes:

> On Tue, 9 Oct 2001, Mark Hadfield wrote:

>

>> From: "Ken Mankoff" <mankoff@lasp.colorado.edu>

>>> I am interested in creating circular arrays, where subscripts that would
>>> be out-of-bounds on a regular array just start indexing on the other side
>>> of the array.

>>

>> You can do quite a lot with ordinary arrays using arrays of indices, eg

>>

>> a = indgen(10)

>> print, a[[0,10,20,100] mod n_elements(a)]

>>

>

> This is the technique I have been using. However there are 2 cases it does
> not cover:

>

```

> 1) negative indexes require a few more lines of code to get your example
> to work. I would recode it as:
>
> a = indgen( 10 )
> indexes = [ 0,10,20,100,-10,-22 ]    ;; or some other values...
> ind = indexes mod n_elements( a )
> neg = where( ind lt 0, num )
> if ( num ne 0 ) then ind[ neg ] = ind[ neg ] + n_elements( a )
> print, a[ ind ]
>
> 2) subscript ranges. You cannot do:
>   print, a[ 8:12 mod n_elements(a) ]
>
> It is these two specific abilities that I would like to have.

```

Hi Ken--

Like Mark, I too have longed for the ability to index "from the vright," so to speak, using negative numbers, or some kind of notation. Unfortunately, negative numbers already have a meaning, or, err, rather, they already have a non-meaning when used in an index list. Negative numbers and too-big numbers are clipped when used in an index list.

However, you can get a little of what you want back by using this notation:

```
print, a[ (ii + na) MOD na ]
```

If ii is guaranteed only to be in the range of [-na to +na] then this will always work. As you pointed out though, you can't do this with index ranges.

Good luck,
Craig

--

 Craig B. Markwardt, Ph.D. EMAIL: craigmnet@cow.physics.wisc.edu
 Astrophysics, IDL, Finance, Derivatives | Remove "net" for better response

Subject: Re: Loop Arrays
 Posted by [R.G.S.](#) on Mon, 15 Oct 2001 16:46:36 GMT
[View Forum Message](#) <> [Reply to Message](#)

Ken Mankoff <mankoff@I.HATE.SPAM.cs.colorado.edu> wrote in message

news:Pine.LNX.4.33.0110141828320.13144-100000@snoe.colorado.edu...

>

> On Tue, 9 Oct 2001, Mark Hadfield wrote:

>> From: "Ken Mankoff" <mankoff@lasp.colorado.edu>

>>> I am interested in creating circular arrays, where subscripts that would

>>> be out-of-bounds on a regular array just start indexing on the other side

>>> of the array.

My 2 cents:

Just write a function ind() that accepts a string, and returns the desired index array.

Then call the array as follows:

```
b = a(ind("-70:-56"))
```

I almost wrote that function, but I had too much work to do.

-bob

Subject: Re: Loop Arrays

Posted by [Ken Mankoff](#) on Tue, 16 Oct 2001 15:11:03 GMT

[View Forum Message](#) <> [Reply to Message](#)

On Mon, 15 Oct 2001, R.G.S. wrote:

>

> My 2 cents:

> Just write a function ind() that accepts a string, and returns the desired

> index array.

> Then call the array as follows:

>

> b = a(ind("-70:-56"))

>

> I almost wrote that function, but I had too much work to do.

>

I like this idea even if its not as transparent as laying a filter over the existing IDL session that re-parses your code for you. Mostly because this could be coded fairly easily and through keywords can even supply multiple behaviors (i.e. circular vs from_right vs reverse...)

However, how would your ind() function know what it is subscripting?

Wouldnt it have to be:

```
b = ind(a, "-70:-56" )
```

Thanks,
-k.

--

Ken Mankoff
LASP://303.492.3264
<http://lasp.colorado.edu/~mankoff/>

Subject: Re: Loop Arrays
Posted by [Ken Mankoff](#) on Tue, 16 Oct 2001 17:49:45 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi Martin,

Yes, the code you supplied come the closest to doing what I want. Thank you.

I believe I will be honing my perl skills a bit to play with an implementation of this idea. I will let you know how it goes...

-k.

On Mon, 15 Oct 2001, Martin Downing wrote:

```
>
>
> "Ken Mankoff" <mankoff@I.HATE.SPAM.cs.colorado.edu> wrote in message
> news:Pine.LNX.4.33.0110091423020.29204-100000@snoe.colorado.edu...
>> On Tue, 9 Oct 2001, Mark Hadfield wrote:
>>
>>> From: "Ken Mankoff" <mankoff@lasp.colorado.edu>
>>>> I am interested in creating circular arrays, where subscripts that
> would
>>>> be out-of-bounds on a regular array just start indexing on the other
> side
>>>> of the array.
>>>
>>> You can do quite a lot with ordinary arrays using arrays of indices, eg
>>>
>>> a = indgen(10)
>>> print, a[ [0,10,20,100] mod n_elements(a)]
>>>
>>
>> This is the technique I have been using. However there are 2 cases it does
```

```

>> not cover:
>>
>> 1) negative indexes require a few more lines of code to get your example
>> to work. I would recode it as:
>>
>> a = indgen( 10 )
>> indexes = [ 0,10,20,100,-10,-22 ]    ;; or some other values...
>> ind = indexes mod n_elements( a )
>> neg = where( ind lt 0, num )
>> if ( num ne 0 ) then ind[ neg ] = ind[ neg ] + n_elements( a )
>> print, a[ ind ]
>>
>> 2) subscript ranges. You cannot do:
>>   print, a[ 8:12 mod n_elements(a) ]
>>
>> It is these two specific abilities that I would like to have.
>>
>> -k.
>
> Hi Ken,
>
> This discussion makes for interesting reading. However, except for arrays
> representing objects with circular indexing logic, such as closed
> polygons for instance, I'm not sure it is productive to prevent IDL from
> pointing out that you have run off the end of an array!
>
> Anyway, there is a way you can code range indexing above for circular
> arrays:
>
> eg for indexing a[b:c] do the following:
>
> IDL> a = indgen(10) ; to be interpreted as a circular array
> IDL> b = 9 & c = 13
> IDL> print, a[ (indgen(c-b)+b) MOD n_elements(a) ]
> ; read as a[b:c]
> 9 0 1 2
>
> IDL> b = 9 & c = 23
> IDL> print, a[ (indgen(c-b)+b) MOD n_elements(a) ] ; read as a_circ[b:c]
> 9 0 1 2 3 4 5 6 7 8 9 0 1 2
>
>
> -Is that of any use to you?
>
> regards
>
> Martin
>

```
