Subject: Adding two strings together
Posted by wdolan on Wed, 18 Nov 2015 02:36:27 GMT
View Forum Message <> Reply to Message

So I'm having a devil of a time with something that should be easy.

So I want the end product to look like 'Run07656_Seq0001_Scan012', for every scan in a file I have
So I put the following in a loop for all of the scans
 x='Run0'+string(variable1)+'_Seq0001_Scan'+string(variable2)
where variable 1 is the run (ex. 7657), and variable 2 is the scan (ex. 012).

But for some reason they all print out like 'Run0   7656_Seq0001_Scan012'

How so I make it so there is no space between the 0 and the 7?

Please help!

Subject: Re: Adding two strings together
Posted by wlandsman on Wed, 18 Nov 2015 03:01:51 GMT
View Forum Message <> Reply to Message

I suspect you have spaces in front of your variable1.   Try setting

variable1 = strtrim(variable1,2)

before your concatenation.   --Wayne

On Tuesday, November 17, 2015 at 9:36:30 PM UTC-5, Wayana Dolan wrote:
> So I'm having a devil of a time with something that should be easy.
>
> So I want the end product to look like 'Run07656_Seq0001_Scan012', for every scan in a file I have
> So I put the following in a loop for all of the scans
> x='Run0'+string(variable1)+'_Seq0001_Scan'+string(variable2)
> where variable 1 is the run (ex. 7657), and variable 2 is the scan (ex. 012).
>
> But for some reason they all print out like 'Run0   7656_Seq0001_Scan012'
>
>
> How so I make it so there is no space between the 0 and the 7?
>
> Please help!

Subject: Re: Adding two strings together
Posted by Jim Pendleton on Wed, 18 Nov 2015 03:05:30 GMT
View Forum Message <> Reply to Message

On Tuesday, November 17, 2015 at 7:36:30 PM UTC-7, Wayana Dolan wrote:
> So I'm having a devil of a time with something that should be easy.
>
> So I want the end product to look like 'Run07656_Seq0001_Scan012', for every scan in a file I have
> So I put the following in a loop for all of the scans
>   x='Run0'+string(variable1)+'_Seq0001_Scan'+string(variable2)
> where variable 1 is the run (ex. 7657), and variable 2 is the scan (ex. 012).
>
> But for some reason they all print out like 'Run0   7656_Seq0001_Scan012'
>
>
> How so I make it so there is no space between the 0 and the 7?
>
> Please help!

When numbers are converted to strings by default using the STRING function, they generally include leading spaces. The traditional way around this is to call the STRTRIM() function, for example STRTRIM(variable1, 2)

If you're running 8.4 or later, I encourage more modern syntax, for example

```
IDL>  x='Run0'+variable1.tostring()+'_Seq0001_Scan'+variable2.tost ring()
IDL> x
Run012345_Seq0001_Scan345
```

The .tostring() static method on IDL_Variable types implies trimming of both leading and trailing blanks.

For more on static methods, see http://www.exelisvis.com/docs/IDL_Variable.html

Jim P.

Subject: modern syntax
Posted by greg.addr on Wed, 18 Nov 2015 11:47:10 GMT
View Forum Message <> Reply to Message

> If you're running 8.4 or later, I encourage more modern syntax, for example
> IDL> x='Run0'+variable1.tostring()+'_Seq0001_Scan'+variable2.tost ring()
> IDL> x
> Run012345_Seq0001_Scan345
>
> The .tostring() static method on IDL_Variable types implies trimming of both leading and trailing blanks.

>
> For more on static methods, see http://www.exelisvis.com/docs/IDL_Variable.html
>
> Jim P.

I've been wondering about this recently. I admit it's rare that I use any language but IDL, so I'm sure I'm behind the times. Consistency is a good thing in programming, so I'm uneasy with a pick and mix approach to the new syntax: it can only leave code less readable. I understand the sense of the object approach, so I'd go with the new, but the notation - to my eyes - is getting worse:

variable1.tostring() vs string(variable1)

- the parentheses have become a relict, symbolically enclosing an argument which has gone elsewhere. To compensate the loss of meaning, we need to tack on a new preposition, 'to'. Ugh.

In some cases, though, the new syntax is elegant:

variable.tname vs size(variable,/tname)

although the old notation was unnecessarily blighted by the need to access this through the size function. I haven't yet grasped why variable.tname doesn't need (or allow) the following parentheses, since it appears to me to return a value and therefore qualify as a function. But I like it better without.

```
IDL> a={b:0,tname:4}
IDL> a.tname
      4
IDL> (a).tname
% Object reference type required in this context: A.
```

I don't know what to do here, though. Isn't 'a' also an object now?

Then there's the grotesque,

PRINT, var1.Equals(var2)

I don't want to write that.

I'd be interested to know how others choose which to use.

cheers,
Greg

Subject: Re: modern syntax
Posted by Fabzi on Wed, 18 Nov 2015 12:06:56 GMT

On 11/18/2015 12:47 PM, greg.addr@googlemail.com wrote:
> variable1.tostring() vs string(variable1)
>
> - the parentheses have become a relict, symbolically enclosing an argument which has gone elsewhere

The parentheses are necessary because tostring() is a function (or
better said a "method") which accepts arguments.

I kind of agree with the rest of your comments about consistency issues,
but I find that the advantages of the new syntax are overwhelming.

Cheers,

Fabien