Subject: Re: Optional parameters
Posted by David Fanning on Tue, 24 Feb 2004 01:05:20 GMT
View Forum Message <> Reply to Message

Michael Wallace writes:

> The learning of IDL continues.... today's question is about optional
> parameters. Specifically, how do you make a parameter optional? I can
> find stuff in the IDL documentation regarding position parameters and
> keywords, but I haven't found anything yet about optional parameters.
> So, how do you create optional parameters and what are common gotchas
> with using them?

An optional parameter can be either a positional parameter or a keyword parameter, although the general rule of thumb is that positional parameters are always required and keyword parameters are always optional. The rule is broken frequently, especially with positional parameters and less so with keyword parameters (having to suffer the wrath of Coyote if you violate the "keyword is optional" dictum probably accounts for the decreased frequency of the latter).

What makes a parameter "required", oddly enough, does not depend on how it is defined or used (e.g., is it an input or output parameter). Rather, what makes a parameter "required" is how you respond to its absence or presence. It is required if you notice it is not there when you need it and issue an error message. It is optional if you notice it is not there when you need it and you arbitrarily define a default value for it.

Here is a simple program with a required positional parameter named "junk":

```
PRO TEST, junk
  On_Error, 1
  IF N_Elements(junk) EQ 0 THEN Message, 'JUNK is required.'
  Print, junk
END
```

Here is the same program with an optional positional parameter named "junk":

```
PRO TEST, junk
  On_Error, 1
  IF N_Elements(junk) EQ 0 THEN junk = 3
  Print, junk
END
```

Here is the same program with an optional keyword parameter

named "junk":

```
PRO TEST, JUNK=junk
  On_Error, 1
  IF N_Elements(junk) EQ 0 THEN junk = 3
  Print, junk
END
```

I'm not going to show you a program with a required keyword parameter because no one in his right mind (with the possible exception of the good folks at RSI) would write one. :-)

Note that if the parameter has a *binary* quality (that is it is 0 or 1, yes or no, on or off, etc.) then you can check the parameter with KEYWORD_SET. All other parameters should be checked with N_ELEMENTS. This is a subtle point that only bites you when your equipment is on it way to Mars. :-)

Good programming!

Cheers,

David

P.S. Let's just say the only gotchas with optional parameters is forgetting to check whether they are defined or not, and using the wrong function to check whether they are defined or not. Remember, you can *never* go wrong with N_ELEMENTS. :-)

--
David Fanning, Ph.D.
Fanning Software Consulting
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Optional parameters
Posted by David Fanning on Tue, 24 Feb 2004 01:39:26 GMT
View Forum Message <> Reply to Message

David Fanning nearly had a cow when he wrote:

> An optional parameter can be either a positional parameter or a keyword
> parameter...

Should have included a reference to the definitive article about how to check for a keyword:

http://www.dfanning.com/tips/keyword_check.html

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Optional parameters
Posted by JD Smith on Tue, 24 Feb 2004 16:10:19 GMT
View Forum Message <> Reply to Message

On Mon, 23 Feb 2004 18:05:20 -0700, David Fanning wrote:

> Michael Wallace writes:
>
>> The learning of IDL continues.... today's question is about optional
>> parameters. Specifically, how do you make a parameter optional? I can
>> find stuff in the IDL documentation regarding position parameters and
>> keywords, but I haven't found anything yet about optional parameters.
>> So, how do you create optional parameters and what are common gotchas
>> with using them?
>
> An optional parameter can be either a positional parameter or a keyword
> parameter, although the general rule of thumb is that positional
> parameters are always required and keyword parameters are always
> optional. The rule is broken frequently, especially with positional
> parameters and less so with keyword parameters (having to suffer
> the wrath of Coyote if you violate the "keyword is optional" dictum
> probably accounts for the decreased frequency of the latter).
>

I would have put it more simply: all parameters, keyword or
positional, are optional in IDL. How you choose to treat missing
parameters defines the behavior of a routine. See David's linked
article for more. I think there are enough useful example (even among
RSI's own offerings) which violate this general rule of thumb that it
shouldn't be considered a rule.

JD

Subject: Re: Optional parameters
Posted by David Fanning on Tue, 24 Feb 2004 16:31:48 GMT
View Forum Message <> Reply to Message

JD Smith writes:

> I would have put it more simply: all parameters, keyword or
> positional, are optional in IDL. How you choose to treat missing
> parameters defines the behavior of a routine. See David's linked
> article for more. I think there are enough useful example (even among
> RSI's own offerings) which violate this general rule of thumb that it
> shouldn't be considered a rule.

I wasn't really going to admit this, but I have even
violated the "no required keywords" rule on occasion.
(Although not in anything I am offering to the public.)
The temptation is really too great when you are about
10 levels up in a superclass and you *really* need something
like keyword inheritance to work! :-)

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Optional parameters
Posted by David Fanning on Tue, 24 Feb 2004 17:08:16 GMT
View Forum Message <> Reply to Message

David Fanning writes:

> Here is the same program with an optional keyword parameter
> named "junk":
>
> PRO TEST, JUNK=junk
> On_Error, 1
> IF N_Elements(junk) EQ 0 THEN junk = 3
> Print, junk
> END

An alert reader, who wishes to remain anonymous, but who
is well-known in the IDL programming community, points out
that there is one possible gotcha in this formulation.
Namely, that an input variable that was previously undefined
is now defined by my example. In other words:

```
IDL> Help, junk
    JUNK           UNDEFINED = <Undefined>
```

```
IDL> Test, JUNK=junk
IDL> Help, junk
    JUNK          INT      =       3
```

If the user were counting on the variable junk to remain
undefined for some reason, there would be a nasty surprise
in store here.

This is, of course, a consequence of parameters being input
or output (or both) according to usage. I guess strictly
speaking, you would need two different variables to represent
parameters that are strictly input variables. The reader suggests
this:

```
PRO TEST, JUNK=junk_local
  On_Error, 1
  junk = n_elements(junk_local) eq 0 ? 3 : junk_local
  Print, junk
END
```

While I absolutely agree with the logic, I find this to be a royal
pain in the neck in practice. Do you read Whinnie the Pooh? I am
one of those "bears of little brain" and having two names for the
same variable sends me into fits of confusion every time. Lord knows
I have enough problems just keeping up with defining the same keywords
over and over again for INIT, SetProperty, and GetProperty methods.
Double names sounds like double work to me. I guess I'd rather track
down the very rare "defined when I didn't want it to be" error than
be frustrated with all the damn typos! :-)

Cheers,

David

P.S. Let's just say I admire the good folks at RSI who are going
to all the trouble to make input keywords strictly input. :-)

--
David Fanning, Ph.D.
Fanning Software Consulting
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Optional parameters
Posted by Pavel Romashkin on Tue, 24 Feb 2004 17:59:29 GMT
View Forum Message <> Reply to Message

I agree. I never had a problem with this because I never put unneeded

variables in a call to a procedure. I fail to see the logic in putting

myfunction, var1=uselessvar

as opposed to just

myfunction

and then worrying about uselessvar getting defined all of a sudden. The
only time this might happen in real life I suppose is if you use
copy-paste to insert function calls that address all of the provided keywords.
Pavel

David Fanning wrote:
>
> David Fanning writes:
>
>> Here is the same program with an optional keyword parameter
>> named "junk":
>>
>> PRO TEST, JUNK=junk
>> On_Error, 1
>> IF N_Elements(junk) EQ 0 THEN junk = 3
>> Print, junk
>> END
>
> An alert reader, who wishes to remain anonymous, but who
> is well-known in the IDL programming community, points out
> that there is one possible gotcha in this formulation.
> Namely, that an input variable that was previously undefined
> is now defined by my example. In other words:
>
> IDL> Help, junk
> JUNK UNDEFINED = <Undefined>
> IDL> Test, JUNK=junk
> IDL> Help, junk
> JUNK INT = 3
>
> If the user were counting on the variable junk to remain
> undefined for some reason, there would be a nasty surprise
> in store here.
>
> This is, of course, a consequence of parameters being input
> or output (or both) according to usage. I guess strictly
> speaking, you would need two different variables to represent
> parameters that are strictly input variables. The reader suggests
> this:
>

> PRO TEST, JUNK=junk_local
> On_Error, 1
> junk = n_elements(junk_local) eq 0 ? 3 : junk_local
> Print, junk
> END
>
> While I absolutely agree with the logic, I find this to be a royal
> pain in the neck in practice. Do you read Whinnie the Pooh? I am
> one of those "bears of little brain" and having two names for the
> same variable sends me into fits of confusion every time. Lord knows
> I have enough problems just keeping up with defining the same keywords
> over and over again for INIT, SetProperty, and GetProperty methods.
> Double names sounds like double work to me. I guess I'd rather track
> down the very rare "defined when I didn't want it to be" error than
> be frustrated with all the damn typos! :-)
>
> Cheers,
>
> David
>
> P.S. Let's just say I admire the good folks at RSI who are going
> to all the trouble to make input keywords strictly input. :-)
>
> --
> David Fanning, Ph.D.
> Fanning Software Consulting
> Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Optional parameters
Posted by JD Smith on Tue, 24 Feb 2004 20:13:01 GMT
View Forum Message <> Reply to Message

On Tue, 24 Feb 2004 10:59:29 -0700, Pavel Romashkin wrote:

> I agree. I never had a problem with this because I never put unneeded
> variables in a call to a procedure. I fail to see the logic in putting
>
> myfunction, var1=uselessvar
>
> as opposed to just
>
> myfunction
>
> and then worrying about uselessvar getting defined all of a sudden. The
> only time this might happen in real life I suppose is if you use
> copy-paste to insert function calls that address all of the provided keywords.

I'd offer another real life example where read-write keyword variables
can get you into trouble. When you call a function several times in a
row, expecting the same results each time, but in actuality the
keyword variable is defined after the first invocation, which can
change the function's behavior:

a=myfunc(test,FUNNY_VARIABLE=foo) ; foo undefined
a=myfunc(test,FUNNY_VARIABLE=foo) ; foo defined
a=myfunc(test,FUNNY_VARIABLE=foo) ; foo defined

For this reason, I try to make keywords either exclusively read
(i.e. pass in a value), or exclusively write (i.e. return a value),
and not both. It is rather convenient, though, to append a keyword
for the sole purpose of keeping some expensive-to-compute data around
between calls to the function: this is usually where I get tripped up.

JD

Subject: Re: Optional parameters
Posted by Pavel Romashkin on Tue, 24 Feb 2004 21:45:09 GMT
View Forum Message <> Reply to Message

And what exactly would be the purpose of such sequence of calls, other
than actually trying to *define* foo in the first function call? If I
didn't need foo re-defined, I'd not even mention it on the first line.
So this *is* the case for 3 copy-paste calls, I guess :-)
I never tried to use keywords for this sort of creative purposes. If I
have a large dataset I don't want to re-compute, I keep it in a pointer
(or object, allrigh, David? :-) and I try to always know when it gets
passed into a function to avoid redefinition and other mishaps.
After all, how is this behavior different than simply

foo = 100 ; foo is now defined, too...

If we did want to get funny I suppose we could try

a=myfunc(test,FUNNY_VARIABLE=[foo]) ; foo didn't change

but if foo is BIG, we'd get a dramatic slowdown.
Pavel

JD Smith wrote:
>
> I'd offer another real life example where read-write keyword variables
> can get you into trouble. When you call a function several times in a
> row, expecting the same results each time, but in actuality the
> keyword variable is defined after the first invocation, which can

> change the function's behavior:
>
> a=myfunc(test,FUNNY_VARIABLE=foo) ; foo undefined
> a=myfunc(test,FUNNY_VARIABLE=foo) ; foo defined
> a=myfunc(test,FUNNY_VARIABLE=foo) ; foo defined
>
> For this reason, I try to make keywords either exclusively read
> (i.e. pass in a value), or exclusively write (i.e. return a value),
> and not both. It is rather convenient, though, to append a keyword
> for the sole purpose of keeping some expensive-to-compute data around
> between calls to the function: this is usually where I get tripped up.
>
> JD

Subject: Re: Optional parameters
Posted by R.G. Stockwell on Tue, 24 Feb 2004 22:02:32 GMT
View Forum Message <> Reply to Message

"Pavel Romashkin" <pavel_romashkin@hotmail.com> wrote in message
news:403BC5E6.CE81249F@hotmail.com...
> And what exactly would be the purpose of such sequence of calls, other
> than actually trying to *define* foo in the first function call?
...

I've had this error (in fact, I think I have committed almost all errors!),
and it comes about when you are interactively calling the routine from
the command line during a session (or running main level scripts) and
the keyword is defined as you repeatedly call the routine. Then
when you leave IDL, and go back in, the keyword is undefined again
and it could behave differently than expected.

Cheers,
bob

PS I almost always run main level scripts in the windows version.
and during development one can trip over persistent main level
variables.

Subject: Re: Optional parameters
Posted by JD Smith on Tue, 24 Feb 2004 22:03:00 GMT
View Forum Message <> Reply to Message

On Tue, 24 Feb 2004 14:45:09 -0700, Pavel Romashkin wrote:

> And what exactly would be the purpose of such sequence of calls, other
> than actually trying to *define* foo in the first function call? If I
> didn't need foo re-defined, I'd not even mention it on the first line.
> So this *is* the case for 3 copy-paste calls, I guess :-)
> I never tried to use keywords for this sort of creative purposes. If I
> have a large dataset I don't want to re-compute, I keep it in a pointer
> (or object, allrigh, David? :-) and I try to always know when it gets
> passed into a function to avoid redefinition and other mishaps.
> After all, how is this behavior different than simply
>
> foo = 100 ; foo is now defined, too...

It's different because I'm overloading a keyword to do two things: 1) take as input a value for foo to override any internal value, and 2) return the value of foo used (default or otherwise). Admittedly, this is bad form, but it's actually very convenient when you want to bring some variable out "over the head" of your function calls, but still need to be able to set the variable via a parameter. This is typically for interactive usage, not for real production code, but the problem still exists.

JD

Subject: Re: Optional parameters
Posted by Craig Markwardt on Wed, 25 Feb 2004 05:50:39 GMT
View Forum Message <> Reply to Message

JD Smith <jdsmith@as.arizona.edu> writes:
> I'd offer another real life example where read-write keyword variables
> can get you into trouble. When you call a function several times in a
> row, expecting the same results each time, but in actuality the
> keyword variable is defined after the first invocation, which can
> change the function's behavior:
>
> a=myfunc(test,FUNNY_VARIABLE=foo) ; foo undefined
> a=myfunc(test,FUNNY_VARIABLE=foo) ; foo defined
> a=myfunc(test,FUNNY_VARIABLE=foo) ; foo defined

Yes, I've encountered this kind of problem before as well, and it's one of the most evil and difficult problems to solve.

That's why I define all of my functions as

```
pro myfunc, value=value0

  ;; Default value  ...
  if n_elements(value0) EQ 0 then value = 1 $
```

```
    ;; ... or requested value
    else                    value = round(value0(0))
  ...
  end
```

The "0" variable is the pristine variable that never gets changed, and the regular variable is the one that is internal to the procedure. I do that as a matter of habit now and it's not too labor intensive.

I do kind of wish that there would be an automatic way for IDL to peel off a local copy of a variable for internal calculations.

Craig

--

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

Subject: Re: Optional parameters
Posted by Paul Van Delst[1] on Wed, 25 Feb 2004 14:49:26 GMT
View Forum Message <> Reply to Message

Craig Markwardt wrote:
>
> That's why I define all of my functions as

You're a big norty, mister. :o) You should define all your functions as:

>
```
>    pro myfunc, value=value0

      COMPILE_OPT STRICTARR
>
>      ;; Default value  ...
>      if n_elements(value0) EQ 0 then value = 1 $
>      ;; ... or requested value
     else                    value = round(value0[0])
>    ...
>    end
```

:oD

paulv

p.s. it's early and I have no $$ to buy coffee... :o(

--
Paul van Delst
CIMSS @ NOAA/NCEP/EMC

Subject: Re: Optional parameters
Posted by David Fanning on Wed, 25 Feb 2004 15:08:45 GMT
View Forum Message <> Reply to Message

Paul van Delst writes:

> You're a big norty, mister. :o) You should define all your functions as:
>
>>
>> pro myfunc, value=value0
>
> COMPILE_OPT STRICTARR

That's for sure! We could understand it when you were
an IDL 4.0 holdout, Craig. But this is the 20th
century. Oh, uh wait, 21st century. Let's get up to date!

Cheers,

David

P.S. Are your customers still running those MicroVAX
computers? :-)

--
David Fanning, Ph.D.
Fanning Software Consulting
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Optional parameters
Posted by Craig Markwardt on Wed, 25 Feb 2004 16:48:50 GMT
View Forum Message <> Reply to Message

David Fanning <david@dfanning.com> writes:

> Paul van Delst writes:
>
>> You're a big norty, mister. :o) You should define all your functions as:

>>
>>>
>>>   pro myfunc, value=value0
>>
>>   COMPILE_OPT STRICTARR
>
> That's for sure! We could understand it when you were
> an IDL 4.0 holdout, Craig. But this is the 20th
> century. Oh, uh wait, 21st century. Let's get up to date!

Huh? Last I checked, round parenthesis are still allowed by default
for array indexing.

I shouldn't have to tell you about old habits dying should I David,
since most of yours seem to come from the late Cretaceous. :-)

Craig

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

Subject: Re: Optional parameters
Posted by David Fanning on Wed, 25 Feb 2004 17:26:31 GMT
View Forum Message <> Reply to Message

Craig Markwardt writes:

> I shouldn't have to tell you about old habits dying should I David,
> since most of yours seem to come from the late Cretaceous. :-)

I really have to subscribe to some kind of programming journal. :-(

Cheers,

David
--
David Fanning, Ph.D.
Fanning Software Consulting
Coyote's Guide to IDL Programming: http://www.dfanning.com/

Subject: Re: Optional parameters
Posted by Pavel Romashkin on Wed, 25 Feb 2004 18:21:37 GMT

I recommend Ranger Rick:
http://www.nwf.org/kidzone/kzPage.cfm?siteId=3&departmen tId=81

I learn something from it all the time :-)
Pavel

David Fanning wrote:
>
> I really have to subscribe to some kind of programming journal. :-(