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