Subject: Re: Preferred way to get multiple returns from a function
Posted by R.G.Stockwell on Sat, 12 Feb 2011 02:19:47 GMT
View Forum Message <> Reply to Message

In article
<6a9748b9-2b9e-4401-ab30-e65d354b6c35@y4g2000prh.googlegroups.com>,
James <donjeezy@gmail.com> wrote:

> I am writing a function that fits an ellipse to a 2*N array of
> points. There are three values to return: the center, semi-major
> axis, and semi-minor axis. This is a simple program, but it brings up
> a more general question: what is the preferred IDL way to return
> multiple values from a function?
>
> Currently, my program returns a structure containing the elements
> {center, major, minor}. However, a lot of built-in IDL routines take
> named variable inputs that are set to the appropriate value on output
> - so instead of something like:
>
> ellipse_struct = fit_ellipse(points)
>
> I would have:
>
> fit_ellipse, points, center, major, minor
>
> I'm not sure which is better. C programming has taught me to
> appreciate structures, but I'd like to code in the conventions of the
> language. Which would you prefer, and why?

My personal preference is to return a structure. They tell you what each
element is, and it has a pleasing "input goes in, output comes out" format.
I have always found the procedure call of inputting outputs somewhat
unpleasant. When some variables are in some out out, some are undefined,
some must defined to be within a certain range and a certain variable type,
then the function call does not make obvious what is happening.

ellipse.center for me is more meaningful and more readable than a,b,c

and frankly, just for job security, I would call that function as
fit_ellipse, in_1, in_2, out_a, out_b

no one would ever be able to debug it. :)