

---

Subject: What is the main difference between a script and a procedure?

Posted by [mystea](#) on Wed, 03 Oct 2007 07:22:35 GMT

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

---

Hello All,

I am pretty much a novice in IDL. Recently I just learned how to write/execute a IDL script file. (i.e. @filename)

I became a little curious(or more properly, confused) about the difference between a procedure and a script. I have my own thoughts but would like to hear opinions from experts if possible.

As far as I can tell, a script:

1. Can't accept any arguments and can't take any extended loops.
2. It can recognize any variable that exists in the current session because it behaves just like a a list of commands in sequence.

On the other hand, a procedure:

1. Can accept arguments, but can't recognize any variables which exist in current IDL session.

However, I often run into the situation that I need a code which can recognize variables in current session \*as well as\* taking arguments. Is it possible to write such a code?

---

---

Subject: Re: What is the main difference between a script and a procedure?

Posted by [David Fanning](#) on Fri, 05 Oct 2007 13:39:37 GMT

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

---

mystea writes:

- > I wonder if there a better way to accomplish this task? In the ideal
- > situation, I wish I could keyin something like:
- > "@qContour, 0.5, 1" to accomplish the task done by:
- > IDL>overplot=1
- > IDL>@qContour
- > IDL>z05=zq

Well written procedures and functions come to mind, but...

Cheers,

David

--

David Fanning, Ph.D.

Fanning Software Consulting, Inc.

---

Subject: Re: What is the main difference between a script and a procedure?

Posted by [Jean H.](#) on Fri, 05 Oct 2007 15:43:00 GMT

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

---

> I don't want to use a procedure because I do want to keep z00, z05 and  
> z08 after I finished plotting. They contain useful information and I  
> might want to shade\_surf them or print them.

Well... but you ARE keeping them!!!

What you can do is to create a function that takes q as the parameter  
(or whatever), creates the variable, plot it and return your Z variable...

You can call this function from the command line

```
Z00 = myPlotFunc(0)
```

then

```
Z05 = myPlotFunc(0.5) etc!
```

If you want to use it in a program, it is fine too!

Jean

PS: always remember the rule of the least effort... if you type the same  
things again and again, there is certainly a way of automating it!

> My own way of doing this task so far is:  
>  
> <setup.pro>  
> .compile make2Darr.pro  
> xaxis=5.0+0.2\*findgen(200)  
> yaxis=3.0+0.1\*findgen(200)  
>  
> <qContour.pro>  
> zq=make2Darr(q)  
> contour, zq, xaxis, yaxis, levels=[0], overplot=overplot  
>  
> IDL>@setup  
> IDL>q=0  
> IDL>overplot=0  
> IDL>@qContour  
> IDL>z00=zq  
> IDL>q=0.5  
> IDL>overplot=1  
> IDL>@qContour  
> IDL>z05=zq

> ...  
> ...  
>  
> I wonder if there a better way to accomplish this task? In the ideal  
> situation, I wish I could keyin something like:  
> "@qContour, 0.5, 1" to accomplish the task done by:  
> IDL>overplot=1  
> IDL>@qContour  
> IDL>z05=zq  
>  
> Sincerely,  
>  
> Gene  
>  
> On Oct 3, 6:40 am, David Fanning <n...@dfanning.com> wrote:  
>> mystea writes:  
>>> As far as I can tell, a script:  
>>> 1. Can't accept any arguments and can't take any extended loops.  
>>> 2. It can recognize any variable that exists in the current session  
>>> because it behaves just like a a list of commands in sequence.  
>>> On the other hand, a procedure:  
>>> 1. Can accept arguments, but can't recognize any variables which exist  
>>> in current IDL session.  
>> What you are calling a "script", most people call a  
>> "batch file". This is a way to execute a series of  
>> commands "as if" you were typing them at the IDL  
>> command line. Since this is just about the most limited  
>> way of using IDL, batch files are typically used infrequently.  
>>  
>> More often people will put the same commands into  
>> a file and add an END statement at the end of the file.  
>> This file is now a "main-level program". It must be  
>> compiled before it can be executed. Normally the  
>> compile and execute is done with the .RUN executive  
>> command. The big advantage of main-level programs over  
>> batch files, is that you can include extended loops, etc.  
>> in a main-level program without all the shenanigans  
>> required to get a loop to work on the IDL command line.  
>>  
>> As you become more sophisticated in your programming, you  
>> will eventually realize that having all your variables  
>> in one big pot is probably not such a great idea. (Especially  
>> if you tend to name all your variables "a" to avoid a lot  
>> of typing.) At that point, you might be interested in writing  
>> procedures and functions (just another term for "IDL commands")  
>> that do particular things for you, while at the same time,  
>> keeping their internal variables from contaminating your  
>> main-level working space.

>>  
>> IDL uses a "pass by reference" method of getting variables  
>> into and out of commands, so it is easy to write procedures  
>> and functions that change main-level variables, if that is  
>> your purpose. You do, in fact, have to pass the variables into  
>> the procedure or function via arguments or keywords, however,  
>> since all the "action" occurs on a level separate from the  
>> main level. (There are ways to access main-level variables  
>> from within procedures and functions that don't involve passing  
>> the variables, but this is rarely done, and only by experienced  
>> programmers who REALLY know what they are doing and why they are  
>> doing it.)  
>>  
>>> However, I often run into the situation that I need a code which can  
>>> recognize variables in current session \*as well as\* taking arguments.  
>>> Is it possible to write such a code?  
>> This is called "having your cake and eating it, too". It is  
>> as easy to do in IDL as it is in life. :-)  
>>  
>> And, anyway, what could you possibly pass to a batch file  
>> that the batch file didn't already know about? The only thing  
>> you can pass are things that exist at the main IDL level, and  
>> the batch file already has access to all of that.  
>>  
>> Cheers,  
>>  
>> David  
>>  
>> --  
>> David Fanning, Ph.D.  
>> Fanning Software Consulting, Inc.  
>> Coyote's Guide to IDL Programming:<http://www.dfanning.com/>  
>> Sepore ma de ni thui. ("Perhaps thou speakest truth.")  
>  
>

---