Subject: Python and IDL
Posted by Michael Wallace on Fri, 28 May 2004 15:38:08 GMT
View Forum Message <> Reply to Message

I'm probably the only person here who'd be interested in this, but I guess I'll throw it out anyway.

I have recently started using Python and love the language. Python is interpreted, interactive, object oriented, and has dynamic data types and dynamic typing. From this perspective, IDL and Python are very similar. There are even projects such as SciPy (http://www.scipy.org) which add some scientific analysis to the language.

Python does have a couple nice benefits, namely the huge number of modules available where there's everything from operating system commands to networking to XML to email etc. One thing I've found myself doing recently is writing my core analysis processing code in IDL and then writing a Python wrapper around the particular code. For example, in one of my programs, I use Python to automatically download files from an FTP site and then call IDL to do the specific processing on the files. In another program, I use Python to process command line arguments and then call the appropriate IDL code based on the arguments.

Currently, in my Python programs I open a pipe to an IDL process and write the IDL commands to the pipe. What I'm wondering is if there is an efficient way to send data from IDL back to Python. The only thing I've found so far is to have IDL write data to stdout and set up the Python side to read this stream. However, this can get pretty inefficient at times. Without any direct conversion between the two languages available, are there any other (i.e. better, efficient) ways to set up bi-directional interprocess communication between Python and IDL?

What'd be really nice is if RSI provided some mechanism for IDL and Python to communicate at a much closer level than interprocess communication. Karl, that's a hint. ;-)

-Mike

Subject: Re: Python and IDL
Posted by idl_toolmaker on Wed, 02 Jun 2004 15:55:56 GMT
View Forum Message <> Reply to Message

Michael Wallace <mwallace.no.spam@no.spam.swri.edu.invalid> wrote in message news:
>
> Currently, in my Python programs I open a pipe to an IDL process and
> write the IDL commands to the pipe. What I'm wondering is if there is
> an efficient way to send data from IDL back to Python.

Does Python pass data by reference, like IDL? You can work with pointers to your data structure, maybe.

- Bob

Subject: Re: Python and IDL
Posted by Stein Vidar Hagfors H[2] on Thu, 03 Jun 2004 13:21:09 GMT
View Forum Message <> Reply to Message

Michael Wallace <mwallace.no.spam@no.spam.swri.edu.invalid> writes:

> I'm probably the only person here who'd be interested in this, but I
> guess I'll throw it out anyway.

Well, you're not quite alone ;-)

[...]
> Currently, in my Python programs I open a pipe to an IDL process and
> write the IDL commands to the pipe. What I'm wondering is if there is
> an efficient way to send data from IDL back to Python. The only thing
> I've found so far is to have IDL write data to stdout and set up the
> Python side to read this stream. However, this can get pretty
> inefficient at times. Without any direct conversion between the two
> languages available, are there any other (i.e. better, efficient) ways
> to set up bi-directional interprocess communication between Python and
> IDL?

My advice would be to use remote procedure calls (RPC) from python to an instance of IDL running as an RPC server. It would need some development of wrapper routines callable from python (written in c). For the IDL side of it, look at the files in $IDL_DIR/external/rpc/. The example.c is a bare-bones example of how to connect & send command lines to IDL. Look at e.g. idl_rpc_test.c to see examples of how to use the numerous features.

With RPC calls, you can set or get variables "directly". For even more efficiency, one could even build IDL & python together using Callable IDL.

> What'd be really nice is if RSI provided some mechanism for IDL and
> Python to communicate at a much closer level than interprocess
> communication. Karl, that's a hint. ;-)

It would be nice, yes. Due to the large body of legacy code that many projects around here are relying on, the likelihood of switching entirely to python is very small. But a number of people would prefer

using python for the "scripting" parts of a data processing pipeline.

--

Stein Vidar Hagfors Haugan
ESA SOHO SOC/European Space Agency Science Operations Coordinator for SOHO

NASA Goddard Space Flight Center, Tel.: 1-301-286-9028
Mail Code 682.3, Bld. 26, Room G-1, Cell: 1-240-354-6066
Greenbelt, Maryland 20771, USA. Fax: 1-301-286-0264

Subject: Re: Python and IDL
Posted by Nigel Wade on Fri, 04 Jun 2004 10:13:43 GMT
View Forum Message <> Reply to Message

On Fri, 28 May 2004 10:38:08 -0500, Michael Wallace wrote:

> I'm probably the only person here who'd be interested in this, but I
> guess I'll throw it out anyway.
>
> I have recently started using Python and love the language. Python is
> interpreted, interactive, object oriented, and has dynamic data types
> and dynamic typing. From this perspective, IDL and Python are very
> similar. There are even projects such as SciPy (http://www.scipy.org)
> which add some scientific analysis to the language.
>
> Python does have a couple nice benefits, namely the huge number of
> modules available where there's everything from operating system
> commands to networking to XML to email etc. One thing I've found myself
> doing recently is writing my core analysis processing code in IDL and
> then writing a Python wrapper around the particular code. For example,
> in one of my programs, I use Python to automatically download files from
> an FTP site and then call IDL to do the specific processing on the
> files. In another program, I use Python to process command line
> arguments and then call the appropriate IDL code based on the arguments.
>
> Currently, in my Python programs I open a pipe to an IDL process and
> write the IDL commands to the pipe. What I'm wondering is if there is
> an efficient way to send data from IDL back to Python. The only thing
> I've found so far is to have IDL write data to stdout and set up the
> Python side to read this stream. However, this can get pretty
> inefficient at times. Without any direct conversion between the two
> languages available, are there any other (i.e. better, efficient) ways
> to set up bi-directional interprocess communication between Python and IDL?
>
> What'd be really nice is if RSI provided some mechanism for IDL and

> Python to communicate at a much closer level than interprocess
> communication. Karl, that's a hint. ;-)
>
> -Mike

The cleanest way would be to create a Python DLM, built using an embedded Python interpreter. The DLM interface routines could pick apart any IDL variables and create Python variables (possibly) using the same memory (you'd need to check on row-major vs. column-major issues for multi-dimensional arrays), then add these Python variables to the global dictionary.

I've built an embedded interpreter into a C program, but not an IDL DLM. I don't think there are any intrinsic reasons why it wouldn't be possible.

--
Nigel Wade, System Administrator, Space Plasma Physics Group,
University of Leicester, Leicester, LE1 7RH, UK
E-mail : nmw@ion.le.ac.uk
Phone : +44 (0)116 2523548, Fax : +44 (0)116 2523555

Subject: Re: Python and IDL
Posted by Bob[2] on Fri, 04 Jun 2004 17:54:13 GMT
View Forum Message <> Reply to Message

Michael Wallace wrote:

>
> What'd be really nice is if RSI provided some mechanism for IDL and
> Python to communicate at a much closer level than interprocess
> communication. Karl, that's a hint. ;-)

Michael, take a look at "External Development Guide" in the IDL help documentation, especially the first chapter. They guide you through the intricacies, e.g., p.13:
--
Supported Inter-Language Communication Techniques in IDL

IDL supports a number of different techniques for communicating with the operating system and programs written in other languages. These methods are described, in brief, below.

Options are presented in approximate order of increasing complexity. We recommend that you favor the simpler options at the head of this list over the more complex ones that follow if they are capable of solving

your problem.
--

-Bob