Subject: Beginner, Question
Posted by justdoit0429 on Tue, 25 Aug 2015 04:49:42 GMT
View Forum Message <> Reply to Message

I'm not programmer.

I have a question.

What is a advantage of idl. comparison with python?

It is not sarcastic. just question.

Subject: Re: Beginner, Question
Posted by Jeremy Bailin on Tue, 25 Aug 2015 14:06:57 GMT
View Forum Message <> Reply to Message

On Monday, August 24, 2015 at 11:49:44 PM UTC-5, justdo...@gmail.com wrote:
> I'm not programmer.
>
> I have a question.
>
> What is a advantage of idl. comparison with python?
>
> It is not sarcastic. just question.

The major reason that it is used a lot in astronomy is because there is a large well-tested codebase out there of routines that make oftentimes very complicated and necessary astronomical tasks fairly easy. There is currently a large effort underway to replicate all of that functionality in Python, and they are making great strides, but for many many tasks, the best and easiest way of doing them is using existing IDL routines.

I suspect the situation is similar in the other fields where IDL is prominent -- it's not so much that the task can't be done in Python, as that the effort required to rewrite routines with the existing sophistication and robustness hasn't been finished yet.

-Jeremy.

Subject: Re: Beginner, Question
Posted by chris_torrence@NOSPAM on Wed, 26 Aug 2015 15:00:27 GMT
View Forum Message <> Reply to Message

On Tuesday, August 25, 2015 at 8:06:59 AM UTC-6, Jeremy Bailin wrote:
> On Monday, August 24, 2015 at 11:49:44 PM UTC-5, justdo...@gmail.com wrote:
>> I'm not programmer.
>>

>> I have a question.
>>
>> What is a advantage of idl. comparison with python?
>>
>> It is not sarcastic. just question.
>
> The major reason that it is used a lot in astronomy is because there is a large well-tested codebase out there of routines that make oftentimes very complicated and necessary astronomical tasks fairly easy. There is currently a large effort underway to replicate all of that functionality in Python, and they are making great strides, but for many many tasks, the best and easiest way of doing them is using existing IDL routines.
>
> I suspect the situation is similar in the other fields where IDL is prominent -- it's not so much that the task can't be done in Python, as that the effort required to rewrite routines with the existing sophistication and robustness hasn't been finished yet.
>
> -Jeremy.

Hi Jeremy,

Good answer. Also, not to start a flame (or religious) war, but I would also say that both languages have their strengths and weaknesses. IDL's interpreter is faster than Python's, so if you are writing lots of general purpose code, or trying to build a large application, then IDL may serve you better. IDL's image processing routines are in general faster, although this may vary depending upon the routine and the platform. Finally, IDL's built-in array routines are much faster than Python's built-in arrays (not numpy!). On the other hand, Python's numerical routines (especially in linear algebra) may be faster on certain platforms (such as Intel), where they have been fine tuned. Finally, Python has a broader selection of modules available, although these can be of varying quality since it is open source.

Finally, one more point - since IDL 8.5 now includes the Python->IDL and IDL->Python bridge, you really can't go wrong either way.

Cheers,
Chris

Subject: Re: Beginner, Question
Posted by penteado on Wed, 02 Sep 2015 20:22:02 GMT
View Forum Message <> Reply to Message

Also not trying to start a war, I can add my view on this. From the perspective of an astronomer who has worked a lot with IDL and Python (disclaimer: I have a lot more experience and knowledge in IDL than in Python). My personal reasons for preferring IDL over Python as a language are, most importantly:

1) Better vectorization. Numpy is very advanced, but because it is just a module, vectorization is not (for now, at least) part of Python's semantics. This makes vector operations (particularly in

arrays that are not 1D) awkward in Python, requiring many transformations between Numpy and native objects, and/or lots of loops.

2) Portability is way better in IDL. Getting some complex program to work on Python, due to their many dependencies on specific versions of other libraries, is often a lot of (platform-specific) trouble. This is what causes several vendors/distributors to package their whole thing into a self-contained Python distribution (Anaconda, Ureka, yt, etc). Which results in one having several independent Python distributions installed, each one usable for one software package, and is a big obstacle to write code that uses functionality from more than one of those distributions. Additionally, code that works now may easily get broken by the next non-backward compatible update in some library. Plus, there is a whole other source of trouble with the Python 2 vs Python 3 issue, which has existed for many years, and probably will keep being an issue for a long time.

3) Better visualization libraries in IDL, which are built-in, unlike Python, which may need messing around with installing several other platform-dependent libraries or getting them to work with the kind of graphics device in use.

4) More functional IDE and debugger in IDL. I have been through many Python IDEs, and I was never satisfied. None seemed to work for debugging as well as IDL's. In Python IDEs there is always some trouble setting the environment or the program execution, or things that seem to decide not to work.

I do recognize that Python has some advantages. Mostly, being free and the greater number general-purpose (as opposed to scientific) modules available. That said, I have to note that languages are not chosen just in an abstract way, considering their intrinsic characteristics. Each project has different constraints, such as availability of useful libraries, existing code, availability of IDL licenses, and language preferences of other people involved in the project or future code users. All of these influence what I find to be the better choice for that particular project.

Paulo

Subject: Re: Beginner, Question
Posted by Fabzi on Thu, 03 Sep 2015 11:43:16 GMT
View Forum Message <> Reply to Message

Thanks Paulo, I totally agree with your analysis. Having tried out
Python myself for a new project, I will add a few things:

1) awkwardness of Numpy

I totally agree with you and dislike the mix between numpy's ndarray's
and python lists. Specially the syntax overhead is not exactly aesthetic
(eg a = np.array([1,2,3]) instead of a simple a = [1, 2, 3]). However,
I've come to really appreciate some of Numpy's broadcasting syntax,
which simplifies some operations, for example:

```
>>> a = np.zeros((3, 4))
```

```
 >>> a += [1, 2, 3, 4]
 >>> a
array([[ 1.,  2.,  3.,  4.],
       [ 1.,  2.,  3.,  4.],
       [ 1.,  2.,  3.,  4.]])
```

To be compared to IDL:

```
IDL> a = FLTARR(4, 3)
IDL> a += [1, 2, 3, 4] # (FLTARR(3)+1)
IDL> a
      1.0000000       2.0000000       3.0000000       4.0000000
      1.0000000       2.0000000       3.0000000       4.0000000
      1.0000000       2.0000000       3.0000000       4.0000000
```

And there are some other advantages (e.g the ellipsis syntax:
a[1:3, ...] which helps you not to bother about variable dimensions any more)

2) Debugger

I agree with you again. One thing that surprised me is how slow Python debuggers are, and that they require a separate running mechanism. IDL in turn has no formal "debug mode" (just put a breakpoint and wait) and is running at the exact same speed. Does anyone know why the two debuggers are so different?

3) Python third party libraries

The major reason for choosing python for my new project (aside of the license issue) is simply that I spared a HUGE amount of work, simply because the things we badly needed were already there. I wont give a full list here but basically, after a year using libraries such as Pandas, Shapely, or Scikit-Learn, going back to IDL for serious data crunching is quite frustrating...

I think that the recent efforts of the IDL devs to make the syntax more flexible are the right ones, and the idea to allow python - IDL bindings is really great. But IMO the next steps should be a revisiting of some unsatisfying basic functionalities:
- the handling of time: IDL's floating julian days are not adapted to today's standards, and plotting timeseries still is a pain.
- it would be great to have indexed and labeled arrays such as in Pandas and xray...
- it would be much nicer to be able to do:

```
IDL> a = list([1, 2, 3], DICTIONARY('a', 1, 'b', 2))
IDL> a[0][1]
       2
IDL> a[1].b
       2
```

instead of the current syntax:

```
IDL> (a[0])[1]
       2
IDL> (a[1])['b']
       2
```

Cheers,

Fabien

On 09/02/2015 10:22 PM, Paulo Penteado wrote:
> snip

Subject: Re: Beginner, Question
Posted by Yngvar Larsen on Thu, 03 Sep 2015 12:31:15 GMT
View Forum Message <> Reply to Message

On Thursday, 3 September 2015 13:43:18 UTC+2, Fabien wrote:

> - it would be much nicer to be able to do:
>
> IDL> a = list([1, 2, 3], DICTIONARY('a', 1, 'b', 2))
> IDL> a[0][1]
>   2
> IDL> a[1].b
>   2
>
> instead of the current syntax:
>
> IDL> (a[0])[1]
>   2
> IDL> (a[1])['b']
>   2

```
IDL> a = list([1, 2, 3], DICTIONARY('a', 1, 'b', 2))
IDL> a[0,1]
      2
IDL> a[1,'b']
      2
```

--
Yngvar

Subject: Re: Beginner, Question
Posted by Fabzi on Thu, 03 Sep 2015 12:32:29 GMT
View Forum Message <> Reply to Message

On 09/03/2015 02:31 PM, Yngvar Larsen wrote:
> IDL> a = list([1, 2, 3], DICTIONARY('a', 1, 'b', 2))
> IDL> a[0,1]
> 2
> IDL> a[1,'b']
> 2

Nice! I missed this one. Since when is it available?

Subject: Re: Beginner, Question
Posted by Yngvar Larsen on Thu, 03 Sep 2015 12:49:21 GMT
View Forum Message <> Reply to Message

On Thursday, 3 September 2015 14:32:31 UTC+2, Fabien wrote:
> On 09/03/2015 02:31 PM, Yngvar Larsen wrote:
>> IDL> a = list([1, 2, 3], DICTIONARY('a', 1, 'b', 2))
>> IDL> a[0,1]
>> 2
>> IDL> a[1,'b']
>> 2
>
> Nice! I missed this one. Since when is it available?

I think since these constructs where introduced, i.e. 8.0 for LIST/HASH and 8.3 for DICTIONARY/ORDEREDHASH.

--
Yngvar

Subject: Re: Beginner, Question
Posted by Jim Pendleton on Thu, 03 Sep 2015 13:04:18 GMT
View Forum Message <> Reply to Message

On Thursday, September 3, 2015 at 6:49:24 AM UTC-6, Yngvar Larsen wrote:
> On Thursday, 3 September 2015 14:32:31 UTC+2, Fabien wrote:
>> On 09/03/2015 02:31 PM, Yngvar Larsen wrote:

>>> IDL> a = list([1, 2, 3], DICTIONARY('a', 1, 'b', 2))
>>> IDL> a[0,1]
>>>   2
>>> IDL> a[1,'b']
>>>   2
>>
>> Nice! I missed this one. Since when is it available?
>
> I think since these constructs where introduced, i.e. 8.0 for LIST/HASH and 8.3 for DICTIONARY/ORDEREDHASH.
>
> --
> Yngvar

There's also new syntax for auto-instantiation of nested hashes in IDL 8.5, which you may find useful.

http://www.exelisvis.com/docs/WhatsNew.html

Jim P.

Subject: Re: Beginner, Question
Posted by Jeremy Bailin on Thu, 03 Sep 2015 17:28:15 GMT
View Forum Message <> Reply to Message

On Wednesday, September 2, 2015 at 3:22:04 PM UTC-5, Paulo Penteado wrote:
> Also not trying to start a war, I can add my view on this. From the perspective of an astronomer who has worked a lot with IDL and Python (disclaimer: I have a lot more experience and knowledge in IDL than in Python). My personal reasons for preferring IDL over Python as a language are, most importantly:
>
> 1) Better vectorization. Numpy is very advanced, but because it is just a module, vectorization is not (for now, at least) part of Python's semantics. This makes vector operations (particularly in arrays that are not 1D) awkward in Python, requiring many transformations between Numpy and native objects, and/or lots of loops.
>
> 2) Portability is way better in IDL. Getting some complex program to work on Python, due to their many dependencies on specific versions of other libraries, is often a lot of (platform-specific) trouble. This is what causes several vendors/distributors to package their whole thing into a self-contained Python distribution (Anaconda, Ureka, yt, etc). Which results in one having several independent Python distributions installed, each one usable for one software package, and is a big obstacle to write code that uses functionality from more than one of those distributions. Additionally, code that works now may easily get broken by the next non-backward compatible update in some library. Plus, there is a whole other source of trouble with the Python 2 vs Python 3 issue, which has existed for many years, and probably will keep being an issue for a long time.
>
> 3) Better visualization libraries in IDL, which are built-in, unlike Python, which may need

messing around with installing several other platform-dependent libraries or getting them to work with the kind of graphics device in use.
>
> 4) More functional IDE and debugger in IDL. I have been through many Python IDEs, and I was never satisfied. None seemed to work for debugging as well as IDL's. In Python IDEs there is always some trouble setting the environment or the program execution, or things that seem to decide not to work.
>
> I do recognize that Python has some advantages. Mostly, being free and the greater number general-purpose (as opposed to scientific) modules available. That said, I have to note that languages are not chosen just in an abstract way, considering their intrinsic characteristics. Each project has different constraints, such as availability of useful libraries, existing code, availability of IDL licenses, and language preferences of other people involved in the project or future code users. All of these influence what I find to be the better choice for that particular project.
>
> Paulo

Yes, all good points.

I'm never sure whether speed issues are intrinsic to the languages/libraries or to my use of them. I know how to write fast IDL code -- I leap straight to the histogram/value_locate/etc tricks immediately. I don't know the equivalents in Python, i.e. what things are fast, what things are slow, what tricks to use. So usually my IDL is faster than my Python. On the other hand, I know Python people who say the exact opposite -- their Python is usually faster than their IDL. I'm not sure how my IDL compares to their Python in speed, though!

-Jeremy.

Subject: Re: Beginner, Question
Posted by penteado on Thu, 03 Sep 2015 21:11:25 GMT
View Forum Message <> Reply to Message

On Thursday, September 3, 2015 at 5:49:24 AM UTC-7, Yngvar Larsen wrote:
> I think since these constructs where introduced, i.e. 8.0 for LIST/HASH and 8.3 for DICTIONARY/ORDEREDHASH.

No, the additional indices were added a little later, in 8.1, after some user feedback on 8.0.

Other useful and often overlooked features are the optional arguments to list::toArray and hash::toStruct. Like the dimensions keyword to list::toArray(), which came in 8.2.3.
