DØ Note 3171

Recommendations on Scripting Language for D0 Experiment

Fritz Bartlett, John Hobbs, Qizhong Li, Lee Lueking, Scott Snyder,
Roy Thatcher, and Gordon Watts

I. THE RECOMMENDATION

The D0 Script Languages Committee strongly recommends that the language Python be used for all scripts with the exception of certain rare and special cases. The committee looked at 18 possible candidates and Python is clearly the best. There is only one small area in which another candidate is superior; Perl has the edge in compiled regular expressions. We strongly recommend that use of anything other than Python be restricted to only three cases:

1. Applications in which all of the existing scripts are written in some other language.
2. Applications for which some other language has a significant advantage. For example, Perl when extensive use of regular expressions is required.
3. Very short shell scripts or script to change the current environment.

What is most strongly DISCOURAGED is writing any scripts in C-shell or any of its variants. C-shell scripts are extremely fragile and untrustworthy. We realize that many existing scripts were written in csh which was mistakenly recommended several years ago. However, we must not perpetuate this unfortunate practice.

II. WHAT IS PYTHON?

Python is a full fledged programming language that is interpreted, byte-compiled, extensible, and embeddable. It is largely procedural but supports objects. Its syntax is very straightforward and promotes clarity of coding. In fact, clarity of code has been one of its principle design criteria since its creation. It interfaces well to TK for graphical interfaces and to C or C++ wherever needed and it can be called from them as well. It even has threads, is good at text processing, and can issue system commands. It currently runs on the following systems:

- UNIX,
- Microsoft Windows - win16,
- Microsoft Windows - win32,
- Microsoft Windows - NT,
- Microsoft Windows 95,
- VMS,
- Next,
- Amiga,
- Atari Mint,
- Atari TOS,
- and QNX.

Python is easy to learn. There are at least two excellent web tutorials for Python, which allow one to get a jump-start. In addition there are two books.:

- an O'Reilly book: "Programming Python" by Mark Lutz and
- a book on using Python for Internet applications: "Internet Programming with Python" by Aavon Watters, Guido van Rossum and James C. Ahlstrom.

The O'Reilly book is remarkably clear and complete. It includes clear expositions of concepts new to those who have used only older languages such as Fortran. We expect to be able to offer classes on site in the near future.

Python is relatively new and therefore modern and its development has until fairly recently been done by one man, Guido van Rossum, formerly of CWI in Amsterdam but now at Corporation for National Research Initiative (CNRI) in Reston, Virginia. Python is freeware. We worried about the support issues at considerable length but are now confident that language support for Python is solid and assured at least the next decade and probably well beyond. There are now a number of other talented people besides Mr. Van Rossum working on the enhancement of Python. However, it will not "grow like Topsy". Any enhancements will clearly be integrated and tightly compatible. There is an enthusiast and vigorous users group (Python Software Activity) which maintains a news group.

III. WHY WE DID NOT PICK THE OTHER CANDIDATES

We cast a very broad net as indicated by the 18 initial candidates listed above. However, most of these had obvious short-comings and were quickly eliminated. Those languages making it past the first cut are Tcl, Perl, Python and Java.

Java was never intended as a scripting language. It is possible to extend Java and to embed it but both processes are more difficult than with Python. Java is much more verbose than Python and less readable. Typically a Python program can do the same thing as a Java program in 1/3 the number of lines.

Tcl is a simplistic language and extremely difficult to use for complex programs. There is no support for complicated data structures and there is no check on arithmetic overflow. Error catching and processing in Tcl is primitive compared to Python. Tcl relies on dynamic scope which makes it hard to reuse code. Tcl does not convert code to an internal format. This means that code has to be parsed every time it is executed. In Tcl everything is a string. In contrast to Tcl Python is byte-compiled (this means that it is converted to an internal format on first read). Python is roughly twice as fast as Tcl. Tcl is easy to learn but so is Python.

Perl is wonderful at certain tasks, namely those text processing tasks for which regular expressions are important. But its syntax is very difficult and confusing, particularly when compared with Python. Many experienced Perl programmers have a hard time modifying their own scripts. In an attempt to find a "Devil's Advocate" we asked Tom Christiansen, who makes his living teaching Perl and who is co-author of the O'Reilly book on Perl 5, "Are there any areas where it significantly harder to use Perl than Python?". His answer:

"Pre-Compiled regex objects."

IV. SUPPORT ISSUES

The support issues are critical and we sought answers from many sources including Python's creator, his employer, and major users. Two places that have committed mission critical projects to Python are NASA (for the shuttle) and LLNL. Both organizations had to do very thorough justification of Python's selection. See appendices A and B.

Mr. Van Rossum's Plans for Python Support: To research these issues we contacted Python's creator, Guido van Rossum, to find out his intentions and who would step in if he were to stop supporting Python. He is currently employed full-time by CNRI to support and enhance Python. Here is what Mr. Van Rossum said about CNRI's plans for Python support:

"...things look very bright.

First of all, CNRI, my current employer, is interested in coming out with a "professional" version of Python in addition to continuing to maintain the free version. The free and non-free versions will be aimed at very different audiences – the free code will be useful for researchers and hackers, the pro version will give managers of big corporations or projects such as yourself support and peace of mind. (Sometimes the free version will be used as a sandbox to experiment with proposed features before they go into the pro version; sometimes the pro version will contain optimized versions of modules available in the free version.) While this is not at all a committed project, there is much interest, and I expect that it will eventually happen – certainly before I move to the Frisian islands to paint Dutch farmer's daughters."

V. WHAT ABOUT THE SPONSOR CNRI?

CNRI looks very solid. It has been around and growing steadily since 1986. CNRI is a non-profit organization dedicated to formulating, planning and carry out national-level research initiative on the use of network based technology. It has contracts with both industry and the US government. CNRI includes on its board of directors, Thomas Everhart,the president of Cal Tech; Joel Birnbaum, a VP of Hewlett-Packard; Richard Cyert, past president of Carnegie Mellon; Joshua Lederberg, Past President of Rockerfeller University. Robert Kahn, the president of the organization is the person responsible for the design of the the ARPANET, the first packet switched network. He is also co-inventor of the TCP/IP protocols and was responsible for originating DARPA's Internet Program. The survival of CNRI looks not to be an issue for us nor does its commitment to Python. However, in the unlikely case in which CNRI would no longer be able to support Python, LLNL could and would take over. (See Appedix B.)

VI. TIME AND EFFORT INSTALLING UPGRADES

We have no support from the Fermilab Computing Division for Python yet. However, that looks to be a rather minor problem at worst. Python is extremely easy to maintain. As described below United Space Alliance & NASA JSC (Rockwell-NASA) has committed to using Python for critical applications. According Robin Friedrich of Rockwell-NASA: "In our case we stated that based of past experience the cost of building new versions of the interpreter when they come out as only 5 man-days per year, (actually it's more like 1, but they wouldn't have believed me)."

VII. HELP WITH PYTHON QUESTIONS:

The on-net documentation is very good. The books are excellent. Perhaps just as important is that the Python news group is reported to give quick and correct answers to questions. Again according to Robin Friedrich of Rockwell-NASA: " Also then questions or issues come up we rely on the net for answers, which have historically been very prompt (even better than response time from commercial software vendors)."

VIII. APPENDIX A: NASA'S COMMITMENT TO PYTHON

Robin Friedrich of Rockwell-NASA wrote: "As you know, Python is a superb language both because of it's clear syntax as well as it's powerful object oriented structure conducive for reuse of code. However, in the real world, technical superiority is not always sufficient for the success of long term solutions. Many managers dwell in the world of problem avoidance and don't give much weight to technical virtues. As your message indicates, it boils down to the best *supportable* solution for the project(s) at hand. We faced the same issues here at NASA. Too often managers have been burned by "hobby shop" software put into production just to find that there is no one with the skills to maintain it. We went through a technology insertion process and how Python is to be maintained was definitely a question that had to be answered. In our case we stated that based of past experience the cost of building new versions of the interpreter when they come out as only 5 man-days per year, (actually it's more like 1, but they wouldn't have believed me). Also then questions or issues come up we rely on the net for answers, which have historically been very prompt (even better than response time from commercial software vendors). This makes Python a very low cost, high productivity language. The alternatives were all much more costly. In fact, as of a few weeks ago Python became the first freeware allowed in Mission Control (a net of DEC Alphas) and will be used in analysis applications before and during shuttle flights. Our experience with Python has been most satisfactory.

In the case of Python, we have a language, interpreter and extension framework, which is actively being supported and enhanced by many people, not just Guido. Guido have been the keeper of the core interpreter since inception in the early 90s, but if he should wish to do other things entirely (which he doesn't, but just for the sake of argument), he would pass the 'sword' to any one of several highly qualified programmers. I can think of three off the top of my head who would probably be willing to take the reins in that unlikely event.

What is more likely to happen in the future, as Guido mentioned, is that CNRI would package a "Python Pro" version with full maintenance support contracts available. It should not be necessary though to wait for this, as I strongly believe in the support available within the active Python community. I have used Python here for three years now and have not come up against a problem that hasn't been quickly addressed by posts to comp.lang.python or email to select people in the community. I have not felt the need for a formalized support contract yet, and if one were available I'm not sure I would use it. So I don't think that waiting for such a formal arrangement is the way to go either.

The kind of thing you would need a support contract for are bugs in the interpreter and new versions. Both of these issues are dealt with nicely with the current community. Bugs are quite rare and solutions are readily available in my experience. Python as a language is not the kind of software which up and breaks one day. Common applications like word processors or browsers can and do exhibit this behavior and for those types of software a support channel is obviously a good thing. But I think there is a categorical difference between say, GUI applications, and language processors. Languages, especially after years of use, are well understood things not subject to unpredictable failure modes. Python itself is only 50K lines of well written C code. Most applications these days are far larger and complex and written in a rush. My point is that the criticality of a support contract is significantly reduced by the nature of this class of software.

You don't mention what the competing language approaches are so I can't provide a contrasting argument. But I can say that in my experience, I have not seen a language with so much going for it as Python. We selected Python as our standard scripting language for Shuttle flight design last year and have not regretted it at all. The language's clean syntax and structure has been easy for our end users (engineers not programmers) to pick up and run with. Being able to understand code written by someone else is a rather important trait for most large shops. If the author of some code is not around or has left, it's important to be able to maintain and enhance it without large investments in time. Perl, for example, doesn't exhibit this trait and consequently maintenance and reuse can be inhibited. These considerations are important to us because we plan to be using Python for a long time.

Python's future is very bright. Two books have been published and more are on the way. O'Reilly and Assoc. has publicly stated that they wish to publish a Python series.

You can stress to management that Python is not solely maintained and enhanced by Guido alone. The Numeric extension (NumPy) for example, which I suppose you will be dipping into heavily, was developed entirely by people other than Guido. The upshot is that, because of Python's maturity and it's large community of devotees, Guido's absence, while missed greatly, would hardly kill the language as a viable tool for production software.

Bottom line, Python will be viable and supported for many years to come. The advent of more publicized languages (like Java) have not dampened the enthusiasm people have for

Python. To the contrary, Python fills a role as a glue framework and scripting language very well now and is getting more powerful all the time. Technical people recognize this easily but the challenge can be to illuminate the non-technical people who may have authority over decisions. I personally do not worry about support for the language into the 21st century.

If you want to read more of my feelings on these topics you might look at http://www.python.org/python/Commercial.html."

IX. APPENDIX B: LLNL COMMITMENT TO PYTHON

Paul Dubois of LLNL wrote:

"Python is being used here at LLNL for the steering language for the project called Kull. This is one half of LLNL's Advanced Strategic Computing Initiative three-dimensional modeling effort. To give you a sense of scale, the computer on which this code is to run costs $92,000,000. The code is expected to be in use for twenty years at least.

We have at least three people on this project with extensive experience writing and maintaining compilers and interpreters. In addition, Lee Busby has recently done some patches that required a deep familiarity with the Python source.

The design of Python's core makes it relatively easy to understand and work on. The truth is, if I had to clean-room recreate Python, the project would not take long at all. The massive library of extensions is another story, as each requires expertise in a particular area. But there is a world-wide community of contributors who help.

The bottom line is that LLNL alone would certainly both be capable of and required to maintain Python should CNRI be unable to do so. We have a commitment from our management that our contributions to Python can be released under the original licensing terms as our part of this world-wide collaboration. So, you can count on us. We expect to actually put many codes under Python, not just Kull."

X. APPENDIX C: WHERE ARE THE PYTHON RESOURCES IN D0?

Python is installed on D0chb. Reference manuals and tutorial are included. The main Python web page's url address is: http://www.python.org.