```
% hep
HEP 4.2.0 (#29, 27 Jan 1994, 14:27:38) [GCC 2.5.7] on sun4-solaris
>>> import python
bailing out at line 1

%
```

Jeff Templon[1]*

Abstract
Python took a while to become an accepted language in High-Energy Physics. This short paper traces some of the history behind this path to acceptance, suggests some reasons for this apparently-slow uptake, and projects this reasoning into the future of our field.

Keywords
High-Energy Physics — Python — Computing

[1] *Nikhef, Science Park 105, 1098 XG Amsterdam, The Netherlands*
*__Corresponding author__: templon@nikhef.nl

Contents

Introduction

This paper contains some personal notes about the "early days" of Python in High-Energy and Nuclear Physics, some thoughts about why it went like it went, and whether this bit of history tells us anything about the future.

1. Before Python

I did my Ph.D. at Indiana (IUCF) in Experimental Nuclear Physics. The main computing environment was VAX/VMS, and on that platform, FORTRAN absolutely rocked. "Scripting" things except for canned command sequences was not common, although the DEC command language DCL could be used to do that. Halfway through my time there, the lab purchased a license for SPEAKEASY[1] (which still exists!). The program was an interpreter containing a large library of math, statistics, and graphics functions, and you could define your own functions as well. I used this often as a sort of desk calculator, usually reserving FORTRAN codes for things that needed to do more complex I/O or calculations that took too long in SPEAKEASY.

At some point I discovered GNU `awk`, which had been ported to VAX/VMS system. After that, most of my analyses were scripted. The Fortran programs were written to Do One Thing Well; the orchestration, data gathering and tabulation of all the hundreds of Fortran runs were done by `awk` scripts.

Like most graduate students, I found ways to avoid writing my dissertation. One of them was playing with exotic little languages, and one of them was called ABC[2], a teaching/prototyping langauge from the CWI in Amsterdam. It was a fun little language, very easy to learn.

Upon graduation I moved to NIKHEF-K in Amsterdam for my first postdoctoral position. Unix (Solaris) instead of VAX/VMS, no SPEAKEASY, no DEC FORTRAN. I got used to Unix and standard FORTRAN-77 fairly quickly, but I missed the interactive calculator and statistics library of SPEAKEASY. The unix calculators `bc` and `dc` didn't work at all for me; I wound up (mis)using `awk` for a lot of things. I write "misuse" the `awk` paradigm is "transform input to output", but I was doing other things with it. My `awk` scripts had all the action in the `END` clause; when I invoked them, I had to type control-d to get them to start running.

2. The first three months

On 27 January 1994, the announcement of Python 1.0.0 came out on comp.lang.misc. I decided to download it and check it out because a) it was available on Solaris, b) it was an interpreted language claiming to come with a large library of standard modules, and c) it came from the same place as that little ABC language and I had liked that one, maybe this one was okay too.

I have a reputation of being the first guy in our field to use Python. I'm not — Jon Eisenberg at the University

[1] `http://www.speakeasy.com/default.htm`

[2] `https://en.wikipedia.org/wiki/ABC_(programming_language)`

of Washington reported in October 1993 that he was "developing a data analysis application using python as the interface and user programming language." I could not discover whether he actually completed it. Also, an David Williams (an ex-colleague from IUCF) became active on the Python mailing list a couple of days before I did.

My first substantive post to the Python mailing list was a request for comments on a little program I'd written while playing around with the language – Friday afternoon of March 4, 1994. The exchange was a lot of fun, with both Guido van Rossum and Tim Peters[3] making most of the comments. At the end of my last message in that thread, I made a joke that became reality:

> I don't think I'll become a class writer just yet. Maybe next year. Although I have this nagging temptation to write a good data analysis program using Python ... all we have here is PAW which is a major pain.

That was 9 March 1994. Twelve days later:

> I did write that data-analysis project. Actually what I wrote was a preprocessor; it takes as input some files which specify how you would like the data to be analyzed, and translates this information into FORTRAN (I hear the Scheme guys coughing) code; I then have a FORTRAN main program which includes the Python FORTRAN output and does just a little bookkeeping. This gets compiled and run. It uses the CERN HBOOK package for histogramming so it takes advantage of all the existing code, but it avoids using PAW (CERN's big hitter data-analysis program); PAW is great for presentation and "analysis" of data in near-final form but not good for analyzing large quantities of data when large numbers of parameters need to be investigated.

The code was named DATAN; not that that's important, but useful since I will refer to it later.

After this, things went very fast:

12 april first kinematics program

21 april four-vector class library

22 april run planning program (count rates, kinematics, beam time estimates for needed statistical accuracy)[4]

25 may DATAN was used as the online analysis and monitoring package for an experiment at NIKHEF-K

3. Resistance to Python

After this point, I was using Python for almost everything. I went a bit too far; I was using it in areas where `bash` would have been more appropriate. Time-critical stuff was still (mostly) written in Fortran and the things that beg to be written in `awk` (yes, they do exist) were still written in `awk`, the rest was Python.

It took a long time before I met anybody else who was actually using Python. For a couple of years, some people literally said I was crazy: "How could you use this language for serious work?? YOU DOWNLOADED IT OFF THE INTERNET!!" Recall that in 1994, you could still have made the argument that WWW stood for Wild West Web. When I left Nikhef in 1995, one of my then ex-colleagues rewrote DATAN. The syntax didn't change at all, but it was rewritten in a "proper language" (C).

About this time, the idea of an "extension language" seemed to take off. The "Tcl war" was in late 1994 - Richard Stallman had written a post entitled "Why you should not use Tcl" (as an extension language). In the ensuing flame war, many alternatives were mentioned, amongst which Python. I jumped into the fray at this point with a user-oriented view. I only mention it because it's the only time in my career that multiple computer luminaries (Guido van Rossum, Tom Christiansen and John Oosterhout) have all responded to a post I wrote. The point of my post was: choose Python as the extension language, your users will thank you.

In late 1995 the ROOT project had its first public release. I was exposed to it about a year later, when the collaboration I was working in at the time started talking about using it. Investigating, I was shocked to find that the CLI was based on C++. I had several long conversations with Fons Rademakers and Rene Brun[5] in this period, trying to convince them to use Python for the interpreter. At some point during these discussions, Rene referred to Python as an "exotic language".

The DØ experiment at Fermilab was, as far as I can tell, the first to use Python as an extension language, as evidenced by the following excerpt from a paper[1] from CHEP 1997.

> DØ has made the decision to move all large software projects to C++. Their framework approach has a set of modules that execute sequentially, each having a specific task. The glue that holds the individual software packages together will be an interpreted script system. The main task of this framework is to "guide" data between the various modules/packages ... prototype framework based on the Python scripting language has been developed and is ready for use.

Working in Nuclear Physics, I didn't know about this work until I visited Fermilab a year later[6] That visit influenced

[3] Of "import this" fame (try importing that module from the python command prompt!

[4] I still have this program, and tested it while researching this paper in early 2018. It still runs under Python 2.7.15 with a single change; the `class.init()` syntax is different.

[5] Rene and I had one of these conversations at a barbecue joint in southeastern Virginia. They had paper tablecloths and we filled the entire tablecloth with sketches about how physics computing and analysis should work. I had an extremely enjoyable, entertaining, educational and inspiring afternoon, one of the most memorable of my career. But I did not manage to convince Rene.

[6] May 26, 1998, to give a talk about programming languages and tools for physics. I was pushing Python for non-computationally-intensive work and Oberon-2 for stuff that needed to be fast.

my own thinking, reflected in this excerpt from a technical note[2] presented at a Hall A (Jefferson Laboratory) collaboration workshop in early 1999.

> Most of the code would be written in C or C++, but the integration would be done through Python. This enables the uninitiated to make simple modifications to the analysis which were perhaps not thought of by the authors; all the neophyte needs to know is how the interfaces work. On the other hand, it will force the code authors to make the analysis subsystems independent of each other (one of the big problems with the current code), and will encourage rigorous testing of subunits.

4. Python into the mainstream

I had hoped to see how things had progressed from looking at the next CHEP proceedings, but the CHEP 1998 site has vanished and procedings don't appear to be available. A search for papers and slides corresponding to talks submitted to that CHEP turned up a few mentioning Python, all from Fermilab.

At the next CHEP (2000), there were two firsts:

1. the first talk mentioning Python in the title, "Dynamic Graphical User Interfaces using XML and JPython"[3] (author based at Fermilab), and

2. the first non-FNAL CHEP reference to Python I could find: "AIDA (LHC++), User Interface in Python"[4], from CERN.

Python seems to have really taken off by the following CHEP, as evidenced by the following excerpt from Philippe Canal's summary talk of the Data Analysis and Visualization track at CHEP 2001 (September) (see fig. 1.)

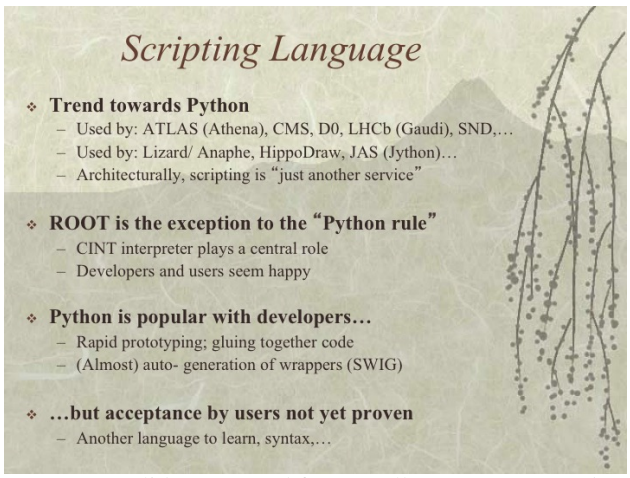

Figure 1. Slide excerpted from a talk at CHEP 2001[5].

By the next CHEP (March 2003), there was an Athena python talk, and LHCb's entire framework was heavily Python based, with DIRAC, GANGA, Gaudi, and Bender. Python was a mainstream HEP language.

5. Reflections

One question that was raised during discussions about holding this 1st PyHEP workshop is: "why did it take so long" for Python to be adopted in HEP? In my opinion, the answer has several components.

Python had all The Right Stuff Python had a lot going for it. The language is

simple, readable, has a large library of useful stuff, can be extended, and there was a clear "standard Python" (contrast Oberon). Benevolent dictator. Oberon had everything except a "standard Oberon" and look what happened.

The platform revolution In the early 1990s, "supported platforms" was the norm. IUCF had VAX/VMS with a handful of Ultrix systems; when I came to Nikhef in 1992, everything was SunOS. I found a CERN Computer Newsletter from September 1994, which mentioned central VM/CMS and VAX/VMS systems, and Sun, DEC, HP, Apollo, and SGI unix workstations. These systems were sold with an operating system and were generally supported, maintained, and operated by the computer systems group.

Python did not come "standard" on any of these systems! "Normal" physicists used the standard software that either came with these systems or was installed on them by the computer group.

I was not normal. As a graduate student in 1989, I was already dissatisfied with the user-oriented tooling[7] on the VAX/VMS systems we had, begged an extra disk allocation from the computer group, downloaded GNU Emacs and GNU `awk` "off the internet", installed them and used them extensively in my analysis. Downloading Python "off the internet" and using it for my research was no different. I am sure there are other "physics computer nerd" outliers like me who took the same route and were using Python before it hit the mainstream. They just weren't making noise about it like I was; in that sense I was a double outlier.

I was also involved in the Linux movement in the mid nineties. With Linux the whole dang thing was "downloaded off the internet", and most Linux distributions had Python installed as a standard package. As Linux became more mainstream, Python became more of a standard package.

To the best of my knowledge, Fermilab was the first big lab using Linux. Fermilab people were involved in the early discussions on the linux-hep mailing list, in late 1995. A bit more than a year later, was that first FNAL contribution at CHEP reporting use of Python. CERN was later to the game; in late 1997 Linux was mentioned as an unsupported platform on the ASIS repository; in late 1999 LXPLUS and LXBATCH (the Linux service at CERN) was announced, and at CHEP 2000 there was the first talk from CERN mentioning Python, 18 months later "the trend is towards Python".

It takes a bit of time So part of the uptake was, moving to a platform where Python was a standard part of the ecosystem; part of it was, once that transition was made, it takes a

[7] As a system itself, VAX/VMS was fantastic. You actually could read the error messages and understand what went wrong, and the system almost never crashed.

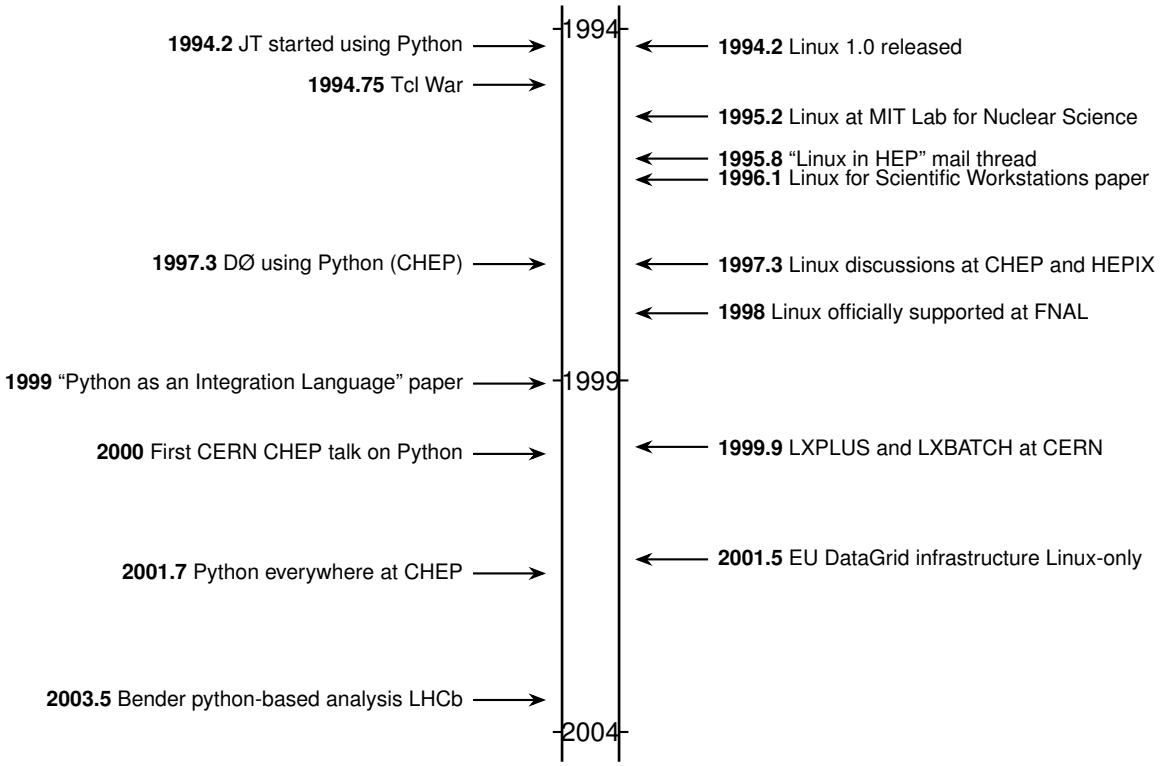

Figure 2. Python and Linux Timeline. The "Linux in HEP" mail thread included people working at Queen Mary University (London), Neils Bohr Institute (Copenhagen), FNAL, DESY, CERN, George Mason University, and JT at MIT.

year or so before a prototype is at the stage where it can be talked about at CHEP. It may even be the 18 month cycle of CHEP we're seeing; possibly the timescale of uptake is more like the months described in my personal experience, but then one has to wait on average 9 months before another CHEP rolls around.

Below is text saved to be used later; the above is a good first draft. I will try to finish the draft in time for the workshop.

benevolent dictator is a good model. look at Oberon for bad example.

Acknowledgments

So long and thanks for all the fish.

References

[1] S. Lammel. Computing models of major HEP experiments: D0 and CDF. In *Proceedings, 9th International Conference on Computing in High-Energy Physics (CHEP 1997): Berlin, Germany, April 7-11, 1997*, 1997.

[2] J. A. Templon. Python as an Integration Language. Technical Report SPAG-1998-02, The University of Georgia, Department of Physics and Astronomy, January 1998.

[3] G. Guglielmo. Dynamic Graphical User Interfaces using XML and JPython. In *Computing in High-Energy Physics (CHEP 2000): Padova, Italy, Febrary 7-11, 2000*, 2000.

[4] A. Pfeiffer. Libraries for HEP Computing (LHC++). In *Computing in High-Energy Physics (CHEP 2000): Padova, Italy, Febrary 7-11, 2000*, 2000.

[5] Philippe Canal and Lucas Taylor. CHEP 2001: Data Analysis & Visualization. Computing in High-Energy Physics (CHEP 2001): Beijing, P.R. China, September 3-7, 2001, 2001.