Python: faster and easier software development

Christian Seberino, Ph.D.
Pythonsoft, 8665 Lake Murray Boulevard Suite 3, San Diego, California 92119-2843

Abstract— **In general, Python software development is faster than Fortran, C, C++, Java and Perl software development. This is partly because of the following Python language features: compactness, high level data structures, automatic portability, type and run development, garbage collection, run time error checking, dynamic typing and extensibility. Fortran, C, C++, Java and Perl may each possess some of these features, but, none has all of them. Python excels in the areas of rapid prototyping, gluing, graphical user interfaces and networking among others. By combining Python with other languages, one can obtain fast development and fast execution.**

I. Introduction

GOOD scientists and engineers are always trying to get more work done with the same or less resources. These men and women are constantly trying to increase their *leverage*. The following is an example of leveraging time. You must write some software in two weeks. No problem! Just work twelve hours every day until the deadline and forgo showers as necessary. Funding is saved. The boss is happy. Everyone breaths a sigh of relief until the next crisis. Notice that upping total weekly work hours from 40 to 60 can only give at most a 50% improvement. Here is an example of leveraging *technology*. The same software is completed five to ten times faster simply by utilizing better tools. Which form of leverage sounds more attractive?

Python is a free and open source computer programming language available for anyone to use. In general, Python software development is faster than Fortran, C, C++, Java and Perl software development. In this paper I will explain why Python provides so much leverage. I will compare Python to other languages and discuss various uses.

II. Maximum Leverage

A. Leverage of compactness

I have often seen Fortran and C used over C++ and Java simply because of a lack of desire to learn object oriented programming. Why learn more complex languages when one can get work done with simpler languages? Fortran and C have the advantage of *compactness*; small feature sets provide a large amount of power. C++ and Perl are famously known for *not* being compact. Python is even more compact than Fortran and C [1] [2]. It was designed for both the programmer and the nonprogrammer. A simple and elegant yet powerful syntax allows anyone to quickly begin writing useful programs that can still be understood when revisited months later.

Please send correspondence to C. Seberino (Phone: 619-554-4071; Email: chris@pythonsoft.com; Internet: www.pythonsoft.com).

B. Leverage of high level data structures

The numerous problems representing strings in C as char (character) type arrays are well known. Have you ever overstepped array bounds? We need to be able to more easily concatenate strings. We need to be able to more easily parse strings and perform search and replace operations. Basically, we need a powerful "string" type.

Also, it is useful to have arrays with elements of multiple types that can more easily grow and shrink as necessary. How about easier methods to: add and delete elements from the middle, reverse the order, combine two arrays, and, the option to use noninteger indices such as element["H"] = "Hydrogen" and element["He"] = "Helium"?

Python provides data structures with all of these features [4]. Python also provides the option, if desired, of using powerful object oriented capabilities as well including polymorphism, operator overloading and multiple inheritance.

C. Leverage of automatic portability

Do you enjoy investing days to make your source code portable between Unix and Windows? Python programs are usually able to run without modification on all platforms that support Python. Python achieves this by compiling to a "bytecode" format similar to Java [5]. Alternatively, you can even compile Python programs to *Java* bytecode and run them in a Java Virtual Machine [6] [7]!

It is possible, if desired, to incorporate nonportable features into your program. For example, one may elect to utilize the Windows only COM libraries. However, portability is usually achievable, if desired, with little if any effort.

D. Leverage of type and run development

Even the smallest changes to Fortran, C and C++ code seem to require an endless loop of recompiling, relinking and rerunning until all is well. Would you save a lot of time if you never had to compile or link again? Python automatically takes care of these steps [4] thereby shortening development time. Programs even start executing while compilation is still proceeding. I refer to this feature as "type and run" development.

E. Leverage of garbage collection

Garbage collection implies automatic memory management [4] [5]. How much faster would software development be if memory leaks were automatically taken care of and pointers eliminated altogether as in Python?

	FORTRAN	C	C++	JAVA	PERL	PYTHON
COMPACTNESS	√	√				√
HIGH LEVEL DATA STRUCTURES			√	√	√	√
AUTOMATIC PORTABILITY				√	√	√
TYPE & RUN DEVELOPMENT					√	√
GARBAGE COLLECTION				√	√	√
RUN TIME ERROR CHECKING				√	√	√
DYNAMIC TYPING					√	√
EXTENSIBILITY		√	√	√	√	√

Fig. 1. Chart shows which features are found in which languages. Python is the only language that has all of these desirable features. See Section III for further discussion.

F. Leverage of run time error checking

Python tries to detect all errors during program execution as soon as possible. Python "raises exceptions" when errors occur. Current positions and paths taken through programs are shown. This makes debugging faster and easier. Without these checks, one is forced to deal with "crashes" with less information.

G. Leverage of dynamic typing

Python eliminates *all* variable declarations with a feature known as "dynamic typing" [8]. It is possible to define functions in a generic format that supports many different types. Programs are shorter [2] [3], more reusable and more easily maintained in addition to being developed faster. Python is nevertheless a strongly typed language.

H. Leverage of extensibility

Python has a large and very functional set of standard libraries. Python also has mechanisms to easily utilize new libraries written in Python and in other languages [9]. This ability to easily "glue" together these components significantly speeds up development.

III. Other Languages

Fortran, C, C++, Java and Perl may each possess some of these features, but, none has all of them. Fortran and C are compact but do not by default support high level data structures. Only Java and Perl also feature automatic portability, garbage collection and run time error checking, but, they are not compact. (There exists garbage collecting libraries for C and C++ but they are still problematic.) Perl is the only other type and run language. Perl is also the only other dynamically typed language. C, C++, Java and Perl are all somewhat extensible. However, accessing code in other languages will not be as easy in general as in Python. Fig. 1 shows which features are found in each language. Fortran, C, C++ and Java typically have faster program execution [4]. This is discussed in the next section.

IV. Common Uses

Python is usually the best choice for applications that are not processor intensive. Prototyping, or creating programs intended to be used only a few times, is not processor intensive [4] [9]. Graphical user interfaces and networking applications involve external interactions; these are therefore also not processor intensive [4] and perfectly suited for Python.

There are at least two ways to leverage Python for performance driven applications. One way is to first quickly construct a prototype in Python. Later, it can be easily converted to C or C++. Yahoo! Mail was designed using this strategy [10]. Major changes were anticipated in response to customer feedback. Yahoo! leveraged Python and later converted to C++ after the design was solidified.

A second way Python can be used where performance is paramount is to use it as a glue language to support or manage fast components written in other languages. For example, graphical user interfaces may be required, input files may need to be parsed, and, results might need to be converted to a different unit system.

For many examples of Python usage and Python software, visit the SciPy web page at www.scipy.org.

V. Conclusion

Easier and greater leverage in general is usually possible with Python. Compactness, high level data structures, automatic portability, type and run development, garbage collection, run time error checking, dynamic typing and extensibility are some reasons increases in productivity are common relative to Fortran, C, C++, Java and Perl. Python excels in the areas of rapid prototyping, gluing, graphical user interfaces and networking among others. By combining Python with other languages, one can obtain fast development and fast execution.

References

[1] E. S. Raymond. Why Python? *Linux Journal*, May 2000.
[2] J. K. Ousterhout. Scripting: higher level programming for the twenty-first century. *IEEE Computer*, March 1998.
[3] L. Prechelt. An empirical comparison of C, C++, Java, Perl, Python, Rexx and Tcl for a search/string-processing program. *Technical Report 2000-5 Universitat Karlsruhe*, March 10, 2000.
[4] M. Lutz and D. Ascher. *Learning Python*. O'Reilly, 1999.
[5] D. Flanagan. *Java in a nutshell*. O'Reilly, 1999.
[6] http://www.jython.org.
[7] S. Pedroni and N. Rappin. *Jython Essentials: rapid scripting in Java*. O'Reilly, 2002.
[8] http://www.artima.com/intv/strongweak.html. B. Venners and F. Sommers. Strong versus weak typing: A conversation with Guido van Rossum, Part V. February 10, 2003.
[9] M. Lutz. *Programming Python*. O'Reilly, 2001.
[10] http://www.artima.com/intv/speed.html. B. Venners. Programming at Python speed: A conversation with Guido van Rossum, Part III. January 27, 2003.

Christian Seberino has been programming for over twenty years. He has worked with numerous languages on personal computers and supercomputers. He earned a Ph.D. in theoretical physics from the University of California, San Diego in 2000. As part of his dissertation, he built a parallel supercomputer to run a parallel magnetic recording simulator he designed. This software utilized then recently developed algorithms and is still in use today. He has developed many other scientific and engineering applications including cryptographic libraries for the National Security Agency (NSA) and stochastic simulation software for the United States Navy. He founded the Pythonsoft company to promote the use of Python among the scientific and engineering communities. Pythonsoft serves these communities by providing software development and software instructional services. Email chris@pythonsoft.com or visit www.pythonsoft.com for more details.