Comparing Python to Other Languages

python.org/doc/essays/comparisons

> **Disclaimer:** This essay was written sometime in 1997. It shows its age. It is retained here merely as a historical artifact. --Guido van Rossum

Python is often compared to other interpreted languages such as Java, JavaScript, Perl, Tcl, or Smalltalk. Comparisons to C++, Common Lisp and Scheme can also be enlightening. In this section I will briefly compare Python to each of these languages. These comparisons concentrate on language issues only. In practice, the choice of a programming language is often dictated by other real-world constraints such as cost, availability, training, and prior investment, or even emotional attachment. Since these aspects are highly variable, it seems a waste of time to consider them much for this comparison.

Java

Python programs are generally expected to run slower than Java programs, but they also take much less time to develop. Python programs are typically 3-5 times shorter than equivalent Java programs. This difference can be attributed to Python's built-in high-level data types and its dynamic typing. For example, a Python programmer wastes no time declaring the types of arguments or variables, and Python's powerful polymorphic list and dictionary types, for which rich syntactic support is built straight into the language, find a use in almost every Python program. Because of the run-time typing, Python's run time must work harder than Java's. For example, when evaluating the expression a+b, it must first inspect the objects a and b to find out their type, which is not known at compile time. It then invokes the appropriate addition operation, which may be an overloaded user-defined method. Java, on the other hand, can perform an efficient integer or floating point addition, but requires variable declarations for a and b, and does not allow overloading of the + operator for instances of user-defined classes.

For these reasons, Python is much better suited as a "glue" language, while Java is better characterized as a low-level implementation language. In fact, the two together make an excellent combination. Components can be developed in Java and combined to form applications in Python; Python can also be used to prototype components until their design can be "hardened" in a Java implementation. To support this type of development, a Python implementation written in Java is under development, which allows calling Python code from Java and vice versa. In this implementation, Python source code is translated to Java bytecode (with help from a run-time library to support Python's dynamic semantics).

Javascript

Python's "object-based" subset is roughly equivalent to JavaScript. Like JavaScript (and unlike Java), Python supports a programming style that uses simple functions and variables without engaging in class definitions. However, for JavaScript, that's all there is. Python, on the other hand, supports writing much larger programs and better code reuse through a true object-oriented programming style, where classes and inheritance play an important role.

Perl

Python and Perl come from a similar background (Unix scripting, which both have long outgrown), and sport many similar features, but have a different philosophy. Perl emphasizes support for common application-oriented tasks, e.g. by having built-in regular expressions, file scanning and report generating features. Python emphasizes support for common programming methodologies such as data structure design and object-oriented programming, and encourages programmers to write readable (and thus maintainable) code by providing an elegant but not overly cryptic notation. As a consequence, Python comes close to Perl but rarely beats it in its original application domain; however Python has an applicability well beyond Perl's niche.

Tcl

Like Python, Tcl is usable as an application extension language, as well as a stand-alone programming language. However, Tcl, which traditionally stores all data as strings, is weak on data structures, and executes typical code much slower than Python. Tcl also lacks features needed for writing large programs, such as modular namespaces. Thus, while a "typical" large application using Tcl usually contains Tcl extensions written in C or C++ that are specific to that application, an equivalent Python application can often be written in "pure Python". Of course, pure Python development is much quicker than having to write and debug a C or C++ component. It has been said that Tcl's one redeeming quality is the Tk toolkit. Python has adopted an interface to Tk as its standard GUI component library.

Tcl 8.0 addresses the speed issuse by providing a bytecode compiler with limited data type support, and adds namespaces. However, it is still a much more cumbersome programming language.

Smalltalk

Perhaps the biggest difference between Python and Smalltalk is Python's more "mainstream" syntax, which gives it a leg up on programmer training. Like Smalltalk, Python has dynamic typing and binding, and everything in Python is an object. However, Python distinguishes built-in object types from user-defined classes, and currently doesn't allow inheritance from built-in types. Smalltalk's standard library of collection data types is more refined, while Python's library has more facilities for dealing with Internet and WWW realities such as email, HTML and FTP.

Python has a different philosophy regarding the development environment and distribution of code. Where Smalltalk traditionally has a monolithic "system image" which comprises both the environment and the user's program, Python stores both standard modules and user modules in individual files which can easily be rearranged or distributed outside the system. One consequence is that there is more than one option for attaching a Graphical User Interface (GUI) to a Python program, since the GUI is not built into the system.

C++

Almost everything said for Java also applies for C++, just more so: where Python code is typically 3-5 times shorter than equivalent Java code, it is often 5-10 times shorter than equivalent C++ code! Anecdotal evidence suggests that one Python programmer can finish in two months what two C++ programmers can't complete in a year. Python shines as a glue language, used to combine components written in C++.

Common Lisp and Scheme

These languages are close to Python in their dynamic semantics, but so different in their approach to syntax that a comparison becomes almost a religious argument: is Lisp's lack of syntax an advantage or a disadvantage? It should be noted that Python has introspective capabilities similar to those of Lisp, and Python programs can construct and execute program fragments on the fly. Usually, real-world properties are decisive: Common Lisp is big (in every sense), and the Scheme world is fragmented between many incompatible versions, where Python has a single, free, compact implementation.

The PSF