Python Tutorial

Release 3.9.1

Guido van Rossum
and the Python development team

January 12, 2021

Python Software Foundation
Email: docs@python.org

CONTENTS

CHAPTER

ONE

WHETTING YOUR APPETITE

If you do much work on computers, eventually you find that there's some task you'd like to automate. For example, you may wish to perform a search-and-replace over a large number of text files, or rename and rearrange a bunch of photo files in a complicated way. Perhaps you'd like to write a small custom database, or a specialized GUI application, or a simple game.

If you're a professional software developer, you may have to work with several C/C++/Java libraries but find the usual write/compile/test/re-compile cycle is too slow. Perhaps you're writing a test suite for such a library and find writing the testing code a tedious task. Or maybe you've written a program that could use an extension language, and you don't want to design and implement a whole new language for your application.

Python is just the language for you.

You could write a Unix shell script or Windows batch files for some of these tasks, but shell scripts are best at moving around files and changing text data, not well-suited for GUI applications or games. You could write a C/C++/Java program, but it can take a lot of development time to get even a first-draft program. Python is simpler to use, available on Windows, Mac OS X, and Unix operating systems, and will help you get the job done more quickly.

Python is simple to use, but it is a real programming language, offering much more structure and support for large programs than shell scripts or batch files can offer. On the other hand, Python also offers much more error checking than C, and, being a *very-high-level language*, it has high-level data types built in, such as flexible arrays and dictionaries. Because of its more general data types Python is applicable to a much larger problem domain than Awk or even Perl, yet many things are at least as easy in Python as in those languages.

Python allows you to split your program into modules that can be reused in other Python programs. It comes with a large collection of standard modules that you can use as the basis of your programs — or as examples to start learning to program in Python. Some of these modules provide things like file I/O, system calls, sockets, and even interfaces to graphical user interface toolkits like Tk.

Python is an interpreted language, which can save you considerable time during program development because no compilation and linking is necessary. The interpreter can be used interactively, which makes it easy to experiment with features of the language, to write throw-away programs, or to test functions during bottom-up program development. It is also a handy desk calculator.

Python enables programs to be written compactly and readably. Programs written in Python are typically much shorter than equivalent C, C++, or Java programs, for several reasons:

- the high-level data types allow you to express complex operations in a single statement;
- statement grouping is done by indentation instead of beginning and ending brackets;
- no variable or argument declarations are necessary.

Python is *extensible*: if you know how to program in C it is easy to add a new built-in function or module to the interpreter, either to perform critical operations at maximum speed, or to link Python programs to libraries that may only be available in binary form (such as a vendor-specific graphics library). Once you are really hooked, you can link the Python interpreter into an application written in C and use it as an extension or command language for that application.

By the way, the language is named after the BBC show "Monty Python's Flying Circus" and has nothing to do with reptiles. Making references to Monty Python skits in documentation is not only allowed, it is encouraged!