Contents

Articles

Python (programming language) 1

CPython 13

Python Software Foundation 15

Guido van Rossum 16

References

Article Sources and Contributors 19

Image Sources, Licenses and Contributors 20

Article Licenses

License 21

Python (programming language)

Paradigm	multi-paradigm: object-oriented, imperative, functional, reflective
Appeared in	1991
Designed by	Guido van Rossum
Developer	Python Software Foundation
Stable release	3.1.2/ March 21, 2010 2.7/ July 3, 2010
Typing discipline	duck, dynamic, strong
Major implementations	**CPython**, IronPython, Jython, Python for S60, PyPy, Unladen Swallow
Dialects	Stackless Python, RPython
Influenced by	ABC, ALGOL 68,[1] C, Haskell, Icon, Lisp, Modula-3, Perl, Java
Influenced	Boo, Cobra, D, Falcon, Groovy, Ruby, JavaScript
OS	Cross-platform
License	Python Software Foundation License
Usual file extensions	.py, .pyw, .pyc, .pyo, .pyd
Website	www.python.org [2]
Python Programming at Wikibooks	

Python is a general-purpose high-level programming language[3] whose design philosophy emphasizes code readability.[4] Python aims to combine "remarkable power with very clear syntax",[5] and its standard library is large and comprehensive. Its use of indentation for block delimiters is unusual among popular programming languages.

Python supports multiple programming paradigms, primarily but not limited to object oriented, imperative and, to a lesser extent, functional programming styles. It features a fully dynamic type system and automatic memory management, similar to that of Scheme, Ruby, Perl, and Tcl. Like other dynamic languages, Python is often used as a scripting language, but is also used in a wide range of non-scripting contexts.

The reference implementation of Python (CPython) is free and open source software and has a community-based development model, as do all or nearly all of its alternative implementations. CPython is managed by the non-profit Python Software Foundation.

History

Python was conceived in the late 1980s[6] and its implementation was started in December 1989[7] by Guido van Rossum at CWI in the Netherlands as a successor to the ABC programming language (itself inspired by SETL)[8] capable of exception handling and interfacing with the Amoeba operating system.[9] Van Rossum is Python's principal author, and his continuing central role in deciding the direction of Python is reflected in the title given to him by the Python community, *Benevolent Dictator for Life* (BDFL).

Python 2.0 was released on 16 October 2000, with many major new features including a full garbage collector and support for Unicode. However, the most important change was to the development process itself, with a shift to a more transparent and community-backed process.[10] Python 3.0, a major, backwards-incompatible release, was released on 3 December 2008[11] after a long period of testing. Many of its major features have been backported to the backwards-compatible Python 2.7.[12]

Programming philosophy

Python is a multi-paradigm programming language. Rather than forcing programmers to adopt a particular style of programming, it permits several styles: object-oriented programming and structured programming are fully supported, and there are a number of language features which support functional programming and aspect-oriented programming (including by metaprogramming[13] and by magic methods).[14] Many other paradigms are supported using extensions, such as pyDBC[15] and Contracts for Python[16] which allow Design by Contract.

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. An important feature of Python is dynamic name resolution (late binding), which binds method and variable names during program execution.

Rather than requiring all desired functionality to be built into the language's core, Python was designed to be highly extensible. New built-in modules can be easily written in C, C++ or Cython. Python can also be used as an extension language for existing modules and applications that need a programmable interface. This design of a small core language with a large standard library and an easily extensible interpreter was intended by Van Rossum from the very start because of his frustrations with ABC (which espoused the opposite mindset).[6]

The design of Python offers only limited support for functional programming in the Lisp tradition. However, Python's design philosophy exhibits significant similarities to those of minimalist Lisp-family languages, such as Scheme. The library has two modules (itertools and functools) that implement proven functional tools borrowed from Haskell and Standard ML.[17]

While offering choice in coding methodology, the Python philosophy rejects exuberant syntax, such as in Perl, in favor of a sparser, less-cluttered grammar. Python's developers expressly promote a particular "culture" or ideology based on what they want the language to be, favoring language forms they see as "beautiful", "explicit" and "simple". As Alex Martelli put it in his *Python Cookbook* (2nd ed., p. 230): "To describe something as clever is NOT considered a compliment in the Python culture." Python's philosophy rejects the Perl "there is more than one way to do it" approach to language design in favor of "there should be one—and preferably only one—obvious way to do it".[18]

Python's developers eschew premature optimization, and moreover, reject patches to non-critical parts of CPython which would offer a marginal increase in speed at the cost of clarity.[19] Python is sometimes described as "slow".[20] However, by the Pareto principle, most problems and sections of programs are not speed critical. When speed is a problem, Python programmers tend to try using a JIT compiler such as Psyco, rewriting the time-critical functions in "closer to the metal" languages such as C, or by translating (a dialect of) Python code to C code using tools like Cython.[21]

The core philosophy of the language is summarized by the document "PEP 20 (The Zen of Python)".[18]

Name and neologisms

An important goal of the Python developers is making Python fun to use. This is reflected in the origin of the name (based on the television series *Monty Python's Flying Circus*), in the common practice of using Monty Python references in example code, and in an occasionally playful approach to tutorials and reference materials.[22] [23] For example, the metasyntactic variables often used in Python literature are *spam* and *eggs*, instead of the traditional *foo* and *bar*.

A common neologism in the Python community is *pythonic*, which can have a wide range of meanings related to program style. To say that a piece of code is pythonic is to say that it uses Python idioms well, that it is natural or shows fluency in the language. Likewise, to say of an interface or language feature that it is pythonic is to say that it works well with Python idioms, that its use meshes well with the rest of the language.

In contrast, a mark of *unpythonic* code is that it attempts to write C++ (or Lisp, Perl, or Java) code in Python—that is, provides a rough transcription rather than an idiomatic translation of forms from another language. The concept of pythonicity is tightly bound to Python's minimalist philosophy of readability and avoiding the "there's more than one way to do it" approach. Unreadable code or incomprehensible idioms are unpythonic.

Users and admirers of Python—most especially those considered knowledgeable or experienced—are often referred to as *Pythonists*, *Pythonistas*, and *Pythoneers*.[24]

The prefix *Py* can be used to show that something is related to Python. Examples of the use of this prefix in names of Python applications or libraries include Pygame, a binding of SDL to Python (commonly used to create games); PyS60, an implementation for the Symbian Series 60 Operating System; PyQt and PyGTK, which bind Qt and GTK, respectively, to Python; and PyPy, a Python implementation written in Python. The prefix is also used outside of naming software packages: the major Python conference is named PyCon.

Usage

Python is often used as a scripting language for web applications, e.g. via mod_wsgi for the Apache web server. With Web Server Gateway Interface a standard API has been developed to facilitate these applications. Web application frameworks like Django, Pylons, TurboGears, web2py, Flask and Zope support developers in the design and maintenance of complex applications. Libraries like NumPy, SciPy and Matplotlib allow Python to be used effectively in scientific computing.

Python has been successfully embedded in a number of software products as a scripting language, including in finite element method software such as Abaqus, 3D animation packages such as Maya, MotionBuilder, Softimage, Cinema 4D, BodyPaint 3D, modo, and Blender, and 2D imaging programs like GIMP, Inkscape, Scribus, and Paint Shop Pro.[25] ESRI is now promoting Python as the best choice for writing scripts in ArcGIS.[26] It has even been used in several videogames,[27] [28] and has been adopted as one of the two available scripting languages in Google Docs.[29]

For many operating systems, Python is a standard component; it ships with most Linux distributions, with NetBSD, and OpenBSD, and with Mac OS X and can be used from the terminal. A number of Linux distributions use installers written in Python: Ubuntu uses the Ubiquity installer, while Red Hat Linux and Fedora use the Anaconda installer. Gentoo Linux uses Python in its package management system, Portage, and the standard tool to access it, emerge. Pardus uses it for administration and during system boot.[30]

Python has also seen extensive use in the information security industry, including exploit development.[31]

Among the users of Python are YouTube[32] and the original BitTorrent client.[33] Large organizations that make use of Python include Google,[34] Yahoo!,[35] CERN,[36] NASA,[37] and ITA.[38] Most of the Sugar software for the One Laptop per Child XO, now developed at Sugar Labs, is written in Python.[39]

Syntax and semantics

Python was intended to be a highly readable language. It is designed to have an uncluttered visual layout, frequently using English keywords where other languages use punctuation. Python requires less boilerplate than traditional manifestly typed structured languages such as C or Pascal, and has a smaller number of syntactic exceptions and special cases than either of these.[40]

```
def add5(x):
   return x+5

def dotwrite(ast):
   nodename = getNodename()
   label=symbol.sym_name.get(int(ast[0]),ast[0])
   print '    %s [label="%s' % (nodename, label),
   if isinstance(ast[1], str):
      if ast[1].strip():
         print '= %s"];' % ast[1]
      else:
         print '"]'
   else:
      print '"];'
      children = []
      for  in n, childenumerate(ast[1:]):
         children.append(dotwrite(child))
      print ,'    %s -> {' % nodename
      for  in :namechildren
         print '%s' % name,
```

Syntax-highlighted Python 2.x code.

Indentation

Python uses whitespace indentation, rather than curly braces or keywords, to delimit blocks (a feature also known as the off-side rule). An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block.[41]

Statements and control flow

Python's statements include (among others):

- The if statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if).
- The for statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.
- The while statement, which executes a block of code as long as its condition is true.
- The try statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits.
- The class statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.
- The def statement, which defines a function or method.
- The with statement (from Python 2.6), which encloses a code block within a context manager (for example, acquiring a lock before the block of code is run, and releasing the lock afterwards).
- The pass statement, which serves as a NOP and can be used in place of a code block.
- The assert statement, used during debugging to check for conditions that ought to apply.
- The yield statement, which returns a value from a generator function. (From Python 2.5, yield is also an operator. This form is used to implement coroutines -- see below.)

Each statement has its own semantics: for example, the def statement does not execute its block immediately, unlike most other statements.

CPython does not support first-class continuations, and according to Guido van Rossum it never will.[42] However, better support for coroutine-like functionality is provided in 2.5, by extending Python's generators.[43] Prior to 2.5, generators were lazy iterators; information was passed unidirectionally out of the generator. As of Python 2.5, it is possible to pass information back into a generator function.

Expressions

Python expressions are similar to languages such as C and Java. Some important notes:

- In Python 2, the */* operator on integers does integer division, i.e. it truncates the result to an integer. In Python 3, however, the result of */* is always a floating-point value, and a new operator *//* is introduced to do integer division. In Python 2, this behavior can be enabled using the statement *from __future__ import division*.
- In Python, == compares by value, in contrast to Java, where it compares by reference (value comparisons in Java use the *equals* method). To compare by reference in Python, use the *is* operator.
- Python uses named *and*, *or*, *not* operators rather than symbolic *&&*, ||, *!*.
- Python has an important type of expression known as a *list comprehension*. Recent versions of Python have extended list comprehensions into a more general expression known as a *generator expression*.
- Anonymous functions are implemented using *lambda expressions*; however, these are limited in that the body can only be a single expression.
- Conditional expressions in Python are written as *y if x else z* (different in order of operands from the *?:* operator common to many other languages).
- Python makes a distinction between lists and tuples. Lists are written as *[1, 2, 3]* are mutable, and cannot be used as the keys of dictionaries (dictionary keys must be immutable in Python). Tuples are written as *(1, 2, 3)*, are immutable and thus can be used as the keys of dictionaries. The parentheses around the tuple are optional in some contexts. Tuples can appear on the left side of an equal sign; hence an expression like *x, y = y, x* can be used to swap two variables.
- Python has a "string format" operator %. This functions analogous to printf expressions in C, e.g. *"foo=%s bar=%d" % ("blah", 2)* evaluates to "foo=blah bar=2".
- Python has various kinds of strings.
  - Either single or double quotes can be used to quote strings. Unlike in Unix shell languages, Perl or Perl-influenced languages such as Ruby or Groovy, single quotes and double quotes function identically, i.e. there is no string interpolation of *$foo* expressions.
  - There are also multi-line strings, which begin and end with a series of three single or double quotes and function like here documents in shell languages, Perl and Ruby.
  - Finally, all of the previously-mentioned string types come in "raw" varieties (denoted by placing a literal *r* before the opening quote), which do no backslash-interpolation and hence are very useful for regular expressions or Windows-style paths; compare "@-quoting" in C#.
- Python has slice expressions on lists, denoted as *...[left:right]* or *...[left:right:stride]*. For example, if the variable *nums* is assigned the list *[1, 3, 5, 7, 8, 13, 20]*, then the following expressions will evaluate True:
  - *nums[2:5] == [5, 7, 8]*, i.e. the slice goes up to, but not including, the right index.
  - *nums[1:] == [3, 5, 7, 8, 13, 20]*, i.e. all elements but the first, because an omitted right index means "the end".
  - *nums[:-3] == [1, 3, 5, 7]*, i.e. an omitted left index means "the start", and a negative index (either left or right) counts from the end.
  - *nums[:]* makes a copy of the list. This means *nums == nums[:]* is true but *nums is nums[:]* is false. Changes to the copy will not affect the original.
  - *nums[1:5:2] == [3, 7]*, i.e. if three numbers are given, the third is the "stride", indicating in this case that every second element will be selected.

In Python, a distinction between expressions and statements is rigidly enforced, in contrast to languages such as Common Lisp, Scheme or Ruby. This leads to some duplication of functionality, e.g.

- list comprehensions vs. "*for*" loops
- conditional expressions vs. "*if*" blocks
- The *eval()* vs. *exec()* builtins (in Python 2, *exec* is a statement declarator); *eval()* is for expressions, *exec()* is for statements.

Statements cannot be a part of an expression and so list and other comprehensions or lambda expressions, all being expressions, cannot contain statements. A particular case of this is that an assignment statement such as 'a =1' cannot form part of the conditional *expression* of a conditional statement, this has the advantage of avoiding a classic C error of mistaking an assignment token '=', for an equality operator '==' which would remain valid C in *if (c = 1) { ... }* but *if c = 1: ...* is *invalid* Python code.

Methods

Methods on objects are functions attached to the object's class; the syntax instance.method(argument) is, for normal methods and functions, syntactic sugar for Class.method(instance, argument). Python methods have an explicit self parameter to access instance data, in contrast to the implicit self in some other object-oriented programming languages (for example, Java, C++ or Ruby).[44]

Typing

Python uses duck typing and has typed objects but untyped variable names. Type constraints are not checked at compile time; rather, operations on an object may fail, signifying that the given object is not of a suitable type. Despite being dynamically typed, Python is strongly typed, forbidding operations that are not well-defined (for example, adding a number to a string) rather than silently attempting to make sense of them.

Python allows programmers to define their own types using classes, which are most often used for object-oriented programming. New instances of classes are constructed by calling the class (for example, SpamClass() or EggsClass()), and the classes themselves are instances of the metaclass type (itself an instance of itself), allowing metaprogramming and reflection.

Prior to version 3.0, Python had two kinds of classes: "old-style" and "new-style". Old-style classes were eliminated in Python 3.0, making all classes new-style. In versions between 2.2 and 3.0, both kinds of classes could be used. The syntax of both styles is the same, the difference being whether the class object is inherited from, directly or indirectly (all new-style classes inherit from object and are instances of type).

Here is a summary of Python's built-in types:

Type	Description	Syntax example
str	An immutable sequence of characters. In Python 2, strings are a sequence of characters. Unicode strings need to be declared by prefixing with the letter u. In Python 3 strings are Unicode by default.	'Wikipedia' "Wikipedia" """Spanning multiple lines"""
bytes	An immutable sequence of bytes	b'Some ASCII' b"Some ASCII"
list	Mutable, can contain mixed types	[4.0, 'string', True]
tuple	Immutable, can contain mixed types	(4.0, 'string', True)
set, frozenset	Unordered, contains no duplicates	{4.0, 'string', True} frozenset([4.0, 'string', True])
dict	A mutable group of key and value pairs	{'key1': 1.0, 3: False}
int	An immutable fixed precision number of unlimited magnitude	42
float	An immutable floating point number (system-defined precision)	3.1415927
complex	An immutable complex number with real number and imaginary parts	3+2.7j
bool	An immutable truth value	True False

Mathematics

Python defines the modulus operator so that the result of a % b is in the open interval [0,b), where b is a positive integer. (When b is negative, the result lies in the interval (b,0]). This is the usual way of defining the modulus operation in mathematics. However, this consequently affects how integer division is defined. To maintain the validity of the equation b * (a // b) + a % b = a, Integer division is defined to round towards minus infinity. Therefore 7 // 3 is 2, but (−7) // 3 is −3. This is different from many programming languages, where the result of integer division rounds towards zero and the modulus operator is consequently defined in a way which can return negative numbers.[45]

Python provides a round function for rounding floats to integers. Version 2.6.1 and lower use round-away-from-zero: round(0.5) is 1.0, round(-0.5) is -1.0. Version 3.0 and higher use round-to-even: round(1.5) is 2.0, round(2.5) is 2.0. The Decimal type/class in module decimal (since version 2.4) provides exact numerical representation and several rounding modes.

Python allows boolean expressions with multiple equality relations in a manner that is consistent with general usage in mathematics. For example, the expression a < b < c tests whether a is less than b and b is less than c. C-derived languages interpret this expression differently: in C, the expression would first evaluate a < b, resulting in 0 or 1, and that result would then be compared with c.[46]

Implementations

CPython

The mainstream Python implementation, known as *CPython*, is written in C meeting the C89 standard.[47] CPython compiles Python programs into intermediate bytecode,[48] which are then executed by the virtual machine.[49] It is distributed with a large standard library written in a mixture of C and Python. CPython ships in versions for many platforms, including Microsoft Windows and most modern Unix-like systems. CPython was intended from almost its very conception to be cross-platform; its use and development on esoteric platforms such as Amoeba, alongside more conventional ones like Unix and Mac OS, has greatly helped in this regard.[50]

Stackless Python is a significant fork of CPython that implements microthreads; it does not use the C memory stack. It can be expected to run on approximately the same platforms that CPython runs on.

Google started a project called Unladen Swallow in 2009 with the aims of increasing the speed of the Python interpreter by 5 times and improving its multithreading ability to scale to thousands of cores.[51]

Alternative implementations

Jython compiles the Python program into Java byte code, which can then be executed by every Java Virtual Machine implementation. This also enables the use of Java class library functions from the Python program. IronPython follows a similar approach in order to run Python programs on the .NET Common Language Runtime. PyPy is an experimental self-hosting implementation of Python, written in Python, that can output several types of bytecode, object code and intermediate languages. There also exist compilers to high-level object languages, with either unrestricted Python, a restricted subset of Python, or a language similar to Python as the source language. PyPy is of this type, compiling RPython to several languages; other examples include Pyjamas compiling to JavaScript; Shed Skin compiling to C++; and Cython and Pyrex compiling to C.

In 2005 Nokia released a Python interpreter for the Series 60 mobile phones called PyS60. It includes many of the modules from the CPython implementations and some additional modules for integration with the Symbian operating system. This project has been kept up to date to run on all variants of the S60 platform and there are several third party modules available. The Nokia N900 also supports Python with gtk windows libraries, with the feature that programs can be both written and run on the device itself. There is also a Python interpreter for Windows

CE devices (including Pocket PC). It is called PythonCE. There are additional tools available for easy application and GUI development.

The PyMite virtual machine began in 2000 and made its first public appearance at PyCon 2003.[52] PyMite was folded into Python-on-a-Chip [53] in 2009.[54] Python-on-a-Chip (p14p) is a project to develop a reduced Python virtual machine (codenamed PyMite) that runs a significant subset of the Python language on microcontrollers without an OS in as little as 4KB of RAM.[55]

Around 2004, the Pyastra [56] project created a specialized translator and assembler that targets very resource-constrained microcontrollers.

ChinesePython (中蟒) is a Python programming language using Chinese language lexicon. Besides reserved words and variable names, most data type operations can be coded in Chinese as well.

Interpretational semantics

Most Python implementations (including CPython) can function as a command line interpreter, for which the user enters statements sequentially and receives the results immediately. In short, Python acts as a shell. While the semantics of the other modes of execution (bytecode compilation, or compilation to native code) preserve the sequential semantics, they offer a speed boost at the cost of interactivity, so they are usually only used outside of a command-line interaction (e.g., when importing a module).

Other shells add capabilities beyond those in the basic interpreter, including IDLE and IPython. While generally following the visual style of the Python shell, they implement features like auto-completion, retention of session state, and syntax highlighting.

Some implementations can compile not only to bytecode, but can turn Python code into machine code. So far, this has only been done for restricted subsets of Python. PyPy takes this approach, naming its restricted compilable version of Python *RPython*.

Psyco is a specialising just in time compiler that integrates with CPython and transforms bytecode to machine code at runtime. The produced code is specialised for certain data types and is faster than standard Python code. Psyco is compatible with all Python code, not only a subset.[57]

Development

Python development is conducted largely through the Python Enhancement Proposal (or "PEP") process. PEPs are standardized design documents providing general information related to Python, including proposals, descriptions, design rationales, and explanations for language features.[58] Outstanding PEPs are reviewed and commented upon by Van Rossum, the Python project's Benevolent Dictator for Life (leader / language architect).[59] CPython's developers also communicate over a mailing list, python-dev, which is the primary forum for discussion about the language's development; specific issues are discussed in the Roundup bug tracker maintained at python.org.[60] Development takes place at the self-hosted `svn.python.org`.

CPython's public releases come in three types, distinguished by which part of the version number is incremented:

- backwards-incompatible versions, where code is expected to break and must be manually ported. The first part of the version number is incremented. These releases happen infrequently—for example, version 3.0 was released 8 years after 2.0.
- major or 'feature' releases, which are largely compatible but introduce new features. The second part of the version number is incremented. These releases are scheduled to occur roughly every 18 months, and each major version is supported by bugfixes for several years after its release.[61]
- bugfix releases, which introduce no new features but fix bugs. The third and final part of the version number is incremented. These releases are made whenever a sufficient number of bugs have been fixed upstream since the last release, or roughly every 3 months. Security vulnerabilities are also patched in bugfix releases.[62]

A number of alpha, beta, and release-candidates are also released as previews and for testing before the final release is made. Although there is a rough schedule for each release, this is often pushed back if the code is not ready. The development team monitor the state of the code by running the large unit test suite during development, and using the BuildBot continuous integration system.[63]

Standard library

Python has a large standard library, commonly cited as one of Python's greatest strengths,[64] providing pre-written tools suited to many tasks. This is deliberate and has been described as a "batteries included"[65] Python philosophy. The modules of the standard library can be augmented with custom modules written in either C or Python. Boost C++ Libraries includes a library, Boost.Python, to enable interoperability between C++ and Python. Because of the wide variety of tools provided by the standard library, combined with the ability to use a lower-level language such as C and C++, which is already capable of interfacing between other libraries, Python can be a powerful glue language between languages and tools.

The standard library is particularly well tailored to writing Internet-facing applications, with a large number of standard formats and protocols (such as MIME and HTTP) already supported. Modules for creating graphical user interfaces, connecting to relational databases, arithmetic with arbitrary precision decimals, manipulating regular expressions, and doing unit testing are also included.[66]

Some parts of the standard library are covered by specifications (for example, the WSGI implementation wsgiref follows PEP 333 [67]), but the majority of the modules are not. They are specified by their code, internal documentation, and test suite (if supplied). However, because most of the standard library is cross-platform Python code, there are only a few modules that must be altered or completely rewritten by alternative implementations.

The standard library is not essential to run python or embed python within an application. Blender 2.49 for instance omits most of the standard library.

Influences on other languages

Python's design and philosophy have influenced several programming languages, including:

- Pyrex and its derivative Cython are code translators that are targeted at writing fast C extensions for the CPython interpreter. The language is mostly Python with syntax extensions for C and C++ features. Both languages produce compilable C code as output.
- Boo uses indentation, a similar syntax, and a similar object model. However, Boo uses static typing and is closely integrated with the .NET framework.[68]
- Cobra uses indentation and a similar syntax. Cobra's "Acknowledgements" document lists Python first among languages that influenced it.[69] However, Cobra directly supports design-by-contract, unit tests and optional static typing.[70]
- ECMAScript borrowed iterators, generators, and list comprehensions from Python.[71]
- Go is described as incorporating the "development speed of working in a dynamic language like Python".[72]
- Groovy was motivated by the desire to bring the Python design philosophy to Java.[73]
- OCaml has an optional syntax, called twt (The Whitespace Thing), inspired by Python and Haskell.[74]

Python's development practices have also been emulated by other languages. The practice of requiring a document describing the rationale for, and issues surrounding, a change to the language (in Python's case, a PEP) is also used in Tcl[75] and Erlang[76] because of Python's influence.

See also

- List of programming languages
- Comparison of computer shells
- Comparison of programming languages
- List of Python software
- List of integrated development environments for Python
- Scripting language

References

[1] "Interview with Guido van Rossum" (http://www.amk.ca/python/writing/gvr-interview). July 1998. . Retrieved 29 2007.

[2] http://www.python.org/

[3] "What is Python Good For?" (http://docs.python.org/faq/general.html#what-is-python-good-for). *General Python FAQ*. Python Foundation. . Retrieved 2008-09-05.

[4] "What is Python? Executive Summary" (http://www.python.org/doc/essays/blurb/). *Python documentation*. Python Foundation. . Retrieved 2007-03-21.

[5] "General Python FAQ" (http://www.python.org/doc/faq/general/#what-is-python). *python.org*. Python Software Foundation. . Retrieved 2009-06-27.

[6] "The Making of Python" (http://www.artima.com/intv/pythonP.html). Artima Developer. . Retrieved 2007-03-22.

[7] "A Brief Timeline of Python" (http://python-history.blogspot.com/2009/01/brief-timeline-of-python.html). Guido van Rossum. . Retrieved 2009-01-20.

[8] van Rossum, Guido. "[Python-Dev] SETL (was: Lukewarm about range literals)" (http://mail.python.org/pipermail/python-dev/2000-August/008881.html). . Retrieved 2009-06-27.

[9] "Why was Python created in the first place?" (http://www.python.org/doc/faq/general/#why-was-python-created-in-the-first-place). Python FAQ. . Retrieved 2007-03-22.

[10] A.M. Kuchling and Moshe Zadka. "What's New in Python 2.0" (http://www.amk.ca/python/2.0/). . Retrieved 2007-03-22.

[11] "Python 3.0 Release" (http://python.org/download/releases/3.0/). Python Software Foundation. . Retrieved 2009-07-08.

[12] van Rossum, Guido (5 April 2006). "PEP 3000 -- Python 3000" (http://www.python.org/dev/peps/pep-3000/). Python Software Foundation. . Retrieved 2009-06-27.

[13] The Cain Gang Ltd.. "Python Metaclasses: Who? Why? When?" (http://www.python.org/community/pycon/dc2004/papers/24/metaclasses-pycon.pdf) (PDF). . Retrieved 2009-06-27.

[14] "3.3. Special method names" (http://docs.python.org/3.0/reference/datamodel.html#special-method-names). *The Python Language Reference*. Python Software Foundation. . Retrieved 2009-06-27.

[15] Contracts for Python (http://www.nongnu.org/pydbc/), PyDBC

[16] Contracts for Python (http://www.wayforward.net/pycontract/), pycontract

[17] "6.5 itertools - Functions creating iterators for efficient looping" (http://docs.python.org/lib/module-itertools.html). Docs.python.org. . Retrieved 2008-11-24.

[18] "PEP 20 - The Zen of Python" (http://www.python.org/dev/peps/pep-0020/). Python Software Foundation. 2004-08-23. . Retrieved 2008-11-24.

[19] Python Culture (http://www.python.org/dev/culture/)

[20] Python is... slow? (http://peter.mapledesign.co.uk/weblog/archives/python-is-slow) December 21st, 2004 — Peter Bowyer's weblog]

[21] Python Patterns - An Optimization Anecdote (http://www.python.org/doc/essays/list2str.html)

[22] Python Tutorial (http://docs.python.org/tut/node3.html)

[23] Python Challenge tutorial (http://www.pythonchallenge.com/)

[24] David Goodger. "Code Like a Pythonista: Idiomatic Python" (http://python.net/~goodger/projects/pycon/2007/idiomatic/handout.html). .; "How to think like a Pythonista" (http://python.net/crew/mwh/hacks/objectthink.html). .

[25] Documentation of the PSP Scripting API can be found at *JASC Paint Shop Pro 9: Additional Download Resources* (http://www.jasc.com/support/customercare/articles/psp9components.asp)

[26] "About getting started with writing geoprocessing scripts" (http://webhelp.esri.com/arcgisdesktop/9.2/index.cfm?TopicName=About_getting_started_with_writing_geoprocessing_scripts). November 2006. . Retrieved April 2007.

[27] porkbelly (2007-07-23). "Stackless Python 2.5" (http://www.webcitation.org/5ru5w3vSR). *Eve Insider Dev Blog*. CCP Games. Archived from the original (http://myeve.eve-online.com/devblog.asp?a=blog&bid=488) on 2010-08-10. . "As you may well know, your favorite space-game owes its existence to the programming language Python"

[28] Caudill, Barry (2005-09-20). "Modding Sid Meier's Civilization IV" (http://www.webcitation.org/5ru5VItfv). *Sid Meier's Civilization IV Developer Blog*. Firaxis Games. Archived from the original (http://www.2kgames.com/civ4/blog_03.htm) on 2010-08-10. . "we created three levels of tools … The next level offers Python and XML support, letting modders with more experience manipulate the game world and everything in it."

[29] "Python Language Guide (v1.0)" (http://www.webcitation.org/5ru5FHxfV). *Google Documents List Data API v1.0*. Google. Archived from the original (http://code.google.com/apis/documents/docs/1.0/developers_guide_python.html) on 2010-08-10. .
[30] ":: Pardus :: TÜBİTAK/UEKAE ::" (http://www.pardus.org.tr/eng/projects/comar/PythonInPardus.html). Pardus.org.tr. . Retrieved 2008-11-24.
[31] Products and discussion of this use of Python include "IMMUNITY : Knowing You're Secure" (http://www.immunitysec.com/products-immdbg.shtml). Immunitysec.com. . Retrieved 2008-11-24.; CORE Security Technologies' open source software repository (http://oss.coresecurity.com/); "Wapiti - Web application security auditor" (http://wapiti.sourceforge.net/). Wapiti.sourceforge.net. . Retrieved 2008-11-24.; "TAOF - theartoffuzzing.com - Home" (http://www.theartoffuzzing.com/joomla/). Theartoffuzzing.com. . Retrieved 2008-11-24.; "[Dailydave] RE: Network Exploitation Tools aka Exploitation Engines" (http://fist.immunitysec.com/pipermail/dailydave/2004-September/000851.html). Fist.immunitysec.com. . Retrieved 2008-11-24.
[32] "Coder Who Says Py: YouTube runs on Python!" (http://sayspy.blogspot.com/2006/12/youtube-runs-on-python.html). Sayspy.blogspot.com. December 12, 2006. . Retrieved 2008-11-24.
[33] Review of original BitTorrent software (http://www.onlamp.com/pub/a/python/2003/7/17/pythonnews.html) at O'Reilly Python Dev Center
[34] "Quotes about Python" (http://python.org/about/quotes/). Python.org. . Retrieved 2008-11-24.
[35] "Organizations Using Python" (http://wiki.python.org/moin/OrganizationsUsingPython). Python.org. . Retrieved 2009-01-15.
[36] CERN Document Server: Record#974627: Python : the holy grail of programming (http://cdsweb.cern.ch/record/974627?ln=no)
[37] "Python Success Stories" (http://www.python.org/about/success/usa/). Python.org. . Retrieved 2008-11-24.
[38] Python Slithers into Systems by Darryl K. Taft (http://www.eweek.com/c/a/Application-Development/Python-Slithers-into-Systems/)
[39] "What is Sugar? - Sugar Labs" (http://sugarlabs.org/go/Sugar). sugarlabs.org. 2008-05-10. . Retrieved 0r-2-11.
[40] "Is Python a good language for beginning programmers?" (http://www.python.org/doc/faq/general/#is-python-a-good-language-for-beginning-programmers). *General Python FAQ*. Python Foundation. March 7, 2005. . Retrieved 2007-03-21.
[41] Myths about indentation in Python (http://www.secnetix.de/~olli/Python/block_indentation.hawk)
[42] van Rossum, Guido (February 9, 2006). "Language Design Is Not Just Solving Puzzles" (http://www.artima.com/weblogs/viewpost.jsp?thread=147358). *Artima forums*. Artima. . Retrieved 2007-03-21.
[43] van Rossum, Guido; Phillip J. Eby (April 21, 2006). "Coroutines via Enhanced Generators" (http://www.python.org/peps/pep-0342.html). *Python Enhancement Proposals*. Python Foundation. . Retrieved 2007-03-21.
[44] "Why must 'self' be used explicitly in method definitions and calls?" (http://www.python.org/doc/faq/general/#why-must-self-be-used-explicitly-in-method-definitions-and-calls). *Python FAQ*. Python Foundation. .
[45] "Why Python's Integer Division Floors" (http://python-history.blogspot.com/2010/08/why-pythons-integer-division-floors.html). . Retrieved 2010-08-25.
[46] Python Essential Reference, David M Beazley
[47] "PEP 7 - Style Guide for C Code" (http://www.python.org/dev/peps/pep-0007/). Python.org. . Retrieved 2008-11-24.
[48] CPython byte code (http://docs.python.org/lib/bytecodes.html)
[49] Python 2.5 internals (http://www.troeger.eu/teaching/pythonvm08.pdf)
[50] "O'Reilly - An Interview with Guido van Rossum" (http://www.oreilly.com/pub/a/oreilly/frank/rossum_1099.html). Oreilly.com. . Retrieved 2008-11-24.
[51] ProjectPlan (http://code.google.com/p/unladen-swallow/wiki/ProjectPlan), Plans for optimizing Python - unladen-swallow
[52] PyMite: Python-on-a-chip (http://wiki.python.org/moin/PyMite)
[53] http://pythononachip.org/
[54] PyMite (http://deanandara.com/PyMite/2010-State.html)
[55] PyMite: Python-on-a-Chip (http://pythononachip.org/)
[56] http://pyastra.sourceforge.net/
[57] Introduction to Psyco (http://psyco.sourceforge.net/introduction.html)
[58] PEP 1 -- PEP Purpose and Guidelines (http://www.python.org/dev/peps/pep-0001/)
[59] "Parade of the PEPs" (http://www.python.org/doc/essays/pepparade.html). Python.org. . Retrieved 2008-11-24.
[60] Cannon, Brett. "Guido, Some Guys, and a Mailing List: How Python is Developed" (http://python.org/dev/intro/). *python.org*. Python Software Foundation. . Retrieved 2009-06-27.
[61] Norwitz, Neal (8 April 2002]]). "[Python-Dev] Release Schedules (was Stability & change)" (http://mail.python.org/pipermail/python-dev/2002-April/022739.html). . Retrieved 2009-06-27.
[62] Baxter, Anthony; Aahz (2001-03-15). "PEP 6 -- Bug Fix Releases" (http://python.org/dev/peps/pep-0006/). Python Software Foundation. . Retrieved 2009-06-27.
[63] Python Buildbot (http://python.org/dev/buildbot/), Python
[64] Przemyslaw Piotrowski, Build a Rapid Web Development Environment for Python Server Pages and Oracle (http://www.oracle.com/technology/pub/articles/piotrowski-pythoncore.html), Oracle Technology Network, July 2006. Retrieved October 21, 2008.
[65] "About Python" (http://www.python.org/about/). *python.org*. Python Software Foundation. . Retrieved 2009-06-27.
[66] "PEP 327 - Decimal Data Type" (http://www.python.org/peps/pep-0327.html). Python.org. . Retrieved 2008-11-24.
[67] http://www.python.org/dev/peps/pep-0333/
[68] "BOO - Gotchas for Python Users" (http://boo.codehaus.org/Gotchas+for+Python+Users). Boo.codehaus.org. . Retrieved 2008-11-24.

[69] "Cobra - Acknowledgements" (http://cobra-language.com/docs/acknowledgements/). cobra-language.com. . Retrieved 2010-04-07.
[70] "Cobra - Comparison to Python" (http://cobra-language.com/docs/python/). cobra-language.com. . Retrieved 2010-04-07.
[71] "proposals:iterators_and_generators [ES4 Wiki]" (http://wiki.ecmascript.org/doku.php?id=proposals:iterators_and_generators). Wiki.ecmascript.org. . Retrieved 2008-11-24.
[72] Kincaid, Jason (2009-11-10). "Google's Go: A New Programming Language That's Python Meets C++" (http://www.techcrunch.com/2009/11/10/google-go-language/). TechCrunch. . Retrieved 2010-01-29.
[73] James Strachan (2003-08-29). "Groovy - the birth of a new dynamic language for the Java platform" (http://radio.weblogs.com/0112098/2003/08/29.html). .
[74] Mike Lin. ""The Whitespace Thing" for OCaml" (http://people.csail.mit.edu/mikelin/ocaml+twt/). . Retrieved 2009-04-12.
[75] "TIP #3: TIP Format" (http://www.tcl.tk/cgi-bin/tct/tip/3.html). Tcl.tk. . Retrieved 2008-11-24.
[76] EEP - Erlang Enhancement Proposal (http://www.erlang.org/eeps/eep-0001.html)

Further reading

- Payne, James (2010). *Beginning Python: Using Python 2.6 and Python3.1* (http://jamesrobertpayne.com) (1st ed.). Wrox. ISBN 978-0470414637.
- Beazley, David M. (2009). *Python Essential Reference* (4th ed.). Addison-Wesley Professional. ISBN 978-0672329784.
- Summerfield, Mark (2009). *Programming in Python 3* (http://www.qtrac.eu/py3book.html) (2nd ed.). Addison-Wesley Professional. ISBN 978-0321680563.
- Lutz, Mark (2009). *Learning Python* (4th ed.). O'Reilly Media. ISBN 978-0596158064.
- Hamilton, Naomi (5 August 2008). "The A-Z of Programming Languages: Python" (http://www.computerworld.com.au/index.php/id;66665771). *Computerworld*. Retrieved 2010-03-31. - An interview with Guido Van Rossum on Python
- Martelli, Alex; Ravenscroft, Anna; Ascher, David (2005). *Python Cookbook* (2nd ed.). O'Reilly Media. ISBN 978-0596007973.
- Pilgrim, Mark (2004). *Dive Into Python* (http://diveintopython.org/toc/index.html). Apress. ISBN 978-1590593561.
- Downey, Allen B.. *Think Python: How to Think Like a Computer Scientist* (http://www.greenteapress.com/thinkpython/html/index.html).
- Sweigart, Al (2010). *Invent Your Own Computer Games with Python* (http://inventwithpython.com) (2nd ed.). CreateSpace. ISBN 978-0982106013.

External links

- Official Python website (http://www.python.org/)
- Python (http://www.dmoz.org/Computers/Programming/Languages/Python/) at the Open Directory Project

CPython

Developer(s)	Python Software Foundation
Stable release	3.1.2 / March 21, 2010
Written in	C
Platform	Cross-platform
Type	Python Programming Language Interpreter
License	Python Software Foundation License
Website	Python.org [2]

CPython is the default, most-widely used implementation of the Python programming language. It is written in C. In addition to CPython, there are two other production-quality Python implementations: Jython, written in Java, and IronPython, which is written for the Common Language Runtime, as well as several experimental implementations.[nutshell]

CPython is a bytecode interpreter. It has a foreign function interface with several languages including C, in which one must explicitly write bindings in a language other than Python.

Supported platforms

Unix-like

- AIX operating system
- BSD
- Darwin
- FreeBSD
- HP-UX
- IRIX 5 and later
- Plan 9 from Bell Labs
- Mac OS X
- NetBSD
- Linux
- OpenBSD
- Solaris
- Tru64
- Other Unixes

Desktop OSes

- AROS
- AtheOS
- BeOS
- Windows
- Windows NT
- OS/2
- RISC OS

Special and embedded

- GP2X
- iPodLinux
- Nintendo DS
- Nintendo Gamecube
- Symbian OS Series60
- Nokia 770 Internet Tablet
- Nokia N800
- Nokia N810
- Palm OS
- PlayStation 2
- PlayStation 3 (Linux)
- Psion
- QNX
- Sharp Zaurus
- Xbox/XBMC
- VxWorks
- Openmoko

Mainframe and other

- OS/390
- VMS
- z/OS

Previously supported platforms

PEP 11 [1] lists platforms which are not supported in CPython by Python Software Foundation. These platforms can still be supported by external ports. See below.

- DOS (unsupported since 2.0)
- IRIX 4 (unsupported since 2.3)
- Mac OS 9 (unsupported since 2.4)
- MINIX (unsupported since 2.3)

External ports

These are ports not integrated to Python Software Foundation's official version of CPython, with links to its main development site. Ports often include additional modules for platform-specific functionalities, like graphics and sound API for PSP and SMS and camera API for S60.

- Amiga: AmigaPython [2]
- AS/400: iSeriesPython [3]
- DOS using DJGPP: PythonD [4]
- PlayStation Portable: Stackless Python for PSP [5]
- Symbian OS: Python for S60 [6]
- Windows CE/Pocket PC: Python Windows CE port [7]

Concurrency issues

A drawback to using CPython on a multiprocessor computer is the presence of a Global Interpreter Lock on each CPython interpreter process, which effectively disables concurrent Python threads within one process.[8] To be truly concurrent in multiprocessor environment, separate CPython interpreter processes have to be run, which makes establishing communication between them a difficult task. There is constant discussion whether to remove the GIL from CPython.[9]

Notes

1. Martelli, Alex (2006). *Python in a Nutshell* (2nd edition ed.). O'Reilly. pp. 5–7. ISBN 0-596-10046-9.

References

[1] http://www.python.org/dev/peps/pep-0011/
[2] http://www.monkeyhouse.eclipse.co.uk/amiga/python/
[3] http://www.iseriespython.com/
[4] http://www.caddit.net/pythond
[5] http://code.google.com/p/pspstacklesspython/
[6] http://opensource.nokia.com/projects/pythonfors60/
[7] http://pythonce.sourceforge.net/
[8] Python/C API Reference Manual: Thread State and the Global Interpreter Lock (http://docs.python.org/api/threads.html)
[9] Python Library and Extension FAQ: Can't we get rid of the Global Interpreter Lock? (http://docs.python.org/faq/library#can-t-we-get-rid-of-the-global-interpreter-lock)

Python Software Foundation

Python Software Foundation	
	
Abbreviation	PSF
Formation	March 6, 2001
Type	Non-profit organization
Purpose/focus	Promote, protect, and advance the Python programming language, and to support and facilitate the growth of the international community of Python programmers.
Headquarters	United States
Region served	Worldwide
President	Guido van Rossum
Key people	Steve Holden, Brett Cannon
Website	Python Software Foundation [1]

The **Python Software Foundation** (**PSF**), is a non-profit organization devoted to the Python programming language. It was launched March 6, 2001. The mission of the foundation is to foster development of the Python community. The PSF is responsible for various processes within the Python community, including developing the core Python distribution, managing intellectual rights, and raising funds.

Python Software Foundation received prestigious 2005 Computerworld Horizon Award for "cutting-edge" technology.

External links

- Python Software Foundation [1]

References

[1] http://www.python.org/psf/

Guido van Rossum

Guido van Rossum	
	
Born	31 January 1956 Netherlands
Nationality	Dutch
Alma mater	University of Amsterdam
Occupation	Computer programmer, author
Employer	Google
Known for	Python programming language
Spouse	Kim Knapp
Children	Orlijn Michiel Knapp-van Rossum[1]
Awards	Award for the Advancement of Free Software (2001)
Website	
www.python.org/~guido/ [2]	

Guido van Rossum (born 31 January[3] 1956) is a Dutch computer programmer who is best known as the author of the Python programming language. In the Python community, Van Rossum is known as a "Benevolent Dictator for Life" (BDFL), meaning that he continues to oversee the Python development process, making decisions where necessary.[4] He is currently employed by Google, where he spends half his time working on Python development.

Biography

Van Rossum was born and grew up in the Netherlands, where he received a masters degree in mathematics and computer science from the University of Amsterdam in 1982. He later worked for various research institutes, including the Dutch Centrum Wiskunde & Informatica (CWI), Amsterdam, the US National Institute of Standards and Technology (NIST), Gaithersburg, Maryland, and the Corporation for National Research Initiatives (CNRI), Reston, Virginia.

In December 2005, Van Rossum was hired by Google.[5] He wrote a web-based code-review tool for Google in Python.[6] [7]

Van Rossum received the 2001 Award for the Advancement of Free Software from the Free Software Foundation (FSF) at the 2002 FOSDEM conference in Brussels, Belgium. Guido received a NLUUG Award in May 2003. In 2006 he was recognized as a Distinguished Engineer by the Association for Computing Machinery.

Personal life

Guido van Rossum is the brother of Just van Rossum, a type designer and also a programmer. Just van Rossum designed the font that is used in the "Python Powered" logo. Currently Guido lives in California together with his American wife Kim Knapp[8] and their son Orlijn.[9] [10] [11]

Work

While working at the *Stichting Mathematisch Centrum* (CWI), Guido van Rossum wrote and contributed a glob() routine to BSD Unix in 1986.[12] [13] Van Rossum also worked on the development of the ABC programming language.

Python

About the origin of Python, Van Rossum wrote in 1996:

> Over six years ago, in December 1989, I was looking for a "hobby" programming project that would keep me occupied during the week around Christmas. My office … would be closed, but I had a home computer, and not much else on my hands. I decided to write an interpreter for the new scripting language I had been thinking about lately: a descendant of ABC that would appeal to Unix/C hackers. I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of Monty Python's Flying Circus).[14]

In 2000 he further wrote:

> Python's predecessor, ABC, was inspired by SETL – Lambert Meertens spent a year with the SETL group at NYU before coming up with the final ABC design![15]

Van Rossum at the *Google I/O* Developer's Conference.

Computer Programming for Everybody

In 1999, Van Rossum submitted a funding proposal to DARPA called *Computer Programming for Everybody*, in which he further defined his goals for Python:

- an easy and intuitive language just as powerful as major competitors
- open source, so anyone can contribute to its development
- code that is as understandable as plain English
- suitability for everyday tasks, allowing for short development times

Arguably, several of these ambitions have since been realized. Python has grown to become a popular programming language, particularly in the Internet environment.

References

[1] Guido van Rossum (http://wiki.codecall.net/Guido_van_Rossum) - CodeCall Programming Wiki

[2] http://www.python.org/~guido/

[3] (Python-Dev) Happy Birthday, Guido! (http://mail.python.org/pipermail/python-dev/2007-January/070849.html), Guido van Rossum, January 31 17:00:29 CET 2007, Python-Dev mailing list

[4] "Benevolent dictator for life" (http://www.linuxformat.co.uk/modules.php?op=modload&name=Sections&file=index&req=viewarticle&artid=10). Linux Format. 2005-02-01. . Retrieved 2007-11-01.

[5] Python Creator Guido van Rossum now working at Google (http://www.oreillynet.com/pub/wlg/8821), December 21, 2005, by Jeremy Jones - O'Reilly ONLamp Blog

[6] Mondrian Google Mondrian: web-based code review and storage (http://www.niallkennedy.com/blog/archives/2006/11/google-mondrian.html), November 30, 2006, Niall Kennedy's Weblog

[7] Code Reviews at Google (http://thebogles.com/blog/2010/06/code-reviews-at-google/), 8 June 2010, Bogle's Blog

[8] (Python-Dev) Guido and Kim married (http://mail.python.org/pipermail/python-dev/2000-June/004497.html), Ken Manheimer, 6 June 2000, Python-Dev -- Python core developers

[9] Guido van Rossum - Brief Bio (http://www.python.org/~guido/bio.html)

[10] (Mailman-Announce) forwarded message from Guido van Rossum (http://mail.python.org/pipermail/mailman-announce/2000-May/000010.html), "Oh, and to top it all off, I'm going on vacation. I'm getting married and will be relaxing on my honeymoon."

[11] What's New in Python? (http://csg.csail.mit.edu/6.893/Handouts/PythonWhatsNew.pdf), "Not your usual list of new features", Stanford CSL Colloquium, October 29, 2003; BayPiggies, November 13, 2003, Guido van Rossum, Elemental Security

[12] 'Globbing' library routine (http://web.archive.org/web/20071219090708/http://www.isc.org/sources/devel/func/glob.txt)

[13] File::Glob - Perl extension for BSD glob routine - search.cpan.org (http://search.cpan.org/~rgarcia/perl-5.10.0/ext/File/Glob/Glob.pm)

[14] Foreword for "Programming Python" (1st ed.) (http://www.python.org/doc/essays/foreword/)

[15] [Python-Dev] SETL (was: Lukewarm about range literals) (http://mail.python.org/pipermail/python-dev/2000-August/008881.html)

External links

- Guido van Rossum's homepage (http://www.python.org/~guido/)
- *Neopythonic* (http://neopythonic.blogspot.com/) (New Weblog)
- *All Things Pythonic* (http://www.artima.com/weblogs/index.jsp?blogger=guido) (Old Weblog)
- *The History of Python* (http://python-history.blogspot.com/) Guido's blog on the History of Python and design decisions
- *Computer Programming for Everybody* (http://www.python.org/doc/essays/cp4e.html)
- Interview with Guido van Rossum (http://www.twit.tv/floss11) on FLOSS Weekly
- Computerworld Interview with Guido van Rossum on Python (http://www.computerworld.com.au/index.php/id;66665771)
- Google App Engine - Run your web applications on Google's infrastructure (http://www.stanford.edu/class/ee380/Abstracts/081105.html) - technical talk on Google App Engine given by Guido van Rossum at Stanford University. (online video archive (http://stanford-online.stanford.edu/courses/ee380/081105-ee380-300.asx))

Article Sources and Contributors

Python (programming language) *Source*: http://en.wikipedia.org/w/index.php?oldid=388845008 *Contributors*: -Barry-, 0x6adb015, 130.94.122.xxx, 207.172.11.xxx, 24.49.40.xxx, 2disbetter, 4th-otaku, A D Monroe III, A plague of rainbows, A. Parrot, A.Ou, ABF, ALbertMietus, AThing, Abednigo, AbstractBeliefs, AdamGomaa, Adamnelson, Aeonimitz, Aflafla1, Agnistus, AhmadH, Ahoerstemeier, Ahy1, Aidan W, Aim Here, Aitias, Aj00200, Akhristov, Akuchling, Akulo, Alan Liefting, Alex Heinz, Alex Libman, AlexLibman, Alexandre.tp, Alsh, AmRadioHed, Amakuha, Andre Engels, Andreas Eisele, Andrejj, Andrew Haji, Andrewjlockley, Andy Dingley, Antaeus Feldspar, Anthon.Eff, Anthoniraj, Antonielly, Anupamsr, Apparition11, Aranel, Archer3, Arichnad, Arite, Arkanosis, Arknascar44, Arny, Aronzak, Arved, Asqueella, Asuffield, Atheuz, Atomique, Auntof6, Avnit, AxelBoldt, Azimuts43, B7T, BartlebyScrivener, Bawolff, Beetle B., Bellhalla, Benjaminhill, Benwing, Bernd vdB, Betterusername, Bevo, Bitmuse, BlindWanderer, Bloodshedder, Blue bear sd, Bmk, Bobo192, Boffob, Bombastrus, Bondates, Boredzo, Borislav, Brainsik, Brainus, Brettlpb, Brion VIBBER, Bryant.cutler, Bsmntbombdood, Bugnot, C. A. Russell, C1932, CFeyecare, CRGreathouse, Cadr, Caim, Cameltrader, Captain panda, Carlj7, Catapult, Cburnett, Cdhitch, Cek, CharlesC, Chconline, Chealer, Cherlin, Chipp, Chmod007, Chowbok, Chri1753, Chris Chittleborough, Chris Roy, Chris the speller, Chrismith, ChuckEsterbrook, Ciphergoth, Cjbprime, Codeczero, Connelly, Conversion script, Coolboy1234, Coolperson3, CorbinSimpson, Cornellier, Creativetechnologist, Crysb, Ctachme, CubicStar, Cvrebert, Cybercobra, Cyp2il, Cícero, DHowell, Da monster under your bed, Dan Ros, Danakil, Daniel Hen, Daniel.Cardenas, Daniel.ugra, Daverose 33, David Woodward, Davide125, Dazmax, Dcoetzee, DeadEyeArrow, Dejavu12, Den fjättrade ankan, Dhart, Digamma, Disavian, Discospinster, Djmackenzie, Dmeranda, Docu, Dogcow, Dolcecars, Dolda2000, Domtyler, Dotancohen, DouglasGreen, Dougofborg, Dp462090, Draicone, Dreftymac, Drosboro, Drummle, Dvarrazzo, Dwheeler, Dysprosia, Eagleal, Eastwind, EdC, Edaelon, Eddie Parker, Edgeprod, EditAnon, Eduardofv, Edward, Egmontaz, Ehheh, Eivanec, Eiwot, El Cubano, Elliskev, Eloquence, Elykyllek, Enchanter, Enochlau, Epl18, Eric B. and Rakim, Erik Zachte, Erikcw, Error, Espen, Etz Haim, Eurleif, Euyyn, Faisal.akeel, Falcorian, Fancypantsss, FatalError, Feedmecereal, FellGleaming, Fergofrog, Ferri, Fetofs, Fibonacci, Finell, FitzySpyder, Flammifer, Flarn2006, Flash200, Flatline, Flauto Dolce, Fleminra, Formulax, Frap, Fred Bradstadt, Fredrik, Freqsh0, Freshraisin, Friday13, Friedo, Fubar Obfusco, Furrykef, Fuzzyman, Fvw, Gamma, Garkbit, Garrett Albright, Garzo, Gauss, GeeksHaveFeelings, Generic Player, Gerrit, Gesslein, Giftlite, Gillwill, Gioto, GirasoleDE, Glenn, Gohst, Gokulmadhavan, GoodSirJava, Goodgerster, Goodone121, Gortsack, Graham87, Graveenib, GreatWhiteNortherner, GreyTeardrop, Gronky, Guaka, Guigolum, Gutworth, H3h, H3llbringer, Habj, Hairy Dude, HalfShadow, Halo, Hannes Röst, Hansamurai, Hao2lian, Happenstance, Harmil, Hawaiian717, Hdante, Hebejebelus, HebrewHammerTime, HeikoEvermann, HelgeStenstrom, Herbee, Hervegirod, Hfastedge, Hom sepanta, Honeyman, Howcheng, HumbertoDiogenes, Hydrargyrum, Hypo, I80and, IW.HG, Iani, Ianozsvald, Idono, Ignacioerrico, Ilya, ImperatorPlinius, Imz, Inquam, Intgr, Iph, Isilanes, Itai, J E Bailey, J.delanoy, JLaTondre, JWB, JYOuyang, Jacob.jose, Jed S, Jeltz, JeremyA, Jerome Charles Potts, Jerryobject, Jerub, Jhylton, Jin, Jkp1187, Jlf23, Jlin, JoaquinFerrero, Joe Chill, Jogloran, JohnElder, JohnWhitlock, Johnuniq, Jonik, Jorge Stolfi, Josh Parris, Josh the Nerd, JoshDuffMan, Joytex, Jsginther, Julesd, KDesk, Karih, Karl Dickman, Karol Langner, Katanzag, Kaustuv, Kazuo Moriwaka, Kbh3rd, Kbk, Kbrose, Kenliu, Kesla, Kim Bruning, KimDabelsteinPetersen, King of Hearts, Kkinder, Kl4m, Kl4m-AWB, Kneiphof, Koavf, KodiakBjörn, Korpios, Kozuch, Kragen, Kris Schnee, Ksn, Kubanczyk, KumpelBert, Kuralyov, Kızılsungur, L Kensington, LFaraone, Lacker, Lambiam, LauriO, Laurusnobilis, Leafman, LeeHunter, Legoktm, Leondz, Levin, LiDaobing, Lightst, Lino Mastrodomenico, LittleDan, Llamadog903, Loggie, LoveEncounterFlow, Lulu of the Lotus-Eaters, LunaticFringe, Lysander89, MER-C, MIT Trekkie, MMuzammils, Mac, Mach5, Macrocoders, Madmardigan53, Maduskis, Magister Mathematicae, Malleus Fatuorum, Marco42, Mark Krueger, Mark Russell, Marko75, Marqueed, Martin.komunide.com, MartinSpacek, Martinkunev, Martinultima, Massysett, Masud1011, Mathnerd314, Matt Crypto, MattGiuca, Matthew Woodcraft, MatthewRayfield, Matusu, McSly, Mcherm, Mdd, Meand, Meandtheshell, Meduz, Memty, MesserWoland, Michal Nebyla, Midnight Madness, Mignon, Mikco, Mike Lin, Miles, Minghong, Mipadi, MisterSheik, Mjb, Mjpieters, MkClark, Modster, MosheZadka, Mpradeep, Mr Minchin, MrOllie, MrStalker, Mrsatori, Mtrinque, MuffinFlavored, Murmurr, Mykhal, NanoTy, Nanowolf, Nanshu, NapoliRoma, NathanBeach, Natrius, Ncmathsadist, Nealmcb, Nearfar, Neilc, Netoholic, Neuralwiki, Nevar stark, NevilleDNZ, Nguyen Thanh Quang, Nick Garvey, Nikai, Ninly, Nir Soffer, Nknight, Nkour, Noamraph, Noldoaran, Nono64, Northgrove, NotAbel, Nummify, Nzk, Obradovic Goran, Oda Mari, Oefe, Ohnoitsjamie, Olathe, OlavN, Oli Filth, Oliverlewis, Oneirist, Ossiemanners, Ozzmosis, P0lyglut, PHaze, Paddy3118, Palfrey, Papna, Patrickjamesmiller, Pcb21, Peak, Pelago, Pengo, Penguinzig, Pepr, Perfecto, PeterReid, Peterhi, Peterl, Peteturtle, Pharos, Philip Trueman, PhilipR, Phoe6, Physicistjedi, Phædrus, Piet Delport, Pillefj, Pmlineditor, Pointillist, Poor Yorick, Poromenos, Positron, Prolog, ProveIt, Prty, Pyritie, Python.tw, Pythonbook, Quadra23, Quartz25, QuicksilverJohny, QuicksilverPhoenix, Qwfp, RP459, Radagast83, Radenski, Radon210, Raghuraman.b, RazielZero, Rbonvall, Rdhettinger, Recent Runes, RedWolf, Rettetast, Rich Farmbrough, RichNick, Richard001, Rjanag, Rjwilmsi, Rmt2m, Robinw77, Rodrigostrauss, Rogper, Roland2, Ronyclau, Rsocol, Rspeer, Rufous, Rummey, Rursus, Ruud Koot, S.Örvarr.S, SF007, Sabre23t, Sachavdk, Salmar, Sam Korn, Sam Pointon, Samohyl Jan, Samuel, Samuel Grant, SandManMattSH, Sander Säde, Sanxiyn, SardonicRick, Sarefo, Sbandrews, SciberDoc, Scorchsaber, Sealican, Sebb, Seidenstud, Sen Mon, Senordefarge, SergeyLitvinov, Serprex, Sgeo, Shadowjams, Shaolinhoward, Shervinafshar, Shimei, Shmorhay, Sidecharm10101, Sietse Snel, Silivrenion, Silverfish, Simeon, Simetrical, Simxp, Sir Isaac, SkyWalker, Slitchfield, Smjg, Snori, Softarch Jon, Sooperman23, SophisticatedPrimate, Spaceyguy, Spebi, SpencerWilson, Squilibob, Sridharinfinity, Ssd, Stangaa, Starnestommy, Stephen Gilbert, Steveha, Strombrg, StuartBrady, Studerby, Style, Sunnan, Sverdrup, Swaroopch, TJRC, TXiKi, TakuyaMurata, Taniquetil, Tarquin, Technobadger, Teehee123, Telemakh0s, Template namespace initialisation script, Tennessee, Thatos, The Anome, The Kid, The Wild Falcon, The best jere, TheDolphin, TheParanoidOne, Thebrid, Thejapanesegeek, Thingg, Thumperward, Tide rolls, TimTay, Timmorgan, Timothy Clemans, Timwi, Tlesher, Tobias Bergemann, Tokigun, Tompsci, Tony Sidaway, Tony1, TonyClarke, Toussaint, Towel401, Trampolineboy, Traxs7, Troeger, Trogdor31, Tualha, TuukkaH, Ucucha, Unixguy, Unyoyega, UserVOBO, Uyuyuy99, VX, Verdlanco, Visor, Vl'hurg, Vorratt, Vsion, Wangi, WatchAndObserve, Waterfox, Wavelength, Wdscxsj, Weel, Wellington, WhatisFeelings?, Whowhat16, Wikiborg, Wimerrill, Wiretse, Wlievens, WoLpH, Wrathchild, Wrp103, Ww, Wws, X96lee15, XDanielx, Xasxas256, Xaxafrad, Xdcdx, Yamaplos, Yath, Yellowstone6, Ygramul, Yoghurt, Yoric, Ytyoun, Yworo, ZeroOne, Zoicon5, Zondor, Zukeeper, Zundark, Пика Пика, ¡مج, 1164 anonymous edits

CPython *Source*: http://en.wikipedia.org/w/index.php?oldid=384061255 *Contributors*: AMAMH, Abednigo, Cybercobra, Duk, DzinX, FatalError, Fiftyquid, Ike-bana, Jonik, K3rb, Kiore, Kocio, Lankier, Leafcat, LittleDan, Lulu of the Lotus-Eaters, MMuzammils, OsamaK, RedWolf, Sanxiyn, Sergey.volk, Ske, Sn0wflake, TMN, Timo Honkasalo, Vedranf, 28 anonymous edits

Python Software Foundation *Source*: http://en.wikipedia.org/w/index.php?oldid=332951927 *Contributors*: Brandon, Chmod007, Fredrik, Ghettoblaster, Greenrd, Kate, LeeHunter, Longouyang, Love Krittaya, M656, Melaen, Pamplelune, Richard001, Sanxiyn, Xezbeth, 1 anonymous edits

Guido van Rossum *Source*: http://en.wikipedia.org/w/index.php?oldid=390399583 *Contributors*: 21655, Afasmit, AhmadSherif, Alex.tan, AlistairMcMillan, Alro, Amakuha, Anirvan, Anoopan, Ayla, Beowulph, Bobblewik, CapitalSasha, CharlotteWebb, Chealer, Cwolfsheep, David Gerard, DePiep, Delpino, Discospinster, Ds13, Dungodung, Ehn, Ejanev, Eurleif, GRuban, Ghettoblaster, GirasoleDE, GregorB, Gronky, HumbertoDiogenes, ImperatorPlinius, Isilanes, Ixtli, Jacoplane, Jamelan, Janm67, Jimpick, Joachim Strombergson, JohnSmith777, José Gnudista, K.lee, Kickboy, Kl4m-AWB, Kocio, Kvangend, Lambiam, Mahjongg, Markvs, Martarius, MattGiuca, Mdd, Minesweeper, Moxfyre, Ms2ger, Muriel Gottrop, Nikai, Noisy, Ocee, Olando, Peterl, Rajah, Raysonho, Reinis, Rentzepopoulos, Rholton, Rich Farmbrough, Richard001, Rjwilmsi, SF007, Sanxiyn, Sealican, Sin-man, Skagedal, SmileyChris, Solitude, SpencerWilson, Stickelberger, TallNapoleon, Taniquetil, TerriersFan, The cattr, Timwi, Tlesher, TonyClarke, Viajero, Warfieldian, Wfeidt, Xinconnu, Youssefsan, 66 anonymous edits

Image Sources, Licenses and Contributors

Image:Python logo.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Python_logo.svg *License*: GNU General Public License *Contributors*: www.python.org

File:Wikibooks-logo-en.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Wikibooks-logo-en.svg *License*: logo *Contributors*: User:Bastique, User:Ramac

Image:Python add5 syntax.svg *Source*: http://en.wikipedia.org/w/index.php?title=File:Python_add5_syntax.svg *License*: unknown *Contributors*: Lulu of the Lotus-Eaters, Nerzhal, Red Rooster, Xander89, 1 anonymous edits

File:Guido_van_Rossum_OSCON_2006.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Guido_van_Rossum_OSCON_2006.jpg *License*: Creative Commons Attribution-Sharealike 2.0 *Contributors*: Doc Searls

Image:Guido van Rossum at Google IO 2008.jpg *Source*: http://en.wikipedia.org/w/index.php?title=File:Guido_van_Rossum_at_Google_IO_2008.jpg *License*: Creative Commons Attribution-Sharealike 2.0 *Contributors*: Alessio Bragadini

License