**Python**

**Python** is a general-purpose, high-level programming language whose design philosophy emphasizes code readability. Python claims to "[combine] remarkable power with very clear syntax", and its standard library is large and comprehensive. Its [use of indentation for block delimiters](http://en.wikipedia.org/wiki/Off-side_rule) is unique among popular programming languages.

Python supports multiple [programming paradigms](http://en.wikipedia.org/wiki/Programming_paradigm), primarily but not limited to  [object - oriented](http://en.wikipedia.org/wiki/Object-oriented_programming),  [imperative](http://en.wikipedia.org/wiki/Imperative_programming)  and, to a lesser extent, [functional programming](http://en.wikipedia.org/wiki/Functional_programming)  styles. It features a fully [dynamic type](http://en.wikipedia.org/wiki/Dynamic_type) system and automatic  [memory management](http://en.wikipedia.org/wiki/Memory_management), similar to that of [Scheme](http://en.wikipedia.org/wiki/Scheme_(programming_language)), [Ruby](http://en.wikipedia.org/wiki/Ruby_(programming_language)), [Perl](http://en.wikipedia.org/wiki/Perl), and [Tcl](http://en.wikipedia.org/wiki/Tcl). Like other [dynamic languages](http://en.wikipedia.org/wiki/Dynamic_language), Python is often used as a [scripting language](http://en.wikipedia.org/wiki/Scripting_language), but is also used in a wide range of non-scripting contexts. Using third-party tools, Python code can be packaged into standalone executable programs. Python interpreters are available for many operating systems.

The [reference implementation](http://en.wikipedia.org/wiki/Reference_implementation_(computing)) of Python ([CPython](http://en.wikipedia.org/wiki/CPython" \o "CPython)) is [free and open source software](http://en.wikipedia.org/wiki/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](http://en.wikipedia.org/wiki/Python_Software_Foundation" \o "Python Software Foundation).

**[](http://en.wikipedia.org/wiki/File:Guido_van_Rossum.jpg)History**

Guido van Rossum, the creator of Python

Python was conceived in the late 1980s and its implementation was started in December 1989 by [Guido van Rossum](http://en.wikipedia.org/wiki/Guido_van_Rossum) at [CWI](http://en.wikipedia.org/wiki/Centrum_Wiskunde_%26_Informatica) in the[Netherlands](http://en.wikipedia.org/wiki/Netherlands) as a successor to the [ABC programming language](http://en.wikipedia.org/wiki/ABC_(programming_language)) (itself inspired by [SETL](http://en.wikipedia.org/wiki/SETL)) capable of [exception handling](http://en.wikipedia.org/wiki/Exception_handling) and interfacing with the [Amoeba operating system](http://en.wikipedia.org/wiki/Amoeba_distributed_operating_system). 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](http://en.wikipedia.org/wiki/Benevolent_Dictator_For_Life" \o "Benevolent Dictator For Life)*[(BDFL)](http://en.wikipedia.org/wiki/Benevolent_Dictator_For_Life" \o "Benevolent Dictator For Life).

Python 2.0 was released on 16 October 2000, with many major new features including a full [garbage collector](http://en.wikipedia.org/wiki/Garbage_collection_(computer_science)) and support for [Unicode](http://en.wikipedia.org/wiki/Unicode). However, the most important change was to the development process itself, with a shift to a more transparent and community-backed process.[[11]](http://en.wikipedia.org/wiki/Python_(programming_language)#cite_note-newin-2.0-10) Python 3.0 (also known as Python 3000 or py3k), a major, backwards-incompatible release, was released on 3 December 2008  after a long period of testing. Many of its major features have been  [backported](http://en.wikipedia.org/wiki/Backporting" \o "Backporting)  to the backwards-compatible Python 2.6 and 2.7. Python has twice been awarded as [TIOBE](http://en.wikipedia.org/wiki/Tiobe_index) Programming Language of the Year (2007, 2010), which is given to the language with the greatest growth in popularity over the course of the year (as measured by the TIOBE index).

**Programming philosophy**

Python is a [multi-paradigm programming language](http://en.wikipedia.org/wiki/Multi-paradigm_programming_language). Rather than forcing programmers to adopt a particular style of programming, it permits several styles:  [object-oriented programming](http://en.wikipedia.org/wiki/Object-oriented_programming) and [structured programming](http://en.wikipedia.org/wiki/Structured_programming) are fully supported, and there are a number of language features which support  [functional programming](http://en.wikipedia.org/wiki/Functional_programming)  and [aspect-oriented programming](http://en.wikipedia.org/wiki/Aspect-oriented_programming) (including by  [metaprogramming](http://en.wikipedia.org/wiki/Metaprogramming)  and by  [magic methods](http://en.wikipedia.org/wiki/Metaobject)). Many other paradigms are supported using extensions, such as pyDBCand Contracts for Python which allow [Design by Contract](http://en.wikipedia.org/wiki/Design_by_Contract).

Python uses [dynamic typing](http://en.wikipedia.org/wiki/Dynamic_typing) and a combination of [reference counting](http://en.wikipedia.org/wiki/Reference_counting) and a cycle-detecting [garbage collector](http://en.wikipedia.org/wiki/Garbage_collection_(computer_science)) for [memory management](http://en.wikipedia.org/wiki/Memory_management). An important feature of Python is dynamic [name resolution](http://en.wikipedia.org/wiki/Name_resolution) ([late binding](http://en.wikipedia.org/wiki/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](http://en.wikipedia.org/wiki/C_(programming_language)), [C++](http://en.wikipedia.org/wiki/C%2B%2B) or [Cython](http://en.wikipedia.org/wiki/Cython" \o "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](http://en.wikipedia.org/wiki/ABC_(programming_language)) (which espoused the opposite mindset).

The design of Python offers only limited support for [functional programming](http://en.wikipedia.org/wiki/Functional_programming)  in the  [Lisp](http://en.wikipedia.org/wiki/Lisp_programming_language" \o "Lisp programming language)  tradition. However, Python's design philosophy exhibits significant similarities to those of minimalistic Lisp-family languages, such as  [Scheme](http://en.wikipedia.org/wiki/Scheme_(programming_language)" \o "Scheme (programming language)). The language has map(), reduce() and filter() functions, and the [list comprehensions](http://en.wikipedia.org/wiki/List_comprehension)   added in Python 2.0 have since been extended with comprehensions for dictionaries and sets, as well as generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from [Haskell](http://en.wikipedia.org/wiki/Haskell_(programming_language)) and [Standard ML](http://en.wikipedia.org/wiki/Standard_ML).

While offering choice in coding methodology, the Python philosophy rejects exuberant syntax, such as in  [Perl](http://en.wikipedia.org/wiki/Perl" \o "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](http://en.wikipedia.org/wiki/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](http://en.wikipedia.org/wiki/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".

Python's developers eschew  [premature optimization](http://en.wikipedia.org/wiki/Optimization_(computer_science)" \l "When_to_optimize" \o "Optimization (computer science)), and moreover, reject patches to non-critical parts of CPython which would offer a marginal increase in speed at the cost of clarity.  Python is sometimes described as "slow". However, by the [Pareto principle](http://en.wikipedia.org/wiki/Pareto_principle), most problems and sections of programs are not speed critical. When speed is important, Python programmers tend to try using a JIT compiler such as [Psyco](http://en.wikipedia.org/wiki/Psyco" \o "Psyco) or using an alternative language implementation such as [PyPy](http://en.wikipedia.org/wiki/PyPy" \o "PyPy). When pure Python code isn't fast enough, time-critical functions can be rewritten in "closer to the metal" languages such as C, or by translating (a dialect of) Python code to C code using tools like  [Cython](http://en.wikipedia.org/wiki/Cython).

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

**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*](http://en.wikipedia.org/wiki/Monty_Python%27s_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. For example, the[metasyntactic variables](http://en.wikipedia.org/wiki/Metasyntactic_variable#Python) often used in Python literature are [*spam* and *eggs*](http://en.wikipedia.org/wiki/Spam_(Monty_Python)), instead of the traditional [*foo* and *bar*](http://en.wikipedia.org/wiki/Foobar).

A common [neologism](http://en.wikipedia.org/wiki/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*.

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](http://en.wikipedia.org/wiki/Pygame" \o "Pygame), a  [binding](http://en.wikipedia.org/wiki/Language_binding)  of [SDL](http://en.wikipedia.org/wiki/Simple_DirectMedia_Layer" \o "Simple DirectMedia Layer)to Python (commonly used to create games);  [PyS60](http://en.wikipedia.org/wiki/PyS60), an implementation for the [Symbian Series 60](http://en.wikipedia.org/wiki/Symbian_Series_60" \o "Symbian Series 60) [Operating System](http://en.wikipedia.org/wiki/Symbian_OS);  [PyQt](http://en.wikipedia.org/wiki/PyQt" \o "PyQt)  and  [PyGTK](http://en.wikipedia.org/wiki/PyGTK), which bind [Qt](http://en.wikipedia.org/wiki/Qt_(toolkit)) and [GTK](http://en.wikipedia.org/wiki/GTK), respectively, to Python; and  [PyPy](http://en.wikipedia.org/wiki/PyPy), a Python implementation written in Python. The prefix is also used outside of naming software packages: the major Python [conference](http://en.wikipedia.org/wiki/Academic_conference) is named  [PyCon](http://en.wikipedia.org/w/index.php?title=PyCon&action=edit&redlink=1" \o "PyCon (page does not exist)).

**Usage**

Python is often used as a [scripting language](http://en.wikipedia.org/wiki/Scripting_language) for [web applications](http://en.wikipedia.org/wiki/Web_application), e.g. via  [mod\_wsgi](http://en.wikipedia.org/wiki/Mod_wsgi" \o "Mod wsgi) for the [Apache web server](http://en.wikipedia.org/wiki/Apache_web_server). With [Web Server Gateway Interface](http://en.wikipedia.org/wiki/Web_Server_Gateway_Interface), a standard API has been developed to facilitate these applications. [Web application frameworks](http://en.wikipedia.org/wiki/Web_application_framework)   like  [Django](http://en.wikipedia.org/wiki/Django_(web_framework)" \o "Django (web framework)),  [Pylons](http://en.wikipedia.org/wiki/Pylons_(web_framework)),  [TurboGears](http://en.wikipedia.org/wiki/TurboGears),  [web2py](http://en.wikipedia.org/wiki/Web2py),  [Flask](http://en.wikipedia.org/wiki/Flask_(programming))  and  [Zope](http://en.wikipedia.org/wiki/Zope)  support developers in the design and maintenance of complex applications. Libraries like [NumPy](http://en.wikipedia.org/wiki/NumPy" \o "NumPy), [SciPy](http://en.wikipedia.org/wiki/SciPy" \o "SciPy) and [Matplotlib](http://en.wikipedia.org/wiki/Matplotlib" \o "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](http://en.wikipedia.org/wiki/Finite_element_method)  software such as  [Abaqus](http://en.wikipedia.org/wiki/Abaqus), 3D animation packages such as  [Houdini](http://en.wikipedia.org/wiki/Houdini_(software)),  [Maya](http://en.wikipedia.org/wiki/Maya_(software)),  [MotionBuilder](http://en.wikipedia.org/wiki/MotionBuilder),  [Softimage](http://en.wikipedia.org/wiki/Softimage_XSI),  [Cinema 4D](http://en.wikipedia.org/wiki/Cinema_4D),  [BodyPaint 3D](http://en.wikipedia.org/w/index.php?title=BodyPaint_3D&action=edit&redlink=1),  [modo](http://en.wikipedia.org/wiki/Modo_(software))  and  [Blender](http://en.wikipedia.org/wiki/Blender_(software))  and 2D imaging programs like  [GIMP](http://en.wikipedia.org/wiki/GIMP), [Inkscape](http://en.wikipedia.org/wiki/Inkscape" \o "Inkscape), [Scribus](http://en.wikipedia.org/wiki/Scribus" \o "Scribus) and [Paint Shop Pro](http://en.wikipedia.org/wiki/Paint_Shop_Pro). [GNU](http://en.wikipedia.org/wiki/GNU)[GDB](http://en.wikipedia.org/wiki/GDB) uses Python as a pretty printer to show complex structures such as C++ containers. [ESRI](http://en.wikipedia.org/wiki/ESRI) is now promoting Python as the best choice for writing scripts in  [ArcGIS](http://en.wikipedia.org/wiki/ArcGIS" \o "ArcGIS). It has even been used in several video games, and has been adopted as first of the three available [programming languages](http://en.wikipedia.org/wiki/Programming_language)  in  [Google App Engine](http://en.wikipedia.org/wiki/Google_App_Engine) (as of May 2011), the other two being[Java](http://en.wikipedia.org/wiki/Java_(software_platform)) and [Go](http://en.wikipedia.org/wiki/Go_(programming_language)).

Because of its similarities to Lisp, Python has also been used in [Artificial Intelligence](http://en.wikipedia.org/wiki/Artificial_Intelligence)  (AI).

For many operating systems, Python is a standard component; it ships with most  [Linux distributions](http://en.wikipedia.org/wiki/Linux_distribution" \o "Linux distribution),  [NetBSD](http://en.wikipedia.org/wiki/NetBSD" \o "NetBSD),  [OpenBSD](http://en.wikipedia.org/wiki/OpenBSD" \o "OpenBSD) and with [Mac OS X](http://en.wikipedia.org/wiki/Mac_OS_X) and can be used from the terminal. A number of Linux distributions use installers written in Python: [Ubuntu](http://en.wikipedia.org/wiki/Ubuntu_(operating_system)" \o "Ubuntu (operating system)) uses the [Ubiquity](http://en.wikipedia.org/wiki/Ubiquity_(software)) installer, while [Red Hat Linux](http://en.wikipedia.org/wiki/Red_Hat_Linux)  and  [Fedora](http://en.wikipedia.org/wiki/Fedora_(operating_system)) use the [Anaconda](http://en.wikipedia.org/wiki/Anaconda_(installer)) installer. [Gentoo Linux](http://en.wikipedia.org/wiki/Gentoo_Linux" \o "Gentoo Linux) uses Python in its  [package management](http://en.wikipedia.org/wiki/Package_management" \o "Package management) system, [Portage](http://en.wikipedia.org/wiki/Portage_(software)) and the standard tool to access it,  [emerge](http://en.wikipedia.org/wiki/Portage_(software)#Emerge).  [Pardus](http://en.wikipedia.org/wiki/Pardus_(operating_system)) uses it for administration and during system boot.

Python has also seen extensive use in the [information security](http://en.wikipedia.org/wiki/Information_security)  industry, including exploit development.

Among the users of Python are [YouTube](http://en.wikipedia.org/wiki/YouTube) and the original [BitTorrent client](http://en.wikipedia.org/wiki/BitTorrent_(software)" \o "BitTorrent (software)).  Large organizations that make use of Python include [Google](http://en.wikipedia.org/wiki/Google), [Yahoo!](http://en.wikipedia.org/wiki/Yahoo!),  [CERN](http://en.wikipedia.org/wiki/CERN),  [NASA](http://en.wikipedia.org/wiki/NASA), [ILM](http://en.wikipedia.org/wiki/Industrial_Light_%26_Magic),  and [ITA](http://en.wikipedia.org/wiki/ITA_Software). Most of the [Sugar](http://en.wikipedia.org/wiki/Sugar_(GUI)) software for the [One Laptop per Child](http://en.wikipedia.org/wiki/One_Laptop_per_Child)  XO, now developed at [Sugar Labs](http://en.wikipedia.org/wiki/Sugar_Labs), is written in Python.

## 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](http://en.wikipedia.org/wiki/Boilerplate_code) than traditional  [manifestly typed](http://en.wikipedia.org/wiki/Manifest_typing) structured languages such as [C](http://en.wikipedia.org/wiki/C_(programming_language)) or [Pascal](http://en.wikipedia.org/wiki/Pascal_programming_language), and has a smaller number of syntactic exceptions and special cases than either of these. For a detailed description of the differences between 2.x and 3.x versions, see  [History of Python](http://en.wikipedia.org/wiki/Python_3000" \o "Python 3000).

### Indentation

Python uses [whitespace](http://en.wikipedia.org/wiki/Whitespace_(computer_science)) indentation, rather than [curly braces](http://en.wikipedia.org/wiki/Curly_bracket_programming_language) or keywords, to delimit [blocks](http://en.wikipedia.org/wiki/Block_(programming)) (a feature also known as the [off-side rule](http://en.wikipedia.org/wiki/Off-side_rule)). An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block.

### Statements and control flow

Python's statements include (among others):

* The [if statement](http://en.wikipedia.org/wiki/If-then-else), which conditionally executes a block of code, along with else and elif (a contraction of else-if).
* The [for statement](http://en.wikipedia.org/wiki/Foreach" \l "Python" \o "Foreach), which iterates over an iterable object, capturing each element to a local variable for use by the attached block.
* The [while statement](http://en.wikipedia.org/wiki/While_loop), which executes a block of code as long as its condition is true.
* The [try](http://en.wikipedia.org/wiki/Exception_handling_syntax#Python) 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](http://en.wikipedia.org/wiki/Class_(computer_programming)), which executes a block of code and attaches its local namespace to a [class](http://en.wikipedia.org/wiki/Class_(computer_science)), for use in [object-oriented programming](http://en.wikipedia.org/wiki/Object-oriented_programming).
* The def statement, which defines a [function](http://en.wikipedia.org/wiki/Function_(computing)) or [method](http://en.wikipedia.org/wiki/Method_(computing)).
* The with statement (from Python 2.5), which encloses a code block within a context manager (for example, acquiring a [lock](http://en.wikipedia.org/wiki/Lock_(computer_science)) before the block of code is run, and releasing the lock afterwards).
* The pass statement, which serves as a [NOP](http://en.wikipedia.org/wiki/NOP) and can be used in place of a code block.
* The  [assert  statement](http://en.wikipedia.org/wiki/Assertion_(programming)" \o "Assertion (programming)), used during debugging to check for conditions that ought to apply.
* The  yield statement, which returns a value from a [generator](http://en.wikipedia.org/wiki/Generator_(computer_science)#Python) function. (From Python 2.5, yield is also an operator. This form is used to implement  [coroutines](http://en.wikipedia.org/wiki/Coroutine" \o "Coroutine) -- 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](http://en.wikipedia.org/wiki/Continuation), and according to Guido van Rossum it never will. However, better support for [coroutine](http://en.wikipedia.org/wiki/Coroutine" \o "Coroutine)-like functionality is provided in 2.5, by extending Python's [generators](http://en.wikipedia.org/wiki/Generator_(computer_science)).  Prior to 2.5, generators were [lazy](http://en.wikipedia.org/wiki/Lazy_evaluation) [iterators](http://en.wikipedia.org/wiki/Iterator" \o "Iterator); 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](http://en.wikipedia.org/wiki/C_(programming_language)) and [Java](http://en.wikipedia.org/wiki/Java_(programming_language)).

* In Python 2, the / operator on integers does [integer division](http://en.wikipedia.org/wiki/Integer_division): 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, == compares by value, in contrast to Java, where it compares by reference. (Value comparisons in Java use the equals() method.) Python's is operator may be used to compare object identities (comparison by reference). Comparisons may be chained, for example a <= b <= c.
* Python uses the words and, or, not for its boolean operators rather than the symbolic &&, ||, ! used in C.
* Python has a type of expression known as a [*list comprehension*](http://en.wikipedia.org/wiki/List_comprehension). Python 2.4 extended list comprehensions into a more general expression known as a [*generator*](http://en.wikipedia.org/wiki/Generator_(computer_science))*expression*.
* [Anonymous functions](http://en.wikipedia.org/wiki/Anonymous_function) are implemented using [lambda expressions](http://en.wikipedia.org/wiki/Lambda_expressions); however, these are limited in that the body can only be a single expression.
* Conditional expressions in Python are written as x if c else y  (different in order of operands from the [?:](http://en.wikipedia.org/wiki/%3F:) operator common to many other languages).
* Python makes a distinction between [lists](http://en.wikipedia.org/wiki/List_(computer_science)) and [tuples](http://en.wikipedia.org/wiki/Tuple" \o "Tuple). Lists are written as [1, 2, 3], are mutable, and cannot be used as the keys of dictionaries (dictionary keys must be[immutable](http://en.wikipedia.org/wiki/Immutable) in Python). Tuples are written as (1, 2, 3), are immutable and thus can be used as the keys of dictionaries, provided all elements of the tuple are immutable. The parentheses around the tuple are optional in some contexts. Tuples can appear on the left side of an equal sign; hence a statement like x, y = y, x can be used to swap two variables.
* Python 2 has a "string format" operator %. This functions analogous to [printf](http://en.wikipedia.org/wiki/Printf)  format strings in [C](http://en.wikipedia.org/wiki/C_(programming_language)), e.g. "foo=%s bar=%d" % ("blah", 2) evaluates to "foo=blah bar=2". In Python 3, this was obsoleted in favour of the format() method of the str class, e.g. "foo={0} bar={1}".format("blah", 2).
* Python has various kinds of [string literals](http://en.wikipedia.org/wiki/String_literal):
  + Strings delimited by single or double quotation marks. Unlike in [Unix shells](http://en.wikipedia.org/wiki/Unix_shell), [Perl](http://en.wikipedia.org/wiki/Perl) and Perl-influenced languages, single quotation marks and double quotation marks function similarly. Both kinds of string use the backslash (\) as an [escape character](http://en.wikipedia.org/wiki/Escape_character) and there is no implicit [string interpolation](http://en.wikipedia.org/wiki/String_interpolation) such as "$foo".
  + Triple-quoted strings, which begin and end with a series of three single or double quotation marks. They may span multiple lines and function like  [here documents](http://en.wikipedia.org/wiki/Here_document" \o "Here document) in shells, Perl and [Ruby](http://en.wikipedia.org/wiki/Ruby_(programming_language)).
  + [Raw string](http://en.wikipedia.org/wiki/Raw_string) varieties, denoted by prefixing the string literal with an r. No escape sequences are interpreted; hence raw strings are useful where literal backslashes are common, such as [regular expressions](http://en.wikipedia.org/wiki/Regular_expression) and [Windows](http://en.wikipedia.org/wiki/Microsoft_Windows)-style paths. Compare "@-quoting" in [C#](http://en.wikipedia.org/wiki/C_Sharp_(programming_language)).
* Python has  [index](http://en.wikipedia.org/wiki/Array_index) and [slice](http://en.wikipedia.org/wiki/Array_slicing) expressions on lists, denoted as a[key],  a [start:stop]  or  a[start:stop:step]. Indexes are [zero-based](http://en.wikipedia.org/wiki/Zero-based), and negative indexes are relative to the end. Slices take elements from the *start* index up to, but not including, the *stop* index. The third slice parameter, called *step* or *stride*, allows elements to be skipped and reversed. Slice indexes may be omitted, for example a[:] returns a [shallow copy](http://en.wikipedia.org/wiki/Shallow_copy) of the entire list.

In Python, a distinction between expressions and statements is rigidly enforced, in contrast to languages such as [Common Lisp](http://en.wikipedia.org/wiki/Common_Lisp), [Scheme](http://en.wikipedia.org/wiki/Scheme_(programming_language)), or [Ruby](http://en.wikipedia.org/wiki/Ruby_(programming_language)). This leads to some duplication of functionality, e.g.

* [list comprehensions](http://en.wikipedia.org/wiki/List_comprehensions) vs. for-loops
* [conditional](http://en.wikipedia.org/wiki/Conditional_(programming)) expressions vs. if blocks
* The  eval() vs. exec() built-in functions (in Python 2, exec is a statement); the former is for expressions, the latter is for statements.

Statements cannot be a part of an expression and so list and other comprehensions or [lambda expressions](http://en.wikipedia.org/wiki/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 operator = for an equality operator == in conditions: if (c = 1) { ... }  is valid C code but if c = 1: ... causes a syntax error in Python.

### Methods

[Methods](http://en.wikipedia.org/wiki/Method_(programming))  on objects are [functions](http://en.wikipedia.org/wiki/Function_(programming)) attached to the object's class; the syntax  instance.method(argument) is, for normal methods and functions, [syntactic sugar](http://en.wikipedia.org/wiki/Syntactic_sugar)  forClass.method(instance, argument). Python methods have an explicit [self](http://en.wikipedia.org/wiki/This_(computer_science))  parameter to access [instance data](http://en.wikipedia.org/wiki/Instance_data), in contrast to the implicit self in some other object-oriented programming languages (for example, [Java](http://en.wikipedia.org/wiki/Java_(programming_language)), [C++](http://en.wikipedia.org/wiki/C%2B%2B) or [Ruby](http://en.wikipedia.org/wiki/Ruby_(programming_language))).

### Typing

Python uses [duck typing](http://en.wikipedia.org/wiki/Duck_typing) and has typed objects but untyped variable names. Type constraints are not checked at [compile time](http://en.wikipedia.org/wiki/Compile_time); rather, operations on an object may fail, signifying that the given object is not of a suitable type. Despite being [dynamically typed](http://en.wikipedia.org/wiki/Dynamic_programming_language), Python is [strongly typed](http://en.wikipedia.org/wiki/Strongly_typed_programming_language), 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](http://en.wikipedia.org/wiki/Class_(computer_science)), which are most often used for [object-oriented programming](http://en.wikipedia.org/wiki/Object-oriented_programming). New [instances](http://en.wikipedia.org/wiki/Object_(computer_science)) of classes are constructed by calling the class (for example, SpamClass() or EggsClass()), and the classes themselves are instances of the [metaclass](http://en.wikipedia.org/wiki/Metaclass" \o "Metaclass) type (itself an instance of itself), allowing [metaprogramming](http://en.wikipedia.org/wiki/Metaprogramming" \o "Metaprogramming) and[reflection](http://en.wikipedia.org/wiki/Reflection_(computer_science)).

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 3's built-in types:

### Mathematics

Python defines the modulus operator so that the result of a % b is in the  [half-open interval](http://en.wikipedia.org/wiki/Half-open_interval" \o "Half-open interval)  [0,*b*), where b is a positive integer. When b is negative, the result lies in the interval (*b*,0]. 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 Python's modulus operator is consequently defined in a way which can return negative numbers.

Python provides a round function for [rounding](http://en.wikipedia.org/wiki/Rounding) floats to integers. Versions before 3 use round-away-from-zero: round(0.5) is 1.0, round(-0.5) is -1.0.  Python 3 uses [round-to-even](http://en.wikipedia.org/wiki/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.

## Implementations

### CPython

The mainstream Python implementation, known as [*CPython*](http://en.wikipedia.org/wiki/CPython), is written in  [C](http://en.wikipedia.org/wiki/C_(programming_language)" \o "C (programming language)) meeting the  [C89](http://en.wikipedia.org/wiki/C89_(C_version))  standard.  CPython compiles Python programs into intermediate  [bytecode](http://en.wikipedia.org/wiki/Bytecode" \o "Bytecode), which are then executed by the virtual machine. 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](http://en.wikipedia.org/wiki/Microsoft_Windows) and most modern [Unix-like](http://en.wikipedia.org/wiki/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](http://en.wikipedia.org/wiki/Amoeba_distributed_operating_system), alongside more conventional ones like  [Unix](http://en.wikipedia.org/wiki/Unix" \o "Unix)  and  [Mac OS](http://en.wikipedia.org/wiki/Mac_OS), has greatly helped in this regard.

[Stackless Python](http://en.wikipedia.org/wiki/Stackless_Python) is a significant fork of CPython that implements [microthreads](http://en.wikipedia.org/wiki/Microthread" \o "Microthread); 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](http://en.wikipedia.org/wiki/Unladen_Swallow" \o "Unladen Swallow) in 2009 with the aims of increasing the speed of the Python interpreter by 5 times by using the [LLVM](http://en.wikipedia.org/wiki/LLVM) and improving its multithreading ability to scale to thousands of cores.

### Alternative implementations

[Jython](http://en.wikipedia.org/wiki/Jython)  compiles the Python program into Java byte code, which can then be executed by every [Java Virtual Machine](http://en.wikipedia.org/wiki/Java_Virtual_Machine) implementation. This also enables the use of Java class library functions from the Python program.  [IronPython](http://en.wikipedia.org/wiki/IronPython)  follows a similar approach in order to run Python programs on the .NET  [Common Language Runtime](http://en.wikipedia.org/wiki/Common_Language_Runtime" \o "Common Language Runtime). [PyPy](http://en.wikipedia.org/wiki/PyPy" \o "PyPy) is a fast [self-hosting](http://en.wikipedia.org/wiki/Self-hosting)implementation of Python, written in Python, that can output several types of [bytecode](http://en.wikipedia.org/wiki/Bytecode" \o "Bytecode), [object code](http://en.wikipedia.org/wiki/Object_code)  and  [intermediate languages](http://en.wikipedia.org/wiki/Intermediate_language" \o "Intermediate language). There also exist compilers to high-level  [object languages](http://en.wikipedia.org/wiki/Object_language" \o "Object language), 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](http://en.wikipedia.org/wiki/RPython) to several languages; other examples include  [Pyjamas](http://en.wikipedia.org/wiki/Pyjamas_(software)" \o "Pyjamas (software))  compiling to [JavaScript](http://en.wikipedia.org/wiki/JavaScript); [Shed Skin](http://en.wikipedia.org/wiki/Shedskin) compiling to [C++](http://en.wikipedia.org/wiki/C%2B%2B); and  [Cython](http://en.wikipedia.org/wiki/Cython" \o "Cython)  and  [Pyrex](http://en.wikipedia.org/wiki/Pyrex_(programming_language))  compiling to [C](http://en.wikipedia.org/wiki/C_(programming_language)).

In 2005  [Nokia](http://en.wikipedia.org/wiki/Nokia" \o "Nokia)  released a Python interpreter for the [Series 60](http://en.wikipedia.org/wiki/Series_60) [mobile phones](http://en.wikipedia.org/wiki/Mobile_phone)  called  [PyS60](http://en.wikipedia.org/wiki/PyS60). It includes many of the modules from the CPython implementations and some additional modules for integration with the  [Symbian](http://en.wikipedia.org/wiki/Symbian" \o "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](http://en.wikipedia.org/wiki/N900" \o "N900)  also supports Python with [gtk](http://en.wikipedia.org/wiki/Gtk" \o "Gtk) widget 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](http://en.wikipedia.org/wiki/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. PyMite was folded into [Python-on-a-Chip](http://pythononachip.org/) in 2009. 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.

Around 2004, the  [Pyastra](http://pyastra.sourceforge.net/)  project created a specialized translator and assembler that targets resource-constrained [microcontrollers](http://en.wikipedia.org/wiki/Microcontroller).

[*ChinesePython*](http://en.wikipedia.org/w/index.php?title=ChinesePython&action=edit&redlink=1)  (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

Python is available on Android as an option as part of the Android Scripting Environment.

### Interpretational semantics

Most Python implementations (including [CPython](http://en.wikipedia.org/wiki/Python_(programming_language)" \l "CPython)) can function as a [command line interpreter](http://en.wikipedia.org/wiki/Command_line_interpreter), for which the user enters statements sequentially and receives the results immediately. In short, Python acts as a [shell](http://en.wikipedia.org/wiki/Shell_(computing)). 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](http://en.wikipedia.org/wiki/IDLE_(Python)" \o "IDLE (Python)) and  [IPython](http://en.wikipedia.org/wiki/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](http://en.wikipedia.org/wiki/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](http://en.wikipedia.org/wiki/RPython" \o "RPython)*.

[Psyco](http://en.wikipedia.org/wiki/Psyco)  is a  [specialising](http://en.wikipedia.org/wiki/Specialising_compiler" \o "Specialising compiler) [just in time compiler](http://en.wikipedia.org/wiki/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](http://en.wikipedia.org/wiki/Data_types" \o "Data types)and is faster than standard Python code. Psyco is compatible with all Python code, not only a subset.

## Development

Python's development is conducted largely through the Python Enhancement Proposal (PEP) process. PEPs are standardized design documents providing general information related to Python, including proposals, descriptions,  [design rationales](http://en.wikipedia.org/wiki/Design_rationale" \o "Design rationale), and explanations for language features.  Outstanding PEPs are reviewed and commented upon by Van Rossum, the Python project's  [Benevolent Dictator for Life](http://en.wikipedia.org/wiki/BDFL" \o "BDFL)  (leader / language architect).  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](http://en.wikipedia.org/wiki/Roundup_(issue_tracker)" \o "Roundup (issue tracker)) [bug tracker](http://en.wikipedia.org/wiki/Bug_tracker) maintained at python.org. Development takes place at the self-hosted[hg](http://en.wikipedia.org/wiki/Mercurial).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](http://en.wikipedia.org/wiki/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.
* 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.

A number of  [alpha, beta, and release-candidates](http://en.wikipedia.org/wiki/Beta_release" \o "Beta release) 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](http://en.wikipedia.org/wiki/Unit_test)  suite during development, and using the  [BuildBot](http://en.wikipedia.org/wiki/BuildBot" \o "BuildBot)[continuous integration](http://en.wikipedia.org/wiki/Continuous_integration)  system.

## Standard library

Python has a large standard library, commonly cited as one of Python's greatest strengths, providing pre-written tools suited to many tasks. This is deliberate and has been described as a "batteries included" Python philosophy. The modules of the standard library can be augmented with custom modules written in either C or Python. [Boost C++ Libraries](http://en.wikipedia.org/wiki/Boost_C%2B%2B_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](http://en.wikipedia.org/wiki/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](http://en.wikipedia.org/wiki/MIME" \o "MIME)  and  [HTTP](http://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol)) already supported. Modules for creating [graphical user interfaces](http://en.wikipedia.org/wiki/Graphical_user_interface), connecting to [relational databases](http://en.wikipedia.org/wiki/Relational_database), arithmetic with arbitrary precision decimals, manipulating [regular expressions](http://en.wikipedia.org/wiki/Regular_expression), and doing [unit testing](http://en.wikipedia.org/wiki/Unit_testing)are also included.

Some parts of the standard library are covered by specifications (for example, the  [WSGI](http://en.wikipedia.org/wiki/Web_Server_Gateway_Interface" \o "Web Server Gateway Interface) implementation  wsgiref  follows [PEP 333](http://www.python.org/dev/peps/pep-0333/)), 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.

For  [software testing](http://en.wikipedia.org/wiki/Software_testing" \o "Software testing), the standard library provides the  unittest  and  [doctest](http://en.wikipedia.org/wiki/Doctest" \o "Doctest)  modules.

## Influences on other languages

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

* [Pyrex](http://en.wikipedia.org/wiki/Pyrex_programming_language)  and its derivative [Cython](http://en.wikipedia.org/wiki/Cython" \o "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](http://en.wikipedia.org/wiki/Boo_(programming_language)) uses indentation, a similar syntax, and a similar object model. However, Boo uses [static typing](http://en.wikipedia.org/wiki/Static_typing) and is closely integrated with the [.NET framework](http://en.wikipedia.org/wiki/.NET_framework).
* [Cobra](http://en.wikipedia.org/wiki/Cobra_(programming_language_from_Cobra_Language_LLC))  uses indentation and a similar syntax. Cobra's "Acknowledgements" document lists Python first among languages that influenced it.  However, Cobra directly supports[design-by-contract](http://en.wikipedia.org/wiki/Design_by_contract), [unit tests](http://en.wikipedia.org/wiki/Unit_testing) and optional [static typing](http://en.wikipedia.org/wiki/Static_typing).
* [ECMAScript](http://en.wikipedia.org/wiki/ECMAScript) borrowed [iterators](http://en.wikipedia.org/wiki/Iterator" \o "Iterator), [generators](http://en.wikipedia.org/wiki/Generator_(computer_science)), and [list comprehensions](http://en.wikipedia.org/wiki/List_comprehension) from Python.
* [Go](http://en.wikipedia.org/wiki/Go_(programming_language)) is described as incorporating the "development speed of working in a dynamic language like Python".
* [Groovy](http://en.wikipedia.org/wiki/Groovy_(programming_language)) was motivated by the desire to bring the Python design philosophy to  [Java](http://en.wikipedia.org/wiki/Java_(programming_language)" \o "Java (programming language)).
* [OCaml](http://en.wikipedia.org/wiki/OCaml) has an optional syntax, called twt (The Whitespace Thing), inspired by Python and [Haskell](http://en.wikipedia.org/wiki/Haskell_(programming_language)).

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](http://en.wikipedia.org/wiki/Tcl" \o "Tcl)  and [Erlang](http://en.wikipedia.org/wiki/Erlang_(programming_language)" \o "Erlang (programming language))  because of Python's influence.