Python (programming language)

From Wikipedia, the free encyclopedia

Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation.[31]

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library.[32][33]

Guido van Rossum began working on Python in the late 1980s as a successor to the ABC programming language and first released it in 1991 as Python 0.9.0.[34] Python 2.0 was released in 2000 and introduced new features such as list comprehensions, cycle-detecting garbage collection, reference counting, and Unicode support. Python 3.0, released in 2008, was a major revision that is not completely backward-compatible with earlier versions. Python 2 was discontinued with version 2.7.18 in 2020.[35]

Python consistently ranks as one of the most popular programming languages.[36][37][38][39]

Python

Paradigm	Multi-paradigm: object-oriented,[1] procedural (imperative), functional, structured, reflective
Designed by	Guido van Rossum
Developer	Python Software Foundation
First appeared	20 February 1991; 31 years ago[2]
Stable release	3.10.6[3] / 2 August 2022; 5 days ago
Typing discipline	Duck, dynamic, strong typing;[4] gradual (since 3.5, but ignored in CPython)[5]
OS	Windows, macOS, Linux/UNIX, Android[6][7] and more[8]
License	Python Software Foundation License
Filename extensions	.py, .pyi, .pyc, .pyd, .pyw, .pyz (since 3.5),[9] .pyo (prior to 3.5)[10]
Website	www.python.org
Major implementations	CPython, PyPy, Stackless Python, MicroPython, CircuitPython, IronPython, Jython
Dialects	Cython, RPython, Starlark[11]
Influenced by	ABC,[12] Ada,[13] ALGOL 68,[14] APL,[15] C,[16] C++,[17] CLU,[18] Dylan,[19] Haskell,[20] Icon,[21] Lisp,[22] Modula-3,[17] Perl, Standard ML, VB[15]
Influenced	Apache Groovy, Boo, Cobra, CoffeeScript,[23] D, F#, Genie,[24] Go, JavaScript,[25][26] Julia,[27] Nim, Ring,[28] Ruby,[29] Swift[30]

Python Programming at Wikibooks

Contents [hide]

History [edit]

Main page
Contents
Current events
Random article
About Wikipedia
Contact us
Donate

Contribute
Help
Learn to edit
Community portal
Recent changes
Upload file

Tools
What links here
Related changes
Special pages
Permanent link
Page information
Cite this page
Wikidata item

Print/export
Download as PDF

In other projects
Wikimedia Commons
MediaWiki
Wikibooks
Wikiquote
Wikiversity

Languages
Afrikaans
Alemannisch
العربية
Aragonés
Asturianu
Azərbaycanca
تۆرکجه
Basa Bali
Bân-lâm-gú
Беларуская
Български
Bosanski
Brezhoneg
Català
Cebuano
Čeština
Cymraeg
Dansk
Deutsch
Eesti

Ελληνικά
Español
Esperanto
Euskara
فارسی
Français
Galego
Հայերեն
Hrvatski
Bahasa Indonesia
Interlingua
Íslenska
Italiano
עברית
ქართული
Қазақша
Kiswahili
Kurdî
Кыргызча
Latina
Latviešu
Lietuvių
La .lojban.
Lombard
Magyar
Македонски
Bahasa Melayu
Монгол
မြန်မာဘာသာ
Na Vosa Vakaviti
Nederlands
日本語
ᓄᓇᕗᑦ
Norsk bokmål
Norsk nynorsk
O'zbekcha/ўзбекча
پنجابی
ភាសាខ្មែរ
Plattdüütsch
Polski
Português
Română
Русский
Саха тыла
Scots
Shqip
සිංහල
Simple English
Slovenčina
Slovenščina
کوردی
Српски / srpski
Srpskohrvatski / српскохрватски
Suomi
Svenska
Tagalog
Татарча/tatarça

Main article: History of Python

The designer of Python, Guido van Rossum, at OSCON 2006

Python was conceived in the late 1980s[40] by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC programming language, which was inspired by SETL,[41] capable of exception handling and interfacing with the Amoeba operating system.[12] Its implementation began in December 1989.[42] Van Rossum shouldered sole responsibility for the project, as the lead developer, until 12 July 2018, when he announced his "permanent vacation" from his responsibilities as Python's "benevolent dictator for life", a title the Python community bestowed upon him to reflect his long-term commitment as the project's chief decision-maker.[43] In January 2019, active Python core developers elected a five-member Steering Council to lead the project.[44][45]

Python 2.0 was released on 16 October 2000, with many major new features.[46] Python 3.0, released on 3 December 2008, with many of its major features backported to Python 2.6.x[47] and 2.7.x. Releases of Python 3 include the `2to3` utility, which automates the translation of Python 2 code to Python 3.[48]

Python 2.7's end-of-life was initially set for 2015, then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3.[49][50] No further security patches or other improvements will be released for it.[51][52] With Python 2's end-of-life, only Python 3.6.x[53] and later were supported. Later, support for 3.6 was also discontinued. In 2021, Python 3.9.2 and 3.8.8 were expedited[54] as all versions of Python (including 2.7[55]) had security issues leading to possible remote code execution[56] and web cache poisoning.[57]

In 2022, Python 3.10.4 and 3.9.12 were expedited[58] and so were older releases including 3.8.13, and 3.7.13 because of many security issues.[59] Python 3.9.13 is the latest 3.9 version, and from now on 3.9 (and older; 3.8 and 3.7) will only get security updates.[60]

Design philosophy and features [edit]

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including metaprogramming[61] and metaobjects [magic methods]).[62] Many other paradigms are supported via extensions, including design by contract[63][64] and logic programming.[65]

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management.[66] It uses dynamic name resolution (late binding), which binds method and variable names during program execution.

Its design offers some support for functional programming in the Lisp tradition. It has `filter`, `map` and `reduce` functions; list comprehensions, dictionaries, sets, and generator expressions.[67] The standard library has two modules (`itertools` and `functools`) that implement functional tools borrowed from Haskell and Standard ML.[68]

Its core philosophy is summarized in the document *The Zen of Python* (*PEP 20*), which includes aphorisms such as:[69]

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Readability counts.

Rather than building all of its functionality into its core, Python was designed to be highly extensible via modules. This compact modularity has made it particularly popular as a means of adding programmable interfaces to existing applications. Van Rossum's vision of a small core language with a large standard library and easily extensible interpreter stemmed from his frustrations with ABC, which espoused the opposite approach.[40]

Python strives for a simpler, less-cluttered syntax and grammar while giving developers a choice in their coding methodology. In contrast to Perl's "there is more than one way to do it" motto, Python embraces a "there should be one—and preferably only one—obvious way to do it" philosophy.[69] Alex Martelli, a Fellow at the Python Software Foundation and Python book author, wrote: "To describe something as 'clever' is *not* considered a compliment in the Python culture."[70]

ไทย
Тоҷикӣ
Türkçe
Українська
اردو
ئۇيغۇرچە / Uyghurche
Tiếng Việt
Walon
文言
Winaray
吴语
粵語
中文
Edit links

Python's developers strive to avoid premature optimization and reject patches to non-critical parts of the CPython reference implementation that would offer marginal increases in speed at the cost of clarity.[71] When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C; or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter.

Python's developers aim for it to be fun to use. This is reflected in its name—a tribute to the British comedy group Monty Python[72]—and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (a reference to a Monty Python sketch) instead of the standard foo and bar.[73][74]

The programming language's name 'Python' came from a BBC Comedy series in the 1970's named Monty Python's Flying Circus. Guido van Rossum thought he needed a name that was short, unique and slightly mysterious, And so, he decided to name the programming language 'Python'.[72]

A common neologism in the Python community is *pythonic*, which has a wide range of meanings related to program style. "Pythonic" code may use Python idioms well, be natural or show fluency in the language, or conform with Python's minimalist philosophy and emphasis on readability. Code that is difficult to understand or reads like a rough transcription from another programming language is called *unpythonic*.[75][76]

Python users and admirers, especially those considered knowledgeable or experienced, are often referred to as *Pythonistas*.[77][78]

Syntax and semantics [edit]

Main article: Python syntax and semantics

Python is meant to be an easily readable language. Its formatting is visually uncluttered and often uses English keywords where other languages use punctuation. Unlike many other languages, it does not use curly brackets to delimit blocks, and semicolons after statements are allowed but rarely used. It has fewer syntactic exceptions and special cases than C or Pascal.[79]

Indentation [edit]

Main article: Python syntax and semantics § Indentation

Python uses whitespace indentation, rather than curly brackets or keywords, to delimit blocks. An increase in indentation comes after certain statements; a decrease in indentation signifies the end of the current block.[80] Thus, the program's visual structure accurately represents its semantic structure.[81] This feature is sometimes termed the off-side rule. Some other languages use indentation this way; but in most, indentation has no semantic meaning. The recommended indent size is four spaces.[82]

Statements and control flow [edit]

Python's statements include:

- The assignment statement, using a single equals sign `=`
- 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 (or new syntax `except*` in Python 3.11 for exception groups[83]); it also ensures that clean-up code in a `finally` block is always run regardless of how the block exits
- The `raise` statement, used to raise a specified exception or re-raise a caught exception
- 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, which encloses a code block within a context manager (for example, acquiring a lock before it is run, then releasing the lock; or opening and closing a file), allowing resource-acquisition-is-initialization (RAII)-like behavior and replacing a common try/finally idiom[84]
- The `break` statement, which exits a loop
- The `continue` statement, which skips the current iteration and continues with the next
- The `del` statement, which removes a variable—deleting the reference from the name to the value, and

producing an error if the variable is referred to before it is redefined
- The `pass` statement, serving as a NOP, syntactically needed to create an empty code block
- The `assert` statement, used in debugging to check for conditions that should apply
- The `yield` statement, which returns a value from a generator function (and also an operator); used to implement coroutines
- The `return` statement, used to return a value from a function
- The `import` statement, used to import modules whose functions or variables can be used in the current program

The assignment statement (`=`) binds a name as a reference to a separate, dynamically-allocated object. Variables may subsequently be rebound at any time to any object. In Python, a variable name is a generic reference holder without a fixed data type; however, it always refers to *some* object with a type. This is called dynamic typing—in contrast to statically-typed languages, where each variable may contain only a value of a certain type.

Python does not support tail call optimization or first-class continuations, and, according to van Rossum, it never will.[85][86] However, better support for coroutine-like functionality is provided by extending Python's generators.[87] Before 2.5, generators were lazy iterators; data was passed unidirectionally out of the generator. From Python 2.5 on, it is possible to pass data back into a generator function; and from version 3.3, it can be passed through multiple stack levels.[88]

Expressions [edit]

Some Python expressions are similar to those in languages such as C and Java, while some are not:

- Addition, subtraction, and multiplication are the same, but the behavior of division differs. There are two types of divisions in Python: floor division (or integer division) `//` and floating-point `/` division.[89] Python also uses the `**` operator for exponentiation.
- The `@` infix operator. It is intended to be used by libraries such as NumPy for matrix multiplication.[90][91]
- The syntax `:=`, called the "walrus operator", was introduced in Python 3.8. It assigns values to variables as part of a larger expression.[92]
- In Python, `==` compares by value, versus Java, which compares numerics by value[93] and objects by reference.[94] Python's `is` operator may be used to compare object identities (comparison by reference), and comparisons may be chained—for example, `a <= b <= c`.
- Python uses `and`, `or`, and `not` as boolean operators rather than the symbolic `&&`, `||`, `!` in Java and C.
- Python has a type of expression called a *list comprehension*, as well as a more general expression called a *generator expression*.[67]
- Anonymous functions are implemented using lambda expressions; however, there may be only one expression in each body.
- Conditional expressions are written as `x if c else y`[95] (different in order of operands from the `c ? x : y` 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, written as `(1, 2, 3)`, are immutable and thus can be used as keys of dictionaries, provided all of the tuple's elements are immutable. The `+` operator can be used to concatenate two tuples, which does not directly modify their contents, but produces a new tuple containing the elements of both. Thus, given the variable `t` initially equal to `(1, 2, 3)`, executing `t = t + (4, 5)` first evaluates `t + (4, 5)`, which yields `(1, 2, 3, 4, 5)`, which is then assigned back to `t`—thereby effectively "modifying the contents" of `t` while conforming to the immutable nature of tuple objects. Parentheses are optional for tuples in unambiguous contexts.[96]
- Python features *sequence unpacking* where multiple expressions, each evaluating to anything that can be assigned (to a variable, writable property, etc.) are associated in an identical manner to that forming tuple literals—and, as a whole, are put on the left-hand side of the equal sign in an assignment statement. The statement expects an *iterable* object on the right-hand side of the equal sign that produces the same number of values as the provided writable expressions; when iterated through them, it assigns each of the produced values to the corresponding expression on the left.[97]
- Python has a "string format" operator `%` that functions analogously to `printf` format strings in C—e.g. `"spam=%s eggs=%d" % ("blah", 2)` evaluates to `"spam=blah eggs=2"`. In Python 2.6+ and 3+, this was supplemented by the `format()` method of the `str` class, e.g. `"spam={0}`

`eggs={1}".format("blah", 2)`. Python 3.6 added "f-strings": `spam = "blah"; eggs = 2; f'spam={spam} eggs={eggs}'`.[98]

- Strings in Python can be concatenated by "adding" them (with the same operator as for adding integers and floats), e.g. `"spam" + "eggs"` returns `"spameggs"`. If strings contain numbers, they are added as strings rather than integers, e.g. `"2" + "2"` returns `"22"`.
- Python has various string literals:
  - Delimited by **single or double quote marks**. Unlike in Unix shells, Perl, and Perl-influenced languages, single and double quote marks function identically. Both use the backslash (`\`) as an escape character. String interpolation became available in Python 3.6 as "formatted string literals".[98]
  - **Triple-quoted** (beginning and ending with three single or double quote marks), which may span multiple lines and function like here documents in shells, Perl, and Ruby.
  - Raw string varieties, denoted by prefixing the string literal with `r`. Escape sequences are not interpreted; hence raw strings are useful where literal backslashes are common, such as regular expressions and Windows-style paths. (Compare "`@`-quoting" in C#.)
- Python has array index and array slicing expressions in lists, denoted as `a[key]`, `a[start:stop]` or `a[start:stop:step]`. Indexes are 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 copy of the entire list. Each element of a slice is a shallow copy.

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 duplicating some functionality. For example:

- List comprehensions vs. `for`-loops
- Conditional 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—so list and other comprehensions or lambda expressions, all being expressions, cannot contain statements. A particular case 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 syntactically valid (but probably unintended) C code, but `if c = 1: ...` causes a syntax error in Python.

Methods [edit]

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 (or `this`) in some other object-oriented programming languages (e.g., C++, Java, Objective-C, Ruby).[99] Python also provides methods, often called *dunder methods* (due to their names beginning and ending with double-underscores), to allow user-defined classes to modify how they are handled by native operations including length, comparison, in arithmetic operations and type conversion.[100]

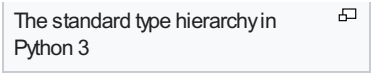
Typing [edit]

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 it 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, 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 are instances of the metaclass `type` (itself an instance of itself), allowing metaprogramming and reflection.

Before version 3.0, Python had two kinds of classes (both using the

same syntax): *old-style* and *new-style*,[101] current Python versions only support the semantics new style.


The standard type hierarchy in Python 3

The long-term plan is to support gradual typing.[102] Python's syntax allows specifying static types, but they are not checked in the default implementation, CPython. An experimental optional static type-checker, *mypy*, supports compile-time type checking.[103]

Summary of Python 3's built-in types

Type	Mutability	Description	Syntax examples
`bool`	immutable	Boolean value	`True` `False`

`bytearray`	mutable	Sequence of bytes	`bytearray(b'Some ASCII')`
			`bytearray(b"Some ASCII")`
			`bytearray([119, 105, 107, 105])`

`bytes`	immutable	Sequence of bytes	`b'Some ASCII'`
			`b"Some ASCII"`
			`bytes([119, 105, 107, 105])`

`complex`	immutable	Complex number with real and imaginary parts	`3+2.7j` `3 + 2.7j`
`dict`	mutable	Associative array (or dictionary) of key and value pairs; can contain mixed types (keys and values), keys must be a hashable type	`{'key1': 1.0, 3: False}` `{}`
`types.EllipsisType`	immutable	An ellipsis placeholder to be used as an index in NumPy arrays	`...` `Ellipsis`
`float`	immutable	Double-precision floating-point number. The precision is machine-dependent but in practice is generally implemented as a 64-bit IEEE 754 number with 53 bits of precision.[104]	`1.33333`

`frozenset`	immutable	Unordered set, contains no duplicates; can contain mixed types, if hashable	`frozenset([4.0, 'string', True])`
`int`	immutable	Integer of unlimited magnitude[105]	`42`
`list`	mutable	List, can contain mixed types	`[4.0, 'string', True]` `[]`
`types.NoneType`	immutable	An object representing the absence of a value, often called null in other languages	`None`
`types.NotImplementedType`	immutable	A placeholder that can be returned from overloaded operators to indicate unsupported operand types.	`NotImplemented`
`range`	immutable	A Sequence of numbers commonly used for looping specific number of times in `for` loops[106]	`range(-1, 10)` `range(10, -5, -2)`
`set`	mutable	Unordered set, contains no duplicates; can contain mixed types, if hashable	`{4.0, 'string', True}` `set()`

`str`	immutable	A character string: sequence of Unicode codepoints	`'Wikipedia'`
			`"Wikipedia"`
			`"""Spanning multiple lines"""`
			`'''Spanning multiple lines'''`

`tuple`	immutable	Can contain mixed types	`(4.0, 'string', True)`
			`('single element',)`
			`()`

Arithmetic operations [edit]

Python has the usual symbols for arithmetic operators (`+`, `-`, `*`, `/`), the floor division operator `//` and the modulo operation `%` (where the remainder can be negative, e.g. `4 % -3 == -2`). It also has `**` for exponentiation, e.g. `5**3 == 125` and `9**0.5 == 3.0`, and a matrix-multiplication operator `@`.[107] These operators work like in traditional math; with the same precedence rules, the operators infix (`+` and `-` can also be unary to represent positive and negative numbers respectively).

The division between integers produces floating-point results. The behavior of division has changed significantly over time:[108]

- Current Python (i.e. since 3.0) changed `/` to always be floating-point division, e.g. `5/2 == 2.5`.
- The floor division `//` operator was introduced. So `7//3 == 2`, `-7//3 == -3`, `7.5//3 == 2.0` and `-7.5//3 == -3.0`. Adding `from __future__ import division` causes a module used in Python 2.7 to use Python 3.0 rules for division (see above).

In Python terms, `/` is *true division* (or simply *division*), and `//` is *floor division*. `/` before version 3.0 is *classic division*.[108]

Rounding towards negative infinity, though different from most languages, adds consistency. For instance, it means that the equation `(a + b)//b == a//b + 1` is always true. It also means that the equation `b*(a//b) + a%b == a` is valid for both positive and negative values of `a`. However, maintaining the validity of this equation means that while the result of `a%b` is, as expected, in the half-open interval [0, *b*), where `b` is a positive integer, it has to lie in the interval (*b*, 0] when `b` is negative.[109]

Python provides a `round` function for rounding a float to the nearest integer. For tie-breaking, Python 3 uses round to even: `round(1.5)` and `round(2.5)` both produce `2`.[110] Versions before 3 used round-away-from-zero: `round(0.5)` is `1.0`, `round(-0.5)` is `-1.0`.[111]

Python allows boolean expressions with multiple equality relations in a manner that is consistent with general use in mathematics. For example, the expression `a < b < c` tests whether `a` is less than `b` and `b` is less than `c`.[112] 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`.[113]

Python uses arbitrary-precision arithmetic for all integer operations. The `Decimal` type/class in the `decimal` module provides decimal floating-point numbers to a pre-defined arbitrary precision and several rounding modes.[114] The `Fraction` class in the `fractions` module provides arbitrary precision for rational numbers.[115]

Due to Python's extensive mathematics library, and the third-party library NumPy that further extends the native capabilities, it is frequently used as a scientific scripting language to aid in problems such as numerical data processing and manipulation.[116][117]

Programming examples [edit]

Hello world program:

```
print('Hello, world!')
```

Program to calculate the factorial of a positive integer:

```
n = int(input('Type a number, and its factorial will be printed: '))

if n < 0:
    raise ValueError('You must enter a non-negative integer')

factorial = 1
for i in range(2, n + 1):
    factorial *= i

print(factorial)
```

Libraries [edit]

Python's large standard library [118] provides tools suited to many tasks, and is commonly cited as one of its greatest strengths. For Internet-facing applications, many standard formats and protocols such as MIME and HTTP are supported. It includes modules for creating graphical user interfaces, connecting to relational databases, generating pseudorandom numbers, arithmetic with arbitrary-precision decimals,[119] manipulating regular expressions, and unit testing.

Some parts of the standard library are covered by specifications—for example, the Web Server Gateway Interface (WSGI) implementation `wsgiref` follows PEP 333[120]—but most are specified by their code, internal documentation, and test suites. However, because most of the standard library is cross-platform Python code, only a few modules need altering or rewriting for variant implementations.

As of June 2022, the Python Package Index (PyPI), the official repository for third-party Python software, contains over 380,000[121] packages with a wide range of functionality, including:

- Automation
- Data analytics
- Databases
- Documentation
- Graphical user interfaces
- Image processing
- Machine learning
- Mobile apps
- Multimedia
- Computer networking
- Scientific computing
- System administration
- Test frameworks
- Text processing
- Web frameworks
- Web scraping

Development environments [edit]

See also: Comparison of integrated development environments § Python

Most Python implementations (including CPython) include a read–eval–print loop (REPL), permitting them to function as a command line interpreter for which users enter statements sequentially and receive results immediately.

Python also comes with an Integrated development environment (IDE) called IDLE, which is more beginner-oriented.

Other shells, including IDLE and IPython, add further abilities such as improved auto-completion, session state retention and syntax highlighting.

As well as standard desktop integrated development environments, there are Web browser-based IDEs, including SageMath, for developing science- and math-related programs; PythonAnywhere, a browser-based IDE and hosting environment; and Canopy IDE, a commercial IDE emphasizing scientific computing.[122]

Implementations [edit]

See also: List of Python software § Python implementations

Reference implementation [edit]

CPython is the reference implementation of Python. It is written in C, meeting the C89 standard (Python 3.11 uses C11[123]) with several select C99 features (With later C versions out, it is considered outdated.[124][125] CPython includes its own C extensions, but third-party extensions are not limited to older C versions—e.g. they can be implemented with C11 or C++.[126][127]) It compiles Python programs into an intermediate bytecode[128] which is then executed by its virtual machine.[129] CPython is distributed with a large standard library written in a mixture of C and native Python, and is available for many platforms, including Windows (starting with Python 3.9, the Python installer deliberately fails to install on Windows 7 and 8;[130][131] Windows XP was

supported until Python 3.5) and most modern Unix-like systems, including macOS (and Apple M1 Macs, since Python 3.9.1, with experimental installer) and unofficial support for e.g. VMS.[132] Platform portability was one of its earliest priorities.[133] (During Python 1 and 2 development, even OS/2 and Solaris were supported,[134] but support has since been dropped for many platforms.)

Other implementations [edit]

- PyPy is a fast, compliant interpreter of Python 2.7 and 3.8.[135] [136] Its just-in-time compiler brings a significant speed improvement over CPython but some libraries written in C cannot be used with it.[137]
- Stackless Python is a significant fork of CPython that implements microthreads; it does not use the call stack in the same way, thus allowing massively concurrent programs. PyPy also has a stackless version.[138]
- MicroPython and CircuitPython are Python 3 variants optimized for microcontrollers, including Lego Mindstorms EV3.[139]
- Pyston is a variant of the Python runtime that uses just-in-time compilation to speed up the execution of Python programs.[140]
- Cinder is a performance-oriented fork of CPython 3.8 that contains a number of optimizations including bytecode inline caching, eager evaluation of coroutines, a method-at-a-time JIT and an experimental bytecode compiler.[141]

Unsupported implementations [edit]

Other just-in-time Python compilers have been developed, but are now unsupported:

- Google began a project named Unladen Swallow in 2009, with the aim of speeding up the Python interpreter fivefold by using the LLVM, and of improving its multithreading ability to scale to thousands of cores,[142] while ordinary implementations suffer from the global interpreter lock.
- Psyco is a discontinued just-in-time specializing compiler that integrates with CPython and transforms bytecode to machine code at runtime. The emitted code is specialized for certain data types and is faster than the standard Python code. Psyco does not support Python 2.7 or later.
- PyS60 was a Python 2 interpreter for Series 60 mobile phones released by Nokia in 2005. It implemented many of the modules from the standard library and some additional modules for integrating with the Symbian operating system. The Nokia N900 also supports Python with GTK widget libraries, enabling programs to be written and run on the target device.[143]

Cross-compilers to other languages [edit]

There are several 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:

- Brython,[144] Transcrypt[145][146] and Pyjs (latest release in 2012) compile Python to JavaScript.
- Cython compiles (a superset of) Python 2.7 to C (while the resulting code is also usable with Python 3 and also e.g. C++).
- Nuitka compiles Python into C.[147]
- Numba uses LLVM to compile a subset of Python to machine code.
- Pythran compiles a subset of Python 3 to C++ (C++11).[148][149][150]
- RPython can be compiled to C, and is used to build the PyPy interpreter of Python.
- The Python → 11l → C++ transpiler[151] compiles a subset of Python 3 to C++ (C++17).

Specialized:

- MyHDL is a Python-based hardware description language (HDL), that converts MyHDL code to Verilog or VHDL code.

Older projects (or not to be used with Python 3.x and latest syntax):

- Google's Grumpy (latest release in 2017) transpiles Python 2 to Go.[152][153][154]
- IronPython allows running Python 2.7 programs (and an alpha, released in 2021, is also available for "Python 3.4, although features and behaviors from later versions may be included"[155]) on the .NET Common Language Runtime.[156]
- Jython compiles Python 2.7 to Java bytecode, allowing the use of the Java libraries from a Python program.[157]
- Pyrex (latest release in 2010) and Shed Skin (latest release in 2013) compile to C and C++ respectively.

Performance [edit]

Performance comparison of various Python implementations on a non-numerical (combinatorial) workload was

presented at EuroSciPy '13.[158] Python's performance compared to other programming languages is also benchmarked by The Computer Language Benchmarks Game.[159]

Development [edit]

Python's development is conducted largely through the *Python Enhancement Proposal* (PEP) process, the primary mechanism for proposing major new features, collecting community input on issues, and documenting Python design decisions.[160] Python coding style is covered in PEP 8.[161] Outstanding PEPs are reviewed and commented on by the Python community and the steering council.[160]

Enhancement of the language corresponds with the development of the CPython reference implementation. The mailing list python-dev is the primary forum for the language's development. Specific issues are discussed in the Roundup bug tracker hosted at bugs.python.org.[162] Development originally took place on a self-hosted source-code repository running Mercurial, until Python moved to GitHub in January 2017.[163]

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

- Backward-incompatible versions, where code is expected to break and needs to be manually ported. The first part of the version number is incremented. These releases happen infrequently—version 3.0 was released 8 years after 2.0. According to Guido van Rossum, a version 4.0 is very unlikely to ever happen.[164]
- Major or "feature" releases are largely compatible with the previous version but introduce new features. The second part of the version number is incremented. Starting with Python 3.9, these releases are expected to happen annually.[165][166] Each major version is supported by bug fixes for several years after its release.[167]
- Bugfix releases,[168] which introduce no new features, occur about every 3 months and are made when a sufficient number of bugs have been fixed upstream since the last release. Security vulnerabilities are also patched in these releases. The third and final part of the version number is incremented.[168]

Many alpha, beta, and release-candidates are also released as previews and for testing before final releases. Although there is a rough schedule for each release, they are often delayed if the code is not ready. Python's development team monitors the state of the code by running the large unit test suite during development.[169]

The major academic conference on Python is PyCon. There are also special Python mentoring programmes, such as Pyladies.

Python 3.10 deprecated `wstr` (to be removed in Python 3.12; meaning Python extensions[170] need to be modified by then),[171] and added pattern matching to the language.[172]

API documentation generators [edit]

Tools that can generate documentation for Python API include pydoc (available as part of the standard library), Sphinx, Pdoc and its forks, Doxygen and Graphviz, among others.[173]

Naming [edit]

Python's name is derived from the British comedy group Monty Python, whom Python creator Guido van Rossum enjoyed while developing the language. Monty Python references appear frequently in Python code and culture;[174] for example, the metasyntactic variables often used in Python literature are *spam* and *eggs* instead of the traditional *foo* and *bar*.[174][175] The official Python documentation also contains various references to Monty Python routines.[176][177]

The prefix *Py-* is 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); PyQt and PyGTK, which bind Qt and GTK to Python respectively; and PyPy, a Python implementation originally written in Python.

Popularity [edit]

Since 2003, Python has consistently ranked in the top ten most popular programming languages in the TIOBE Programming Community Index where, as of October 2021, it is the most popular language (ahead of Java, and C).[178] It was selected Programming Language of the Year (for "the highest rise in ratings in a year") in 2007, 2010, 2018, and 2020 (the only language to do so four times[179]).[180]

An empirical study found that scripting languages, such as Python, are more productive than conventional

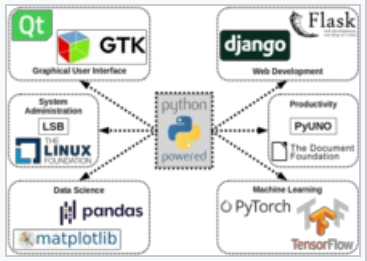
languages, such as C and Java, for programming problems involving string manipulation and search in a dictionary, and determined that memory consumption was often "better than Java and not much worse than C or C++".[181]

Large organizations that use Python include Wikipedia, Google,[182] Yahoo!,[183] CERN,[184] NASA,[185] Facebook,[186] Amazon, Instagram,[187] Spotify,[188] and some smaller entities like ILM[189] and ITA.[190] The social news networking site Reddit was written mostly in Python.[191]

Uses [edit]

Main article: List of Python software

Python can serve as a scripting language for web applications, e.g., via mod_wsgi for the Apache webserver.[192] With Web Server Gateway Interface, a standard API has evolved to facilitate these applications. Web frameworks like Django, Pylons, Pyramid, TurboGears, web2py, Tornado, Flask, Bottle, and Zope support developers in the design and maintenance of complex applications. Pyjs and IronPython can be used to develop the client-side of Ajax-based applications. SQLAlchemy can be used as a data mapper to a relational database. Twisted is a framework to program communications between computers, and is used (for example) by Dropbox.



Python Powered

Libraries such as NumPy, SciPy, and Matplotlib allow the effective use of Python in scientific computing,[193][194] with specialized libraries such as Biopython and Astropy providing domain-specific functionality. SageMath is a computer algebra system with a notebook interface programmable in Python: its library covers many aspects of mathematics, including algebra, combinatorics, numerical mathematics, number theory, and calculus.[195] OpenCV has Python bindings with a rich set of features for computer vision and image processing.[196]

Python is commonly used in artificial intelligence projects and machine learning projects with the help of libraries like TensorFlow, Keras, Pytorch, and Scikit-learn.[197][198][199][200] As a scripting language with a modular architecture, simple syntax, and rich text processing tools, Python is often used for natural language processing.[201]

Python can also be used to create games, with libraries such as Pygame, which can make 2D games.

Python has been successfully embedded in many software products as a scripting language, including in finite element method software such as Abaqus, 3D parametric modeler like FreeCAD, 3D animation packages such as 3ds Max, Blender, Cinema 4D, Lightwave, Houdini, Maya, modo, MotionBuilder, Softimage, the visual effects compositor Nuke, 2D imaging programs like GIMP,[202] Inkscape, Scribus and Paint Shop Pro,[203] and musical notation programs like scorewriter and capella. GNU Debugger uses Python as a pretty printer to show complex structures such as C++ containers. Esri promotes Python as the best choice for writing scripts in ArcGIS.[204] It has also been used in several video games,[205][206] and has been adopted as first of the three available programming languages in Google App Engine, the other two being Java and Go.[207]

Many operating systems include Python as a standard component. It ships with most Linux distributions,[208] AmigaOS 4 (using Python 2.7), FreeBSD (as a package), NetBSD, and OpenBSD (as a package) and can be used from the command line (terminal). Many Linux distributions use installers written in Python: Ubuntu uses the Ubiquity installer, while Red Hat Linux and Fedora Linux use the Anaconda installer. Gentoo Linux uses Python in its package management system, Portage.

Python is used extensively in the information security industry, including in exploit development.[209][210]

Most of the Sugar software for the One Laptop per Child XO, now[*when?*] developed at Sugar Labs, is written in Python.[211] The Raspberry Pi single-board computer project has adopted Python as its main user-programming language.

LibreOffice includes Python and intends to replace Java with Python. Its Python Scripting Provider is a core feature[212] since Version 4.0 from 7 February 2013.

Languages influenced by Python [edit]

Python's design and philosophy have influenced many other programming languages:

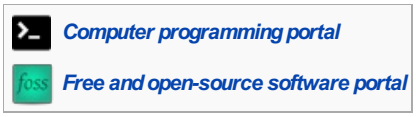
- Boo uses indentation, a similar syntax, and a similar object model.[213]
- Cobra uses indentation and a similar syntax, and its *Acknowledgements* document lists Python first among languages that influenced it.[214]
- CoffeeScript, a programming language that cross-compiles to JavaScript, has Python-inspired syntax.

- ECMAScript/JavaScript borrowed iterators and generators from Python.[215]
- GDScript, a scripting language very similar to Python, built-in to the Godot game engine.[216]
- Go is designed for the "speed of working in a dynamic language like Python"[217] and shares the same syntax for slicing arrays.
- Groovy was motivated by the desire to bring the Python design philosophy to Java.[218]
- Julia was designed to be "as usable for general programming as Python".[27]
- Nim uses indentation and similar syntax.[219]
- Ruby's creator, Yukihiro Matsumoto, has said: "I wanted a scripting language that was more powerful than Perl, and more object-oriented than Python. That's why I decided to design my own language."[220]
- Swift, a programming language developed by Apple, has some Python-inspired syntax.[221]

Python's development practices have also been emulated by other languages. For example, the practice of requiring a document describing the rationale for, and issues surrounding, a change to the language (in Python, a PEP) is also used in Tcl,[222] Erlang,[223] and Swift.[224]

See also [edit]

- Python syntax and semantics
- pip (package manager)
- List of programming languages
- History of programming languages
- Comparison of programming languages



References [edit]

1. ^ "General Python FAQ — Python 3.9.2 documentation". *docs.python.org*. Archived from the original on 24 October 2012. Retrieved 28 March 2021.
2. ^ "Python 0.9.1 part 01/21". alt.sources archives. Archived from the original on 11 August 2021. Retrieved 11 August 2021.
3. ^ "Python 3.10.6 is available". 2 August 2022. Retrieved 2 August 2022.
4. ^ "Why is Python a dynamic language and also a strongly typed language – Python Wiki". *wiki.python.org*. Archived from the original on 14 March 2021. Retrieved 27 January 2021.
5. ^ "PEP 483 – The Theory of Type Hints". *Python.org*. Archived from the original on 14 June 2020. Retrieved 14 June 2018.
6. ^ "test — Regression tests package for Python — Python 3.7.13 documentation". *docs.python.org*. Retrieved 17 May 2022.
7. ^ "platform — Access to underlying platform's identifying data — Python 3.10.4 documentation". *docs.python.org*. Retrieved 17 May 2022.
8. ^ "Download Python". *Python.org*. Archived from the original on 8 August 2018. Retrieved 24 May 2021.
9. ^ Holth, Moore (30 March 2014). "PEP 0441 – Improving Python ZIP Application Support". Archived from the original on 26 December 2018. Retrieved 12 November 2015.
10. ^ File extension .pyo was removed in Python 3.5. See PEP 0488 Archived 1 June 2020 at the Wayback Machine
11. ^ "Starlark Language". Archived from the original on 15 June 2020. Retrieved 25 May 2019.
12. ^ [a] [b] "Why was Python created in the first place?". *General Python FAQ*. Python Software Foundation. Archived from the original on 24 October 2012. Retrieved 22 March 2007.
13. ^ "Ada 83 Reference Manual (raise statement)". Archived from the original on 22 October 2019. Retrieved 7 January 2020.
14. ^ Kuchling, Andrew M. (22 December 2006). "Interview with Guido van Rossum (July 1998)". *amk.ca*. Archived from the original on 1 May 2007. Retrieved 12 March 2012.
15. ^ [a] [b] "itertools — Functions creating iterators for efficient looping — Python 3.7.1 documentation". *docs.python.org*. Archived from the original on 14 June 2020. Retrieved 22 November 2016.
16. ^ van Rossum, Guido (1993). "An Introduction to Python for UNIX/C Programmers". *Proceedings of the NLUUG Najaarsconferentie (Dutch UNIX Users Group)*. CiteSeerX 10.1.1.38.2023. "even though the design of C is far from ideal, its influence on Python is considerable."
17. ^ [a] [b] "Classes". *The Python Tutorial*. Python Software Foundation. Archived from the original on 23 October 2012. Retrieved 20 February 2012. "It is a mixture of the class mechanisms found in C++ and Modula-3"
18. ^ Lundh, Fredrik. "Call By Object". *effbot.org*. Archived from the original on 23 November 2019. Retrieved 21 November 2017. "replace "CLU" with "Python", "record" with "instance", and "procedure" with "function or method", and you get a pretty accurate description of Python's object model."
19. ^ Simionato, Michele. "The Python 2.3 Method Resolution Order". Python Software Foundation. Archived from the original on 20 August 2020. Retrieved 29 July 2014. "The C3 method itself has nothing to do with Python, since it was invented by people working on Dylan and it is described in a paper intended for lispers"

20. ^ Kuchling, A. M. "Functional Programming HOWTO". *Python v2.7.2 documentation*. Python Software Foundation. Archived from the original on 24 October 2012. Retrieved 9 February 2012.
21. ^ Schemenauer, Neil; Peters, Tim; Hetland, Magnus Lie (18 May 2001). "PEP 255 – Simple Generators". *Python Enhancement Proposals*. Python Software Foundation. Archived from the original on 5 June 2020. Retrieved 9 February 2012.
22. ^ "More Control Flow Tools". *Python 3 documentation*. Python Software Foundation. Archived from the original on 4 June 2016. Retrieved 24 July 2015.
23. ^ "CoffeeScript". *coffeescript.org*. Archived from the original on 12 June 2020. Retrieved 3 July 2018.
24. ^ "The Genie Programming Language Tutorial". Archived from the original on 1 June 2020. Retrieved 28 February 2020.
25. ^ "Perl and Python influences in JavaScript". *www.2ality.com*. 24 February 2013. Archived from the original on 26 December 2018. Retrieved 15 May 2015.
26. ^ Rauschmayer, Axel. "Chapter 3: The Nature of JavaScript; Influences". *O'Reilly, Speaking JavaScript*. Archived from the original on 26 December 2018. Retrieved 15 May 2015.
27. ^ [a] [b] "Why We Created Julia". *Julia website*. February 2012. Archived from the original on 2 May 2020. Retrieved 5 June 2014. "We want something as usable for general programming as Python [...]"
28. ^ Ring Team (4 December 2017). "Ring and other languages". *ring-lang.net*. ring-lang. Archived from the original on 25 December 2018. Retrieved 4 December 2017.
29. ^ Bini, Ola (2007). *Practical JRuby on Rails Web 2.0 Projects: bringing Ruby on Rails to the Java platform*. Berkeley: APress. p. 3. ISBN 978-1-59059-881-8.
30. ^ Lattner, Chris (3 June 2014). "Chris Lattner's Homepage". Chris Lattner. Archived from the original on 25 December 2018. Retrieved 3 June 2014. "The Swift language is the product of tireless effort from a team of language experts, documentation gurus, compiler optimization ninjas, and an incredibly important internal dogfooding group who provided feedback to help refine and battle-test ideas. Of course, it also greatly benefited from the experiences hard-won by many other languages in the field, drawing ideas from Objective-C, Rust, Haskell, Ruby, Python, C#, CLU, and far too many others to list."
31. ^ Kuhlman, Dave. "A Python Book: Beginning Python, Advanced Python, and Python Exercises". Section 1.1. Archived from the original (PDF) on 23 June 2012.
32. ^ "About Python". Python Software Foundation. Archived from the original on 20 April 2012. Retrieved 24 April 2012., second section "Fans of Python use the phrase "batteries included" to describe the standard library, which covers everything from asynchronous processing to zip files."
33. ^ "PEP 206 – Python Advanced Library". *Python.org*. Archived from the original on 5 May 2021. Retrieved 11 October 2021.
34. ^ Rossum, Guido Van (20 January 2009). "The History of Python: A Brief Timeline of Python". *The History of Python*. Archived from the original on 5 June 2020. Retrieved 5 March 2021.
35. ^ Peterson, Benjamin (20 April 2020). "Python Insider: Python 2.7.18, the last release of Python 2". *Python Insider*. Archived from the original on 26 April 2020. Retrieved 27 April 2020.
36. ^ "Stack Overflow Developer Survey 2020". *Stack Overflow*. Archived from the original on 2 March 2021. Retrieved 5 March 2021.
37. ^ "The State of Developer Ecosystem in 2020 Infographic". *JetBrains: Developer Tools for Professionals and Teams*. Archived from the original on 1 March 2021. Retrieved 5 March 2021.
38. ^ "index | TIOBE – The Software Quality Company". *www.tiobe.com*. Archived from the original on 25 February 2018. Retrieved 2 February 2021. "Python has won the TIOBE programming language of the year award! This is for the fourth time in the history, which is a record! The title is awarded to the programming language that has gained most popularity in one year."
39. ^ "PYPL PopularitY of Programming Language index". *pypl.github.io*. Archived from the original on 14 March 2017. Retrieved 26 March 2021.
40. ^ [a] [b] Venners, Bill (13 January 2003). "The Making of Python". *Artima Developer*. Artima. Archived from the original on 1 September 2016. Retrieved 22 March 2007.
41. ^ van Rossum, Guido (29 August 2000). "SETL (was: Lukewarm about range literals)". *Python-Dev* (Mailing list). Archived from the original on 14 July 2018. Retrieved 13 March 2011.
42. ^ van Rossum, Guido (20 January 2009). "A Brief Timeline of Python". *The History of Python*. Archived from the original on 5 June 2020. Retrieved 20 January 2009.
43. ^ Fairchild, Carlie (12 July 2018). "Guido van Rossum Stepping Down from Role as Python's Benevolent Dictator For Life". *Linux Journal*. Archived from the original on 13 July 2018. Retrieved 13 July 2018.
44. ^ "PEP 8100". Python Software Foundation. Archived from the original on 4 June 2020. Retrieved 4 May 2019.
45. ^ "PEP 13 – Python Language Governance". *Python.org*. Archived from the original on 27 May 2021. Retrieved 25 August 2021.
46. ^ Kuchling, A. M.; Zadka, Moshe (16 October 2000). "What's New in Python 2.0". Python Software Foundation. Archived from the original on 23 October 2012. Retrieved 11 February 2012.
47. ^ van Rossum, Guido (5 April 2006). "PEP 3000 – Python 3000". *Python Enhancement Proposals*. Python Software Foundation. Archived from the original on 3 March 2016. Retrieved 27 June 2009.
48. ^ "2to3 – Automated Python 2 to 3 code translation". *docs.python.org*. Archived from the original on 4 June 2020. Retrieved 2 February 2021.

49. ^ "PEP 373 – Python 2.7 Release Schedule". *python.org*. Archived from the original on 19 May 2020. Retrieved 9 January 2017.
50. ^ "PEP 466 – Network Security Enhancements for Python 2.7.x". *python.org*. Archived from the original on 4 June 2020. Retrieved 9 January 2017.
51. ^ "Sunsetting Python 2". *Python.org*. Archived from the original on 12 January 2020. Retrieved 22 September 2019.
52. ^ "PEP 373 – Python 2.7 Release Schedule". *Python.org*. Archived from the original on 13 January 2020. Retrieved 22 September 2019.
53. ^ "Python Developer's Guide — Python Developer's Guide". *devguide.python.org*. Archived from the original on 9 November 2020. Retrieved 17 December 2019.
54. ^ Langa, Łukasz (19 February 2021). "Python Insider: Python 3.9.2 and 3.8.8 are now available". *Python Insider*. Archived from the original on 25 February 2021. Retrieved 26 February 2021.
55. ^ "Red Hat Customer Portal – Access to 24x7 support and knowledge". *access.redhat.com*. Archived from the original on 6 March 2021. Retrieved 26 February 2021.
56. ^ "CVE – CVE-2021-3177". *cve.mitre.org*. Archived from the original on 27 February 2021. Retrieved 26 February 2021.
57. ^ "CVE – CVE-2021-23336". *cve.mitre.org*. Archived from the original on 24 February 2021. Retrieved 26 February 2021.
58. ^ Langa, Łukasz (24 March 2022). "Python Insider: Python 3.10.4 and 3.9.12 are now available out of schedule". *Python Insider*. Retrieved 19 April 2022.
59. ^ Langa, Łukasz (16 March 2022). "Python Insider: Python 3.10.3, 3.9.11, 3.8.13, and 3.7.13 are now available with security content". *Python Insider*. Retrieved 19 April 2022.
60. ^ Langa, Łukasz (17 May 2022). "Python Insider: Python 3.9.13 is now available". *Python Insider*. Retrieved 21 May 2022.
61. ^ The Cain Gang Ltd. "Python Metaclasses: Who? Why? When?" (PDF). Archived from the original (PDF) on 30 May 2009. Retrieved 27 June 2009.
62. ^ "3.3. Special method names". *The Python Language Reference*. Python Software Foundation. Archived from the original on 15 December 2018. Retrieved 27 June 2009.
63. ^ "PyDBC: method preconditions, method postconditions and class invariants for Python". Archived from the original on 23 November 2019. Retrieved 24 September 2011.
64. ^ "Contracts for Python". Archived from the original on 15 June 2020. Retrieved 24 September 2011.
65. ^ "PyDatalog". Archived from the original on 13 June 2020. Retrieved 22 July 2012.
66. ^ "Extending and Embedding the Python Interpreter: Reference Counts". Docs.python.org. Archived from the original on 18 October 2012. Retrieved 5 June 2020. "Since Python makes heavy use of `malloc()` and `free()`, it needs a strategy to avoid memory leaks as well as the use of freed memory. The chosen method is called *reference counting*."
67. ^ [a] [b] Hettinger, Raymond (30 January 2002). "PEP 289 – Generator Expressions". *Python Enhancement Proposals*. Python Software Foundation. Archived from the original on 14 June 2020. Retrieved 19 February 2012.
68. ^ "6.5 itertools – Functions creating iterators for efficient looping". Docs.python.org. Archived from the original on 14 June 2020. Retrieved 22 November 2016.
69. ^ [a] [b] Peters, Tim (19 August 2004). "PEP 20 – The Zen of Python". *Python Enhancement Proposals*. Python Software Foundation. Archived from the original on 26 December 2018. Retrieved 24 November 2008.
70. ^ Martelli, Alex; Ravenscroft, Anna; Ascher, David (2005). *Python Cookbook, 2nd Edition*. O'Reilly Media. p. 230. ISBN 978-0-596-00797-3. Archived from the original on 23 February 2020. Retrieved 14 November 2015.
71. ^ "Python Culture". *ebeab*. 21 January 2014. Archived from the original on 30 January 2014.
72. ^ [a] [b] "General Python FAQ". *Python v2.7.3 documentation*. Docs.python.org. Archived from the original on 24 October 2012. Retrieved 4 June 2020.
73. ^ "15 Ways Python Is a Powerful Force on the Web". Archived from the original on 11 May 2019. Retrieved 3 July 2018.
74. ^ "8.18. pprint — Data pretty printer — Python 3.8.3 documentation". *docs.python.org*. Archived from the original on 22 January 2021. Retrieved 4 June 2020.
75. ^ Clark, Robert (26 April 2019). "How to be Pythonic and why you should care". *Medium*. Archived from the original on 13 August 2021. Retrieved 20 January 2021.
76. ^ "Code Style — The Hitchhiker's Guide to Python". *docs.python-guide.org*. Archived from the original on 27 January 2021. Retrieved 20 January 2021.
77. ^ Goodger, David. "Code Like a Pythonista: Idiomatic Python". Archived from the original on 27 May 2014. Retrieved 24 March 2009.
78. ^ "How to think like a Pythonista". Archived from the original on 23 March 2018. Retrieved 24 March 2009.
79. ^ "Is Python a good language for beginning programmers?". *General Python FAQ*. Python Software Foundation. Archived from the original on 24 October 2012. Retrieved 21 March 2007.
80. ^ "Myths about indentation in Python". Secnetix.de. Archived from the original on 18 February 2018. Retrieved 19 April 2011.

81. ^ Guttag, John V. (12 August 2016). *Introduction to Computation and Programming Using Python: With Application to Understanding Data*. MIT Press. ISBN 978-0-262-52962-4.
82. ^ "PEP 8 – Style Guide for Python Code". *Python.org*. Archived from the original on 17 April 2019. Retrieved 26 March 2019.
83. ^ "8. Errors and Exceptions — Python 3.12.0a0 documentation". *docs.python.org*. Retrieved 9 May 2022.
84. ^ "Highlights: Python 2.5". *Python.org*. Archived from the original on 4 August 2019. Retrieved 20 March 2018.
85. ^ van Rossum, Guido (22 April 2009). "Tail Recursion Elimination". Neopythonic.blogspot.be. Archived from the original on 19 May 2018. Retrieved 3 December 2012.
86. ^ van Rossum, Guido (9 February 2006). "Language Design Is Not Just Solving Puzzles". *Artima forums*. Artima. Archived from the original on 17 January 2020. Retrieved 21 March 2007.
87. ^ van Rossum, Guido; Eby, Phillip J. (10 May 2005). "PEP 342 – Coroutines via Enhanced Generators". *Python Enhancement Proposals*. Python Software Foundation. Archived from the original on 29 May 2020. Retrieved 19 February 2012.
88. ^ "PEP 380". Python.org. Archived from the original on 4 June 2020. Retrieved 3 December 2012.
89. ^ "division". *python.org*. Archived from the original on 20 July 2006. Retrieved 30 July 2014.
90. ^ "PEP 0465 – A dedicated infix operator for matrix multiplication". *python.org*. Archived from the original on 4 June 2020. Retrieved 1 January 2016.
91. ^ "Python 3.5.1 Release and Changelog". *python.org*. Archived from the original on 14 May 2020. Retrieved 1 January 2016.
92. ^ "What's New in Python 3.8". Archived from the original on 8 June 2020. Retrieved 14 October 2019.
93. ^ "Chapter 15. Expressions – 15.21.1. Numerical Equality Operators == and !=". Oracle Corporation. Archived from the original on 7 June 2020. Retrieved 28 August 2016.
94. ^ "Chapter 15. Expressions – 15.21.3. Reference Equality Operators == and !=". Oracle Corporation. Archived from the original on 7 June 2020. Retrieved 28 August 2016.
95. ^ van Rossum, Guido; Hettinger, Raymond (7 February 2003). "PEP 308 – Conditional Expressions". *Python Enhancement Proposals*. Python Software Foundation. Archived from the original on 13 March 2016. Retrieved 13 July 2011.
96. ^ "4. Built-in Types — Python 3.6.3rc1 documentation". *python.org*. Archived from the original on 14 June 2020. Retrieved 1 October 2017.
97. ^ "5.3. Tuples and Sequences — Python 3.7.1rc2 documentation". *python.org*. Archived from the original on 10 June 2020. Retrieved 17 October 2018.
98. ^ [a] [b] "PEP 498 – Literal String Interpolation". *python.org*. Archived from the original on 15 June 2020. Retrieved 8 March 2017.
99. ^ "Why must 'self' be used explicitly in method definitions and calls?". *Design and History FAQ*. Python Software Foundation. Archived from the original on 24 October 2012. Retrieved 19 February 2012.
100. ^ Sweigart, Al (2020). *Beyond the Basic Stuff with Python: Best Practices for Writing Clean Code*. No Starch Press. p. 322. ISBN 978-1-59327-966-0. Archived from the original on 13 August 2021. Retrieved 7 July 2021.
101. ^ "The Python Language Reference, section 3.3. New-style and classic classes, for release 2.7.1". Archived from the original on 26 October 2012. Retrieved 12 January 2011.
102. ^ "Type hinting for Python". LWN.net. 24 December 2014. Archived from the original on 20 June 2019. Retrieved 5 May 2015.
103. ^ "mypy – Optional Static Typing for Python". Archived from the original on 6 June 2020. Retrieved 28 January 2017.
104. ^ "15. Floating Point Arithmetic: Issues and Limitations — Python 3.8.3 documentation". *docs.python.org*. Archived from the original on 6 June 2020. Retrieved 6 June 2020. "Almost all machines today (November 2000) use IEEE-754 floating point arithmetic, and almost all platforms map Python floats to IEEE-754 “double precision”."
105. ^ Zadka, Moshe; van Rossum, Guido (11 March 2001). "PEP 237 – Unifying Long Integers and Integers". *Python Enhancement Proposals*. Python Software Foundation. Archived from the original on 28 May 2020. Retrieved 24 September 2011.
106. ^ "Built-in Types". Archived from the original on 14 June 2020. Retrieved 3 October 2019.
107. ^ "PEP 465 – A dedicated infix operator for matrix multiplication". *python.org*. Archived from the original on 29 May 2020. Retrieved 3 July 2018.
108. ^ [a] [b] Zadka, Moshe; van Rossum, Guido (11 March 2001). "PEP 238 – Changing the Division Operator". *Python Enhancement Proposals*. Python Software Foundation. Archived from the original on 28 May 2020. Retrieved 23 October 2013.
109. ^ "Why Python's Integer Division Floors". 24 August 2010. Archived from the original on 5 June 2020. Retrieved 25 August 2010.
110. ^ "round", *The Python standard library, release 3.2, §2: Built-in functions*, archived from the original on 25 October 2012, retrieved 14 August 2011
111. ^ "round", *The Python standard library, release 2.7, §2: Built-in functions*, archived from the original on 27 October 2012, retrieved 14 August 2011
112. ^ Beazley, David M. (2009). *Python Essential Reference* (4th ed.). p. 66. ISBN 9780672329784.
113. ^ Kernighan, Brian W.; Ritchie, Dennis M. (1988). *The C Programming Language* (2nd ed.). p. 206.

114. ^ Batista, Facundo. "PEP 0327 – Decimal Data Type". *Python.org*. Archived from the original on 4 June 2020. Retrieved 26 September 2015.
115. ^ "What's New in Python 2.6 — Python v2.6.9 documentation". *docs.python.org*. Archived from the original on 23 December 2019. Retrieved 26 September 2015.
116. ^ "10 Reasons Python Rocks for Research (And a Few Reasons it Doesn't) – Hoyt Koepke". *www.stat.washington.edu*. Archived from the original on 31 May 2020. Retrieved 3 February 2019.
117. ^ Shell, Scott (17 June 2014). "An introduction to Python for scientific computing" (PDF). Archived (PDF) from the original on 4 February 2019. Retrieved 3 February 2019.
118. ^ Piotrowski, Przemyslaw (July 2006). "Build a Rapid Web Development Environment for Python Server Pages and Oracle". *Oracle Technology Network*. Oracle. Archived from the original on 2 April 2019. Retrieved 12 March 2012.
119. ^ Batista, Facundo (17 October 2003). "PEP 327 – Decimal Data Type". *Python Enhancement Proposals*. Python Software Foundation. Archived from the original on 4 June 2020. Retrieved 24 November 2008.
120. ^ Eby, Phillip J. (7 December 2003). "PEP 333 – Python Web Server Gateway Interface v1.0". *Python Enhancement Proposals*. Python Software Foundation. Archived from the original on 14 June 2020. Retrieved 19 February 2012.
121. ^ "Modulecounts". *Modulecounts*. 26 June 2022. Archived from the original on 26 June 2022. Retrieved 26 June 2022.
122. ^ Enthought, Canopy. "Canopy". *www.enthought.com*. Archived from the original on 15 July 2017. Retrieved 20 August 2016.
123. ^ "PEP 7 – Style Guide for C Code | peps.python.org". *peps.python.org*. Retrieved 28 April 2022.
124. ^ "Mailman 3 Why aren't we allowing the use of C11? - Python-Dev - python.org". *mail.python.org*. Archived from the original on 14 April 2021. Retrieved 1 March 2021.
125. ^ "Issue 35473: Intel compiler (icc) does not fully support C11 Features, including atomics – Python tracker". *bugs.python.org*. Archived from the original on 14 April 2021. Retrieved 1 March 2021.
126. ^ "4. Building C and C++ Extensions — Python 3.9.2 documentation". *docs.python.org*. Archived from the original on 3 March 2021. Retrieved 1 March 2021.
127. ^ van Rossum, Guido (5 June 2001). "PEP 7 – Style Guide for C Code". *Python Enhancement Proposals*. Python Software Foundation. Archived from the original on 1 June 2020. Retrieved 24 November 2008.
128. ^ "CPython byte code". Docs.python.org. Archived from the original on 5 June 2020. Retrieved 16 February 2016.
129. ^ "Python 2.5 internals" (PDF). Archived (PDF) from the original on 6 August 2012. Retrieved 19 April 2011.
130. ^ "Changelog — Python 3.9.0 documentation". *docs.python.org*. Archived from the original on 7 February 2021. Retrieved 8 February 2021.
131. ^ "Download Python". *Python.org*. Archived from the original on 8 December 2020. Retrieved 13 December 2020.
132. ^ "history [vmspython]". *www.vmspython.org*. Archived from the original on 2 December 2020. Retrieved 4 December 2020.
133. ^ "An Interview with Guido van Rossum". Oreilly.com. Archived from the original on 16 July 2014. Retrieved 24 November 2008.
134. ^ "Download Python for Other Platforms". *Python.org*. Archived from the original on 27 November 2020. Retrieved 4 December 2020.
135. ^ "PyPy compatibility". Pypy.org. Archived from the original on 6 June 2020. Retrieved 3 December 2012.
136. ^ Team, The PyPy (28 December 2019). "Download and Install". *PyPy*. Retrieved 8 January 2022.
137. ^ "speed comparison between CPython and Pypy". Speed.pypy.org. Archived from the original on 10 May 2021. Retrieved 3 December 2012.
138. ^ "Application-level Stackless features — PyPy 2.0.2 documentation". Doc.pypy.org. Archived from the original on 4 June 2020. Retrieved 17 July 2013.
139. ^ "Python-for-EV3". *LEGO Education*. Archived from the original on 7 June 2020. Retrieved 17 April 2019.
140. ^ Yegulalp, Serdar (29 October 2020). "Pyston returns from the dead to speed Python". *InfoWorld*. Archived from the original on 27 January 2021. Retrieved 26 January 2021.
141. ^ "cinder: Instagram's performance oriented fork of CPython". *GitHub*. Archived from the original on 4 May 2021. Retrieved 4 May 2021.
142. ^ "Plans for optimizing Python". *Google Project Hosting*. 15 December 2009. Archived from the original on 11 April 2016. Retrieved 24 September 2011.
143. ^ "Python on the Nokia N900". *Stochastic Geometry*. 29 April 2010. Archived from the original on 20 June 2019. Retrieved 9 July 2015.
144. ^ "Brython". *brython.info*. Archived from the original on 3 August 2018. Retrieved 21 January 2021.
145. ^ "Transcrypt – Python in the browser". *transcrypt.org*. Archived from the original on 19 August 2018. Retrieved 22 December 2020.
146. ^ "Transcrypt: Anatomy of a Python to JavaScript Compiler". *InfoQ*. Archived from the original on 5 December 2020. Retrieved 20 January 2021.

147. ^ "Nuitka Home | Nuitka Home". *nuitka.net*. Archived from the original on 30 May 2020. Retrieved 18 August 2017.
148. ^ Borderies, Olivier (24 January 2019). "Pythran: Python at C++ speed !". *Medium*. Archived from the original on 25 March 2020. Retrieved 25 March 2020.
149. ^ "Pythran — Pythran 0.9.5 documentation". *pythran.readthedocs.io*. Archived from the original on 19 February 2020. Retrieved 25 March 2020.
150. ^ "Archived copy" (PDF). Archived (PDF) from the original on 18 April 2021. Retrieved 20 January 2021.
151. ^ The Python → 11l → C++ transpiler
152. ^ "google/grumpy". 10 April 2020. Archived from the original on 15 April 2020. Retrieved 25 March 2020 – via GitHub.
153. ^ "Projects". *opensource.google*. Archived from the original on 24 April 2020. Retrieved 25 March 2020.
154. ^ Francisco, Thomas Claburn in San. "Google's Grumpy code makes Python Go". *www.theregister.com*. Archived from the original on 7 March 2021. Retrieved 20 January 2021.
155. ^ "GitHub – IronLanguages/ironpython3: Implementation of Python 3.x for .NET Framework that is built on top of the Dynamic Language Runtime". *GitHub*. Archived from the original on 28 September 2021.
156. ^ "IronPython.net /". *ironpython.net*. Archived from the original on 17 April 2021.
157. ^ "Jython FAQ". *www.jython.org*. Archived from the original on 22 April 2021. Retrieved 22 April 2021.
158. ^ Murri, Riccardo (2013). *Performance of Python runtimes on a non-numeric scientific code*. European Conference on Python in Science (EuroSciPy). arXiv:1404.6388. Bibcode:2014arXiv1404.6388M.
159. ^ "The Computer Language Benchmarks Game". Archived from the original on 14 June 2020. Retrieved 30 April 2020.
160. ^ [a] [b] Warsaw, Barry; Hylton, Jeremy; Goodger, David (13 June 2000). "PEP 1 – PEP Purpose and Guidelines". *Python Enhancement Proposals*. Python Software Foundation. Archived from the original on 6 June 2020. Retrieved 19 April 2011.
161. ^ "PEP 8 – Style Guide for Python Code". *Python.org*. Archived from the original on 17 April 2019. Retrieved 26 March 2019.
162. ^ Cannon, Brett. "Guido, Some Guys, and a Mailing List: How Python is Developed". *python.org*. Python Software Foundation. Archived from the original on 1 June 2009. Retrieved 27 June 2009.
163. ^ "Python Developer's Guide — Python Developer's Guide". *devguide.python.org*. Archived from the original on 9 November 2020. Retrieved 17 December 2019.
164. ^ Hughes, Owen (24 May 2021). "Programming languages: Why Python 4.0 might never arrive, according to its creator". *TechRepublic*. Retrieved 16 May 2022.
165. ^ "PEP 602 – Annual Release Cycle for Python". *Python.org*. Archived from the original on 14 June 2020. Retrieved 6 November 2019.
166. ^ "Changing the Python release cadence [LWN.net]". *lwn.net*. Archived from the original on 6 November 2019. Retrieved 6 November 2019.
167. ^ Norwitz, Neal (8 April 2002). "[Python-Dev] Release Schedules (was Stability & change)". Archived from the original on 15 December 2018. Retrieved 27 June 2009.
168. ^ [a] [b] Aahz; Baxter, Anthony (15 March 2001). "PEP 6 – Bug Fix Releases". *Python Enhancement Proposals*. Python Software Foundation. Archived from the original on 5 June 2020. Retrieved 27 June 2009.
169. ^ "Python Buildbot". *Python Developer's Guide*. Python Software Foundation. Archived from the original on 5 June 2020. Retrieved 24 September 2011.
170. ^ "1. Extending Python with C or C++ — Python 3.9.1 documentation". *docs.python.org*. Archived from the original on 23 June 2020. Retrieved 14 February 2021.
171. ^ "PEP 623 – Remove wstr from Unicode". *Python.org*. Archived from the original on 5 March 2021. Retrieved 14 February 2021.
172. ^ "PEP 634 – Structural Pattern Matching: Specification". *Python.org*. Archived from the original on 6 May 2021. Retrieved 14 February 2021.
173. ^ "Documentation Tools". *Python.org*. Archived from the original on 11 November 2020. Retrieved 22 March 2021.
174. ^ [a] [b] "Whetting Your Appetite". *The Python Tutorial*. Python Software Foundation. Archived from the original on 26 October 2012. Retrieved 20 February 2012.
175. ^ "In Python, should I use else after a return in an if block?". *Stack Overflow*. Stack Exchange. 17 February 2011. Archived from the original on 20 June 2019. Retrieved 6 May 2011.
176. ^ Lutz, Mark (2009). *Learning Python: Powerful Object-Oriented Programming*. O'Reilly Media, Inc. p. 17. ISBN 9781449379322. Archived from the original on 17 July 2017. Retrieved 9 May 2017.
177. ^ Fehily, Chris (2002). *Python*. Peachpit Press. p. xv. ISBN 9780201748840. Archived from the original on 17 July 2017. Retrieved 9 May 2017.
178. ^ "TIOBE Index". TIOBE – The Software Quality Company. Archived from the original on 12 October 2021. Retrieved 13 October 2021.
179. ^ Blake, Troy (18 January 2021). "TIOBE Index for January 2021". *Technology News and Information by SeniorDBA*. Archived from the original on 21 March 2021. Retrieved 26 February 2021.

180. ^ TIOBE Software Index (2015). "TIOBE Programming Community Index Python". Archived from the original on 7 September 2015. Retrieved 10 September 2015.
181. ^ Prechelt, Lutz (14 March 2000). "An empirical comparison of C, C++, Java, Perl, Python, Rexx, and Tcl" (PDF). Archived (PDF) from the original on 3 January 2020. Retrieved 30 August 2013.
182. ^ "Quotes about Python". Python Software Foundation. Archived from the original on 3 June 2020. Retrieved 8 January 2012.
183. ^ "Organizations Using Python". Python Software Foundation. Archived from the original on 21 August 2018. Retrieved 15 January 2009.
184. ^ "Python : the holy grail of programming". *CERN Bulletin*. CERN Publications (31/2006). 31 July 2006. Archived from the original on 15 January 2013. Retrieved 11 February 2012.
185. ^ Shafer, Daniel G. (17 January 2003). "Python Streamlines Space Shuttle Mission Design". Python Software Foundation. Archived from the original on 5 June 2020. Retrieved 24 November 2008.
186. ^ "Tornado: Facebook's Real-Time Web Framework for Python – Facebook for Developers". *Facebook for Developers*. Archived from the original on 19 February 2019. Retrieved 19 June 2018.
187. ^ "What Powers Instagram: Hundreds of Instances, Dozens of Technologies". Instagram Engineering. 11 December 2016. Archived from the original on 15 June 2020. Retrieved 27 May 2019.
188. ^ "How we use Python at Spotify". *Spotify Labs*. 20 March 2013. Archived from the original on 10 June 2020. Retrieved 25 July 2018.
189. ^ Fortenberry, Tim (17 January 2003). "Industrial Light & Magic Runs on Python". Python Software Foundation. Archived from the original on 6 June 2020. Retrieved 11 February 2012.
190. ^ Taft, Darryl K. (5 March 2007). "Python Slithers into Systems". *eWeek.com*. Ziff Davis Holdings. Archived from the original on 13 August 2021. Retrieved 24 September 2011.
191. ^ *GitHub – reddit-archive/reddit: historical code from reddit.com.*, The Reddit Archives, archived from the original on 1 June 2020, retrieved 20 March 2019
192. ^ "Usage statistics and market share of Python for websites". 2012. Archived from the original on 13 August 2021. Retrieved 18 December 2012.
193. ^ Oliphant, Travis (2007). "Python for Scientific Computing". *Computing in Science and Engineering*. **9** (3): 10–20. Bibcode:2007CSE.....9c..10O. CiteSeerX 10.1.1.474.6460. doi:10.1109/MCSE.2007.58. S2CID 206457124. Archived from the original on 15 June 2020. Retrieved 10 April 2015.
194. ^ Millman, K. Jarrod; Aivazis, Michael (2011). "Python for Scientists and Engineers". *Computing in Science and Engineering*. **13** (2): 9–12. Bibcode:2011CSE....13b...9M. doi:10.1109/MCSE.2011.36. Archived from the original on 19 February 2019. Retrieved 7 July 2014.
195. ^ *Science education with SageMath*, Innovative Computing in Science Education, archived from the original on 15 June 2020, retrieved 22 April 2019
196. ^ "OpenCV: OpenCV-Python Tutorials". *docs.opencv.org*. Archived from the original on 23 September 2020. Retrieved 14 September 2020.
197. ^ Dean, Jeff; Monga, Rajat; et al. (9 November 2015). "TensorFlow: Large-scale machine learning on heterogeneous systems" (PDF). *TensorFlow.org*. Google Research. Archived (PDF) from the original on 20 November 2015. Retrieved 10 November 2015.
198. ^ Piatetsky, Gregory. "Python eats away at R: Top Software for Analytics, Data Science, Machine Learning in 2018: Trends and Analysis". *KDnuggets*. KDnuggets. Archived from the original on 15 November 2019. Retrieved 30 May 2018.
199. ^ "Who is using scikit-learn? — scikit-learn 0.20.1 documentation". *scikit-learn.org*. Archived from the original on 6 May 2020. Retrieved 30 November 2018.
200. ^ Jouppi, Norm. "Google supercharges machine learning tasks with TPU custom chip". *Google Cloud Platform Blog*. Archived from the original on 18 May 2016. Retrieved 19 May 2016.
201. ^ "Natural Language Toolkit — NLTK 3.5b1 documentation". *www.nltk.org*. Archived from the original on 13 June 2020. Retrieved 10 April 2020.
202. ^ "Installers for GIMP for Windows – Frequently Asked Questions". 26 July 2013. Archived from the original on 17 July 2013. Retrieved 26 July 2013.
203. ^ "jasc psp9components". Archived from the original on 19 March 2008.
204. ^ "About getting started with writing geoprocessing scripts". *ArcGIS Desktop Help 9.2*. Environmental Systems Research Institute. 17 November 2006. Archived from the original on 5 June 2020. Retrieved 11 February 2012.
205. ^ CCP porkbelly (24 August 2010). "Stackless Python 2.7". *EVE Community Dev Blogs*. CCP Games. Archived from the original on 11 January 2014. Retrieved 11 January 2014. "As you may know, EVE has at its core the programming language known as Stackless Python."
206. ^ Caudill, Barry (20 September 2005). "Modding Sid Meier's Civilization IV". *Sid Meier's Civilization IV Developer Blog*. Firaxis Games. Archived from the original on 2 December 2010. "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."
207. ^ "Python Language Guide (v1.0)". *Google Documents List Data API v1.0*. Archived from the original on 15 July 2010.
208. ^ "Python Setup and Usage". Python Software Foundation. Archived from the original on 17 June 2020. Retrieved 10 January 2020.

209. ^ "Immunity: Knowing You're Secure". Archived from the original on 16 February 2009.
210. ^ "Core Security". *Core Security*. Archived from the original on 9 June 2020. Retrieved 10 April 2020.
211. ^ "What is Sugar?". Sugar Labs. Archived from the original on 9 January 2009. Retrieved 11 February 2012.
212. ^ "4.0 New Features and Fixes". *LibreOffice.org*. The Document Foundation. 2013. Archived from the original on 9 February 2014. Retrieved 25 February 2013.
213. ^ "Gotchas for Python Users". *boo.codehaus.org*. Codehaus Foundation. Archived from the original on 11 December 2008. Retrieved 24 November 2008.
214. ^ Esterbrook, Charles. "Acknowledgements". *cobra-language.com*. Cobra Language. Archived from the original on 8 February 2008. Retrieved 7 April 2010.
215. ^ "Proposals: iterators and generators [ES4 Wiki]". wiki.ecmascript.org. Archived from the original on 20 October 2007. Retrieved 24 November 2008.
216. ^ "Frequently asked questions". *Godot Engine documentation*. Archived from the original on 28 April 2021. Retrieved 10 May 2021.
217. ^ Kincaid, Jason (10 November 2009). "Google's Go: A New Programming Language That's Python Meets C++". *TechCrunch*. Archived from the original on 18 January 2010. Retrieved 29 January 2010.
218. ^ Strachan, James (29 August 2003). "Groovy – the birth of a new dynamic language for the Java platform". Archived from the original on 5 April 2007. Retrieved 11 June 2007.
219. ^ Yegulalp, Serdar (16 January 2017). "Nim language draws from best of Python, Rust, Go, and Lisp". *InfoWorld*. Archived from the original on 13 October 2018. Retrieved 7 June 2020. "Nim's syntax is strongly reminiscent of Python's, as it uses indented code blocks and some of the same syntax (such as the way if/elif/then/else blocks are constructed)."
220. ^ "An Interview with the Creator of Ruby". Linuxdevcenter.com. Archived from the original on 28 April 2018. Retrieved 3 December 2012.
221. ^ Lattner, Chris (3 June 2014). "Chris Lattner's Homepage". Chris Lattner. Archived from the original on 22 December 2015. Retrieved 3 June 2014. "I started work on the Swift Programming Language in July of 2010. I implemented much of the basic language structure, with only a few people knowing of its existence. A few other (amazing) people started contributing in earnest late in 2011, and it became a major focus for the Apple Developer Tools group in July 2013 [...] drawing ideas from Objective-C, Rust, Haskell, Ruby, Python, C#, CLU, and far too many others to list."
222. ^ Kupries, Andreas; Fellows, Donal K. (14 September 2000). "TIP #3: TIP Format". *tcl.tk*. Tcl Developer Xchange. Archived from the original on 13 July 2017. Retrieved 24 November 2008.
223. ^ Gustafsson, Per; Niskanen, Raimo (29 January 2007). "EEP 1: EEP Purpose and Guidelines". erlang.org. Archived from the original on 15 June 2020. Retrieved 19 April 2011.
224. ^ "Swift Evolution Process". *Swift Programming Language Evolution repository on GitHub*. 18 February 2020. Archived from the original on 27 April 2020. Retrieved 27 April 2020.

Sources [edit]

- "Python for Artificial Intelligence". Wiki.python.org. 19 July 2012. Archived from the original on 1 November 2012. Retrieved 3 December 2012.
- Paine, Jocelyn, ed. (August 2005). "AI in Python". *AI Expert Newsletter*. Amzi!. Archived from the original on 26 March 2012. Retrieved 11 February 2012.
- "PyAIML 0.8.5 : Python Package Index". Pypi.python.org. Retrieved 17 July 2013.
- Russell, Stuart J. & Norvig, Peter (2009). *Artificial Intelligence: A Modern Approach* (3rd ed.). Upper Saddle River, NJ: Prentice Hall. ISBN 978-0-13-604259-4.

Further reading [edit]

- Downey, Allen B. (May 2012). *Think Python: How to Think Like a Computer Scientist* (Version 1.6.6 ed.). ISBN 978-0-521-72596-5.
- Hamilton, Naomi (5 August 2008). "The A-Z of Programming Languages: Python". *Computerworld*. Archived from the original on 29 December 2008. Retrieved 31 March 2010.
- Lutz, Mark (2013). *Learning Python* (5th ed.). O'Reilly Media. ISBN 978-0-596-15806-4.
- Pilgrim, Mark (2004). *Dive into Python*. Apress. ISBN 978-1-59059-356-1.
- Pilgrim, Mark (2009). *Dive into Python 3*. Apress. ISBN 978-1-4302-2415-0.
- Summerfield, Mark (2009). *Programming in Python 3* (2nd ed.). Addison-Wesley Professional. ISBN 978-0-321-68056-3.

External links [edit]

Python
at Wikipedia's sister projects

Media from Commons

Quotations from Wikiquote

Textbooks from Wikibooks

Resources from Wikiversity

Data from Wikidata

- Official website

v · t · e	**Python**
Implementations	CircuitPython · CLPython · CPython · Cython · MicroPython · Numba · IronPython · Jython · Psyco · PyPy · Python for S60 · Shed Skin · Stackless Python · Unladen Swallow · *more...*
IDE	Boa · Eric Python IDE · IDLE · PyCharm · PyDev · Ninja-IDE · *more...*
Topics	WSGI · ASGI
software (list) · Python Software Foundation · PyCon	

v · t · e	**Programming languages**
Comparison · Timeline · History	
Ada · ALGOL · APL · Assembly · BASIC · C · C++ · C# · COBOL · Erlang · Forth · Fortran · Go · Haskell · Java · JavaScript · Kotlin · Lisp · Lua · ML · Pascal · Perl · PHP · Prolog · **Python** · R · Ruby · Rust · SQL · Shell · Simula · Smalltalk · Swift · *more...*	
Category · **Lists:** Alphabetical · Categorical · Generational · Non-English-based	

v · t · e	**Python web frameworks**
Bottle · CherryPy · CubicWeb · Django · FastAPI · Flask · Grok · Nagare · Nevow · Pylons · Pyramid · Quixote · TACTIC · Tornado · TurboGears · TwistedWeb · web2py · Zope 2 · *more...*	

v · t · e	**Differentiable computing**	
General	**Differentiable programming** · Information geometry · Statistical manifold · Neural Turing machine · Differentiable neural computer · Automatic differentiation · Neuromorphic engineering · Cable theory · Pattern recognition · Tensor calculus · Computational learning theory	
Concepts	Gradient descent (SGD) · Clustering · Regression (Overfitting) · Adversary · Attention · Convolution · Loss functions · Backpropagation · Normalization · Activation (Softmax · Sigmoid · Rectifier) · Regularization · Datasets (Augmentation) · Diffusion · Autoregression	
Programming languages	**Python** · Julia	
Application	Machine learning · Artificial neural network (Deep learning) · Scientific computing · Artificial Intelligence	
Hardware	IPU · TPU · VPU · Memristor · SpiNNaker	
Software library	TensorFlow · PyTorch · Keras · Theano	
Implementation	**Audio–visual**	AlexNet · WaveNet · Human image synthesis · HWR · OCR · Speech synthesis · Speech recognition · Facial recognition · AlphaFold · DALL-E
	Verbal	Word2vec · Transformer · BERT · LaMDA · NMT · Project Debater · IBM Watson · GPT-2 · GPT-3
	Decisional	AlphaGo · AlphaZero · Q-learning · SARSA · OpenAI Five · Self-driving car · MuZero · Action selection · Robot control
People	Alex Graves · Ian Goodfellow · Yoshua Bengio · Geoffrey Hinton · Yann LeCun · Andrew Ng · Demis Hassabis · David Silver · Fei-Fei Li	
Organizations	DeepMind · OpenAI · MIT CSAIL · Mila · Google Brain · Meta AI	
Architectures	Recurrent neural network (RNN) · Long short-term memory (LSTM) · Gated recurrent unit (GRU) · Echo state network · Multilayer perceptron (MLP) · Autoencoder · Variational autoencoder (VAE) · Generative adversarial network (GAN)	
Portals (Computer programming · Technology) · Category (Artificial neural networks · Machine learning)		

v · t · e	**Free and open-source software**
General	Alternative terms for free software · Comparison of open-source and closed-source software · Comparison of source-code-hosting facilities · Free software · Free software project directories · Gratis versus libre · Long-term support · Open-source software · Open-source software development · Outline · Timeline

Software packages		Audio · Bioinformatics · Codecs · Configuration management · Drivers (Graphics · Wireless) · Geophysics · Health · Mathematics · Office Suites · Operating systems · Programming languages · Routing · Television · Video games · Web applications (E-commerce) · Android apps · iOS apps · Commercial · Trademarked · Formerly proprietary · Formerly open-source
Community		Free software movement · History · Open-source-software movement · Events
Organisations		Free Software Movement of India · Free Software Foundation
Licenses		AFL · Apache · APSL · Artistic · Beerware · BSD · Creative Commons · CDDL · EPL · Free Software Foundation (GNU GPL · GNU LGPL) · ISC · MIT · MPL · Python · Python Software Foundation License · Shared Source Initiative · Sleepycat · Unlicense · WTFPL · zlib
	Types and standards	Comparison of licenses · Contributor License Agreement · Copyleft · Debian Free Software Guidelines · Definition of Free Cultural Works · Free license · The Free Software Definition · The Open Source Definition · Open-source license · Permissive software license · Public domain · Viral license
Challenges		Digital rights management · Hardware restrictions · License proliferation · Mozilla software rebranding · Proprietary device drivers · Proprietary firmware · Proprietary software · SCO/Linux controversies · Software patents · Software security · Trusted Computing
Related topics		Forking · *GNU Manifesto* · Microsoft Open Specification Promise · Open-core model · Open-source hardware · Shared Source Initiative · Source-available software · *The Cathedral and the Bazaar* · *Revolution OS*
		Portal · **Category**

v · t · e		**Statistical software**
Public domain		Dataplot · Epi Info · CSPro · X-12-ARIMA
Open-source		ADMB · DAP · gretl · JASP · JAGS · JMulTi · Julia · Jupyter (*Julia*, *Python*, *R*) · GNU Octave · OpenBUGS · Orange · PSPP · **Python** (statsmodels, PyMC3, IPython, IDLE) · R (RStudio) · SageMath · SimFiT · SOFA Statistics · Stan · XLispStat
Freeware		BV4.1 · CumFreq · SegReg · XploRe · WinBUGS
Commercial	**Cross-platform**	Data Desk · GAUSS · GraphPad InStat · GraphPad Prism · IBM SPSS Statistics · IBM SPSS Modeler · JMP · Maple · Mathcad · Mathematica · MATLAB · OxMetrics · RATS · Revolution Analytics · SAS · SmartPLS · Stata · StatView · SUDAAN · S-PLUS · TSP · World Programming System (WPS)
	Windows only	BMDP · EViews · GenStat · LIMDEP · LISREL · MedCalc · Microfit · Minitab · MLwiN · NCSS · SHAZAM · SigmaStat · Statistica · StatsDirect · StatXact · SYSTAT · The Unscrambler · UNISTAT
	Excel add-ons	Analyse-it · UNISTAT for Excel · XLfit · RExcel
		Category · **Comparison**

v · t · e		**Numerical-analysis software**
Free		Advanced Simulation Library · ADMB · Chapel · Euler · Fortress · FreeFem++ · FreeMat · Genius · Gmsh · GNU Octave · gretl · Julia · Jupyter (*Julia*, *Python*, *R*; IPython) · MFEM · OpenFOAM · **Python** · R · SageMath · Salome · ScicosLab · Scilab · X10 · Weka
Proprietary		DADiSP · FEATool Multiphysics · GAUSS · LabVIEW · Maple · Mathcad · Mathematica · MATLAB · Speakeasy · VisSim
		Comparison

Authority control	
National libraries	France (data) · Germany · Israel · United States · Czech Republic
Other	Faceted Application of Subject Terminology · SUDOC (France) (1)

.

Categories: Python (programming language) | Class-based programming languages | Notebook interface | Computer science in the Netherlands | Concurrent programming languages | Cross-platform free software | Cross-platform software | Dutch inventions | Dynamically typed programming languages | Educational programming languages | High-level programming languages | Information technology in the Netherlands | Multi-paradigm programming languages | Object-oriented programming languages | Programming languages | Programming languages created in 1991 | Scripting languages | Text-oriented programming languages

This page was last edited on 7 August 2022, at 20:48 (UTC).

Text is available under the Creative Commons Attribution-ShareAlike License 3.0; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.

Privacy policy About Wikipedia Disclaimers Contact Wikipedia Mobile view Developers Statistics Cookie statement

a WIKIMEDIA project Powered by MediaWiki