Python (programming language)

From Wikipedia, the free encyclopedia

Jump to navigationJump to search

For other uses, see Python (disambiguation).

Python

Python logo and wordmark.svg

Paradigm Multi-paradigm: object-oriented,[1] procedural (imperative), functional, structured, reflective

Designed by Guido van Rossum

Developer Python Software Foundation

First appeared February 20, 1991; 30 years ago[2]

Stable release

3.9.7[3] Edit this on Wikidata / 30 August 2021; 32 days ago

Typing discipline Duck, dynamic, strong typing;[4] gradual (since 3.5, but ignored in CPython)[5]

OS Windows, Linux/UNIX, macOS and more[6]

License Python Software Foundation License

Filename extensions .py, .pyi, .pyc, .pyd, .pyo (prior to 3.5),[7] .pyw, .pyz (since 3.5)[8]

Website www.python.org

Major implementations

CPython, PyPy, Stackless Python, MicroPython, CircuitPython, IronPython, Jython

Dialects

Cython, RPython, Starlark[9]

Influenced by

ABC,[10] Ada,[11] ALGOL 68,[12] APL,[13] C,[14] C++,[15] CLU,[16] Dylan,[17] Haskell,[18] Icon,[19] Java,[20] Lisp,[21] Modula-3,[15] Perl, Standard ML[13]

Influenced

Apache Groovy, Boo, Cobra, CoffeeScript,[22] D, F#, Genie,[23] Go, JavaScript,[24][25] Julia,[26] Nim, Ring,[27] Ruby,[28] Swift[29]

Python Programming at Wikibooks

Python is an interpreted high-level general-purpose programming language. Its design philosophy emphasizes code readability with its use of significant indentation. Its language constructs as well as its object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.[30]

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.[31]

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.[32] Python 2.0 was released in 2000 and introduced new features, such as list comprehensions and a garbage collection system using reference counting. Python 3.0 was released in 2008 and was a major revision of the language that is not completely backward-compatible. Python 2 was discontinued with version 2.7.18 in 2020.[33]

Python consistently ranks as one of the most popular programming languages.[34][35][36][37]

Contents

1 History

2 Design philosophy and features

3 Syntax and semantics

3.1 Indentation

3.2 Statements and control flow

3.3 Expressions

3.4 Methods

3.5 Typing

3.6 Arithmetic operations

4 Programming examples

5 Libraries

6 Development environments

7 Implementations

7.1 Reference implementation

7.2 Other implementations

7.3 Unsupported implementations

7.4 Cross-compilers to other languages

7.5 Performance

8 Development

9 API documentation generators

10 Naming

11 Popularity

12 Uses

13 Languages influenced by Python

14 See also

15 References

15.1 Sources

16 Further reading

17 External links

History

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

Main article: History of Python

Python was conceived in the late 1980s[38] by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to ABC programming language, which was inspired by SETL,[39] capable of exception handling and interfacing with the Amoeba operating system.[10] Its implementation began in December 1989, and it added some functional programming features in the January 1994 release, such as the lambda operator.[40] 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.[41] In January 2019, active Python core developers elected a 5-member "Steering Council" to lead the project.[42] As of 2021, the current members of this council are Barry Warsaw, Brett Cannon, Carol Willing, Thomas Wouters, and Pablo Galindo Salgado.[43]

Python 2.0 was released on 16 October 2000, with many major new features, including a cycle-detecting garbage collector and support for Unicode.[44]

Python 3.0 was released on 3 December 2008. It was a major revision of the language that is not completely backward-compatible.[45] Many of its major features were backported to Python 2.6.x[46] and 2.7.x version series. Releases of Python 3 include the 2to3 utility, which automates (at least partially) the translation of Python 2 code to Python 3.[47]

Python 2.7's end-of-life date was initially set at 2015 then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3.[48][49] No more security patches or other improvements will be released for it.[50][51] With Python 2's end-of-life, only Python 3.6.x[52] and later are supported.

Python 3.9.2 and 3.8.8 were expedited[53] as all versions of Python (including 2.7[54]) had security issues, leading to possible remote code execution[55] and web cache poisoning.[56]

Design philosophy and features

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 by metaprogramming[57] and metaobjects (magic methods)).[58] Many other paradigms are supported via extensions, including design by contract[59][60] and logic programming.[61]

Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management.[62] It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's design offers some support for functional programming in the Lisp tradition. It has filter,mapandreduce functions; list comprehensions, dictionaries, sets, and generator expressions.[63] The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.[64]

The language's core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:[65]

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Readability counts.

Rather than having all of its functionality built into its core, Python was designed to be highly extensible (with 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.[38]

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" design philosophy.[65] Alex Martelli, a Fellow at the Python Software Foundation and Python book author, writes that "To describe something as 'clever' is not considered a compliment in the Python culture."[66]

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.[67] 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 to keep the language fun to use. This is reflected in its name—a tribute to the British comedy group Monty Python[68]—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.[69][70]

A common neologism in the Python community is pythonic, which can have a wide range of meanings related to program style. To say that code is pythonic is to say that it uses Python idioms well, that it is natural or shows fluency in the language, that it conforms with Python's minimalist philosophy and emphasis on readability. In contrast, code that is difficult to understand or reads like a rough transcription from another programming language is called unpythonic.[71][72]

Users and admirers of Python, especially those considered knowledgeable or experienced, are often referred to as Pythonistas.[73][74]

Syntax and semantics

Main article: Python syntax and semantics

Python is meant to be an easily readable language. Its formatting is visually uncluttered, and it 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 are rarely, if ever, used. It has fewer syntactic exceptions and special cases than C or Pascal.[75]

Indentation

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.[76] Thus, the program's visual structure accurately represents the program's semantic structure.[77] This feature is sometimes termed the off-side rule, which some other languages share, but in most languages indentation does not have any semantic meaning. The recommended indent size is four spaces.[78]

Statements and control flow

Python's statements include (among others):

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; it also ensures that clean-up code in a finally block will always be 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 the block of code is run and releasing the lock afterwards, or opening a file and then closing it), allowing resource-acquisition-is-initialization (RAII)-like behavior and replaces a common try/finally idiom.[79]

The break statement, exits from a loop.

The continue statement, skips this iteration and continues with the next item.

The del statement, removes a variable, which means the reference from the name to the value is deleted and trying to use that variable will cause an error. A deleted variable can be reassigned.

The pass statement, which serves as a NOP. It is syntactically needed to create an empty code block.

The assert statement, used during debugging to check for conditions that should apply.

The yield statement, which returns a value from a generator function and yield is also an operator. This form is used to implement coroutines.

The return statement, used to return a value from a function.

The import statement, which is used to import modules whose functions or variables can be used in the current program.

The assignment statement (=) operates by binding a name as a reference to a separate, dynamically-allocated object. Variables may be subsequently rebound at any time to any object. In Python, a variable name is a generic reference holder and does not have a fixed data type associated with it. However, at a given time, a variable will refer to some object, which will have a type. This is referred to as dynamic typing and is contrasted with statically-typed programming languages, where each variable may only contain values of a certain type.

Python does not support tail call optimization or first-class continuations, and, according to Guido van Rossum, it never will.[80][81] However, better support for coroutine-like functionality is provided, by extending Python's generators.[82] Before 2.5, generators were lazy iterators; information was passed unidirectionally out of the generator. From Python 2.5, it is possible to pass information back into a generator function, and from Python 3.3, the information can be passed through multiple stack levels.[83]

Expressions

Some Python expressions are similar to those found 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. They are floor division (or integer division) // and floating-point/division.[84] Python also uses the \*\* operator for exponentiation.

From Python 3.5, the new @ infix operator was introduced. It is intended to be used by libraries such as NumPy for matrix multiplication.[85][86]

From Python 3.8, the syntax :=, called the 'walrus operator' was introduced. It assigns values to variables as part of a larger expression.[87]

In Python, == compares by value, versus Java, which compares numerics by value[88] and objects by reference.[89] (Value comparisons in Java on objects can be performed with the equals() method.) Python's is operator may be used to compare object identities (comparison by reference). In Python, 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 Java and C.

Python has a type of expression termed a list comprehension as well as a more general expression termed a generator expression.[63]

Anonymous functions are implemented using lambda expressions; however, these are limited in that the body can only be one expression.

Conditional expressions in Python are written as x if c else y[90] (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 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 + operator can be used to concatenate two tuples, which does not directly modify their contents, but rather produces a new tuple containing the elements of both provided tuples. 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.[91]

Python features sequence unpacking wherein multiple expressions, each evaluating to anything that can be assigned to (a variable, a 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 and will iterate through it, assigning each of the produced values to the corresponding expression on the left.[92]

Python has a "string format" operator %. This functions analogously to printf format strings in C, e.g. "spam=%s eggs=%d" % ("blah", 2) evaluates to "spam=blah eggs=2". In Python 3 and 2.6+, 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": blah = "blah"; eggs = 2; f'spam={blah} eggs={eggs}'.[93]

Strings in Python can be concatenated, by "adding" them (same operator as for adding integers and floats). E.g. "spam" + "eggs" returns "spameggs". Even if your strings contain numbers, they are still added as strings rather than integers. E.g. "2" + "2" returns "22".

Python has various kinds of string literals:

Strings delimited by single or double quote marks. Unlike in Unix shells, Perl and Perl-influenced languages, single quote marks and double quote marks function identically. Both kinds of string use the backslash (\) as an escape character. String interpolation became available in Python 3.6 as "formatted string literals".[93]

Triple-quoted strings, which begin and end with a series of three single or double quote marks. They may span multiple lines and function like here documents in shells, Perl and Ruby.

Raw string varieties, denoted by prefixing the string literal with an 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 on 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 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 syntactically valid (but probably unintended) C code but if c = 1: ... causes a syntax error in Python.

Methods

Methods on objects are functions attached to the object's class; the syntax instance.method(argument) is, for normal methods and functions, syntactic sugar for Class.method(instance, argument). Python methods have an explicit self parameter to access instance data, in contrast to the implicit self (or this) in some other object-oriented programming languages (e.g., C++, Java, Objective-C, or Ruby).[94] Apart from this, 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 such as length, comparison, in arithmetic operations, type conversion, and many more.[95]