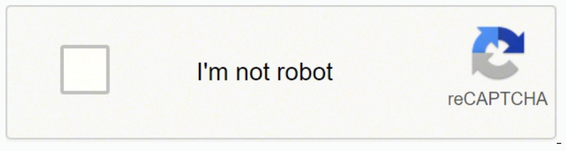

Next

What is python easy definition

General-purpose programming language PythonParadigmMulti-paradigm: object-oriented,[1] procedural (imperative), functional, structured, reflectiveDesigned byGuido van RossumDeveloperPython Software FoundationFirst appeared20 February 1991; 31 years ago (1991-02-20)[2]Stable release3.10.4[3] / 24 March 2022; 2 days ago (24 March 2022)Preview release3.11.0a6[4] / 7 March 2022; 19 days ago (7 March 2022) Typing disciplineDuck, dynamic, strong typing;[5] gradual (since 3.5, but ignored in CPython)[6]OSWindows, Linux/UNIX, macOS and more[7]LicensePython Software Foundation LicenseFilename extensions.py, .pyi, .pyc, .pyd, .pyo (prior to 3.5),[8] .pyw, .pyz (since 3.5)[9]Websitewww.python.orgMajor implementationsCPython, PyPy, Stackless Python, MicroPython, CircuitPython, IronPython, JythonDialectsCython, RPython, Starlark[10]Influenced byABC,[11] Ada,[12] ALGOL 68,[13] APL,[14] C,[15] C++,[16] CLU,[17] Dylan,[18] Haskell,[19] Icon,[20] Lisp,[21] Modula-3,[16] Perl, Standard ML[14]InfluencedApache 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 a high-level, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Its language constructs and 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][32] 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.[33] 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.[34] Python consistently ranks as one of the most popular programming languages.[35][36][37][38] History The designer of Python, Guido van Rossum, at OSCON 2006 Main article: History of Python Python was conceived in the late 1980s[39] 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,[40] capable of exception handling and interfacing with the Amoeba operating system.[11] Its implementation began in December 1989.[41] 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.[42] In January 2019, active Python core developers elected a five-member Steering Council to lead the project.[43][44] Python 2.0 was released on 16 October 2000, with many major new features.[45] Python 3.0, released on 3 December 2008, with many of its major features backported to Python 2.6.x[46] and 2.7.x. Releases of Python 3 include the 2to3 utility, which automates the translation of Python 2 code to Python 3.[47] 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.[48][49] No further 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 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,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] Its 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 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.[39] 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.[65] 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."[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 for it to be 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 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.[71][72] Python users and admirers, 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 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.[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 its semantic structure.[77] 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.[78] Statements and control flow 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; 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[79] 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.[80][81] However, better support for coroutine-like functionality is provided by extending Python's generators.[82] 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.[83] Expressions 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.[84] Python also uses the ** operator for exponentiation. The @ infix operator was introduced in Python 3.5. It is intended to be used by libraries such as NumPy for matrix multiplication.[85][86] The syntax :=, called the "walrus operator", was introduced in Python 3.8. 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] Python's is operator may be used to compare object identities (comparison by reference), and comparisons may be chained—for example, a