Finolex Academy of Management and Technology, Ratnagiri

Department of Information Technology

Subject:	**Python Lab. (ITL404)**		
Class:	**SE IT / Semester – IV (Rev-2016) / Academic year: 2017-18**		
Name of Student:	**Kazi Jawwad A Rahim**		
Roll No:	**28**	**Date of performance (DOP) :**	
Assignment No:	**1**	**Date of checking (DOC) :**	
Title: Looking back and ahead of Python (Programming language evolution)			
Marks:		**Teacher's Signature:**	

1. Learning Objectives Applicable: LO 1, LO3

2. Program Outcomes Applicable: PO6, PO8, PO12

3. Program Education Objectives Applicable: PEO2, PEO4, PEO6

5. Assignment Evaluation:

SR	Parameters	Weight	Excellent	Good	Average	Poor	Not as per requirement
		Scale Factor ->	5	4	3	2	0
1	Technical Understanding	25					
2	Performance / Execution	25					
3	Question Answers	20					
4	Punctuality	20					
5	Presentation	10					
	Total out of 5 --> #(to be converted as per term-work evaluation applicable to the subject)		**∑ (Weight * Scale Factor)/100 = …….. / 5**				

PYTHON

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation.

Python

Paradigm	Object-oriented, imperative, functional, procedural, reflective
Designed by	Guido van Rossum
Developer	Python Software Foundation
First appeared	1990
Stable release	3.6.5 / 28 March 2018
	2.7.14 / 16 September 2017
Preview release	3.7.0b3 / 29 March 2018
Typing discipline	Duck, dynamic, strong
License	Python Software Foundation License
Filename extensions	.py, .pyc, .pyd, .pyo (prior to 3.5), .pyw, .pyz (since 3.5)
Website	python.org
Major implementations	CPython, IronPython, Jython, MicroPython, Numba, PyPy, Stackless Python
Dialects	Cython, RPython
Influenced by	ABC, ALGOL 68, C, C++, CLU, Dylan, Haskell, Icon, Java, Lisp, Modula-3, Perl
Influenced	Boo, Cobra, Coconut, CoffeeScript, D, F#, Falcon, Genie,Go, Groovy, JavaScript, Julia, Nim, Ring, Ruby, Swift

History

Python was conceived in the late 1980s, and its implementation began in December 1989 by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL) capable of exception handling and interfacing with the Amoeba operating system. Van Rossum remains Python's principal author. His continuing central role in Python's development is reflected in the title given to him by the Python community: Benevolent Dictator For Life (BDFL). Python 2.0 was released on 16 October 2000 and had many major new features, including a cycle-detecting garbage

collector and support for Unicode. With this release, the development process became more transparent and community-backed.Python 3.0 (initially called Python 3000 or py3k) was released on 3 December 2008 after a long testing period. It is a major revision of the language that is not completely backward-compatible with previous versions. However, many of its major features have been backported to the Python 2.6.x and 2.7.x version series, and releases of Python 3 include the 2 to 3 utility, which automates the translation of Python 2 code to Python 3.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.Python 3.6 had changes regarding UTF-8 (in Windows, PEP 528 and PEP 529) and Python 3.7.0b1 (PEP 540) adds a new "UTF-8 Mode" (and overrides POSIX locale). In January 2017, Google announced work on a Python 2.7 to Go transcompiler to improve performance under concurrent workloads.

Features and philosophy

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 and metaobjects (magic methods)). Many other paradigms are supported via extensions, including design by contract and logic programming.Python uses dynamic typing, and a combination of reference counting and a cycle-detecting garbage collector for memory management. 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(), map(), and reduce() functions; list comprehensions, dictionaries, and sets; and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML. The language's core philosophy is summarized in the document The Zen of Python (PEP 20), which includes aphorisms such as:

- 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. 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. While offering choice in coding methodology, the Python philosophy rejects exuberant syntax (such as that of Perl) in favor of a simpler, less-cluttered grammar. As Alex Martelli put it: "To describe something as 'clever' is not considered a compliment in the Python culture."[48] Python's philosophy rejects the Perl "there is more than one way to do it" approach to language design in favor of "there should be one—and preferably only one—obvious way to do it". Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of CPython that would offer marginal increases in speed at the cost of clarity. 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. An important goal of Python's developers is keeping it fun to use. This is reflected in the language's name—a tribute to the British comedy group Monty Python—and in occasionally playful approaches to tutorials and reference materials, such as examples that refer to spam and eggs (from a famous Monty Python sketch) instead of the standard foo and bar. A common neologism in the Python community is pythonic, which can have a wide range of meanings related to program style. To say that 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. Users and admirers of Python, especially those considered knowledgeable or experienced, are often referred to as Pythonists, Pythonistas, and Pythoneers.

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 optional. It has fewer syntactic exceptions and special cases than C or Pascal.

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. Thus, the program's visual structure accurately represents the program's semantic structure. This feature is also sometimes termed the off-side rule.

Statements and control flow

Python's statements include (among others):

- The assignment statement (token '=', the equals sign). This operates differently than in traditional imperative programming languages, and this fundamental mechanism (including the nature of Python's version of variables) illuminates many other features of the language. Assignment in C, e.g., x = 2, translates to "typed variable name x receives a copy of numeric value 2". The (right-hand) value is copied into an allocated storage location for which the (left-hand) variable name is the symbolic address. The memory allocated to the variable is large enough (potentially quite large) for the declared type. In the simplest case of Python assignment, using the same example, x = 2, translates to "(generic) name x receives a reference to a separate, dynamically allocated object of numeric (int) type of value 2." This is termed binding the name to the object. Since the name's storage location doesn't contain the indicated value, it is improper to call it a variable. Names may be subsequently rebound at any time to objects of greatly varying types, including strings, procedures, complex objects with data and methods, etc. Successive assignments of a common value to multiple names, e.g., x = 2; y = 2; z = 2 result in allocating storage to (at most) three names and one numeric object, to which all three names are bound. Since a name is a generic reference holder it is unreasonable to associate a fixed data type with it. However at a given time a name will be bound to some object, which will have a type; thus there is dynamic typing.
- The if statement, which conditionally executes a block of code, along with else and elif (a contraction of else-if).
- The for statement, which iterates over an iterable object, capturing each element to a local variable for use by the attached block.
- The while statement, which executes a block of code as long as its condition is true.
- The try statement, which allows exceptions raised in its attached code block to be caught and handled by except clauses; it also ensures that clean-up code in a finally block will always be run regardless of how the block exits.
- The class statement, which executes a block of code and attaches its local namespace to a class, for use in object-oriented programming.

- The def statement, which defines a function or method.
- The with statement, from Python 2.5 released on September 2006, 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.
- 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 ought to apply.
- The yield statement, which returns a value from a generator function. From Python 2.5, yield is also an operator. This form is used to implement coroutines.
- The import statement, which is used to import modules whose functions or variables can be used in the current program. There are four ways of using import: import <module name> or from <module name> import * or import numpy as np or from numpy import pi as Pie.
- The print statement was changed to the print() function in Python 3.

Python does not support tail call optimization or first-class continuations, and, according to Guido van Rossum, it never will. However, better support for coroutine-like functionality is provided in 2.5, by extending Python's generators. 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.

Expressions

Some Python expressions are similar to 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 and integer division. Python also added 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.
- In Python, == compares by value, versus Java, which compares numerics by value and objects by reference. (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. Python 2.4 extended list comprehensions into a more general expression termed a generator expression.
- 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 (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.

- Python features sequence unpacking where multiple expressions, each evaluating to anything that can be assigned to (a variable, a writable property, etc.), are associated in the 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.[citation needed]
- Python has a "string format" operator %. This functions analogous 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}'.
- 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".
    - 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).

Typing

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

Python allows programmers to define their own types using classes, which are most often used for object-oriented programming. New instances of classes are constructed by calling the class (for example, SpamClass() or EggsClass()), and the classes 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: old-style and new-style. The syntax of both styles is the same, the difference being whether the class object is inherited from, directly or indirectly (all new-style classes inherit from object and are instances of type). In versions of Python 2 from Python 2.2 onwards, both kinds of classes can be used. Old-style classes were eliminated in Python 3.0.

The long term plan is to support gradual typing and from Python 3.5, the syntax of the language allows specifying static types but they are not checked in the default implementation, CPython. An experimental optional static type checker named mypy supports compile-time type checking.

Summary of Python 3's built-in types

Type	Mutable	Description	Syntax example
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
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}
ellipsis		An ellipsis placeholder to be used as an index in NumPy arrays	...
float	immutable	Floating point number, system-defined precision	3.1415927
frozenset	immutable	Unordered set, contains no duplicates; can contain mixed types, if hashable	frozenset([4.0, 'string', True])

int	immutable	Integer of unlimited magnitude	42
list	mutable	List, can contain mixed types	[4.0, 'string', True]
set	mutable	Unordered set, contains no duplicates; can contain mixed types, if hashable	{4.0, 'string', True}
str	immutable	A character string: sequence of Unicode codepoints	'Wikipedia' "Wikipedia" """Spanning multiple lines"""
tuple	immutable	Can contain mixed types	(4.0, 'string', True) But we can append elements using __add__ . a = (4.0, 'string' , True).__add__(('hi' ,)) now a gives (4.0, 'string', True ,'hi')

Mathematics

Python has the usual C arithmetic operators (+, -, *, /, %). It also has ** for exponentiation, e.g. 5**3 == 125 and 9**0.5 == 3.0, and a new matrix multiply @ operator is included in version 3.5. Additionally, it has a unary operator (~), which essentially inverts all the bits of its one argument. For integers, this means ~x=-x-1. Other operators include bitwise shift operators x << y, which shifts x to the left y places, the same as x*(2**y) , and x >> y, which shifts x to the right y places, the same as x/(2**y) .

The behavior of division has changed significantly over time:

- Python 2.1 and earlier use the C division behavior. The / operator is integer division if both operands are integers, and floating-point division otherwise. Integer division rounds towards 0, e.g. 7/3 == 2 and -7/3 == -2.
- Python 2.2 changes integer division to round towards negative infinity, e.g. 7/3 == 2 and -7/3 == -3. The floor division // operator is 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 to use Python 3.0 rules for division (see next).
- Python 3.0 changes / to be always floating-point division. In Python terms, the pre-3.0 / is classic division, the version-3.0 / is real division, and // is floor division.

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.

Python provides a round function for rounding a float to the nearest integer. For tie-breaking, versions before 3 use round-away-from-zero: round(0.5) is 1.0, round(-0.5) is −1.0. Python 3 uses round to even: round(1.5) is 2, round(2.5) is 2.

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. 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.
Python has extensive built-in support for arbitrary precision arithmetic. Integers are transparently switched from the machine-supported maximum fixed-precision (usually 32 or 64 bits), belonging to the python type int, to arbitrary precision, belonging to the python type long, where needed. The latter have an "L" suffix in their textual representation. (In Python 3, the distinction between the int and long types was eliminated; this behavior is now entirely contained by the int class.) The Decimal type/class in module decimal (since version 2.4) provides decimal floating point numbers to arbitrary precision and several rounding modes. The Fraction type in module fractions (since version 2.6) provides arbitrary precision for rational numbers.
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.

Libraries

Python's large standard library, commonly cited as one of its greatest strengths, provides tools suited to many tasks. 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, 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), but most modules are not. They are specified by their code, internal documentation, and test suites (if supplied). 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 March 2018, the Python Package Index (PyPI), the official repository for third-party Python software, contains over 130,000 packages with a wide range of functionality, including:

- Graphical user interfaces
- Web frameworks
- Multimedia
- Databases
- Networking
- Test frameworks
- Automation
- Web scraping
- Documentation
- System administration
- Scientific computing
- Text processing
- Image processing

Development environments

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

Other shells, including IDLE and IPython, add further abilities such as auto-completion, session state retention and syntax highlighting.
As well as standard desktop integrated development environments (see Wikipedia's "Python IDE" article), there are Web browser-based IDEs; SageMath (intended for developing science and math-related Python programs); PythonAnywhere, a browser-based IDE and hosting environment; and Canopy IDE, a commercial Python IDE emphasizing scientific computing.

Implementations

Reference implementation

CPython is the reference implementation of Python. It is written in C, meeting the C89 standard with several select C99 features. It compiles Python programs into an intermediate bytecode which is then executed by its virtual machine. CPython is distributed with a large standard library written in a mixture of C and native Python. It is available for many platforms, including Windows and most modern Unix-like systems. Platform portability was one of its earliest priorities.

Other implementations

PyPy is a fast, compliant interpreter of Python 2.7 and 3.5. Its just-in-time compiler brings a significant speed improvement over CPython.
Stackless Python is a significant fork of CPython that implements microthreads; it does not use the C memory stack, thus allowing massively concurrent programs. PyPy also has a stackless version.
MicroPython is a Python 3 variant optimised for microcontrollers.

Unsupported implementations

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.

Psyco is a just-in-time specialising compiler that integrates with CPython and transforms bytecode to machine code at runtime. The emitted code is specialised for certain data types and is faster than standard Python code.

In 2005, Nokia released a Python interpreter for the Series 60 mobile phones named PyS60. It includes many of the modules from the CPython implementations and some additional modules to integrate with the Symbian operating system. The project has been kept up-to-date to run on all variants of the S60 platform, and several third-party modules are available. The Nokia N900 also supports Python with GTK widget libraries, enabling programs to be written and run on the target device.

Cross-compilers to other languages

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:

- Jython compiles into Java byte code, which can then be executed by every Java virtual machine implementation. This also enables the use of Java class library functions from the Python program.
- IronPython follows a similar approach in order to run Python programs on the .NET Common Language Runtime.
- The RPython language can be compiled to C, Java bytecode, or Common Intermediate Language, and is used to build the PyPy interpreter of Python.
- Pyjs compiles Python to JavaScript.
- Cython compiles Python to C and C++.
- Pythran compiles Python to C++.
- Somewhat dated Pyrex (latest release in 2010) and Shed Skin (latest release in 2013) compile to C and C++ respectively.

- Google's Grumpy compiles Python to Go.
- Nuitka compiles Python into C++

Performance

A performance comparison of various Python implementations on a non-numerical (combinatorial) workload was presented at EuroSciPy '13.

Development

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. Outstanding PEPs are reviewed and commented on by the Python community and Guido Van Rossum, Python's Benevolent Dictator For Life.

Enhancement of the language corresponds with 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 maintained at python.org. Development originally took place on a self-hosted source-code repository running Mercurial, until Python moved to GitHub in January 2017.

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 need to be manually ported. The first part of the version number is incremented. These releases happen infrequently—for example, version 3.0 was released 8 years after 2.0.
- Major or "feature" releases, about every 18 months, are largely compatible but introduce new features. The second part of the version number is incremented. Each major version is supported by bugfixes for several years after its release.
- Bugfix releases, 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.

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, and using the BuildBot continuous integration system.

The community of Python developers has also contributed over 86,000 software modules (as of 20 August 2016) to the Python Package Index (PyPI), the official repository of third-party Python libraries.

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

Naming

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; for example, the metasyntactic variables often used in Python literature are spam and eggs instead of the traditional foo and bar. The official Python documentation also contains various references to Monty Python routines.

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.

Uses

Since 2003, Python has consistently ranked in the top ten most popular programming languages in the TIOBE Programming Community Index where, as of January 2018, it is the fourth most popular language (behind Java, C, and C++). It was selected Programming Language of the Year in 2007 and 2010.

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++".

Large organizations that use Python include Wikipedia, Google, Yahoo!, CERN, NASA, Facebook, Amazon, Instagram, Spotify and some smaller entities like ILM and ITA. The social news networking site Reddit is written entirely in Python.

Python can serve as a scripting language for web applications, e.g., via mod_wsgi for the Apache web server. With Web Server Gateway Interface, a standard API has 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 data mapper to a relational database. Twisted is a framework to program communications between computers, and is used (for example) by Dropbox.

Libraries such as NumPy, SciPy and Matplotlib allow the effective use of Python in scientific computing, with specialized libraries such as Biopython and Astropy providing domain-specific functionality. SageMath is a mathematical software with a "notebook" programmable in Python: its library covers many aspects of mathematics, including algebra, combinatorics, numerical mathematics, number theory, and calculus. The Python language re-implemented in Java platform is used for numeric and statistical calculations with 2D/3D visualization by the DMelt project.

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, Inkscape, Scribus and Paint Shop Pro, 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. It has also been used in several video games, and has been adopted as first of the three available programming languages in Google App Engine, the other two being Java and Go. Python is also used in algorithmic trading and quantitative finance. Python can also be implemented in APIs of online brokerages that run on other languages by using wrappers.

Python has been used in artificial intelligence projects. As a scripting language with modular architecture, simple syntax and rich text processing tools, Python is often used for natural language processing.

Many operating systems include Python as a standard component. It ships with most Linux distributions, AmigaOS 4, FreeBSD, NetBSD, OpenBSD and macOS, 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 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.

Most of the Sugar software for the One Laptop per Child XO, now developed at Sugar Labs, is written in Python.

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 since Version 4.0 from 7 February 2013.

Languages influenced by Python

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

- Boo uses indentation, a similar syntax, and a similar object model.
- Cobra uses indentation and a similar syntax, and its "Acknowledgements" document lists Python first among languages that influenced it. However, Cobra directly supports design-by-contract, unit tests, and optional static typing.
- CoffeeScript, a programming language that cross-compiles to JavaScript, has Python-inspired syntax.
- ECMAScript borrowed iterators, generators and list comprehensions from Python.
- Go is designed for the "speed of working in a dynamic language like Python" and shares the same syntax for slicing arrays.
- Groovy was motivated by the desire to bring the Python design philosophy to Java.
- Julia was designed "with true macros [.. and to be] as usable for general programming as Python [and] should be as fast as C". Calling to or from Julia is possible; to with PyCall.jl and a Python package pyjulia allows calling, in the other direction, from Python.
- Kotlin (programming language) is a functional programming language with an interactive shell similar to python. However, Kotlin is strongly typed with access to standard Java libraries.
- 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."
- Swift, a programming language developed by Apple, has some Python-inspired syntax.

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 and Erlang.

Python received TIOBE's Programming Language of the Year awards in 2007 and 2010. The award is given to the language with the greatest growth in popularity over the year, as measured by the TIOBE index.