
Distributing Python Modules

发布 2.7.18

Guido van Rossum
and the Python development team

五月 20, 2020

Python Software Foundation
Email: docs@python.org

Contents

1	关键术语	3
2	开源许可与协作	5
3	安装工具	7
4	Reading the guide	9
5	我该如何…？	11
5.1	…为我的项目选择一个名字？	11
5.2	…创建和分发二进制扩展？	11
A	术语对照表	13
B	文档说明	21
B.1	Python 文档的贡献者	21
C	历史和许可证	23
C.1	该软件的历史	23
C.2	获取或以其他方式使用 Python 的条款和条件	24
C.3	被收录软件的许可证与鸣谢	27
D	Copyright	39
	索引	41

电子邮箱 distutils-sig@python.org

作为一个流行的开源开发项目，Python 拥有一个活跃的贡献者和用户支持社区，这些社区也可以让他们的软件可供其他 Python 开发人员在开源许可条款下使用。

这允许 Python 用户有效地共享和协作，从其他人已经创建的解决方案中受益于常见（有时甚至是罕见的）问题，以及可以提供他们自己的解决方案。

本指南涵盖了分发部分的流程。有关安装其他 Python 项目的指南，请参阅 [安装指南](#)。

注解： 对于企业和其他机构用户，请注意许多组织都有自己的政策来使用和贡献开源软件。在使用 Python 提供的分发和安装工具时，请考虑这些政策。

关键术语

- [Python Packaging Index](#) 是一个开源许可的软件包公共存储库，可供所有 Python 用户使用。
- the [Python Packaging Authority](#) are the group of developers and documentation authors responsible for the maintenance and evolution of the standard packaging tools and the associated metadata and file format standards. They maintain a variety of tools, documentation and issue trackers on both [GitHub](#) and [BitBucket](#).
- `distutils` 是 1998 年首次添加到 Python 标准库的原始构建和分发系统。虽然直接使用 `distutils` 正在逐步淘汰，但它仍然为当前的打包和分发基础架构奠定了基础它不仅仍然是标准库的一部分，而且它的名称还以其他方式存在（例如用于协调 Python 包装标准开发的邮件列表的名称）。
- `setuptools`（在很大程序上）是作为 `distutils` 的取代者，于 2004 年首次发布。它对未经修改的 `distutils` 工具最重要的补充是能够声明对其他包的依赖。目前它被推荐用来替代 `distutils`，其更新更为频繁，在更为多样的 Python 版本之上为最新的包标准提供持续支持。
- `wheel`（在此上下文中）是一个将 `bdist_wheel` 命令添加到 `distutils/setuptools` 的项目。这产生了一个跨平台的二进制打包格式（称为“轮子”或“轮子文件”，并在 [PEP 427](#) 中定义），它允许在系统上安装 Python 库，甚至包括二进制扩展的库，而不需在本地进行构建。

开源许可与协作

在世界上大多数地方，软件自动受版权保护。这意味着其他开发人员需要明确的权限来复制，使用，修改和重新分发软件。

开源许可是一种以相对一致的方式明确授予此类权限的方式，允许开发人员通过为各种问题免费提供通用解决方案来有效地共享和协作。这使得许多开发人员可以将更多时间用于关注他们特定情况相对独特的问题。

Python 提供的分发工具旨在使开发人员选择开源时，可以合理地直接将其自己的贡献回馈到该公共软件池。

无论该软件是否作为开源软件发布，相同的分发工具也可用于在组织内分发软件。

安装工具

标准库不包括支持现代 Python 打包标准的构建工具，因为核心开发团队已经发现，即使在旧版本的 Python 上，使用一致工作的标准工具也很重要。

可以通过在命令行调用 pip 模块来安装当前推荐的构建和分发工具：

```
python -m pip install setuptools wheel twine
```

注解：对于 POSIX 用户（包括 Mac OS X 和 Linux 用户），这些说明假定使用 *virtual environment*。

对于 Windows 用户，这些说明假定在安装 Python 时选择了调整系统 PATH 环境变量的选项。

Python 打包指南包含有关 *currently recommended tools* 的更多详细信息。

CHAPTER 4

Reading the guide

The Python Packaging User Guide covers the various key steps and elements involved in creating a project:

- 项目的结构
- 项目的构建与打包
- 上传项目到 [Python Packaging Index](#)

我该如何…？

这是一些常见任务的快速解答或相关链接。

5.1 … 为我的项目选择一个名字？

这不是一个简单的主题，但这里有一些提示：

- 检查 [Python Packaging Index](#) 以查看该名称是否已被使用
- check popular hosting sites like GitHub, BitBucket, etc to see if there is already a project with that name
- 检查您正在考虑的名称在网络搜索中出现的内容
- 避免使用特别常见的单词，尤其是具有多重含义的单词，因为它们会使用户在搜索时难以找到您的软件

5.2 … 创建和分发二进制扩展？

这实际上是一个非常复杂的主题，根据您的目标，可以提供各种替代方案。有关更多信息和建议，请参阅 [Python 打包指南](#)。

参见：

[Python Packaging User Guide: Binary Extensions](#)

术语对照表

>>> 交互式终端中默认的 Python 提示符。往往会显示于能以交互方式在解释器里执行的样例代码之前。

... The default Python prompt of the interactive shell when entering code for an indented code block, when within a pair of matching left and right delimiters (parentheses, square brackets, curly braces or triple quotes), or after specifying a decorator.

2to3 一个将 Python 2.x 代码转换为 Python 3.x 代码的工具，能够处理大部分通过解析源码并遍历解析树可检测到的不兼容问题。

2to3 包含在标准库中，模块名为 `lib2to3`；并提供一个独立入口点 `Tools/scripts/2to3`。参见 `2to3-reference`。

abstract base class – 抽象基类 Abstract base classes complement *duck-typing* by providing a way to define interfaces when other techniques like `hasattr()` would be clumsy or subtly wrong (for example with magic methods). ABCs introduce virtual subclasses, which are classes that don't inherit from a class but are still recognized by `isinstance()` and `issubclass()`; see the `abc` module documentation. Python comes with many built-in ABCs for data structures (in the `collections` module), numbers (in the `numbers` module), and streams (in the `io` module). You can create your own ABCs with the `abc` module.

argument – 参数 A value passed to a *function* (or *method*) when calling the function. There are two types of arguments:

- 关键字参数: 在函数调用中前面带有标识符（例如 `name=`）或者作为包含在前面带有 `**` 的字典里的值传入。举例来说，3 和 5 在以下对 `complex()` 的调用中均属于关键字参数：

```
complex(real=3, imag=5)
complex(**{'real': 3, 'imag': 5})
```

- 位置参数: 不属于关键字参数的参数。位置参数可出现于参数列表的开头以及/或者作为前面带有 `*` 的 *iterable* 里的元素被传入。举例来说，3 和 5 在以下调用中均属于位置参数：

```
complex(3, 5)
complex(*(3, 5))
```

参数会被赋值给函数体中对应的局部变量。有关赋值规则参见 `calls` 一节。根据语法，任何表达式都可用来表示一个参数；最终算出的值会被赋给对应的局部变量。

See also the [parameter](#) glossary entry and the FAQ question on the difference between arguments and parameters.

attribute –属性 关联到一个对象的值，可以使用点号表达式通过其名称来引用。例如，如果一个对象 *o* 具有一个属性 *a*，就可以用 *o.a* 来引用它。

BDFL Benevolent Dictator For Life, a.k.a. [Guido van Rossum](#), Python’s creator.

bytes-like object –字节类对象 An object that supports the buffer protocol, like `str`, `bytearray` or `memoryview`. Bytes-like objects can be used for various operations that expect binary data, such as compression, saving to a binary file or sending over a socket. Some operations need the binary data to be mutable, in which case not all bytes-like objects can apply.

bytecode –字节码 Python source code is compiled into bytecode, the internal representation of a Python program in the CPython interpreter. The bytecode is also cached in `.pyc` and `.pyo` files so that executing the same file is faster the second time (recompilation from source to bytecode can be avoided). This “intermediate language” is said to run on a [virtual machine](#) that executes the machine code corresponding to each bytecode. Do note that bytecodes are not expected to work between different Python virtual machines, nor to be stable between Python releases.

字节码指令列表可以在 `dis` 模块的文档中查看。

class –类 用来创建用户定义对象的模板。类定义通常包含对该类的实例进行操作的方法定义。

classic class Any class which does not inherit from `object`. See [new-style class](#). Classic classes have been removed in Python 3.

coercion –强制类型转换 The implicit conversion of an instance of one type to another during an operation which involves two arguments of the same type. For example, `int(3.15)` converts the floating point number to the integer 3, but in `3+4.5`, each argument is of a different type (one int, one float), and both must be converted to the same type before they can be added or it will raise a `TypeError`. Coercion between two operands can be performed with the `coerce` built-in function; thus, `3+4.5` is equivalent to calling `operator.add(*coerce(3, 4.5))` and results in `operator.add(3.0, 4.5)`. Without coercion, all arguments of even compatible types would have to be normalized to the same value by the programmer, e.g., `float(3)+4.5` rather than just `3+4.5`.

complex number –复数 对普通实数系统的扩展，其中所有数字都被表示为一个实部和一个虚部的和。虚数是虚数单位（-1 的平方根）的实倍数，通常在数学中写为 *i*，在工程学中写为 *j*。Python 内置了对复数的支持，采用工程学标记方式；虚部带有一个 *j* 后缀，例如 `3+1j`。如果需要 `math` 模块内对象的对应复数版本，请使用 `cmath`，复数的使用是一个比较高级的数学特性。如果你感觉没有必要，忽略它们也几乎不会有任何问题。

context manager –上下文管理器 在 `with` 语句中使用，通过定义 `__enter__()` 和 `__exit__()` 方法来控制环境状态的对象。参见 [PEP 343](#)。

CPython Python 编程语言的规范实现，在 [python.org](#) 上发布。”CPython” 一词用于在必要时将此实现与其他实现例如 Jython 或 IronPython 相区别。

decorator –装饰器 返回值为另一个函数的函数，通常使用 `@wrapper` 语法形式来进行函数变换。装饰器的常见例子包括 `classmethod()` 和 `staticmethod()`。

装饰器语法只是一种语法糖，以下两个函数定义在语义上完全等价：

```
def f(...):
    ...
f = staticmethod(f)

@staticmethod
def f(...):
    ...
```

同样的概念也适用于类，但通常较少这样使用。有关装饰器的详情可参见 [函数定义](#) 和 [类定义](#) 的文档。

descriptor –描述器 Any *new-style* object which defines the methods `__get__()`, `__set__()`, or `__delete__()`. When a class attribute is a descriptor, its special binding behavior is triggered upon attribute lookup. Normally, using `a.b` to get, set or delete an attribute looks up the object named `b` in the class dictionary for `a`, but if `b` is a descriptor, the respective descriptor method gets called. Understanding descriptors is a key to a deep understanding of Python because they are the basis for many features including functions, methods, properties, class methods, static methods, and reference to super classes.

有关描述符的方法的详情可参看 `descriptors`。

dictionary –字典 An associative array, where arbitrary keys are mapped to values. The keys can be any object with `__hash__()` and `__eq__()` methods. Called a hash in Perl.

dictionary view –字典视图 The objects returned from `dict.viewkeys()`, `dict.viewvalues()`, and `dict.viewitems()` are called dictionary views. They provide a dynamic view on the dictionary's entries, which means that when the dictionary changes, the view reflects these changes. To force the dictionary view to become a full list use `list(dictview)`. See `dict-views`.

docstring –文档字符串 作为类、函数或模块之内的第一个表达式出现的字符串字面值。它在代码执行时会被忽略，但会被解释器识别并放入所在类、函数或模块的 `__doc__` 属性中。由于它可用于代码内省，因此是对象存放文档的规范位置。

duck-typing –鸭子类型 指一种编程风格，它并不依靠查找对象类型来确定其是否具有正确的接口，而是直接调用或使用其方法或属性（“看起来像鸭子，叫起来也像鸭子，那么肯定就是鸭子。”）由于强调接口而非特定类型，设计良好的代码可通过允许多态替代来提升灵活性。鸭子类型避免使用 `type()` 或 `isinstance()` 检测。（但要注意鸭子类型可以使用[抽象基类](#)作为补充。）而往往会采用 `hasattr()` 检测或是[EAFP](#)编程。

EAFP “求原谅比求许可更容易”的英文缩写。这种 Python 常用代码编写风格会假定所需的键或属性存在，并在假定错误时捕获异常。这种简洁快速风格的特点就是大量运用 `try` 和 `except` 语句。于其相对的则是所谓[LBYL](#)风格，常见于 C 等许多其他语言。

expression –表达式 A piece of syntax which can be evaluated to some value. In other words, an expression is an accumulation of expression elements like literals, names, attribute access, operators or function calls which all return a value. In contrast to many other languages, not all language constructs are expressions. There are also *statements* which cannot be used as expressions, such as `print` or `if`. Assignments are also statements, not expressions.

extension module –扩展模块 以 C 或 C++ 编写的模块，使用 Python 的 C API 来与语言核心以及用户代码进行交互。

file object –文件对象 对外提供面向文件 API 以使用下层资源的对象（带有 `read()` 或 `write()` 这样的方法）。根据其创建方式的不同，文件对象可以处理对真实磁盘文件，对其他类型存储，或是对通讯设备的访问（例如标准输入/输出、内存缓冲区、套接字、管道等等）。文件对象也被称为 文件类对象或流。

There are actually three categories of file objects: raw binary files, buffered binary files and text files. Their interfaces are defined in the `io` module. The canonical way to create a file object is by using the `open()` function.

file-like object –文件类对象 *file object* 的同义词。

finder –查找器 An object that tries to find the *loader* for a module. It must implement a method named `find_module()`. See [PEP 302](#) for details.

floor division –向下取整除法 向下舍入到最接近的整数的数学除法。向下取整除法的运算符是 `//`。例如，表达式 `11 // 4` 的计算结果是 2，而与之相反的是浮点数的真正除法返回 2.75。注意 `(-11) // 4` 会返回 -3 因为这是 -2.75 向下舍入得到的结果。见 [PEP 238](#)。

function –函数 可以向调用者返回某个值的一组语句。还可以向其传入零个或多个[参数](#)并在函数体执行中被使用。另见[parameter](#), [method](#) 和 `function` 等节。

__future__ A pseudo-module which programmers can use to enable new language features which are not compatible with the current interpreter. For example, the expression `11 / 4` currently evaluates to 2. If the module in which it

is executed had enabled *true division* by executing:

```
from __future__ import division
```

the expression `11/4` would evaluate to `2.75`. By importing the `__future__` module and evaluating its variables, you can see when a new feature was first added to the language and when it will become the default:

```
>>> import __future__
>>> __future__.division
_Feature((2, 2, 0, 'alpha', 2), (3, 0, 0, 'alpha', 0), 8192)
```

garbage collection –垃圾回收 The process of freeing memory when it is not used anymore. Python performs garbage collection via reference counting and a cyclic garbage collector that is able to detect and break reference cycles.

generator –生成器 A function which returns an iterator. It looks like a normal function except that it contains `yield` statements for producing a series of values usable in a `for`-loop or that can be retrieved one at a time with the `next()` function. Each `yield` temporarily suspends processing, remembering the location execution state (including local variables and pending try-statements). When the generator resumes, it picks up where it left off (in contrast to functions which start fresh on every invocation).

generator expression –生成器表达式 An expression that returns an iterator. It looks like a normal expression followed by a `for` expression defining a loop variable, range, and an optional `if` expression. The combined expression generates values for an enclosing function:

```
>>> sum(i*i for i in range(10))           # sum of squares 0, 1, 4, ... 81
285
```

GIL 参见 *global interpreter lock*。

global interpreter lock –全局解释器锁 *CPython* 解释器所采用的一种机制，它确保同一时刻只有一个线程在执行 Python *bytecode*。此机制通过设置对象模型（包括 `dict` 等重要内置类型）针对并发访问的隐式安全简化了 *CPython* 实现。给整个解释器加锁使得解释器多线程运行更方便，其代价则是牺牲了在多处理器上的并行性。

不过，某些标准库或第三方库的扩展模块被设计为在执行计算密集型任务如压缩或哈希时释放 GIL。此外，在执行 I/O 操作时也总是会释放 GIL。

创建一个（以更精细粒度来锁定共享数据的）“自由线程”解释器的努力从未获得成功，因为这会牺牲在普通单处理器情况下的性能。据信克服这种性能问题的措施将导致实现变得更复杂，从而更难以维护。

hashable –可哈希 An object is *hashable* if it has a hash value which never changes during its lifetime (it needs a `__hash__()` method), and can be compared to other objects (it needs an `__eq__()` or `__cmp__()` method). Hashable objects which compare equal must have the same hash value.

可哈希性使得对象能够作为字典键或集合成员使用，因为这些数据结构要在内部使用哈希值。

All of Python's immutable built-in objects are hashable, while no mutable containers (such as lists or dictionaries) are. Objects which are instances of user-defined classes are hashable by default; they all compare unequal (except with themselves), and their hash value is derived from their `id()`.

IDLE Python 的 IDE，“集成开发与学习环境”的英文缩写。是 Python 标准发行版附带的基本编程器和解释器环境。

immutable –不可变 具有固定值的对象。不可变对象包括数字、字符串和元组。这样的对象不能被改变。如果必须存储一个不同的值，则必须创建新的对象。它们在需要常量哈希值的地方起着重要作用，例如作为字典中的键。

integer division Mathematical division discarding any remainder. For example, the expression `11/4` currently evaluates to 2 in contrast to the `2.75` returned by float division. Also called *floor division*. When dividing two integers the outcome will always be another integer (having the floor function applied to it). However, if one of the operands is

another numeric type (such as a `float`), the result will be coerced (see [coercion](#)) to a common type. For example, an integer divided by a float will result in a float value, possibly with a decimal fraction. Integer division can be forced by using the `//` operator instead of the `/` operator. See also [__future__](#).

importing –导入 令一个模块中的 Python 代码能为另一个模块中的 Python 代码所使用的过程。

importer –导入器 查找并加载模块的对象；此对象既属于 [finder](#) 又属于 [loader](#)。

interactive –交互 Python 带有一个交互式解释器，即你可以在解释器提示符后输入语句和表达式，立即执行并查看其结果。只需不带参数地启动 `python` 命令（也可以在你的计算机开始菜单中选择相应菜单项）。在测试新想法或检验模块和包的时候用这种方式会非常方便（请记得使用 `help(x)`）。

interpreted –解释型 Python 一是种解释型语言，与之相对的是编译型语言，虽然两者的区别由于字节码编译器的存在而会有所模糊。这意味着源文件可以直接运行而不必显式地创建可执行文件再运行。解释型语言通常具有比编译型语言更短的开发/调试周期，但是其程序往往运行得更慢。参见 [interactive](#)。

iterable –可迭代对象 An object capable of returning its members one at a time. Examples of iterables include all sequence types (such as `list`, `str`, and `tuple`) and some non-sequence types like `dict` and `file` and objects of any classes you define with an `__iter__()` or `__getitem__()` method. Iterables can be used in a `for` loop and in many other places where a sequence is needed (`zip()`, `map()`, ...). When an iterable object is passed as an argument to the built-in function `iter()`, it returns an iterator for the object. This iterator is good for one pass over the set of values. When using iterables, it is usually not necessary to call `iter()` or deal with iterator objects yourself. The `for` statement does that automatically for you, creating a temporary unnamed variable to hold the iterator for the duration of the loop. See also [iterator](#), [sequence](#), and [generator](#).

iterator –迭代器 An object representing a stream of data. Repeated calls to the iterator's `next()` method return successive items in the stream. When no more data are available a `StopIteration` exception is raised instead. At this point, the iterator object is exhausted and any further calls to its `next()` method just raise `StopIteration` again. Iterators are required to have an `__iter__()` method that returns the iterator object itself so every iterator is also iterable and may be used in most places where other iterables are accepted. One notable exception is code which attempts multiple iteration passes. A container object (such as a `list`) produces a fresh new iterator each time you pass it to the `iter()` function or use it in a `for` loop. Attempting this with an iterator will just return the same exhausted iterator object used in the previous iteration pass, making it appear like an empty container.

更多信息可查看 [typeiter](#)。

key function –键函数 键函数或称整理函数，是能够返回用于排序或排位的值的可调用对象。例如，`locale.strxfrm()` 可用于生成一个符合特定区域排序约定的排序键。

A number of tools in Python accept key functions to control how elements are ordered or grouped. They include `min()`, `max()`, `sorted()`, `list.sort()`, `heapq.nsmallest()`, `heapq.nlargest()`, and `itertools.groupby()`.

There are several ways to create a key function. For example, the `str.lower()` method can serve as a key function for case insensitive sorts. Alternatively, an ad-hoc key function can be built from a `lambda` expression such as `lambda r: (r[0], r[2])`. Also, the `operator` module provides three key function constructors: `attrgetter()`, `itemgetter()`, and `methodcaller()`. See the Sorting HOW TO for examples of how to create and use key functions.

keyword argument –关键字参数 参见 [argument](#)。

lambda 由一个单独 [expression](#) 构成的匿名内联函数，表达式会在调用时被求值。创建 `lambda` 函数的句法为 `lambda [parameters]: expression`

LBYL “先查看后跳跃”的英文缩写。这种代码编写风格会在进行调用或查找之前显式地检查前提条件。此风格与 [EAFP](#) 方式恰成对比，其特点是大量使用 `if` 语句。

在多线程环境中，LBYL 方式会导致“查看”和“跳跃”之间发生条件竞争风险。例如，以下代码 `if key in mapping: return mapping[key]` 可能由于在检查操作之后其他线程从 `mapping` 中移除了 `key` 而出错。这种问题可通过加锁或使用 [EAFP](#) 方式来解决。

list –列表 Python 内置的一种 *sequence*。虽然名为列表，但更类似于其他语言中的数组而非链接列表，因为访问元素的时间复杂度为 $O(1)$ 。

list comprehension –列表推导式 A compact way to process all or part of the elements in a sequence and return a list with the results. `result = ["0x%02x" % x for x in range(256) if x % 2 == 0]` generates a list of strings containing even hex numbers (0x..) in the range from 0 to 255. The `if` clause is optional. If omitted, all elements in `range(256)` are processed.

loader –加载器 An object that loads a module. It must define a method named `load_module()`. A loader is typically returned by a *finder*. See [PEP 302](#) for details.

magic method –魔法方法 *special method* 的非正式同义词。

mapping –映射 A container object that supports arbitrary key lookups and implements the methods specified in the Mapping or MutableMapping abstract base classes. Examples include `dict`, `collections.defaultdict`, `collections.OrderedDict` and `collections.Counter`.

metaclass –元类 一种用于创建类的类。类定义包含类名、类字典和基类列表。元类负责接受上述三个参数并创建相应的类。大部分面向对象的编程语言都会提供一个默认实现。Python 的特别之处在于可以创建自定义元类。大部分用户永远不需要这个工具，但当需要出现时，元类可提供强大而优雅的解决方案。它们已被用于记录属性访问日志、添加线程安全性、跟踪对象创建、实现单例，以及其他许多任务。

更多详情参见 `metaclasses`。

method 方法 在类内部定义的函数。如果作为该类的实例的一个属性来调用，方法将会获取实例对象作为其第一个 *argument* (通常命名为 `self`)。参见 *function* 和 *nested scope*。

method resolution order –方法解析顺序 方法解析顺序就是在查找成员时搜索全部基类所用的先后顺序。请查看 [Python 2.3 方法解析顺序](#) 了解自 2.3 版起 Python 解析器所用相关算法的详情。

module 模块 此对象是 Python 代码的一种组织单位。各模块具有独立的命名空间，可包含任意 Python 对象。模块可通过 *importing* 操作被加载到 Python 中。

另见 *package*。

MRO 参见 *method resolution order*。

mutable –可变 可变对象可以在其 `id()` 保持固定的情况下改变其取值。另请参见 *immutable*。

named tuple –具名元组 Any tuple-like class whose indexable elements are also accessible using named attributes (for example, `time.localtime()` returns a tuple-like object where the *year* is accessible either with an index such as `t[0]` or with a named attribute like `t.tm_year`).

A named tuple can be a built-in type such as `time.struct_time`, or it can be created with a regular class definition. A full featured named tuple can also be created with the factory function `collections.namedtuple()`. The latter approach automatically provides extra features such as a self-documenting representation like `Employee(name='jones', title='programmer')`.

namespace –命名空间 The place where a variable is stored. Namespaces are implemented as dictionaries. There are the local, global and built-in namespaces as well as nested namespaces in objects (in methods). Namespaces support modularity by preventing naming conflicts. For instance, the functions `__builtin__.open()` and `os.open()` are distinguished by their namespaces. Namespaces also aid readability and maintainability by making it clear which module implements a function. For instance, writing `random.seed()` or `itertools.izip()` makes it clear that those functions are implemented by the `random` and `itertools` modules, respectively.

nested scope –嵌套作用域 The ability to refer to a variable in an enclosing definition. For instance, a function defined inside another function can refer to variables in the outer function. Note that nested scopes work only for reference and not for assignment which will always write to the innermost scope. In contrast, local variables both read and write in the innermost scope. Likewise, global variables read and write to the global namespace.

new-style class –新式类 Any class which inherits from `object`. This includes all built-in types like `list` and `dict`. Only new-style classes can use Python’s newer, versatile features like `__slots__`, descriptors, properties, and `__getattr__()`.

More information can be found in `newstyle`.

object –对象 任何具有状态（属性或值）以及预定义行为（方法）的数据。`object` 也是任何 *new-style class* 的最顶层基类名。

package –包 一种可包含子模块或递归地包含子包的 Python *module*。从技术上说，包是带有 `__path__` 属性的 Python 模块。

parameter –形参 A named entity in a *function* (or method) definition that specifies an *argument* (or in some cases, arguments) that the function can accept. There are four types of parameters:

- *positional-or-keyword*: 位置或关键字，指定一个可以作为位置参数传入也可以作为关键字参数传入的实参。这是默认的形参类型，例如下面的 `foo` 和 `bar`:

```
def func(foo, bar=None): ...
```

- *positional-only*: 仅限位置，指定一个只能按位置传入的参数。Python 中没有定义仅限位置形参的语法。但是一些内置函数有仅限位置形参（比如 `abs()`）。
- *var-positional*: 可变位置，指定可以提供由一个任意数量的位置参数构成的序列（附加在其他形参已接受的位置参数之后）。这种形参可通过在形参名称前加缀 `*` 来定义，例如下面的 `args`:

```
def func(*args, **kwargs): ...
```

- *var-keyword*: 可变关键字，指定可以提供任意数量的关键字参数（附加在其他形参已接受的关键字参数之后）。这种形参可通过在形参名称前加缀 `**` 来定义，例如上面的 `kwargs`。

形参可以同时指定可选和必选参数，也可以为某些可选参数指定默认值。

See also the *argument* glossary entry, the FAQ question on the difference between arguments and parameters, and the function section.

PEP “Python 增强提议”的英文缩写。一个 PEP 就是一份设计文档，用来向 Python 社区提供信息，或描述一个 Python 的新增特性及其进度或环境。PEP 应当提供精确的技术规格和所提议特性的原理说明。

PEP 应被作为提出主要新特性建议、收集社区对特定问题反馈以及为必须加入 Python 的设计决策编写文档的首选机制。PEP 的作者有责任在社区内部建立共识，并应将不同意见也记入文档。

参见 **PEP 1**。

positional argument –位置参数 参见 *argument*。

Python 3000 Python 3.x 发布路线的昵称（这个名字在版本 3 的发布还遥遥无期的时候就已出现了）。有时也被缩写为“Py3k”。

Pythonic 指一个思路或一段代码紧密遵循了 Python 语言最常用的风格和理念，而不是使用其他语言中通用的概念来实现代码。例如，Python 的常用风格是使用 `for` 语句循环来遍历一个可迭代对象中的所有元素。许多其他语言没有这样的结构，因此不熟悉 Python 的人有时会选择使用一个数字计数器：

```
for i in range(len(food)):
    print food[i]
```

而相应的更简洁更 Pythonic 的方法是这样的：

```
for piece in food:
    print piece
```

reference count –引用计数 对特定对象的引用的数量。当一个对象的引用计数降为零时，所分配资源将被释放。引用计数对 Python 代码来说通常是不可见的，但它是 CPython 实现的一个关键元素。sys 模块定义了一个 `getrefcount()` 函数，程序员可调用它来返回特定对象的引用计数。

__slots__ A declaration inside a *new-style class* that saves memory by pre-declaring space for instance attributes and eliminating instance dictionaries. Though popular, the technique is somewhat tricky to get right and is best reserved for rare cases where there are large numbers of instances in a memory-critical application.

sequence –序列 An *iterable* which supports efficient element access using integer indices via the `__getitem__()` special method and defines a `len()` method that returns the length of the sequence. Some built-in sequence types are `list`, `str`, `tuple`, and `unicode`. Note that `dict` also supports `__getitem__()` and `__len__()`, but is considered a mapping rather than a sequence because the lookups use arbitrary *immutable* keys rather than integers.

slice –切片 An object usually containing a portion of a *sequence*. A slice is created using the subscript notation, `[]` with colons between numbers when several are given, such as in `variable_name[1:3:5]`. The bracket (subscript) notation uses `slice` objects internally (or in older versions, `__getslice__()` and `__setslice__()`).

special method –特殊方法 一种由 Python 隐式调用的方法，用来对某个类型执行特定操作例如相加等等。这种方法名称的首尾都为双下划线。特殊方法的文档参见 `specialnames`。

statement –语句 语句是程序段（一个代码“块”）的组成单位。一条语句可以是一个 *expression* 或某个带有关键字的结构，例如 `if`、`while` 或 `for`。

struct sequence A tuple with named elements. Struct sequences expose an interface similar to *named tuple* in that elements can be accessed either by index or as an attribute. However, they do not have any of the named tuple methods like `_make()` or `_asdict()`. Examples of struct sequences include `sys.float_info` and the return value of `os.stat()`.

triple-quoted string –三引号字符串 首尾各带三个连续双引号 (") 或者单引号 (') 的字符串。它们在功能上与首尾各用一个引号标注的字符串没有什么不同，但是有多种用处。它们允许你在字符串内包含未经转义的单引号和双引号，并且可以跨越多行而无需使用连接符，在编写文档字符串时特别好用。

type –类型 类型决定一个 Python 对象属于什么种类；每个对象都具有一种类型。要知道对象的类型，可以访问它的 `__class__` 属性，或是通过 `type(obj)` 来获取。

universal newlines –通用换行 A manner of interpreting text streams in which all of the following are recognized as ending a line: the Unix end-of-line convention `'\n'`, the Windows convention `'\r\n'`, and the old Macintosh convention `'\r'`. See [PEP 278](#) and [PEP 3116](#), as well as `str.splitlines()` for an additional use.

virtual environment –虚拟环境 一种采用协作式隔离的运行环境，允许 Python 用户和应用程序在安装和升级 Python 分发版时不会干扰到同一系统上运行的其他 Python 应用程序的行为。

virtual machine –虚拟机 一台完全通过软件定义的计算机。Python 虚拟机可执行字节码编译器所生成的 *bytecode*。

Zen of Python –Python 之禅 列出 Python 设计的原则与哲学，有助于理解与使用这种语言。查看其具体内容可在交互模式提示符中输入 `"import this"`。

文档说明

这些文档生成自 `reStructuredText` 原文档，由 `Sphinx`（一个专门为 Python 文档写的文档生成器）创建。

本文档和它所用工具链的开发完全是由志愿者完成的，这和 Python 本身一样。如果您想参与进来，请阅读 `reporting-bugs` 了解如何参与。我们随时欢迎新的志愿者！

特别鸣谢：

- Fred L. Drake, Jr., 创造了用于早期 Python 文档的工具链，以及撰写了非常多的文档；
- `Docutils` 软件包 项目，创建了 `reStructuredText` 文本格式和 `Docutils` 软件套件；
- Fredrik Lundh, `Sphinx` 从他的 `Alternative Python Reference` 项目中获得了很多好的想法。

B.1 Python 文档的贡献者

有很多对 Python 语言，Python 标准库和 Python 文档有贡献的人，随 Python 源代码发布的 `Misc/ACKS` 文件列出了部分贡献者。

有了 Python 社区的输入和贡献，Python 才有了如此出色的文档 - 谢谢你们！

历史和许可证

C.1 该软件的历史

Python 由荷兰数学和计算机科学研究学会（CWI，见 <https://www.cwi.nl/>）的 Guido van Rossum 于 1990 年代初设计，作为一门叫做 ABC 的语言的替代品。尽管 Python 包含了许多来自其他人的贡献，Guido 仍是其主要作者。

1995 年，Guido 在弗吉尼亚州的国家创新研究公司（CNRI，见 <https://www.cnri.reston.va.us/>）继续他在 Python 上的工作，并在那里发布了该软件的多个版本。

2000 年五月，Guido 和 Python 核心开发团队转到 BeOpen.com 并组建了 BeOpen PythonLabs 团队。同年十月，PythonLabs 团队转到 Digital Creations（现为 Zope Corporation；见 <https://www.zope.org/>）。2001 年，Python 软件基金会（PSF，见 <https://www.python.org/psf/>）成立，这是一个专为拥有 Python 相关知识产权而创建的非营利组织。Zope Corporation 现在是 PSF 的赞助成员。

所有的 Python 版本都是开源的（有关开源的定义参阅 <https://opensource.org/>）。历史上，绝大多数 Python 版本是 GPL 兼容的；下表总结了各个版本情况。

发布版本	源自	年份	所有者	GPL 兼容？
0.9.0 至 1.2	n/a	1991-1995	CWI	是
1.3 至 1.5.2	1.2	1995-1999	CNRI	是
1.6	1.5.2	2000	CNRI	否
2.0	1.6	2000	BeOpen.com	否
1.6.1	1.6	2001	CNRI	否
2.1	2.0+1.6.1	2001	PSF	否
2.0.1	2.0+1.6.1	2001	PSF	是
2.1.1	2.1+2.0.1	2001	PSF	是
2.1.2	2.1.1	2002	PSF	是
2.1.3	2.1.2	2002	PSF	是
2.2 及更高	2.1.1	2001 至今	PSF	是

注解： GPL 兼容并不意味着 Python 在 GPL 下发布。与 GPL 不同，所有 Python 许可证都允许您分发修改后

的版本，而无需开源所做的更改。GPL 兼容的许可证使得 Python 可以与其它在 GPL 下发布的软件结合使用；但其它的许可证则不行。

感谢众多在 Guido 指导下工作的外部志愿者，使得这些发布成为可能。

C.2 获取或以其他方式使用 Python 的条款和条件

C.2.1 用于 PYTHON 2.7.18 的 PSF 许可协议

1. This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"),
→and
the Individual or Organization ("Licensee") accessing and otherwise using
→Python
2.7.18 software in source or binary form and its associated documentation.
2. Subject to the terms and conditions of this License Agreement, PSF hereby
grants Licensee a nonexclusive, royalty-free, world-wide license to
→reproduce,
analyze, test, perform and/or display publicly, prepare derivative works,
distribute, and otherwise use Python 2.7.18 alone or in any derivative
version, provided, however, that PSF's License Agreement and PSF's notice
→of
copyright, i.e., "Copyright © 2001-2020 Python Software Foundation; All
→Rights
Reserved" are retained in Python 2.7.18 alone or in any derivative version
prepared by Licensee.
3. In the event Licensee prepares a derivative work that is based on or
incorporates Python 2.7.18 or any part thereof, and wants to make the
derivative work available to others as provided herein, then Licensee
→hereby
agrees to include in any such work a brief summary of the changes made to
→Python
2.7.18.
4. PSF is making Python 2.7.18 available to Licensee on an "AS IS" basis.
PSF MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF
EXAMPLE, BUT NOT LIMITATION, PSF MAKES NO AND DISCLAIMS ANY REPRESENTATION
→OR
WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT
→THE
USE OF PYTHON 2.7.18 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.
5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 2.7.18
FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT
→OF
MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 2.7.18, OR ANY
→DERIVATIVE
THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.

6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.
7. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between PSF and Licensee. This License Agreement does not grant permission to use PSF trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.
8. By copying, installing or otherwise using Python 2.7.18, Licensee agrees to be bound by the terms and conditions of this License Agreement.

C.2.2 用于 PYTHON 2.0 的 BEOPEN.COM 许可协议

BEOPEN PYTHON 开源许可协议第 1 版

1. This LICENSE AGREEMENT is between BeOpen.com ("BeOpen"), having an office at 160 Saratoga Avenue, Santa Clara, CA 95051, and the Individual or Organization ("Licensee") accessing and otherwise using this software in source or binary form and its associated documentation ("the Software").
2. Subject to the terms and conditions of this BeOpen Python License Agreement, BeOpen hereby grants Licensee a non-exclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use the Software alone or in any derivative version, provided, however, that the BeOpen Python License is retained in the Software, alone or in any derivative version prepared by Licensee.
3. BeOpen is making the Software available to Licensee on an "AS IS" basis. BEOPEN MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, BEOPEN MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF THE SOFTWARE WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.
4. BEOPEN SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF THE SOFTWARE FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE SOFTWARE, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.
5. This License Agreement will automatically terminate upon a material breach of its terms and conditions.
6. This License Agreement shall be governed by and interpreted in all respects by the law of the State of California, excluding conflict of law provisions. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between BeOpen and Licensee. This License Agreement does not grant permission to use BeOpen trademarks or trade names in a trademark sense to endorse or promote products or services of Licensee, or any third party. As an exception, the "BeOpen Python" logos available at <http://www.pythonlabs.com/logos.html> may be used according to the permissions

(下页继续)

(续上页)

granted on that web page.

7. By copying, installing or otherwise using the software, Licensee agrees to be bound by the terms and conditions of this License Agreement.

C.2.3 用于 PYTHON 1.6.1 的 CNRI 许可协议

1. This LICENSE AGREEMENT is between the Corporation for National Research Initiatives, having an office at 1895 Preston White Drive, Reston, VA 20191 ("CNRI"), and the Individual or Organization ("Licensee") accessing and otherwise using Python 1.6.1 software in source or binary form and its associated documentation.
2. Subject to the terms and conditions of this License Agreement, CNRI hereby grants Licensee a nonexclusive, royalty-free, world-wide license to reproduce, analyze, test, perform and/or display publicly, prepare derivative works, distribute, and otherwise use Python 1.6.1 alone or in any derivative version, provided, however, that CNRI's License Agreement and CNRI's notice of copyright, i.e., "Copyright © 1995-2001 Corporation for National Research Initiatives; All Rights Reserved" are retained in Python 1.6.1 alone or in any derivative version prepared by Licensee. Alternately, in lieu of CNRI's License Agreement, Licensee may substitute the following text (omitting the quotes): "Python 1.6.1 is made available subject to the terms and conditions in CNRI's License Agreement. This Agreement together with Python 1.6.1 may be located on the Internet using the following unique, persistent identifier (known as a handle): 1895.22/1013. This Agreement may also be obtained from a proxy server on the Internet using the following URL: <http://hdl.handle.net/1895.22/1013>."
3. In the event Licensee prepares a derivative work that is based on or incorporates Python 1.6.1 or any part thereof, and wants to make the derivative work available to others as provided herein, then Licensee hereby agrees to include in any such work a brief summary of the changes made to Python 1.6.1.
4. CNRI is making Python 1.6.1 available to Licensee on an "AS IS" basis. CNRI MAKES NO REPRESENTATIONS OR WARRANTIES, EXPRESS OR IMPLIED. BY WAY OF EXAMPLE, BUT NOT LIMITATION, CNRI MAKES NO AND DISCLAIMS ANY REPRESENTATION OR WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PARTICULAR PURPOSE OR THAT THE USE OF PYTHON 1.6.1 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.
5. CNRI SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON 1.6.1 FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON 1.6.1, OR ANY DERIVATIVE THEREOF, EVEN IF ADVISED OF THE POSSIBILITY THEREOF.
6. This License Agreement will automatically terminate upon a material breach of its terms and conditions.
7. This License Agreement shall be governed by the federal intellectual property law of the United States, including without limitation the federal copyright law, and, to the extent such U.S. federal law does not apply, by the law of the Commonwealth of Virginia, excluding Virginia's conflict of law provisions. Notwithstanding the foregoing, with regard to derivative works based on Python 1.6.1 that incorporate non-separable material that was previously distributed under the GNU General Public License (GPL), the law of the Commonwealth of

(下页继续)

(续上页)

Virginia shall govern this License Agreement only as to issues arising under or with respect to Paragraphs 4, 5, and 7 of this License Agreement. Nothing in this License Agreement shall be deemed to create any relationship of agency, partnership, or joint venture between CNRI and Licensee. This License Agreement does not grant permission to use CNRI trademarks or trade name in a trademark sense to endorse or promote products or services of Licensee, or any third party.

8. By clicking on the "ACCEPT" button where indicated, or by copying, installing or otherwise using Python 1.6.1, Licensee agrees to be bound by the terms and conditions of this License Agreement.

C.2.4 用于 PYTHON 0.9.0 至 1.2 的 CWI 许可协议

Copyright © 1991 - 1995, Stichting Mathematisch Centrum Amsterdam, The Netherlands. All rights reserved.

Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation, and that the name of Stichting Mathematisch Centrum or CWI not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission.

STICHTING MATHEMATISCH CENTRUM DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN NO EVENT SHALL STICHTING MATHEMATISCH CENTRUM BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

C.3 被收录软件的许可证与鸣谢

本节是 Python 发行版中收录的第三方软件的许可和致谢清单，该清单是不完整且不断增长的。

C.3.1 Mersenne Twister

`_random` 模块包含基于 <http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/MT2002/emt19937ar.html> 下载的代码。以下是原始代码的完整注释（声明）：

A C-program for MT19937, with initialization improved 2002/1/26.
Coded by Takuji Nishimura and Makoto Matsumoto.

Before using, initialize the state by using `init_genrand(seed)`
or `init_by_array(init_key, key_length)`.

Copyright (C) 1997 - 2002, Makoto Matsumoto and Takuji Nishimura,
All rights reserved.

(下页继续)

(续上页)

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The names of its contributors may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Any feedback is very welcome.

<http://www.math.sci.hiroshima-u.ac.jp/~m-mat/MT/emt.html>

email: m-mat @ math.sci.hiroshima-u.ac.jp (remove space)

C.3.2 套接字

socket 模块使用 `getaddrinfo()` 和 `getnameinfo()` 函数, 这些函数源代码在 WIDE 项目 (<http://www.wide.ad.jp/>) 的单独源文件中。

Copyright (C) 1995, 1996, 1997, and 1998 WIDE Project.
All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. Neither the name of the project nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE PROJECT AND CONTRIBUTORS ``AS IS'' AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE

(下页继续)

(续上页)

```

IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED.  IN NO EVENT SHALL THE PROJECT OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.

```

C.3.3 Floating point exception control

The source for the `fpectl` module includes the following notice:

```

-----
/                               Copyright (c) 1996.                               \
|                               The Regents of the University of California.          |
|                               All rights reserved.                                |
|                                                                                     |
|  Permission to use, copy, modify, and distribute this software for                |
|  any purpose without fee is hereby granted, provided that this en-               |
|  tire notice is included in all copies of any software which is or               |
|  includes a copy or modification of this software and in all                   |
|  copies of the supporting documentation for such software.                      |
|                                                                                     |
|  This work was produced at the University of California, Lawrence                 |
|  Livermore National Laboratory under contract no. W-7405-ENG-48                 |
|  between the U.S. Department of Energy and The Regents of the                 |
|  University of California for the operation of UC LLNL.                        |
|                                                                                     |
|                               DISCLAIMER                                             |
|                                                                                     |
|  This software was prepared as an account of work sponsored by an                |
|  agency of the United States Government. Neither the United States              |
|  Government nor the University of California nor any of their em-               |
|  ployees, makes any warranty, express or implied, or assumes any                |
|  liability or responsibility for the accuracy, completeness, or                 |
|  usefulness of any information, apparatus, product, or process                  |
|  disclosed, or represents that its use would not infringe                      |
|  privately-owned rights. Reference herein to any specific commer-               |
|  cial products, process, or service by trade name, trademark,                   |
|  manufacturer, or otherwise, does not necessarily constitute or                 |
|  imply its endorsement, recommendation, or favoring by the United              |
|  States Government or the University of California. The views and               |
|  opinions of authors expressed herein do not necessarily state or               |
|  reflect those of the United States Government or the University                 |
|  of California, and shall not be used for advertising or product                |
|  endorsement purposes.                                                           |
\                                                                                     /
-----

```

C.3.4 MD5 message digest algorithm

The source code for the md5 module contains the following notice:

```
Copyright (C) 1999, 2002 Aladdin Enterprises. All rights reserved.

This software is provided 'as-is', without any express or implied
warranty. In no event will the authors be held liable for any damages
arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it
freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not
   claim that you wrote the original software. If you use this software
   in a product, an acknowledgment in the product documentation would be
   appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be
   misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

L. Peter Deutsch
ghost@aladdin.com

Independent implementation of MD5 (RFC 1321).

This code implements the MD5 Algorithm defined in RFC 1321, whose
text is available at
    http://www.ietf.org/rfc/rfc1321.txt
The code is derived from the text of the RFC, including the test suite
(section A.5) but excluding the rest of Appendix A. It does not include
any code or documentation that is identified in the RFC as being
copyrighted.

The original and principal author of md5.h is L. Peter Deutsch
<ghost@aladdin.com>. Other authors are noted in the change history
that follows (in reverse chronological order):

2002-04-13 lpd Removed support for non-ANSI compilers; removed
    references to Ghostscript; clarified derivation from RFC 1321;
    now handles byte order either statically or dynamically.
1999-11-04 lpd Edited comments slightly for automatic TOC extraction.
1999-10-18 lpd Fixed typo in header comment (ansi2knr rather than md5);
    added conditionalization for C++ compilation from Martin
    Purschke <purschke@bnl.gov>.
1999-05-03 lpd Original version.
```

C.3.5 异步套接字服务

asynchat and asyncore 模块包含以下声明:

```
Copyright 1996 by Sam Rushing
```

```
    All Rights Reserved
```

```
Permission to use, copy, modify, and distribute this software and
its documentation for any purpose and without fee is hereby
granted, provided that the above copyright notice appear in all
copies and that both that copyright notice and this permission
notice appear in supporting documentation, and that the name of Sam
Rushing not be used in advertising or publicity pertaining to
distribution of the software without specific, written prior
permission.
```

```
SAM RUSHING DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE,
INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS, IN
NO EVENT SHALL SAM RUSHING BE LIABLE FOR ANY SPECIAL, INDIRECT OR
CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS
OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT,
NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN
CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
```

C.3.6 Cookie 管理

The Cookie module contains the following notice:

```
Copyright 2000 by Timothy O'Malley <timo@alum.mit.edu>
```

```
    All Rights Reserved
```

```
Permission to use, copy, modify, and distribute this software
and its documentation for any purpose and without fee is hereby
granted, provided that the above copyright notice appear in all
copies and that both that copyright notice and this permission
notice appear in supporting documentation, and that the name of
Timothy O'Malley not be used in advertising or publicity
pertaining to distribution of the software without specific, written
prior permission.
```

```
Timothy O'Malley DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS
SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY
AND FITNESS, IN NO EVENT SHALL Timothy O'Malley BE LIABLE FOR
ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS,
WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS
ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR
PERFORMANCE OF THIS SOFTWARE.
```

C.3.7 执行追踪

trace 模块包含以下声明:

```
portions copyright 2001, Autonomous Zones Industries, Inc., all rights...
err... reserved and offered to the public under the terms of the
Python 2.2 license.
Author: Zooko O'Whielacronx
http://zooko.com/
mailto:zooko@zooko.com
```

```
Copyright 2000, Mojam Media, Inc., all rights reserved.
Author: Skip Montanaro
```

```
Copyright 1999, Bioreason, Inc., all rights reserved.
Author: Andrew Dalke
```

```
Copyright 1995-1997, Automatrix, Inc., all rights reserved.
Author: Skip Montanaro
```

```
Copyright 1991-1995, Stichting Mathematisch Centrum, all rights reserved.
```

```
Permission to use, copy, modify, and distribute this Python software and
its associated documentation for any purpose without fee is hereby
granted, provided that the above copyright notice appears in all copies,
and that both that copyright notice and this permission notice appear in
supporting documentation, and that the name of neither Automatrix,
Bioreason or Mojam Media be used in advertising or publicity pertaining to
distribution of the software without specific, written prior permission.
```

C.3.8 UUencode 与 UUdecode 函数

uu 模块包含以下声明:

```
Copyright 1994 by Lance Ellinghouse
Cathedral City, California Republic, United States of America.
    All Rights Reserved
Permission to use, copy, modify, and distribute this software and its
documentation for any purpose and without fee is hereby granted,
provided that the above copyright notice appear in all copies and that
both that copyright notice and this permission notice appear in
supporting documentation, and that the name of Lance Ellinghouse
not be used in advertising or publicity pertaining to distribution
of the software without specific, written prior permission.
LANCE ELLINGHOUSE DISCLAIMS ALL WARRANTIES WITH REGARD TO
THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND
FITNESS, IN NO EVENT SHALL LANCE ELLINGHOUSE CENTRUM BE LIABLE
FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT
OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

Modified by Jack Jansen, CWI, July 1995:
- Use binascii module to do the actual line-by-line conversion
  between ascii and binary. This results in a 1000-fold speedup. The C
```

(下页继续)

(续上页)

```
version is still 5 times faster, though.
- Arguments more compliant with Python standard
```

C.3.9 XML 远程过程调用

The xmlrpclib module contains the following notice:

```
The XML-RPC client interface is

Copyright (c) 1999-2002 by Secret Labs AB
Copyright (c) 1999-2002 by Fredrik Lundh

By obtaining, using, and/or copying this software and/or its
associated documentation, you agree that you have read, understood,
and will comply with the following terms and conditions:

Permission to use, copy, modify, and distribute this software and
its associated documentation for any purpose and without fee is
hereby granted, provided that the above copyright notice appears in
all copies, and that both that copyright notice and this permission
notice appear in supporting documentation, and that the name of
Secret Labs AB or the author not be used in advertising or publicity
pertaining to distribution of the software without specific, written
prior permission.

SECRET LABS AB AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD
TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANT-
ABILITY AND FITNESS.  IN NO EVENT SHALL SECRET LABS AB OR THE AUTHOR
BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY
DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS,
WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS
ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE
OF THIS SOFTWARE.
```

C.3.10 test_epoll

The test_epoll contains the following notice:

```
Copyright (c) 2001-2006 Twisted Matrix Laboratories.

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be
included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
```

(下页继续)

(续上页)

```
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE
LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION
OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION
WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

C.3.11 Select queue

The select and contains the following notice for the queue interface:

```
Copyright (c) 2000 Doug White, 2006 James Knight, 2007 Christian Heimes
All rights reserved.
```

```
Redistribution and use in source and binary forms, with or without
modification, are permitted provided that the following conditions
are met:
```

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

```
THIS SOFTWARE IS PROVIDED BY THE AUTHOR AND CONTRIBUTORS ``AS IS'' AND
ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
SUCH DAMAGE.
```

C.3.12 strtod and dtoa

Python/dtoa.c 文件提供了 C 语言的 dtoa 和 strtod 函数，用于将 C 语言的双精度型和字符串进行转换，该文件由 David M. Gay 的同名文件派生而来，当前可从 <http://www.netlib.org/fp/> 下载。2009 年 3 月 16 日检索到的原始文件包含以下版权和许可声明：

```
/*
 * *****
 *
 * The author of this software is David M. Gay.
 *
 * Copyright (c) 1991, 2000, 2001 by Lucent Technologies.
 *
 * Permission to use, copy, modify, and distribute this software for any
 * purpose without fee is hereby granted, provided that this entire notice
 * is included in all copies of any software which is or includes a copy
 * or modification of this software and in all copies of the supporting
 * documentation for such software.
 *
 * THIS SOFTWARE IS BEING PROVIDED "AS IS", WITHOUT ANY EXPRESS OR IMPLIED
```

(下页继续)

(续上页)

```
* WARRANTY.  IN PARTICULAR, NEITHER THE AUTHOR NOR LUCENT MAKES ANY
* REPRESENTATION OR WARRANTY OF ANY KIND CONCERNING THE MERCHANTABILITY
* OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR PURPOSE.
*
*****/
```

C.3.13 OpenSSL

如果操作系统可用, 则 `hashlib`, `posix`, `ssl`, `crypt` 模块使用 **OpenSSL** 库来提高性能。此外, 适用于 Python 的 Windows 和 Mac OS X 安装程序可能包括 **OpenSSL** 库的拷贝, 所以在此处也列出了 **OpenSSL** 许可证的拷贝:

LICENSE ISSUES

=====

The OpenSSL toolkit stays under a dual license, i.e. both the conditions of the OpenSSL License and the original SSLeay license apply to the toolkit. See below for the actual license texts. Actually both licenses are BSD-style Open Source licenses. In case of any license issues related to OpenSSL please contact openssl-core@openssl.org.

OpenSSL License

```
/* =====
 * Copyright (c) 1998-2008 The OpenSSL Project.  All rights reserved.
 *
 * Redistribution and use in source and binary forms, with or without
 * modification, are permitted provided that the following conditions
 * are met:
 *
 * 1. Redistributions of source code must retain the above copyright
 *    notice, this list of conditions and the following disclaimer.
 *
 * 2. Redistributions in binary form must reproduce the above copyright
 *    notice, this list of conditions and the following disclaimer in
 *    the documentation and/or other materials provided with the
 *    distribution.
 *
 * 3. All advertising materials mentioning features or use of this
 *    software must display the following acknowledgment:
 *    "This product includes software developed by the OpenSSL Project
 *    for use in the OpenSSL Toolkit. (http://www.openssl.org/)"
 *
 * 4. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to
 *    endorse or promote products derived from this software without
 *    prior written permission. For written permission, please contact
 *    openssl-core@openssl.org.
 *
 * 5. Products derived from this software may not be called "OpenSSL"
 *    nor may "OpenSSL" appear in their names without prior written
 *    permission of the OpenSSL Project.
 *
 * 6. Redistributions of any form whatsoever must retain the following
```

(下页继续)

(续上页)

```

*   acknowledgment:
*   "This product includes software developed by the OpenSSL Project
*   for use in the OpenSSL Toolkit (http://www.openssl.org/)"
*
* THIS SOFTWARE IS PROVIDED BY THE OpenSSL PROJECT ``AS IS'' AND ANY
* EXPRESSED OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR
* PURPOSE ARE DISCLAIMED.  IN NO EVENT SHALL THE OpenSSL PROJECT OR
* ITS CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL,
* SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT
* NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;
* LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT,
* STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE)
* ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED
* OF THE POSSIBILITY OF SUCH DAMAGE.
* =====
*
* This product includes cryptographic software written by Eric Young
* (eay@cryptsoft.com).  This product includes software written by Tim
* Hudson (tjh@cryptsoft.com).
*
*/

```

Original SSLeay License

```

/* Copyright (C) 1995-1998 Eric Young (eay@cryptsoft.com)
* All rights reserved.
*
* This package is an SSL implementation written
* by Eric Young (eay@cryptsoft.com).
* The implementation was written so as to conform with Netscapes SSL.
*
* This library is free for commercial and non-commercial use as long as
* the following conditions are aheared to.  The following conditions
* apply to all code found in this distribution, be it the RC4, RSA,
* lhash, DES, etc., code; not just the SSL code.  The SSL documentation
* included with this distribution is covered by the same copyright terms
* except that the holder is Tim Hudson (tjh@cryptsoft.com).
*
* Copyright remains Eric Young's, and as such any Copyright notices in
* the code are not to be removed.
* If this package is used in a product, Eric Young should be given attribution
* as the author of the parts of the library used.
* This can be in the form of a textual message at program startup or
* in documentation (online or textual) provided with the package.
*
* Redistribution and use in source and binary forms, with or without
* modification, are permitted provided that the following conditions
* are met:
* 1. Redistributions of source code must retain the copyright
*    notice, this list of conditions and the following disclaimer.
* 2. Redistributions in binary form must reproduce the above copyright
*    notice, this list of conditions and the following disclaimer in the
*    documentation and/or other materials provided with the distribution.

```

(下页继续)

(续上页)

```

* 3. All advertising materials mentioning features or use of this software
* must display the following acknowledgement:
* "This product includes cryptographic software written by
* Eric Young (eay@cryptsoft.com)"
* The word 'cryptographic' can be left out if the routines from the library
* being used are not cryptographic related :-).
* 4. If you include any Windows specific code (or a derivative thereof) from
* the apps directory (application code) you must include an acknowledgement:
* "This product includes software written by Tim Hudson (tjh@cryptsoft.com)"
*
* THIS SOFTWARE IS PROVIDED BY ERIC YOUNG ``AS IS'' AND
* ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE
* IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE
* ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR OR CONTRIBUTORS BE LIABLE
* FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL
* DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS
* OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION)
* HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT
* LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY
* OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF
* SUCH DAMAGE.
*
* The licence and distribution terms for any publically available version or
* derivative of this code cannot be changed. i.e. this code cannot simply be
* copied and put under another distribution licence
* [including the GNU Public Licence.]
*/

```

C.3.14 expat

除非使用 `--with-system-expat` 配置了构建, 否则 `pyexpat` 扩展都是用包含 `expat` 源的拷贝构建的:

```

Copyright (c) 1998, 1999, 2000 Thai Open Source Software Center Ltd
and Clark Cooper

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
"Software"), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be included
in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT.
IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY
CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT,
TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE
SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

```

C.3.15 libffi

除非使用 `--with-system-libffi` 配置了构建, 否则 `_ctypes` 扩展都是包含 `libffi` 源的拷贝构建的:

```
Copyright (c) 1996-2008 Red Hat, Inc and others.

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the
`Software'), to deal in the Software without restriction, including
without limitation the rights to use, copy, modify, merge, publish,
distribute, sublicense, and/or sell copies of the Software, and to
permit persons to whom the Software is furnished to do so, subject to
the following conditions:

The above copyright notice and this permission notice shall be included
in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED ``AS IS'', WITHOUT WARRANTY OF ANY KIND,
EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF
MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND
NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT
HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER
DEALINGS IN THE SOFTWARE.
```

C.3.16 zlib

如果系统上找到的 `zlib` 版本太旧而无法用于构建, 则使用包含 `zlib` 源代码的拷贝来构建 `zlib` 扩展:

```
Copyright (C) 1995-2010 Jean-loup Gailly and Mark Adler

This software is provided 'as-is', without any express or implied
warranty. In no event will the authors be held liable for any damages
arising from the use of this software.

Permission is granted to anyone to use this software for any purpose,
including commercial applications, and to alter it and redistribute it
freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not
   claim that you wrote the original software. If you use this software
   in a product, an acknowledgment in the product documentation would be
   appreciated but is not required.

2. Altered source versions must be plainly marked as such, and must not be
   misrepresented as being the original software.

3. This notice may not be removed or altered from any source distribution.

Jean-loup Gailly          Mark Adler
jloup@gzip.org            madler@alumni.caltech.edu
```

APPENDIX D

Copyright

Python 与这份文档：

Copyright © 2001-2020 Python Software Foundation。保留所有权利。

版权所有 © 2000 BeOpen.com。保留所有权利。

版权所有 © 1995-2000 Corporation for National Research Initiatives。保留所有权利。

版权所有 © 1991-1995 Stichting Mathematisch Centrum。保留所有权利。

有关完整的许可证和许可信息，参见[历史](#)和[许可证](#)。

非字母

..., [13](#)
2to3, [13](#)
>>>, [13](#)
__future__, [15](#)
__slots__, [20](#)

A

abstract base class -- 抽象基类, [13](#)
argument -- 参数, [13](#)
attribute -- 属性, [14](#)

B

BDFL, [14](#)
bytecode -- 字节码, [14](#)
bytes-like object -- 字节类对象, [14](#)

C

class -- 类, [14](#)
classic class, [14](#)
coercion -- 强制类型转换, [14](#)
complex number -- 复数, [14](#)
context manager -- 上下文管理器, [14](#)
CPython, [14](#)

D

decorator -- 装饰器, [14](#)
descriptor -- 描述器, [15](#)
dictionary -- 字典, [15](#)
dictionary view -- 字典视图, [15](#)
docstring -- 文档字符串, [15](#)
duck-typing -- 鸭子类型, [15](#)

E

EAFP, [15](#)
expression -- 表达式, [15](#)
extension module -- 扩展模块, [15](#)

F

file object -- 文件对象, [15](#)
file-like object -- 文件类对象, [15](#)
finder -- 查找器, [15](#)
floor division -- 向下取整除法, [15](#)
function -- 函数, [15](#)

G

garbage collection -- 垃圾回收, [16](#)
generator, [16](#)
generator -- 生成器, [16](#)
generator expression, [16](#)
generator expression -- 生成器表达式, [16](#)
GIL, [16](#)
global interpreter lock -- 全局解释器锁, [16](#)

H

hashable -- 可哈希, [16](#)

I

IDLE, [16](#)
immutable -- 不可变, [16](#)
importer -- 导入器, [17](#)
importing -- 导入, [17](#)
integer division, [16](#)
interactive -- 交互, [17](#)
interpreted -- 解释型, [17](#)
iterable -- 可迭代对象, [17](#)
iterator -- 迭代器, [17](#)

K

key function -- 键函数, [17](#)
keyword argument -- 关键字参数, [17](#)

L

lambda, [17](#)
LBYL, [17](#)
list -- 列表, [18](#)

list comprehension -- 列表推导式, [18](#)
loader -- 加载器, [18](#)

M

magic
 method, [18](#)
magic method -- 魔术方法, [18](#)
mapping -- 映射, [18](#)
metaclass -- 元类, [18](#)
method
 magic, [18](#)
 special, [20](#)
method resolution order -- 方法解析顺序, [18](#)
method 方法, [18](#)
module 模块, [18](#)
MRO, [18](#)
mutable -- 可变, [18](#)

N

named tuple -- 具名元组, [18](#)
namespace -- 命名空间, [18](#)
nested scope -- 嵌套作用域, [18](#)
new-style class -- 新式类, [19](#)

O

object -- 对象, [19](#)

P

package -- 包, [19](#)
parameter -- 形参, [19](#)
PEP, [19](#)
positional argument -- 位置参数, [19](#)
Python 3000, [19](#)
Python 提高建议
 PEP 1, [19](#)
 PEP 238, [15](#)
 PEP 278, [20](#)
 PEP 302, [15](#), [18](#)
 PEP 343, [14](#)
 PEP 427, [3](#)
 PEP 3116, [20](#)
Pythonic, [19](#)

R

reference count -- 引用计数, [20](#)

S

sequence -- 序列, [20](#)
slice -- 切片, [20](#)
special
 method, [20](#)
special method -- 特殊方法, [20](#)

statement -- 语句, [20](#)
struct sequence, [20](#)

T

triple-quoted string -- 三引号字符串, [20](#)
type -- 类型, [20](#)

U

universal newlines -- 通用换行, [20](#)

V

virtual environment -- 虚拟环境, [20](#)
virtual machine -- 虚拟机, [20](#)

Z

Zen of Python -- Python 之禅, [20](#)