

---

# Installing Python Modules

发行版本 3.13.1

Guido van Rossum and the Python development team

一月 29, 2025

Python Software Foundation  
Email: [docs@python.org](mailto:docs@python.org)



<b>1</b>	<b>关键术语</b>	<b>3</b>
<b>2</b>	<b>基本使用</b>	<b>5</b>
<b>3</b>	<b>我应如何... ?</b>	<b>7</b>
3.1	... 在 Python 3.4 之前的 Python 版本中安装 pip ?	7
3.2	... 只为当前用户安装软件包?	7
3.3	... 安装科学计算类 Python 软件包?	7
3.4	... 使用并行安装的多个 Python 版本?	7
<b>4</b>	<b>常见的安装问题</b>	<b>9</b>
4.1	在 Linux 的系统 Python 版本上安装	9
4.2	未安装 pip	9
4.3	安装二进制编译扩展	9
<b>A</b>	<b>术语对照表</b>	<b>11</b>
<b>B</b>	<b>关于本文档</b>	<b>27</b>
B.1	Python 文档的贡献者	27
<b>C</b>	<b>历史和许可证</b>	<b>29</b>
C.1	该软件的历史	29
C.2	获取或以其他方式使用 Python 的条款和条件	30
C.2.1	PYTHON SOFTWARE FOUNDATION LICENSE VERSION 2	30
C.2.2	用于 PYTHON 2.0 的 BEOPEN.COM 许可协议	31
C.2.3	用于 PYTHON 1.6.1 的 CNRI 许可协议	31
C.2.4	用于 PYTHON 0.9.0 至 1.2 的 CWI 许可协议	32
C.2.5	ZERO-CLAUSE BSD LICENSE FOR CODE IN THE PYTHON DOCUMENTATION	33
C.3	收录软件的许可与鸣谢	33
C.3.1	Mersenne Twister	33
C.3.2	套接字	34
C.3.3	异步套接字服务	34
C.3.4	Cookie 管理	35
C.3.5	执行追踪	35
C.3.6	UUencode 与 UUdecode 函数	36
C.3.7	XML 远程过程调用	36
C.3.8	test_epoll	37
C.3.9	Select kqueue	37
C.3.10	SipHash24	38
C.3.11	strtod 和 dtoa	38
C.3.12	OpenSSL	38

C.3.13	expat . . . . .	41
C.3.14	libffi . . . . .	42
C.3.15	zlib . . . . .	42
C.3.16	cfuhash . . . . .	43
C.3.17	libmpdec . . . . .	43
C.3.18	W3C C14N 测试套件 . . . . .	44
C.3.19	mimalloc . . . . .	44
C.3.20	asyncio . . . . .	45
C.3.21	Global Unbounded Sequences (GUS) . . . . .	45
<b>D</b>	<b>版权所有</b>	<b>47</b>
	<b>索引</b>	<b>49</b>

**电子邮箱**[distutils-sig@python.org](mailto: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。

本指南涵盖了安装部分的流程。有关创建和共享自己的 Python 项目的指导，请参阅 [Python packaging user guide](#)。

**i 备注**

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

- `pip` 是首选的安装程序。从 Python 3.4 开始，它默认包含在 Python 二进制安装程序中。
- *virtual environment* 是一种半隔离的 Python 环境，允许为特定的应用安装各自的包，而不是安装到整个系统。
- `venv` 是创建虚拟环境的标准工具，从 Python 3.3 开始成为 Python 的组成部分。从 Python 3.4 开始，它会默认安装 `pip` 到所创建的全部虚拟环境。
- `virtualenv` 是 `venv` 的第三方替代（及其前身）。它允许在 Python 3.4 之前的版本中使用虚拟环境，那些版本或是完全不提供 `venv`，或是不会自动安装 `pip` 到所创建的虚拟环境。
- [Python Package 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。
- [Python Packaging Authority](#) 是负责标准打包工具以及相关元数据和文件格式标准维护与改进的开发人员和文档作者团队。他们基于 [GitHub](#) 维护着各种工具、文档和问题追踪系统。
- `distutils` 是最初的构建和分发系统，于 1998 年首次加入 Python 标准库。虽然直接使用 `distutils` 的方式已被淘汰，它仍然是当前打包和分发架构的基础，而且它不仅仍然是标准库的一部分，这个名称还以其他方式存在（例如用于协调 Python 打包标准开发流程的邮件列表就以此命名）。

在 3.5 版本发生变更：现在推荐使用 `venv` 来创建虚拟环境。

### 参见

[Python 软件包用户指南：创建和使用虚拟环境](#)





---

## 基本使用

---

标准打包工具完全是针对命令行使用方式来设计的。

以下命令将从 Python Package Index 安装一个模块的最新版本及其依赖项：

```
python -m pip install SomePackage
```

### 备注

对于 POSIX 用户（包括 macOS 和 Linux 用户）本指南中的示例假定使用了 *virtual environment*。

对于 Windows 用户，本指南中的示例假定在安装 Python 时选择了修改系统 PATH 环境变量。

在命令行中指定一个准确或最小版本也是可以的。当使用比较运算符例如 `>`, `<` 或其他某些可以被终端所解析的特殊字符时，包名称与版本号应当用双引号括起来：

```
python -m pip install SomePackage==1.0.4    # 特定版本
python -m pip install "SomePackage>=1.0.4"  # 最小版本
```

通常，如果一个匹配的模块已安装，尝试再次安装将不会有任何效果。要升级现有模块必须显式地发出请求：

```
python -m pip install --upgrade SomePackage
```

更多有关 pip 及其功能的信息和资源可以在 [Python 软件包用户指南](#) 中找到。

虚拟环境的创建可使用 `venv` 模块来完成。向已激活虚拟环境安装软件包可使用上文所介绍的命令。

### 参见

[Python 软件包用户指南：安装 Python 分发包](#)



---

## 我应如何... ?

---

这是一些常见任务的快速解答或相关链接。

### 3.1 ... 在 Python 3.4 之前的 Python 版本中安装 pip ?

Python 捆绑 pip 是从 Python 3.4 才开始的。对于更早的版本, pip 需要“引导安装”, 具体说明参见 Python 软件包用户指南。

 参见

Python 软件包用户指南: 安装软件包的前提要求

### 3.2 ... 只为当前用户安装软件包 ?

将 `--user` 选项传入 `python -m pip install` 将只为当前用户而非为系统中的所有用户安装软件包。

### 3.3 ... 安装科学计算类 Python 软件包 ?

许多科学计算类 Python 软件包都有复杂的二进制编译文件依赖, 直接使用 pip 安装目前并不太容易。在当前情况下, 通过 [其他方式](#) 而非尝试用 pip 安装这些软件包对用户来说通常会更容易。

 参见

Python 软件包用户指南: 安装科学计算类软件包

### 3.4 ... 使用并行安装的多个 Python 版本 ?

在 Linux, macOS 以及其他 POSIX 系统中, 使用带版本号的 Python 命令配合 `-m` 开关选项来运行特定版本的 pip:

```
python2 -m pip install SomePackage # 默认 Python 2
python2.7 -m pip install SomePackage # 指明 Python 2.7
python3 -m pip install SomePackage # 默认 Python 3
python3.4 -m pip install SomePackage # 指明 Python 3.4
```

也可以使用带特定版本号的 `pip` 命令。

在 Windows 中，使用 `py Python` 启动器命令配合 `-m` 开关选项：

```
py -2 -m pip install SomePackage # 默认 Python 2
py -2.7 -m pip install SomePackage # 指明 Python 2.7
py -3 -m pip install SomePackage # 默认 Python 3
py -3.4 -m pip install SomePackage # 指明 Python 3.4
```

---

## 常见的安装问题

---

### 4.1 在 Linux 的系统 Python 版本上安装

Linux 系统通常会将某个 Python 版本作为发行版的一部分包含在内。将软件包安装到这个 Python 版本上需要系统 root 权限，并可能会干扰到系统包管理器和其他系统组件的运作，如果这些组件在使用 pip 时被意外升级的话。

在这样的系统上，通过 pip 安装软件包通常最好是使用虚拟环境或分用户安装。

### 4.2 未安装 pip

默认情况下可能未安装 pip，一种可选解决方案是：

```
python -m ensurepip --default-pip
```

还有其他附加资源可用来 [安装 pip](#)。

### 4.3 安装二进制编译扩展

Python 通常非常依赖基于源代码的发布方式，也就是期望最终用户在安装过程中使用源码来编译生成扩展模块。

随着对二进制码 wheel 格式支持的引入，以及通过 Python Package Index 至少发布 Windows 和 macOS 版的 wheel 文件，预计此问题将逐步得到解决，因为用户将能够更频繁地安装预编译扩展，而不再需要自己编译它们。

某些用来安装 [科学计算类软件包](#) 的解决方案对于尚未提供预编译 wheel 文件的那些扩展模块来说，也有助于用户在无需进行本机编译的情况下获取二进制码扩展模块。

#### 参见

[Python 软件包用户指南：二进制码扩展](#)



## 术语对照表

&gt;&gt;&gt;

*interactive* shell 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。

...

具有以下含义：

- *interactive* shell 中输入特殊代码时默认的 Python 提示符，特殊代码包括缩进的代码块，左右成对分隔符（圆括号、方括号、花括号或三重引号等）之内，或是在指定一个装饰器之后。
- Ellipsis 内置常量。

**abstract base class -- 抽象基类**

抽象基类简称 ABC，是对 *duck-typing* 的补充，它提供了一种定义接口的新方式，相比之下其他技巧例如 `hasattr()` 显得过于笨拙或有微妙错误（例如使用 魔术方法）。ABC 引入了虚拟子类，这种类并非继承自其他类，但却仍能被 `isinstance()` 和 `issubclass()` 所认可；详见 `abc` 模块文档。Python 自带许多内置的 ABC 用于实现数据结构（在 `collections.abc` 模块中）、数字（在 `numbers` 模块中）、流（在 `io` 模块中）、导入查找器和加载器（在 `importlib.abc` 模块中）。你可以使用 `abc` 模块来创建自己的 ABC。

**annotation -- 标注**

关联到某个变量、类属性、函数形参或返回值的标签，被约定作为 *类型注解* 来使用。

局部变量的标注在运行时不可访问，但全局变量、类属性和函数的标注会分别存放模块、类和函数的 `__annotations__` 特殊属性中。

参见 *variable annotation*、*function annotation*、**PEP 484** 和 **PEP 526**，对此功能均有介绍。另请参见 `annotations-howto` 了解使用标注的最佳实践。

**argument -- 参数**

在调用函数时传给 *function*（或 *method*）的值。参数分为两种：

- 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。

另参见 [parameter](#) 术语表条目，常见问题中 参数与形参的区别以及 [PEP 362](#)。

### asynchronous context manager -- 异步上下文管理器

此种对象通过定义 `__aenter__()` 和 `__aexit__()` 方法来对 `async with` 语句中的环境进行控制。由 [PEP 492](#) 引入。

### asynchronous generator -- 异步生成器

返回值为 [asynchronous generator iterator](#) 的函数。它与使用 `async def` 定义的协程函数很相似，不同之处在于它包含 `yield` 表达式以产生一系列可在 `async for` 循环中使用的值。

此术语通常是指异步生成器函数，但在某些情况下则可能是指 异步生成器迭代器。如果需要清楚表达具体含义，请使用全称以避免歧义。

一个异步生成器函数可能包含 `await` 表达式或者 `async for` 以及 `async with` 语句。

### asynchronous generator iterator -- 异步生成器迭代器

[asynchronous generator](#) 函数所创建的对象。

此对象属于 [asynchronous iterator](#)，当使用 `__anext__()` 方法调用时会返回一个可等待对象来执行异步生成器函数的函数体直到下一个 `yield` 表达式。

每个 `yield` 会临时暂停处理过程，记住执行状态（包括局部变量和挂起的 `try` 语句）。当 异步生成器迭代器通过 `__anext__()` 所返回的另一个可等待对象有效地恢复时，它会从离开位置恢复处理过程。参见 [PEP 492](#) 和 [PEP 525](#)。

### asynchronous iterable -- 异步可迭代对象

一个可以在 `async for` 语句中使用的对象。必须通过它的 `__aiter__()` 方法返回一个 [asynchronous iterator](#)。由 [PEP 492](#) 引入。

### asynchronous iterator -- 异步迭代器

一个实现了 `__aiter__()` 和 `__anext__()` 方法的对象。`__anext__()` 必须返回一个 [awaitable](#) 对象。`async for` 会处理异步迭代器的 `__anext__()` 方法所返回的可等待对象直到其引发一个 `StopAsyncIteration` 异常。由 [PEP 492](#) 引入。

### attribute -- 属性

关联到一个对象的值，通常使用点号表达式按名称来引用。举例来说，如果对象 `o` 具有属性 `a` 则可以用 `o.a` 来引用它。

如果对象允许，将未被定义为 `identifiers` 的非标识名称用作一个对象的属性也是可以的，例如使用 `setattr()`。这样的属性将无法使用点号表达式来访问，而是必须通过 `getattr()` 来获取。

### awaitable -- 可等待对象

一个可在 `await` 表达式中使用的对象。可以是 [coroutine](#) 或是具有 `__await__()` 方法的对象。参见 [PEP 492](#)。

### BDFL

“终身仁慈独裁者”的英文缩写，即 [Guido van Rossum](#)，Python 的创造者。

### binary file -- 二进制文件

[file object](#) 能够读写字节型对象。二进制文件的例子包括以二进制模式 (`'rb'`, `'wb'` 或 `'rb+'`) 打开的文件、`sys.stdin.buffer`、`sys.stdout.buffer` 以及 `io.BytesIO` 和 `gzip.GzipFile` 的实例。

另请参见 [text file](#) 了解能够读写 `str` 对象的文件对象。

### borrowed reference -- 借入引用

在 Python 的 C API 中，借用引用是指一种对象引用，使用该对象的代码并不持有该引用。如果对象被销毁则它就会变成一个悬空指针。例如，垃圾回收器可以移除对象的最后一个 [strong reference](#) 来销毁它。



推荐在 *borrowed reference* 上调用 `Py_INCREF()` 以将其原地转换为 *strong reference*，除非是当该对象无法在借入引用的最后一次使用之前被销毁。`Py_NewRef()` 函数可以被用来创建一个新的 *strong reference*。

### bytes-like object -- 字节型对象

支持 `bufferobjects` 并且能导出 *C-contiguous* 缓冲的对象。这包括所有 `bytes`、`bytearray` 和 `array.array` 对象，以及许多普通 `memoryview` 对象。字节型对象可在多种二进制数据操作中使用；这些操作包括压缩、保存为二进制文件以及通过套接字发送等。

某些操作需要可变的二进制数据。这种对象在文档中常被称为“可读写字节类对象”。可变缓冲对象的例子包括 `bytearray` 以及 `bytearray` 的 `memoryview`。其他操作要求二进制数据存放于不可变对象（“只读字节类对象”）；这种对象的例子包括 `bytes` 以及 `bytes` 对象的 `memoryview`。

### bytecode -- 字节码

Python 源代码会被编译为字节码，即 CPython 解释器中表示 Python 程序的内部代码。字节码还会缓存在 `.pyc` 文件中，这样第二次执行同一文件时速度更快（可以免去将源码重新编译为字节码）。这种“中间语言”运行在根据字节码执行相应机器码的 *virtual machine* 之上。请注意不同 Python 虚拟机上的字节码不一定通用，也不一定能在不同 Python 版本上兼容。

字节码指令列表可以在 `dis` 模块的文档中查看。

### callable -- 可调用对象

可调用对象就是可以执行调用运算的对象，并可能附带一组参数（参见 *argument*），使用以下语法：

```
callable(argument1, argument2, argumentN)
```

*function*，还可扩展到 *method* 等，就属于可调用对象。实现了 `__call__()` 方法的类的实例也属于可调用对象。

### callback -- 回调

一个作为参数被传入以用在未来的某个时刻被调用的子例程函数。

### 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。

### class variable -- 类变量

在类中定义的变量，并且仅限在类的层级上修改（而不是在类的实例中修改）。

### closure variable -- 闭包变量

引用自 *nested scope* 的 *free variable*，它是在外层作用域中定义而不是在运行时自全局或内置命名空间中取得。可能使用 `nonlocal` 关键字显式地定义以允许写入访问，或者如果变量仅供读取则只需隐式地定义。

例如，在以下代码的 `inner` 函数中，`x` 和 `print` 均为自由变量，但只有 `x` 属于闭包变量：

```
def outer():
    x = 0
    def inner():
        nonlocal x
        x += 1
        print(x)
    return inner
```

由于 `codeobject.co_freevars` 属性的存在（虽然名称如此，但其仅包括闭包变量名称而非列出所有被引用的自由变量），更一般化的术语 *free variable* 有时在即便本意是专指闭包变量时也会被使用。

### 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 -- 上下文**

此术语根据其所在场合和使用方式的不同而具有不同的含义。一些常见的含义为：

- 通过 `with` 语句由一个 *context manager* 所创建的临时环境或状态。
- 一组关联到特定 `contextvars.Context` 对象并通过 `ContextVar` 对象来访问的键值绑定。另请参见 *context variable*。
- 一个 `contextvars.Context` 对象。另请参见 *current context*。

**上下文管理协议**

`__enter__()` 和 `__exit__()` 方法将由 `with` 语句来调用。参见 [PEP 343](#)。

**context manager -- 上下文管理器**

一个实现了 *context management protocol* 并负责控制某个 `with` 语句内的环境的对象。参见 [PEP 343](#)。

**context variable -- 上下文变量**

一个具体值取决于哪个上下文是 *current context* 的变量。这些值是通过 `contextvars.ContextVar` 对象来访问的。上下文变量主要被用来隔离并发的异步任务之间的状态。

**contiguous -- 连续**

一个缓冲如果是 *C* 连续或 *Fortran* 连续就会被认为是连续的。零维缓冲是 *C* 和 *Fortran* 连续的。在一维数组中，所有条目必须在内存中彼此相邻地排列，采用从零开始的递增索引顺序。在多维 *C*-连续数组中，当按内存地址排列时用最后一个索引访问条目时速度最快。但是在 *Fortran* 连续数组中则是用第一个索引最快。

**coroutine -- 协程**

协程是子例程的更一般形式。子例程可以在某一点进入并在另一点退出。协程则可以在许多不同的点上进入、退出和恢复。它们可通过 `async def` 语句来实现。参见 [PEP 492](#)。

**coroutine function -- 协程函数**

返回一个 *coroutine* 对象的函数。协程函数可通过 `async def` 语句来定义，并可能包含 `await`、`async for` 和 `async with` 关键字。这些特性是由 [PEP 492](#) 引入的。

**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。

**current context -- 当前上下文**

在当前被 `ContextVar` 对象用来访问（获取或设置）*上下文变量* 的值的 *context* (`contextvars.Context` 对象)。每个线程都具有它自己的当前上下文。用于执行异步任务（参见 *asyncio*）的框架会在每个任务开始或恢复执行时将任务关联到一个成为当前上下文的上下文。

**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(arg):
    ...
f = staticmethod(f)

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

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**

任何定义了 `__get__()`、`__set__()` 或 `__delete__()` 方法的对象。当一个类属性为描述器时，它的特殊绑定行为就会在属性查找时被触发。通常情况下，使用 *a.b* 来获取、设置或删除一个属性时会在 *a* 类的字典中查找名称为 *b* 的对象，但如果 *b* 是一个描述器，则会调用对应的描述器方法。理解描述器的概念是更深层次理解 Python 的关键，因为这是许多重要特性的基础，包括函数、方法、特征属性、类方法、静态方法以及对超类的引用等等。

有关描述器的方法的更多信息，请参阅 `descriptors` 或 描述器使用指南。

### dictionary -- 字典

一个关联数组，其中的任意键都映射到相应的值。键可以是任何具有 `__hash__()` 和 `__eq__()` 方法的对象。在 Perl 中称为 `hash`。

### dictionary comprehension -- 字典推导式

处理一个可迭代对象中的所有或部分元素并返回结果字典的一种紧凑写法。`results = {n: n ** 2 for n in range(10)}` 将生成一个由键 `n` 到值 `n ** 2` 的映射构成的字典。参见 `comprehensions`。

### dictionary view -- 字典视图

从 `dict.keys()`、`dict.values()` 和 `dict.items()` 返回的对象被称为字典视图。它们提供了字典条目的一个动态视图，这意味着当字典改变时，视图也会相应改变。要将字典视图强制转换为真正的列表，可使用 `list(dictview)`。参见 `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。它在代码块被执行时将被忽略，但会被编译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，因此是存放对象的文档的规范位置。

### 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

可以求出某个值的语法单元。换句话说，一个表达式就是表达元素例如字面值、名称、属性访问、运算符或函数调用的汇总，它们最终都会返回一个值。与许多其他语言不同，并非所有语言构件都是表达式。还存在不能被用作表达式的 *statement*，例如 `while`。赋值也是属于语句而非表达式。

### 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。

### f-string -- f-字符串

带有 `'f'` 或 `'F'` 前缀的字符串字面值通常被称为“f-字符串”即 格式化字符串字面值的简写。参见 [PEP 498](#)。

### file object -- 文件对象

对外公开面向文件的 API（带有 `read()` 或 `write()` 等方法）以使用下层资源的对象。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、其他类型的存储或通信设备的访问（例如标准输入/输出、内存缓冲区、套接字、管道等）。文件对象也被称为 文件型对象 或 流。

实际上共有三种类别的文件对象：原始二进制文件、缓冲二进制文件 以及 文本文件。它们的接口定义均在 `io` 模块中。创建文件对象的规范方式是使用 `open()` 函数。

### file-like object -- 文件型对象

[file object](#) 的同义词。

### filesystem encoding and error handler -- 文件系统编码格式与错误处理器

Python 用来从操作系统解码字节串和向操作系统编码 Unicode 的编码格式与错误处理器。

文件系统编码格式必须保证能成功解码长度在 128 以下的所有字节串。如果文件系统编码格式无法提供此保证，则 API 函数可能会引发 `UnicodeError`。

`sys.getfilesystemencoding()` 和 `sys.getfilesystemencodeerrors()` 函数可被用来获取文件系统编码格式与错误处理器。

[filesystem encoding and error handler](#) 是在 Python 启动时通过 `PyConfig_Read()` 函数来配置的：请参阅 `PyConfig` 的 `filesystem_encoding` 和 `filesystem_errors` 等成员。

另请参见 *locale encoding*。

### finder -- 查找器

一种会尝试查找被导入模块的 *loader* 的对象。

存在两种类型的查找器: *元路径查找器* 配合 `sys.meta_path` 使用, 以及 *路径条目查找器* 配合 `sys.path_hooks` 使用。

请参阅 `finders-and-loaders` 和 `importlib` 以了解更多细节。

### 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](#)。

### free threading -- 自由线程

一种线程模型, 在同一个解释器内部的多个线程可以同时运行 Python 字节码。与此相对的是 *global interpreter lock*, 在同一时刻只允许一个线程执行 Python 字节码。参见 [PEP 703](#)。

### free variable -- 自由变量

在正式场合下, 如语言执行模型所定义的, 自由变量是指在某个命名空间中被使用的不属于该命名空间中的局部变量的任何变量。参见 *closure variable* 中的样例。在实际应用中, 由于 `codeobject.co_freevars` 属性被如此命名, 该术语有时也被用作 *closure variable* 的同义词。

### 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。

### function annotation -- 函数标注

即针对函数形参或返回值的 *annotation*。

函数标注通常用于 *类型提示*: 例如以下函数预期接受两个 `int` 参数并预期返回一个 `int` 值:

```
def sum_two_numbers(a: int, b: int) -> int:
    return a + b
```

函数标注语法的详解见 *function* 一节。

参见 *variable annotation* 和 [PEP 484](#), 其中描述了此功能。另请参阅 *annotations-howto* 以了解使用标的最佳实践。

### \_\_future\_\_

`future` 语句, `from __future__ import <feature>` 指示编译器使用将在未来的 Python 发布版中成为标准的语法和语义来编译当前模块。`__future__` 模块文档记录了可能的 *feature* 取值。通过导入此模块并对其变量求值, 你可以看到每项新特性在何时被首次加入到该语言中以及它将 (或已) 在何时成为默认:

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

### garbage collection -- 垃圾回收

释放不再被使用的内存空间的过程。Python 是通过引用计数和一个能够检测和打破循环引用的循环垃圾回收器来执行垃圾回收的。可以使用 `gc` 模块来控制垃圾回收器。

### generator -- 生成器

返回一个 *generator iterator* 的函数。它看起来很像普通函数, 不同点在于其包含 `yield` 表达式以便产生一系列值供给 `for`-循环使用或是通过 `next()` 函数逐一获取。

通常是指生成器函数, 但在某些情况下也可能是指 *生成器迭代器*。如果需要清楚表达具体含义, 请使用全称以避免歧义。

### generator iterator -- 生成器迭代器

*generator* 函数所创建的对象。



每个 `yield` 会临时暂停处理过程，记住执行状态（包括局部变量和挂起的 `try` 语句）。当生成器迭代器恢复时，它会从离开位置继续执行（不同于每次被唤起时都从新开始的普通函数）。

### generator expression -- 生成器表达式

返回一个 *iterator* 的 *expression*。它看起来很像普通表达式后带有定义了一个循环变量、范围的 `for` 子句，以及一个可选的 `if` 子句。以下复合表达式会为外层函数生成一系列值：

```
>>> sum(i*i for i in range(10))      # 平方值 0, 1, 4, ... 81 之和
285
```

### generic function -- 泛型函数

为不同的类型实现相同操作的多个函数所组成的函数。在调用时会由调度算法来确定应该使用哪个实现。

另请参见 *single dispatch* 术语表条目、`functools.singledispatch()` 装饰器以及 [PEP 443](#)。

### generic type -- 泛型

可参数化的 *type*；通常为 `list` 或 `dict` 这样的容器类。用于类型提示和注解。

更多细节参见 泛型别名类型、[PEP 483](#)、[PEP 484](#)、[PEP 585](#) 和 `typing` 模块。

## 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。

在 Python 3.13 中，GIL 可以使用 `--disable-gil` 构建配置选项来禁用。在使用此选项构建 Python 之后，代码必须附带 `-X gil=0` 或在设置 `PYTHON_GIL=0` 环境变量后运行。此特性将为多线程应用程序启用性能提升并让高效率地使用多核 CPU 更加容易。要了解详情，请参阅 [PEP 703](#)。

### hash-based pyc -- 基于哈希的 pyc

使用对应源文件的哈希值而非最后修改时间来确定其有效性的字节码缓存文件。参见 `pyc-invalidation`。

### hashable -- 可哈希

一个对象如果具有在其生命周期内绝不改变的哈希值（它需要有 `__hash__()` 方法），并可以同其他对象进行比较（它需要有 `__eq__()` 方法）就被称为可哈希对象。可哈希对象必须具有相同的哈希值比较结果才会相等。

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。

大多数 Python 中的不可变内置对象都是可哈希的；可变容器（例如列表或字典）都不可哈希；不可变容器（例如元组和 `frozenset`）仅当它们的元素均为可哈希时才是可哈希的。用户定义类的实例对象默认是可哈希的。它们在比较时一定不相同（除非是与自己比较），它们的哈希值的生成是基于它们的 `id()`。

## IDLE

Python 的集成开发与学习环境。idle 是 Python 标准发行版附带的基本编辑器和解释器环境。

### immortal -- 永生对象

永生对象是在 [PEP 683](#) 中引入的 *CPython* 实现细节。

如果对象属于永生对象，则它的 *reference count* 永远不会被修改，因而它在解释器运行期间永远不会被取消分配。例如，`True` 和 `None` 在 *CPython* 中都属于永生对象。

### immutable -- 不可变对象

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。

**import path -- 导入路径**

由多个位置（或[路径条目](#)）组成的列表，会被模块的[path based finder](#)用来查找导入目标。在导入时，此位置列表通常来自 `sys.path`，但对次级包来说也可能来自上级包的 `__path__` 属性。

**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，这意味着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主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这会非常方便（记住 `help(x)` 函数）。有关交互模式的详情，参见 [tut-interac](#)。

**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](#)。

**interpreter shutdown -- 解释器关闭**

当被要求关闭时，Python 解释器将进入一个特殊运行阶段并逐步释放所有已分配资源，例如模块和各种关键内部结构等。它还会多次调用[垃圾回收器](#)。这会触发用户定义析构器或弱引用回调中的代码执行。在关闭阶段执行的代码可能会遇到各种异常，因为其所依赖的资源已不再有效（常见的例子有库模块或警告机制等）。

解释器需要关闭的主要原因有 `__main__` 模块或所运行的脚本已完成执行。

**iterable -- 可迭代对象**

一种能够逐个返回其成员项的对象。可迭代对象的例子包括所有序列类型（如 `list`, `str` 和 `tuple` 等）以及某些非序列类型如 `dict`、[文件对象](#) 以及任何定义了 `__iter__()` 方法或实现了[sequence](#) 语义的 `__getitem__()` 方法的自定义类的对象。

可迭代对象可被用于 `for` 循环以及许多其他需要一个序列的地方（`zip()`, `map()`, ...）。当一个可迭代对象作为参数被传给内置函数 `iter()` 时，它会返回该对象的迭代器。这种迭代器适用于对值集合的一次性遍历。在使用可迭代对象时，你通常不需要调用 `iter()` 或者自己处理迭代器对象。`for` 语句会自动为你处理那些操作，创建一个临时的未命名变量用来在循环期间保存迭代器。参见[iterator](#)、[sequence](#) 和[generator](#)。

**iterator -- 迭代器**

用来表示一连串数据流的对象。重复调用迭代器的 `__next__()` 方法（或将其传给内置函数 `next()`）将逐个返回流中的项。当没有数据可用时则将引发 `StopIteration` 异常。到这时迭代器对象中的数据项已耗尽，继续调用其 `__next__()` 方法只会再次引发 `StopIteration`。迭代器必须具有 `__iter__()` 方法用来返回该迭代器对象自身，因此迭代器必定也是可迭代对象，可被用于其他可迭代对象适用的大部分场合。一个显著的例外是那些会多次重复访问迭代项的代码。容器对象（例如 `list`）在你每次将其传入 `iter()` 函数或是在 `for` 循环中使用时都会产生一个新的迭代器。如果在此情况下你尝试用迭代器则会返回在之前迭代过程中被耗尽的同一迭代器对象，使其看起来就像是一个空容器。

更多信息可查看 [typeiter](#)。

**CPython 实现细节：**CPython 没有强制推行迭代器定义 `__iter__()` 的要求。还要注意的是自由线程 CPython 并不保证迭代器操作的线程安全性。

**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。

Python 中有许多工具都允许用键函数来控制元素的排位或分组方式。其中包括 `min()`, `max()`, `sorted()`, `list.sort()`, `heapq.merge()`, `heapq.nsmallest()`, `heapq.nlargest()` 以及 `itertools.groupby()`。

有多种方式可以创建一个键函数。例如，`str.lower()` 方法可以用作忽略大小写排序的键函数。或者，键函数也可通过 `lambda` 表达式来创建例如 `lambda r: (r[0], r[2])`。此外，`operator`。

`attrgetter()`, `operator.itemgetter()` 和 `operator.methodcaller()` 是键函数的三个构造器。请查看 排序指引 来获取创建和使用键函数的示例。

### 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

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

### list comprehension -- 列表推导式

处理一个序列中的所有或部分元素并返回结果列表的一种紧凑写法。`result = ['{:04x}'.format(x) for x in range(256) if x % 2 == 0]` 将生成一个 0 到 255 范围内的十六进制偶数对应字符串 (0x..) 的列表。其中 `if` 子句是可选的，如果省略则 `range(256)` 中的所有元素都会被处理。

### loader -- 加载器

负责加载模块的对象。它必须定义 `exec_module()` 和 `create_module()` 方法以实现 Loader 接口。加载器通常由一个 *finder* 返回。另请参阅：

- `finders-and-loaders`
- `importlib.abc.Loader`
- **PEP 302**

### locale encoding -- 语言区域编码格式

在 Unix 上，它是 `LC_CTYPE` 语言区域的编码格式。它可以通过 `locale.setlocale(locale.LC_CTYPE, new_locale)` 来设置。

在 Windows 上，它是 ANSI 代码页 (如: "cp1252")。

在 Android 和 VxWorks 上，Python 使用 "utf-8" 作为语言区域编码格式。

`locale.getencoding()` 可被用来获取语言区域编码格式。

另请参阅 *filesystem encoding and error handler*。

### magic method -- 魔术方法

*special method* 的非正式同义词。

### mapping -- 映射

一种支持任意键查找并实现了 `collections.abc.Mapping` 或 `collections.abc.MutableMapping` 抽象基类所规定方法的容器对象。此类对象的例子包括 `dict`, `collections.defaultdict`, `collections.OrderedDict` 以及 `collections.Counter`。

### meta path finder -- 元路径查找器

`sys.meta_path` 的搜索所返回的 *finder*。元路径查找器与 *path entry finders* 存在关联但并不相同。

请查看 `importlib.abc.MetaPathFinder` 了解元路径查找器所实现的方法。

### 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基类的顺序。请参阅 `python_2.3_mro` 了解自 2.3 发布版起 Python 解释器所使用算法的详情。

### 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](#)。

### module spec -- 模块规格

一个命名空间，其中包含用于加载模块的相关导入信息。是 `importlib.machinery.ModuleSpec` 的实例。

另请参阅 `module-specs`。

### MRO

参见 [method resolution order](#)。

### mutable -- 可变对象

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

术语“具名元组”可用于任何继承自元组，并且其中的可索引元素还能使用名称属性来访问的类型或类。这样的类型或类还可能拥有其他特性。

有些内置类型属于具名元组，包括 `time.localtime()` 和 `os.stat()` 的返回值。另一个例子是 `sys.float_info`:

```
>>> sys.float_info[1]           # 索引访问
1024
>>> sys.float_info.max_exp      # 命名字段访问
1024
>>> isinstance(sys.float_info, tuple) # 属于元组
True
```

有些具名元组是内置类型（比如上面的例子）。此外，具名元组还可通过常规类定义从 `tuple` 继承并定义指定名称的字段的方式来创建。这样的类可以手工编号，或者也可以通过继承 `typing.NamedTuple`，或者使用工厂函数 `collections.namedtuple()` 来创建。后一种方式还会添加一些手工编写或内置的具名元组所没有的额外方法。

### namespace -- 命名空间

命名空间是存放变量的场所。命名空间有局部、全局和内置的，还有对象中的嵌套命名空间（在方法之内）。命名空间通过防止命名冲突来支持模块化。例如，函数 `builtins.open` 与 `os.open()` 可通过各自的命名空间来区分。命名空间还通过明确哪个模块实现那个函数来帮助提高可读性和可维护性。例如，`random.seed()` 或 `itertools.islice()` 这种写法明确了这些函数是由 `random` 与 `itertools` 模块分别实现的。

### namespace package -- 命名空间包

[PEP 420](#) 所引入的一种仅被用作子包的容器的 *package*，命名空间包可以没有实体表示物，其描述方式与 *regular package* 不同，因为它们没有 `__init__.py` 文件。

另可参见 [module](#)。

### nested scope -- 嵌套作用域

在一个定义范围内引用变量的能力。例如，在另一函数之内定义的函数可以引用前者的变量。请注意嵌套作用域默认只对引用有效而对赋值无效。局部变量的读写都受限于最内层作用域。类似的，全局变量的读写则作用于全局命名空间。通过 `nonlocal` 关键字可允许写入外层作用域。



**new-style class -- 新式类**

对目前已被应用于所有类对象的类形式的旧称谓。在较早的 Python 版本中，只有新式类能够使用 Python 新增的更灵活我，如 `__slots__`、描述器、特征属性、`__getattr__()`、类方法和静态方法等。

**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。

**optimized scope -- 已优化的作用域**

当代码被编译时编译器已充分知晓目标局部变量名称的作用域，这允许对这些名称的读写进行优化。针对函数、生成器、协程、推导式和生成器表达式的局部命名空间都是这样的已优化作用域。注意：大部分解释器优化将应用于所有作用域，只有那些依赖于已知的局部和非局部变量名称集合的优化会仅限于已优化的作用域。

**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，包是具有 `__path__` 属性的 Python 模块。

另参见 *regular package* 和 *namespace package*。

**parameter -- 形参**

*function*（或方法）定义中的命名实体，它指定函数可以接受的一个 *argument*（或在某些情况下，多个实参）。有五种形参：

- *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，指定一个只能通过位置传入的参数。仅限位置形参可通过在函数定义的形参列表中它们之后包含一个 `/` 字符来定义，例如下面的 *posonly1* 和 *posonly2*：

```
def func(posonly1, posonly2, /, positional_or_keyword): ...
```

- *keyword-only*：仅限关键字，指定一个只能通过关键字传入的参数。仅限关键字形参可通过在函数定义的形参列表中包含单个可变位置形参或者在多个可变位置形参之前放一个 `*` 来定义，例如下面的 *kw\_only1* 和 *kw\_only2*：

```
def func(arg, *, kw_only1, kw_only2): ...
```

- *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。

另参见 *argument* 术语表条目、参数与形参的区别中的常见问题、`inspect.Parameter` 类、*function* 一节以及 **PEP 362**。

**path entry -- 路径入口**

*import path* 中的一个单独位置，会被 *path based finder* 用来查找要导入的模块。

**path entry finder -- 路径入口查找器**

任一可调用对象使用 `sys.path_hooks`（即 *path entry hook*）返回的 *finder*，此种对象能通过 *path entry* 来定位模块。

请参看 `importlib.abc.PathEntryFinder` 以了解路径入口查找器所实现的各个方法。

**path entry hook -- 路径入口钩子**

一种可调用对象，它在知道如何查找特定 *path entry* 中的模块的情况下能够使用 `sys.path_hooks` 列表返回一个 *path entry finder*。

**path based finder -- 基于路径的查找器**

默认的一种元路径查找器，可在一个 *import path* 中查找模块。

**path-like object -- 路径类对象**

代表一个文件系统路径的对象。路径类对象可以是一个表示路径的 `str` 或者 `bytes` 对象，还可以是一个实现了 `os.PathLike` 协议的对象。一个支持 `os.PathLike` 协议的对象可通过调用 `os.fspath()` 函数转换为 `str` 或者 `bytes` 类型的文件系统路径；`os.fsdecode()` 和 `os.fsencode()` 可被分别用来确保获得 `str` 或 `bytes` 类型的结果。此对象是由 **PEP 519** 引入的。

**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**。

**portion -- 部分**

构成一个命名空间包的单个目录内文件集合（也可能存放于一个 `zip` 文件内），具体定义见 **PEP 420**。

**positional argument -- 位置参数**

参见 *argument*。

**provisional API -- 暂定 API**

暂定 API 是指被有意排除在标准库的向后兼容性保证之外的应用编程接口。虽然此类接口通常不会再有重大改变，但只要其被标记为暂定，就可能在核心开发者确定有必要的情况下进行向后不兼容的更改（甚至包括移除该接口）。此种更改并不会随意进行 -- 仅在 API 被加入之前未考虑到的严重基础性缺陷被发现时才可能会这样做。

即便是对暂定 API 来说，向后不兼容的更改也会被视为“最后的解决方案”——任何问题被确认时都会尽可能先尝试找到一种向后兼容的解决方案。

这种处理过程允许标准库持续不断地演进，不至于被有问题的长期性设计缺陷所困。详情见 **PEP 411**。

**provisional package -- 暂定包**

参见 *provisional API*。

**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)
```

**qualified name -- 限定名称**

一个以点号分隔的名称，显示从模块的全局作用域到该模块中定义的某个类、函数或方法的“路径”，相关定义见 **PEP 3155**。对于最高层级的函数和类，限定名称与对象名称一致：

```
>>> class C:
...     class D:
...         def meth(self):
...             pass
...
>>> C.__qualname__
'C'
>>> C.D.__qualname__
'C.D'
>>> C.D.meth.__qualname__
'C.D.meth'
```

当被用于引用模块时，完整限定名称意为标示该模块的以点号分隔的整个路径，其中包含其所有的父包，例如 `email.mime.text`：

```
>>> import email.mime.text
>>> email.mime.text.__name__
'email.mime.text'
```

### reference count -- 引用计数

指向某个对象的引用的数量。当一个对象的引用计数降为零时，它就会被释放。特殊的 *immortal* 对象具有永远不会被修改的引用计数，因此这种对象永远不会被释放。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。程序员可以调用 `sys.getrefcount()` 函数来返回特定对象的引用计数。

### regular package -- 常规包

传统型的 *package*，例如包含有一个 `__init__.py` 文件的目录。

另参见 *namespace package*。

### REPL

“读取-求值-打印循环” read-eval-print loop 的缩写，*interactive* 解释器 shell 的另一个名字。

### \_\_slots\_\_

一种写在类内部的声明，通过预先声明实例属性等对象并移除实例字典来节省内存。虽然这种技巧很流行，但想要用好却并不容易，最好是只保留在少数情况下采用，例如极耗内存的应用程序，并且其中包含大量实例。

### sequence -- 序列

一种 *iterable*，它支持通过 `__getitem__()` 特殊方法来使用整数索引进行高效的元素访问，并定义了一个返回序列长度的 `__len__()` 方法。内置序列类型有 `list`, `str`, `tuple` 和 `bytes` 等。请注意虽然 `dict` 也支持 `__getitem__()` 和 `__len__()`，但它被归类为映射而非序列，因为它使用任意的 *hashable* 键而不是整数来查找元素。

`collections.abc.Sequence` 抽象基类定义了一个更丰富的接口，它在 `__getitem__()` 和 `__len__()` 之外，还添加了 `count()`, `index()`, `__contains__()` 和 `__reversed__()`。实现此扩展接口的类型可以使用 `register()` 来显式地注册。要获取有关通用序列方法的更多文档，请参阅 通用序列操作。

### set comprehension -- 集合推导式

处理一个可迭代对象中的所有或部分元素并返回结果集合的一种紧凑写法。`results = {c for c in 'abracadabra' if c not in 'abc'}` 将生成字符串集合 `{'r', 'd'}`。参见 *comprehensions*。

### single dispatch -- 单分派

一种 *generic function* 分派形式，其实现是基于单个参数的类型来选择的。

### slice -- 切片

通常只包含了特定 *sequence* 的一部分的对象。切片是通过使用下标标记来创建的，在 `[]` 中给出几个以冒号分隔的数字，例如 `variable_name[1:3:5]`。方括号（下标）标记在内部使用 *slice* 对象。

### soft deprecated -- 软弃用

被软弃用的 API 不应在新编写的代码中使用，但在已有代码中使用仍是安全的。这样的 API 将保留在文档中并被测试，但不会再获得进一步的功能增强。

与普通的弃用不同，软弃用没有 API 移除计划也不会发出弃用警告。

参见 [PEP 387: Soft Deprecation](#)。

### 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。这种方法的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`。

### static type checker -- 静态类型检查器

读取 Python 代码并进行分析，以查找问题例如拼写错误的外部工具。另请参阅 [类型提示](#) 以及 `typing` 模块。

### strong reference -- 强引用

在 Python 的 C API 中，强引用是指为持有引用的代码所拥有的对象的引用。在创建引用时可通过调用 `Py_INCREF()` 来获取强引用而在删除引用时可通过 `Py_DECREF()` 来释放它。

`Py_NewRef()` 函数可被用于创建一个对象的强引用。通常，必须在退出某个强引用的作用域时在该强引用上调用 `Py_DECREF()` 函数，以避免引用的泄漏。

另请参阅 [borrowed reference](#)。

### text encoding -- 文本编码格式

在 Python 中，一个字符串是一串 Unicode 代码点（范围为 U+0000--U+10FFFF）。为了存储或传输一个字符串，它需要被序列化为一串字节。

将一个字符串序列化为一个字节序列被称为“编码”，而从字节序列中重新创建字符串被称为“解码”。

有各种不同的文本序列化 编码器，它们被统称为“文本编码格式”。

### text file -- 文本文件

一种能够读写 `str` 对象的 [file object](#)。通常一个文本文件实际是访问一个面向字节的数据流并自动处理 [text encoding](#)。文本文件的例子包括以文本模式（`'r'` 或 `'w'`）打开的文件、`sys.stdin`、`sys.stdout` 以及 `io.StringIO` 的实例。

另请参看 [binary file](#) 了解能够读写 [字节型对象](#) 的文件对象。

### 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

Python 对象的类型决定它属于什么种类；每个对象都具有特定的类型。对象的类型可通过其 `__class__` 属性来访问或是用 `type(obj)` 来获取。

### type alias -- 类型别名

一个类型的同义词，创建方式是把类型赋值给特定的标识符。

类型别名的作用是简化 [类型注解](#)。例如：

```
def remove_gray_shades(
    colors: list[tuple[int, int, int]]) -> list[tuple[int, int, int]]:
    pass
```

可以这样提高可读性：

```
Color = tuple[int, int, int]

def remove_gray_shades(colors: list[Color]) -> list[Color]:
    pass
```

参见 `typing` 和 [PEP 484](#)，其中有对此功能的详细描述。

**type hint -- 类型注解**

*annotation* 为变量、类属性、函数的形参或返回值指定预期的类型。

类型提示是可选的而不是 Python 的强制要求，但它们对静态类型检查器很有用处。它们还能协助 IDE 实现代码补全与重构。

全局变量、类属性和函数的类型注解可以使用 `typing.get_type_hints()` 来访问，但局部变量则不可以。

参见 `typing` 和 [PEP 484](#)，其中有对此功能的详细描述。

**universal newlines -- 通用换行**

一种解读文本流的方式，将以下所有符号都识别为行结束标志：Unix 的行结束约定 `'\n'`、Windows 的约定 `'\r\n'` 以及旧版 Macintosh 的约定 `'\r'`。参见 [PEP 278](#) 和 [PEP 3116](#) 和 `bytes.splitlines()` 了解更多用法说明。

**variable annotation -- 变量标注**

对变量或类属性的 *annotation*。

在标注变量或类属性时，还可选择为其赋值：

```
class C:
    field: 'annotation'
```

变量标注通常被用作类型提示：例如以下变量预期接受 `int` 类型的值：

```
count: int = 0
```

变量标注语法的详细解释见 `annassign` 一节。

参见 *function annotation*、[PEP 484](#) 和 [PEP 526](#)，其中描述了此功能。另请参阅 `annotations-howto` 以了解使用标注的最佳实践。

**virtual environment -- 虚拟环境**

一种采用协作式隔离的运行时环境，允许 Python 用户和应用程序在安装和升级 Python 分发包时不会干扰到同一系统上运行的其他 Python 应用程序的行为。

另参见 `venv`。

**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"`。



---

## 关于本文档

---

Python 的文档是用 [Sphinx](#) 从 [reStructuredText](#) 源生成的，*Sphinx* 是一个专为处理 Python 文档而编写的文档生成器。

本文档及其工具链之开发，皆在于志愿者之努力，亦恰如 Python 本身。如果您想为此作出贡献，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.，原始 Python 文档工具集之创造者，众多文档之作者；
- 用于创建 [reStructuredText](#) 和 [Docutils](#) 套件的 [Docutils](#) 项目；
- Fredrik Lundh 的 [Alternative Python Reference](#) 项目，为 *Sphinx* 提供许多好的点子。

## 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 源代码分发的 [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 was created in the early 1990s by Guido van Rossum at Stichting Mathematisch Centrum (CWI, see <https://www.cwi.nl>) in the Netherlands as a successor of a language called ABC. Guido remains Python's principal author, although it includes many contributions from others.

In 1995, Guido continued his work on Python at the Corporation for National Research Initiatives (CNRI, see <https://www.cnri.reston.va.us>) in Reston, Virginia where he released several versions of the software.

In May 2000, Guido and the Python core development team moved to BeOpen.com to form the BeOpen PythonLabs team. In October of the same year, the PythonLabs team moved to Digital Creations, which became Zope Corporation. In 2001, the Python Software Foundation (PSF, see <https://www.python.org/psf/>) was formed, a non-profit organization created specifically to own Python-related Intellectual Property. Zope Corporation was a sponsoring member of the PSF.

All Python releases are Open Source (see <https://opensource.org> for the Open Source Definition). Historically, most, but not all, Python releases have also been GPL-compatible; the table below summarizes the various releases.

发布版本	源自	年份	所有者	GPL-compatible? (1)
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	yes (2)
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

**i 备注**

(1) GPL-compatible doesn't mean that we're distributing Python under the GPL. All Python licenses, unlike the GPL, let you distribute a modified version without making your changes open source. The GPL-

compatible licenses make it possible to combine Python with other software that is released under the GPL; the others don't.

- (2) According to Richard Stallman, 1.6.1 is not GPL-compatible, because its license has a choice of law clause. According to CNRI, however, Stallman's lawyer has told CNRI's lawyer that 1.6.1 is "not incompatible" with the GPL.

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

Python software and documentation are licensed under the Python Software Foundation License Version 2.

Starting with Python 3.8.6, examples, recipes, and other code in the documentation are dual licensed under the PSF License Version 2 and the *Zero-Clause BSD license*.

某些包含在 Python 中的软件基于不同的许可。这些许可会与相应许可之下的代码一同列出。有关这些许可的不完整列表请参阅收录软件的许可与鸣谢。

### C.2.1 PYTHON SOFTWARE FOUNDATION LICENSE VERSION 2

1. This LICENSE AGREEMENT is between the Python Software Foundation ("PSF"), and the Individual or Organization ("Licensee") accessing and otherwise using this software ("Python") 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 alone or in any derivative version, provided, however, that PSF's License Agreement and PSF's notice of copyright, i.e., "Copyright © 2001-2024 Python Software Foundation; All Rights Reserved" are retained in Python 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 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.
4. PSF is making Python 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 WILL NOT INFRINGE ANY THIRD PARTY RIGHTS.
5. PSF SHALL NOT BE LIABLE TO LICENSEE OR ANY OTHER USERS OF PYTHON FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES OR LOSS AS A RESULT OF MODIFYING, DISTRIBUTING, OR OTHERWISE USING PYTHON, 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.

(续下页)

(接上页)

8. By copying, installing or otherwise using Python, 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 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,

(续下页)

(接上页)

- 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 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.

(续下页)

(接上页)

```
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.2.5 ZERO-CLAUSE BSD LICENSE FOR CODE IN THE PYTHON DOCUMENTATION

Permission to use, copy, modify, and/or distribute this software for any purpose with or without fee is hereby granted.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY SPECIAL, DIRECT, 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 收录软件的许可与鸣谢

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

作为 random 模块下层的 \_random C 扩展包括基于从 <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：

```
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.
```

```
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
```

(续下页)

(接上页)

```
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 使用了 `getaddrinfo()` 和 `getnameinfo()` WIDE 项目的不同源文件中: <https://www.wide.ad.jp/>

```
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
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 异步套接字服务

`test.support.asyncchat` 和 `test.support.asyncore` 模块包含以下说明。:

```
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
```

(续下页)

(接上页)

```
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.3.4 Cookie 管理

http.cookies 模块包含以下声明:

```
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.3.5 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.
```

(续下页)

(接上页)

```
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.3.6 UUencode 与 UUdecode 函数

uu 编解码器包含以下声明:

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

### C.3.7 XML 远程过程调用

xmlrpc.client 模块包含以下声明:

```
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-
```

(续下页)



(接上页)

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.3.8 test\_epoll

test.test\_epoll 模块包含以下说明:

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 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.3.9 Select kqueue

select 模块关于 kqueue 的接口包含以下声明:

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.3.10 SipHash24

Python/pyhash.c 文件包含 Marek Majkowski 对 Dan Bernstein 的 SipHash24 算法的实现。它包含以下声明:

```
<MIT License>
Copyright (c) 2013  Marek Majkowski <marek@popcount.org>

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.
</MIT License>

Original location:
    https://github.com/majek/csiphash/

Solution inspired by code from:
    Samuel Neves (supercop/crypto_auth/siphhash24/little)
    djb (supercop/crypto_auth/siphhash24/little2)
    Jean-Philippe Aumasson (https://131002.net/siphhash/siphhash24.c)
```

### C.3.11 strtod 和 dtoa

Python/dtoa.c 文件提供了 C 函数 dtoa 和 strtod, 用于 C 双精度数值和字符串之间的转换, 它派生自由 David M. Gay 编写的同名文件。目前该文件可在 <https://web.archive.org/web/20220517033456/http://www.netlib.org/fp/dtoa.c> 访问。在 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
 * 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.3.12 OpenSSL

如果操作系统有支持则 hashlib, posix 和 ssl 会使用 OpenSSL 库来提升性能。此外, Python 的 Windows 和 macOS 安装程序可能会包括 OpenSSL 库的副本, 所以我们也在其中包括一份 OpenSSL 许可证的副本。对于 OpenSSL 3.0 发布版, 及其后续衍生版本, 均使用 Apache License v2:

```
Apache License
Version 2.0, January 2004
https://www.apache.org/licenses/
```

(续下页)

(接上页)

## TERMS AND CONDITIONS FOR USE, REPRODUCTION, AND DISTRIBUTION

## 1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 9 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication sent to the Licensor or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

(续下页)

(接上页)

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.
3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable (except as stated in this section) patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted. If You institute patent litigation against any entity (including a cross-claim or counterclaim in a lawsuit) alleging that the Work or a Contribution incorporated within the Work constitutes direct or contributory patent infringement, then any patent licenses granted to You under this License for that Work shall terminate as of the date such litigation is filed.
4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:
  - (a) You must give any other recipients of the Work or Derivative Works a copy of this License; and
  - (b) You must cause any modified files to carry prominent notices stating that You changed the files; and
  - (c) You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and
  - (d) If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. The contents of the NOTICE file are for informational purposes only and do not modify the License. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work, provided that such additional attribution notices cannot be construed as modifying the License.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or

(续下页)

(接上页)

for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.
6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.
7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.
8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.
9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License. However, in accepting such obligations, You may act only on Your own behalf and on Your sole responsibility, not on behalf of any other Contributor, and only if You agree to indemnify, defend, and hold each Contributor harmless for any liability incurred by, or claims asserted against, such Contributor by reason of your accepting any such warranty or additional liability.

END OF TERMS AND CONDITIONS

### C.3.13 expat

pyexpat 扩展是使用所包括的 expat 源副本来构建的, 除非配置了 `--with-system-expat`:

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

(续下页)

(接上页)

```
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.3.14 libffi

作为 ctypes 模块下层的 \_ctypes C 扩展是使用包括了 libffi 源的副本构建的, 除非构建时配置了 --with-system-libffi:

```
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.3.15 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) 1995-2011 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
```

(续下页)

(接上页)

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  
jloup@gzip.org

Mark Adler  
madler@alumni.caltech.edu

### C.3.16 cfuhash

tracemalloc 使用的哈希表的实现基于 cfuhash 项目:

Copyright (c) 2005 Don Owens  
All rights reserved.

This code is released under the BSD license:

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

- \* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- \* 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.
- \* Neither the name of the author 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 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.

### C.3.17 libmpdec

作为 decimal 模块下层的 \_decimal C 扩展是使用包括了 libmpdec 库的副本构建的, 除非构建时配置了 --with-system-libmpdec:

Copyright (c) 2008-2020 Stefan Krah. All rights reserved.

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

(续下页)

(接上页)

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.3.18 W3C C14N 测试套件

test 包中的 C14N 2.0 测试集 (Lib/test/xmltestdata/c14n-20/) 提取自 W3C 网站 <https://www.w3.org/TR/xml-c14n2-testcases/> 并根据 3 条款版 BSD 许可证发行:

Copyright (c) 2013 W3C(R) (MIT, ERCIM, Keio, Beihang),  
All Rights Reserved.

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

- \* Redistributions of works must retain the original copyright notice, this list of conditions and the following disclaimer.
- \* Redistributions in binary form must reproduce the original copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- \* Neither the name of the W3C nor the names of its contributors may be used to endorse or promote products derived from this work 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.

### C.3.19 mimalloc

MIT 许可证:

Copyright (c) 2018-2021 Microsoft Corporation, Daan Leijen

Permission is hereby granted, free of charge, to any person obtaining a copy

(续下页)



(接上页)

```
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.3.20 asyncio

asyncio 模块的某些部分来自 `uvloop 0.16`，它是基于 MIT 许可证发行的：

```
Copyright (c) 2015-2021 MagicStack Inc. http://magic.io
```

```
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.3.21 Global Unbounded Sequences (GUS)

文件 `Python/qsbr.c` 改编自 `subr_smr.c` 中 FreeBSD 的“Global Unbounded Sequences”安全内存回收方案。该文件是基于 2 条款 BSD 许可证分发的：

```
Copyright (c) 2019,2020 Jeffrey Roberson <jeff@FreeBSD.org>
```

```
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 unmodified, 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 ``AS IS'' AND ANY EXPRESS OR
```

(续下页)

(接上页)

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 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.

---

### 版权所有

---

Python 与这份文档：

版权所有 © 2001-2024 Python 软件基金会。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，请参见[历史](#)和[许可证](#)。



## 非字母

..., [11](#)

>>>, [11](#)

\_\_future\_\_, [16](#)

\_\_slots\_\_, [23](#)

上下文管理协议, [14](#)

特殊

method -- 方法, [24](#)

环境变量

PYTHON\_GIL, [17](#)

魔术

method -- 方法, [19](#)

## A

abstract base class -- 抽象基类, [11](#)

annotation -- 标注, [11](#)

argument -- 参数, [11](#)

asynchronous context manager -- 异步上下文管理器, [12](#)

asynchronous generator -- 异步生成器, [12](#)

asynchronous generator iterator -- 异步生成器迭代器, [12](#)

asynchronous iterable -- 异步可迭代对象, [12](#)

asynchronous iterator -- 异步迭代器, [12](#)

attribute -- 属性, [12](#)

awaitable -- 可等待对象, [12](#)

## B

BDFL, [12](#)

binary file -- 二进制文件, [12](#)

borrowed reference -- 借入引用, [12](#)

bytecode -- 字节码, [13](#)

bytes-like object -- 字节型对象, [13](#)

## C

C 连续, [14](#)

callable -- 可调用对象, [13](#)

callback -- 回调, [13](#)

class -- 类, [13](#)

class variable -- 类变量, [13](#)

closure variable -- 闭包变量, [13](#)

complex number -- 复数, [13](#)

context -- 上下文, [14](#)

context manager -- 上下文管理器, [14](#)

context variable -- 上下文变量, [14](#)

contiguous -- 连续, [14](#)

coroutine -- 协程, [14](#)

coroutine function -- 协程函数, [14](#)

CPython, [14](#)

current context -- 当前上下文, [14](#)

## D

decorator -- 装饰器, [14](#)

descriptor -- 描述器, [14](#)

dictionary -- 字典, [15](#)

dictionary comprehension -- 字典推导式, [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

f-string -- f-字符串, [15](#)

file object -- 文件对象, [15](#)

file-like object -- 文件型对象, [15](#)

filesystem encoding and error handler -- 文件系统编码格式与错误处理器, [15](#)

finder -- 查找器, [16](#)

floor division -- 向下取整除法, [16](#)

Fortran 连续, [14](#)

free threading -- 自由线程, [16](#)

free variable -- 自由变量, [16](#)

function -- 函数, [16](#)

function annotation -- 函数标注, [16](#)

## G

garbage collection -- 垃圾回收, [16](#)

generator -- 生成器, [16](#)

generator -- 生成器, [16](#)

generator expression -- 生成器表达式, [17](#)

generator expression -- 生成器表达式, [17](#)

generator iterator -- 生成器迭代器, [16](#)

generic function -- 泛型函数, [17](#)  
 generic type -- 泛型, [17](#)  
 GIL, [17](#)  
 global interpreter lock -- 全局解释器锁, [17](#)

## H

hash-based pyc -- 基于哈希的 pyc, [17](#)  
 hashable -- 可哈希, [17](#)

## I

IDLE, [17](#)  
 immortal -- 永生对象, [17](#)  
 immutable -- 不可变对象, [17](#)  
 import path -- 导入路径, [18](#)  
 importer -- 导入器, [18](#)  
 importing -- 导入, [18](#)  
 interactive -- 交互, [18](#)  
 interpreted -- 解释型, [18](#)  
 interpreter shutdown -- 解释器关闭, [18](#)  
 iterable -- 可迭代对象, [18](#)  
 iterator -- 迭代器, [18](#)

## K

key function -- 键函数, [18](#)  
 keyword argument -- 关键字参数, [19](#)

## L

lambda, [19](#)  
 LBYL, [19](#)  
 list -- 列表, [19](#)  
 list comprehension -- 列表推导式, [19](#)  
 loader -- 加载器, [19](#)  
 locale encoding -- 语言区域编码格式, [19](#)

## M

magic method -- 魔术方法, [19](#)  
 mapping -- 映射, [19](#)  
 meta path finder -- 元路径查找器, [19](#)  
 metaclass -- 元类, [19](#)  
 method -- 方法  
     特殊, [24](#)  
     魔术, [19](#)  
 method -- 方法, [20](#)  
 method resolution order -- 方法解析顺序, [20](#)  
 module -- 模块, [20](#)  
 module spec -- 模块规格, [20](#)  
 MRO, [20](#)  
 mutable -- 可变对象, [20](#)

## N

named tuple -- 具名元组, [20](#)  
 namespace -- 命名空间, [20](#)  
 namespace package -- 命名空间包, [20](#)  
 nested scope -- 嵌套作用域, [20](#)  
 new-style class -- 新式类, [21](#)

## O

object -- 对象, [21](#)

optimized scope -- 已优化的作用域, [21](#)

## P

package -- 包, [21](#)  
 parameter -- 形参, [21](#)  
 path based finder -- 基于路径的查找器, [22](#)  
 path entry -- 路径入口, [21](#)  
 path entry finder -- 路径入口查找器, [21](#)  
 path entry hook -- 路径入口钩子, [22](#)  
 path-like object -- 路径类对象, [22](#)  
 PEP, [22](#)  
 portion -- 部分, [22](#)  
 positional argument -- 位置参数, [22](#)  
 provisional API -- 暂定 API, [22](#)  
 provisional package -- 暂定包, [22](#)  
 Python 3000, [22](#)  
 Python 增强建议; PEP 1, [22](#)  
 Python 增强建议; PEP 238, [16](#)  
 Python 增强建议; PEP 278, [25](#)  
 Python 增强建议; PEP 302, [19](#)  
 Python 增强建议; PEP 343, [14](#)  
 Python 增强建议; PEP 362, [12, 21](#)  
 Python 增强建议; PEP 411, [22](#)  
 Python 增强建议; PEP 420, [20, 22](#)  
 Python 增强建议; PEP 443, [17](#)  
 Python 增强建议; PEP 483, [17](#)  
 Python 增强建议; PEP 484, [11, 16, 17, 24, 25](#)  
 Python 增强建议; PEP 492, [12, 14](#)  
 Python 增强建议; PEP 498, [15](#)  
 Python 增强建议; PEP 519, [22](#)  
 Python 增强建议; PEP 525, [12](#)  
 Python 增强建议; PEP 526, [11, 25](#)  
 Python 增强建议; PEP 585, [17](#)  
 Python 增强建议; PEP 683, [17](#)  
 Python 增强建议; PEP 703, [16, 17](#)  
 Python 增强建议; PEP 3116, [25](#)  
 Python 增强建议; PEP 3155, [22](#)  
 PYTHON\_GIL, [17](#)  
 Pythonic, [22](#)

## Q

qualified name -- 限定名称, [22](#)

## R

reference count -- 引用计数, [23](#)  
 regular package -- 常规包, [23](#)  
 REPL, [23](#)

## S

sequence -- 序列, [23](#)  
 set comprehension -- 集合推导式, [23](#)  
 single dispatch -- 单分派, [23](#)  
 slice -- 切片, [23](#)  
 soft deprecated -- 软弃用, [23](#)  
 special method -- 特殊方法, [24](#)  
 statement -- 语句, [24](#)  
 static type checker -- 静态类型检查器, [24](#)  
 strong reference -- 强引用, [24](#)

## T

text encoding -- 文本编码格式, [24](#)  
text file -- 文本文件, [24](#)  
triple-quoted string -- 三引号字符串, [24](#)  
type -- 类型, [24](#)  
type alias -- 类型别名, [24](#)  
type hint -- 类型注解, [25](#)

## U

universal newlines -- 通用换行, [25](#)

## V

variable annotation -- 变量标注, [25](#)  
virtual environment -- 虚拟环境, [25](#)  
virtual machine -- 虚拟机, [25](#)

## Z

Zen of Python -- Python 之禅, [25](#)