What's New in Python

发布 *3.7.8*

A. M. Kuchling

六月 29, 2020

Python Software Foundation
Email: docs@python.org

Contents

编者 Elvis Pranskevichus <elvis@magic.io>

本文解释了 Python 3.7 相比 3.6 的新增特性。Python 3.7 于 2018 年 6 月 27 日发布。完整的详情可参阅 更新日志。

1 摘要 - 发布重点

新的语法特性：

- *PEP 563*，类型标注延迟求值。

向后不兼容的语法更改：

- `async` 和 `await` 现在是保留的关键字。

新的库模块：

- `contextvars`: *PEP 567 -- 上下文变量*
- `dataclasses`: *PEP 557 -- 数据类*
- *importlib.resources*

新的内置特性：

- *PEP 553*, 新的 `breakpoint()` 函数。

对 Python 数据模型的改进：

- *PEP 562*, 自定义可访问的模块属性。
- *PEP 560*, typing 模块和泛型类型的核心支持。
- dict 对象会保持插入时的顺序这个特性 正式宣布 成为 Python 语言官方规范的一部分。

标准库中的重大改进：

- `asyncio` 模块添加了新的功能，重大改进请参阅*可用性与性能提升*。
- `time` 模块现在提供*纳秒级精度函数* 的支持。

CPython 实现的改进：

- 避免使用 ASCII 作为默认的文本编码：
  - *PEP 538*，传统 C 区域强制转换
  - *PEP 540*，强制 UTF-8 运行时模式
- *PEP 552*，确定性的.pyc 文件
- *新的开发运行时模式*
- *PEP 565*，改进的 `DeprecationWarning` 处理

C API 的改进：

- *PEP 539*，用于线程本地存储的新 C API

文档的改进：

- *PEP 545*, Python 文档翻译
- 新的文档翻译：Japanese，French 和 Korean。

此版本在诸多方面有显著的性能改进。*性能优化* 章节详细列出了它们。

和之前的 Python 版本存在兼容性的更改列表，请参阅*移植到 Python 3.7* 章节。

2 新的特性

2.1 PEP 563：延迟的标注求值

随着 **PEP 3107** 加入标注功能并在 **PEP 526** 进一步细化，Python 中类型提示的出现揭示了两个明显的可用性问题：

- 标注只能使用在当前作用域中已经存在的名称，也就是说，它们不支持任何形式的前向引用；而且——
- 标注源码对 Python 程序的启动时间有不利的影响。

这两个问题都可以通过延迟标注求值来解决。在定义标注的时候，编译器并不会编译执行相应表达式的代码，而是保存与相应表达式的 AST 等价的字符串形式。如果有需要，标注可以在运行时使用 `typing.get_type_hints()` 进行解析。在不需要这种解析的通常情况下，标注的存储成本更低（因为解析器只需处理较短的字符串）且启动时间更短。

在可用性方面，标注现在支持向前引用，以使以下句法有效：

```
class C:
    @classmethod
    def from_string(cls, source: str) -> C:
        ...

    def validate_b(self, obj: B) -> bool:
        ...

class B:
    ...
```

由于此修改会破坏兼容性，在 Python 3.7 中此种新的行为需要在每个模块层级上使用 `__future__` 导入来启用：

```
from __future__ import annotations
```

它将在 Python 4.0 中成为默认行为。

参见：

PEP 563 -- 延迟的标注求值 PEP 由 Łukasz Langa 撰写并实现。

2.2 PEP 538: 传统 C 区域强制转换

Python 3 系列有一个持续的挑战就是为处理 7 比特位 ASCII 文本的假定编码确定合理的默认策略，目前的设定是在非 Windows 平台上使用默认的 C 或 POSIX 区域设置。

PEP 538 更新了默认的解释器命令行接口，以自动将上述区域强制转换为可用的基于 UTF-8 的区域，具体描述可参见有关新增环境变量 `PYTHONCOERCECLOCALE` 的文档。以这种方式自动设置 `LC_CTYPE` 意味着核心解释器和能感知区域的 C 扩展 (例如 `readline`) 都将会假定 UTF-8 已被用作默认的文本编码，而不再是 ASCII。

PEP 11 中的平台支持定义也已被更新以限制完整文本处理支持适当配置的基于非 ASCII 的语言区域。

作为此更改的一部分，当使用任何已定义的强制转换目标区域时 (目前为 `C.UTF-8`, `C.utf8` 和 `UTF-8`) `stdin` 和 `stdout` 默认的处理器现在将为 `surrogateescape` (而不是 `strict`)。而无论是什么区域，`stderr` 默认的处理器仍为 `backslashreplace`。

默认情况下区域强制转换会静默进行，但为了辅助调试潜在的区域相关集成问题，可以通过设置 `PYTHONCOERCECLOCALE=warn` 来请求显式地启用警告信息（直接在 `stderr` 上发出）。此设置还会使得 Python 运行时在核心解释器初始化时如果传统 C 区域仍然处于激活状态时发出警告。

虽然 **PEP 538** 的区域强制转换的好处在于它还会同时影响扩展模块 (例如 GNU `readline`) 以及子进程 (包括运行非 Python 应用和旧版本 Python 的子进程)，但它也存在需要所运行系统必须存在适合的目标区域的缺点。为了更好地处理没有可用适合的目标区域的情况 (例如在 RHEL/CentOS 7 上就会出现此情况)，Python 3.7 还实现了*PEP 540: 强制 UTF-8 运行时模式*。

参见:

PEP 538 -- 强制转换传统 C 区域到基于 UTF-8 的区域 PEP 由 Nick Coghlan 撰写并实现。

2.3 PEP 540: 强制 UTF-8 运行时模式

新的 `-X utf8` 命令行选项和 `PYTHONUTF8` 环境变量可被用来启用 CPython 的 *UTF-8* 模式。

当处于 UTF-8 模式时，CPython 会忽略区域设置，并默认使用 UTF-8 编码。用于 `sys.stdin` 和 `sys.stdout` 流的错误处理器将设置为 `surrogateescape`。

强制 UTF-8 模式可被用来在嵌入的 Python 解释器中改变文本处理行为，而不会改变嵌入方应用的区域设置。

PEP 540 的 UTF-8 模式的好处是不必关心运行所在系统中有哪些可用区域即可工作，但它也存在对扩展模块 (例如 GNU `readline`)、运行非 Python 应用的子进程以及运行旧版本 Python 的子进程不起作用的缺点。为了减小与这些组件通信时破坏文本数据的风险，Python 3.7 还实现了*PEP 540: 强制 UTF-8 运行时模式*)。

UTF-8 模式在语言区域为 `C` 或 `POSIX` 并且 **PEP 538** 区域强制转换特性无法将其修改为某种基于 UTF-8 的替代项时会被默认启用（无论修改失败是由于设置了 `PYTHONCOERCECLOCALE=0`, `LC_ALL` 还是由于缺少适合的目标区域）。

参见:

PEP 540 -- 增加了新的 UTF-8 模式 PEP 由 Victor Stinner 撰写并实现

2.4 PEP 553: 内置的 `breakpoint()`

Python 3.7 包含了新的内置 `breakpoint()` 函数，作为一种简单方便地进入 Python 调试器的方式。

内置 `breakpoint()` 会调用 `sys.breakpointhook()`。在默认情况下后者会导入 `pdb` 然后再调用 `pdb.set_trace()`，但是通过将 `sys.breakpointhook()` 绑定到你选定的函数，`breakpoint()` 可以进入任何调试器。此外，环境变量 `PYTHONBREAKPOINT` 可被设置为你选定的调试器的可调用对象。设置 `PYTHONBREAKPOINT=0` 会完全禁用内置 `breakpoint()`。

参见:

PEP 553 -- 内置的 breakpoint() PEP 由 Barry Warsaw 撰写并实现

2.5 PEP 539: 用于线程局部存储的新 C API

虽然 Python 已提供了用于线程局部存储支持的 C API；但原有的 线程局部存储 (TLS) API 使用 `int` 来表示所有平台上的 TLS 密钥。对于官方支持的平台而言这通常不是问题，但这既不符合 POSIX 标准，也不具备任何实际意义上的可移植性。

PEP 539 通过向 CPython 提供了一个新的 线程特定存储 (TSS) API 来改变这一点，它取代了原有的 CPython 内部 TLS API 的使用，并且原有 API 已弃用。TSS API 使用一种新类型 `Py_tss_t` 而非 `int` 来表示 TSS 密钥 -- 这是一种不透明类型，其定义可能依赖于下层的 TLS 实现。因此，这将允许在以无法安全地转换为 `int` 的方式定义原生 TLS 密钥的平台上构建 CPython。

请注意在原生 TLS 密钥定义方式无法被安全地转换为 int 的平台上，原有 TLS API 中的全部函数将无法执行并会立即返回失败信息。这样能清楚地表明原有 API 在无法可靠使用的平台上不受支持，并且不会再尝试添加此类支持。

参见:

PEP 539 -- 在 CPython 中用于线程局部存储的新 C-API PEP 由 Erik M. Bray 撰写；由 Masayuki Yamamoto 实现。

2.6 PEP 562: 定制对模块属性的访问

Python 3.7 允许在模块上定义 __getattr__() 并且当以其他方式找不到某个模块属性时将会调用它。在模块上定义 __dir__() 现在也是允许的。

一个典型的可能有用的例子是已弃用模块属性和惰性加载。

参见:

PEP 562 -- 模块的 __getattr__ 和 __dir__ PEP 由 Ivan Levkivskyi 撰写并实现

2.7 PEP 564: 具有纳秒级精度的新时间函数

现代系统的时钟精度可以超过由 time.time() 函数及其变化形式所返回的浮点数的有限精度。为了避免精度损失，**PEP 564** 在 time 模块中增加了原有计时器函数的六个新“纳秒版”变化形式:

- time.clock_gettime_ns()
- time.clock_settime_ns()
- time.monotonic_ns()
- time.perf_counter_ns()
- time.process_time_ns()
- time.time_ns()

这些新函数会以整数值的形式返回纳秒数。

测量 表明在 Linux 和 Windows 上 time.time_ns() 的精度大约比 time.time() 要高 3 倍。

参见:

PEP 564 -- 增加具有纳秒级精度的新时间函数 PEP 由 Victor Stinner 撰写并实现

2.8 PEP 565: 在 __main__ 中显示 DeprecationWarning

DeprecationWarning 的默认处理方式已经被更改，这此警告默认只显示一次，仅有当直接在 __main__ 模块中运行的代码触发它们时才会再次显示。因此，单文件脚本开发者以及 Python 交互模式使用者应该会再次开始看到针对他们所使用 API 的已弃用警告，但被导入应用、库和框架模块所触发的已弃用警告默认将继续隐藏。

作为此项更改的结果，标准库现在允许开发者在三种不同的已弃用警告行为之间进行选择:

- FutureWarning: 默认情况下总是会显示，建议用于应用程序最终用户应该看到的警告信息（例如对于已弃用的应用程序配置的设置选项）。
- DeprecationWarning: 默认情况下仅在 __main__ 中以及当运行测试时会显示，建议用于其他 Python 开发者应该看到的警告信息，提示版本升级可能导致行为改变或者错误。

- `PendingDeprecationWarning`: 默认情况下仅在运行测试时会显示，可用于提示未来版本升级将会改变警告类别为 `DeprecationWarning` 或 `FutureWarning` 的情况。

在此之前 `DeprecationWarning` 和 `PendingDeprecationWarning` 都仅在运行测试时可见，这意味着主要编写单文件脚本或使用 Python 交互模式的开发者可能会因他们所用 API 突然出现的改变而感到惊讶。

参见:

PEP 565 -- 在 `__main__` 中显示 DeprecationWarning PEP 由 Nick Coghlan 撰写并实现

2.9 PEP 560: 对 `typing` 模块和泛型类型的核心支持

PEP 484 最初的设计方式使其不会向核心 CPython 解释器引入 任何更改。现在类型提示和 `typing` 模块已被社区广泛使用，因此这个限制已被取消。这个 PEP 引入了两个特殊方法 `__class_getitem__()` 和 `__mro_entries__`，这些方法现在被 `typing` 中的大多数类和特殊构造所使用。结果就是与类型相关的各类操作的速度提升了 7 倍，泛型类型可以在没有元类冲突的情况下被使用，而 `typing` 模块中几个长期存在的错误也已被修正。

参见:

PEP 560 -- 对 typing 模块和泛型类型的核心支持 PEP 由 Ivan Levkivskyi 撰写并实现

2.10 PEP 552: 基于哈希值的.pyc 文件

传统上 Python 检查字节码缓存文件 (即 `.pyc` 文件) 是否最新的方式是通过对源码元数据 (最后更改的时间戳和大小）和生成缓存时保存在其文件头中的源码元数据进行比较。这种检查方法虽然有效，但也存在缺点。当文件系统的时间戳太粗糙时，Python 有可能错过源码更新，导致用户感到困惑。此外，在缓存文件中存在时间戳对于 构建可再现 并且基于内容的构建系统来说是有问题的。

PEP 552 扩展了 pyc 格式以允许使用源文件的哈希值而非源文件的时间戳来检查有效性。这种 `.pyc` 文件就称为“基于哈希值的”。默认情况下，Python 仍然使用基于时间戳的有效性检查，不会在运行时生成基于哈希值的 `.pyc` 文件。基于哈希值的 `.pyc` 文件可以使用 `py_compile` 或 `compileall` 来生成。

基于哈希值的 `.pyc` 文件包含两种变体：已选定和未选定。Python 会在运行时针对相应源码文件验证已选定基于哈希值的 `.pyc` 文件，但对未选定基于哈希值的 pyc 文件则不会这样做。未选定基于哈希值的 `.pyc` 文件对于需要由 Python 外部的系统（例如构建系统）负责使 `.pyc` 文件保持最新的环境来说是一种有用的性能优化。

请参阅 pyc-invalidation 了解更多信息。

参见:

PEP 552 -- 确定性的 pyc 文件 PEP 由 Benjamin Peterson 撰写并实现

2.11 PEP 545: Python 文档翻译

PEP 545 描述了创建和维护 Python 文档翻译的整个过程。

新增了三个新的翻译版本:

- 日语: https://docs.python.org/ja/
- 法语: https://docs.python.org/fr/
- 韩语: https://docs.python.org/ko/

参见:

PEP 545 -- Python 文档翻译 PEP 由 Julien Palard, Inada Naoki 和 Victor Stinner 撰写并实现。

2.12 开发运行时模式: -X dev

新的 `-X dev` 命令行选项或新的 `PYTHONDEVMODE` 环境变量可被用来启用 CPython 的 *开发模式*。在开发模式下，CPython 会执行额外的在默认情况下开销过大的运行时检查。请参阅 `-X dev` 文档查看对于此模式效果的完整描述。

3 其他语言特性修改

- `await` 表达式和包含 `async for` 子句的推导式不允许在 格式化字符串字面值的表达式中使用。在 Python 3.7 中此限制已被取消。
- 现在可以将超过 255 个参数传递给一个函数，而现在一个函数也可以拥有超过 255 个形参。(由 Serhiy Storchaka 在 bpo-12844 和 bpo-18896 中贡献。)
- 现在 `bytes.fromhex()` 和 `bytearray.fromhex()` 会忽略所有 ASCII 空白符，而非仅是空格符. (由 Robert Xiao 在 bpo-28927 中贡献。)
- `str`, `bytes` 和 `bytearray` 获得了对新 `isascii()` 方法的支持，该方法可被用来测试是个字符串或字节串是否仅包含 ASCII 字符。(由 INADA Naoki 在 bpo-32677 中贡献。)
- 现在当 `from ... import ...` 失败时 `ImportError` 会显示模块名称和模块 `__file__` 路径。(由 Matthias Bussonnier 在 bpo-29546 中贡献。)
- 现在已支持涉及将子模块绑定到一个名称的绝对导入的循环导入。(由 Serhiy Storchaka 在 bpo-30024 中贡献。)
- 现在 `object.__format__(x, '')` 等价于 `str(x)` 而非 `format(str(self), '')`。(由 Serhiy Storchaka d bpo-28974 中贡献。)
- 为更好地支持栈跟踪的动态创建，现在 `types.TracebackType` 可以从 Python 代码中被实例化，并且 回溯对象的 `tb_next` 属性现在是可写的。(由 Nathaniel J. Smith 在 bpo-30579 中贡献。)
- 当使用 `-m` 开关时，现在 `sys.path[0]` 会主动扩展为完整的起始目录路径，而不是保持为空目录（这将允许在发生导入时从 当前工作目录导入）(由 Nick Coghlan 在 bpo-33053 中贡献。)
- 新的 `-X importtime` 选项或 `PYTHONPROFILEIMPORTTIME` 环境变量可被用来显示每次模块导入的时间。(由 Victor Stinner 在 bpo-31415 中贡献。)

4 新增模块

4.1 contextvars

新的 `contextvars` 模块和一组 新的 C API 引入了对 上下文变量的支持。上下文变量在概念上类似于线程局部变量。与 TLS 不同，上下文变量能正确地支持异步代码。

`asyncio` 和 `decimal` 已得到更新以使用和支持开箱即用的上下文变量。特别是激活的 decimal 上下文现在将存储在上下文变量中，它允许十进制运算在异步代码中使用正确的上下文。

参见:

PEP 567 -- 上下文变量 PEP 由 Yury Selivanov 撰写并实现

4.2 dataclasses

新的 `dataclass()` 装饰器提供了一种声明 *数据类* 的方式。数据类使用变量标注来描述其属性。它的构造器和其他魔术方法例如 `__repr__()`, `__eq__()` 以及 `__hash__()` 会自动地生成。

示例:

```
@dataclass
class Point:
    x: float
    y: float
    z: float = 0.0

p = Point(1.5, 2.5)
print(p)   # produces "Point(x=1.5, y=2.5, z=0.0)"
```

参见:

PEP 557 -- 数据类 PEP 由 Eric V. Smith 撰写并实现

4.3 importlib.resources

新的 `importlib.resources` 模块提供了一些新的 API 和一个新的 ABC 用于访问、打开和读取包内的 *资源*。资源基本上类似于包内的文件，但它们不一定是物理文件系统中实际的文件。模块加载器可以提供 `get_resource_reader()` 函数，它会返回一个 `importlib.abc.ResourceReader` 实例来支持这个新 API。内置的文件路径加载器和 zip 文件加载器都支持此特性。

由 Barry Warsaw 和 Brett Cannon 在 bpo-32248 中贡献。

参见:

importlib_resources -- 用于先前 Python 版本的 PyPI 下层接口。

5 改进的模块

5.1 argparse

新的 `ArgumentParser.parse_intermixed_args()` 方法允许混合选项与位置参数。（由 paul.j3 在 bpo-14191 中提供。）

5.2 asyncio

`asyncio` 模块获得了许多新的特性、可用性和性能提升。重要的改变包括:

- 新的 暂定 `asyncio.run()` 函数可被用于通过自动创建和销毁事件循环以基于同步代码运行协程。(由 Yury Selivanov 在 bpo-32314 中贡献。)
- asyncio 增加支持 `contextvars`. `loop.call_soon()`, `loop.call_soon_threadsafe()`, `loop.call_later()`, `loop.call_at()` 并且 `Future.add_done_callback()` 具有新的可选仅关键字参数 *context*。现在 `Tasks` 会自动跟踪其上下文。详情参见 **PEP 567**。(由 Yury Selivanov 在 bpo-32436 中贡献。)
- 增加了新的 `asyncio.create_task()` 函数作为 `asyncio.get_event_loop().create_task()` 的快捷方式。(由 Andrew Svetlov 在 bpo-32311 中贡献。)

- 新的 `loop.start_tls()` 方法可用于升级现有的 TLS 连接。(由 Yury Selivanov 在 bpo-23749 中贡献。)
- 新的 `loop.sock_recv_into()` 方法允许直接从套接字读取数据放入所提供的缓冲区，从而可以减少数据复制。(由 Antoine Pitrou 在 bpo-31819 中贡献。)
- 新的 `asyncio.current_task()` 函数可返回当前运行的 `Task` 实例，以及新的 `asyncio.all_tasks()` 函数可返回给定循环中所有现存 `Task` 实例的集合。`Task.current_task()` 和 `Task.all_tasks()` 方法已弃用。(由 Andrew Svetlov 在 bpo-32250 中贡献。)
- 新的 暂定 `BufferedProtocol` 类允许通过手动控制接收缓冲区来实现流式协议。(由 Yury Selivanov 在 bpo-32251 中贡献。)
- 新的 `asyncio.get_running_loop()` 函数可返回当前运行的循环，如果没有循环在运行则引发 `RuntimeError`。这与 `asyncio.get_event_loop()` 不同，后者在没有循环在运行时将 创建一个新的事件循环。(由 Yury Selivanov 在 bpo-32269 中提供。)
- 新的 `StreamWriter.wait_closed()` 协程方法允许执行等待直到流写入器被关闭。新的 `StreamWriter.is_closing()` 方法可用于确定写入器是否被关闭。(由 Andrew Svetlov 在 bpo-32391 中贡献。)
- 新的 `loop.sock_sendfile()` 协程方法允许在可能的情况下使用 `os.sendfile` 发送文件。(由 Andrew Svetlov 在 bpo-32410 中贡献。)
- 新的 `Future.get_loop()` 和 `Task.get_loop()` 方法会返回创建 task 或 future 对象的事件循环的实例。`Server.get_loop()` 允许为 `asyncio.Server` 对象执行同样操作。(由 Yury Selivanov 在 bpo-32415 中，以及由 Srinivas Reddy Thatiparthy 在 bpo-32418 中贡献。)
- 现在可以控制 `asyncio.Server` 的实例如何开启服务。之前，服务在创建后将立即开启服务。新的 *start_serving* 关键字参数已添加到 `loop.create_server()` 和 `loop.create_unix_server()`，并且 `Server.start_serving()`，和 `Server.serve_forever()` 可被用来分离服务的实例化和服务的开启。新的 `Server.is_serving()` 方法会在服务开启时返回 `True`。现在 `Server` 对象已是异步上下文管理器:

```
srv = await loop.create_server(...)

async with srv:
    # some code

# At this point, srv is closed and no longer accepts new connections.
```

(由 Yury Selivanov 在 bpo-32662 中贡献。)

- 由 `loop.call_later()` 所返回的回调对象已获得新的 `when()` 方法，该方法会返回一个排入计划日程的绝对时间戳。(由 Andrew Svetlov 在 bpo-32741 中贡献。)
- `loop.create_datagram_endpoint()` 方法已获得对 Unix 套接字的支持。(由 Quentin Dawans 在 bpo-31245 中贡献。)
- `asyncio.open_connection()`, `asyncio.start_server()` functions, `loop.create_connection()`, `loop.create_server()`, `loop.create_accepted_socket()` 方法及其对应的 UNIX 套接字变体现在接受 *ssl_handshake_timeout* 关键字参数。(由 Neil Aspinall 在 bpo-29970 中贡献。)
- 新的 `Handle.cancelled()` 方法会在回调被取消时返回 `True`。(由 Marat Sharafutdinov 在 bpo-31943 中贡献。)
- asyncio 源已被转换为使用 `async`/`await` 语法。(由 Andrew Svetlov 在 bpo-32193 中贡献。)
- 新的 `ReadTransport.is_reading()` 方法可用于确定传输的读取状态。此外，对 `ReadTransport.resume_reading()` 和 `ReadTransport.pause_reading()` 的调用现在是幂等的。(由 Yury Selivanov 在 bpo-32356 中贡献。)

- 接受套接字路径的循环方法现在支持传入 路径类对象。(由 Yury Selivanov 在 bpo-32066 中贡献。)
- 在 asyncio 中，Linux 上的 TCP 套接字现在创建时默认带有 TCP_NODELAY 旗标设置。(由 Yury Selivanov 和 Victor Stinner 在 bpo-27456 中贡献。)
- 在被取消任务中发生的异常不会再被记录。(由 Yury Selivanov 在 bpo-30508 中贡献。)
- 新的 WindowsSelectorEventLoopPolicy 和 WindowsProactorEventLoopPolicy 类。(由 Yury Selivanov 在 bpo-33792 中贡献。)

部分 asyncio API 改为已弃用。

5.3 binascii

b2a_uu() 函数现在接受可选的 *backtick* 关键字参数。当其为真值时，零会以 '`' 而非空格来表示。(由 Xiang Zhang 在 bpo-30103 中贡献。)

5.4 calendar

HTMLCalendar 类具有新的类属性，可以简化所生成 HTML 日历中 CSS 类的自定义。(由 Oz Tiram 在 bpo-30095 中贡献。)

5.5 collections

collections.namedtuple() 现在支持默认值。(由 Raymond Hettinger 在 bpo-32320 中贡献。)

5.6 compileall

compileall.compile_dir() 增加了新的 *invalidation_mode* 形参，可用于启用基于哈希值的.pyc 有效性检测。失效模式也可以在命令行中使用新的 --invalidation-mode 参数来指定。(由 Benjamin Peterson 在 bpo-31650 中贡献。)

5.7 concurrent.futures

ProcessPoolExecutor 和 ThreadPoolExecutor 现在支持新的 *初始化器*以及 *initargs* 构造器参数。(由 Antoine Pitrou 在 bpo-21423 中贡献。)

ProcessPoolExecutor 现在能通过新的 *mp_context* 参数来接受多进程上下文。(由 Thomas Moreau 在 bpo-31540 中贡献。)

5.8 contextlib

新的 nullcontext() 是一个比 ExitStack 更简单和快速的无操作上下文管理器。(由 Jesse-Bakker 在 bpo-10049 中贡献。)

增加了新的 asynccontextmanager(), AbstractAsyncContextManager 和 AsyncExitStack 以补充它们所对应的同步项。(由 Jelle Zijlstra 在 bpo-29679 和 bpo-30241 中，以及由 Alexander Mohr 和 Ilya Kulakov 在 bpo-29302 中贡献。)

5.9 cProfile

`cProfile` 命令行现在接受 `-m module_name` 作为脚本路径的替代。（由 Sanyam Khurana 在 bpo-21862 中贡献。）

5.10 crypt

`crypt` 模块现在支持 Blowfish 哈希方法。（由 Serhiy Storchaka 在 bpo-31664 中贡献。）

`mksalt()` 函数现在允许指定哈希操作的轮数。（由 Serhiy Storchaka 在 bpo-31702 中贡献。）

5.11 datetime

新的 `datetime.fromisoformat()` 方法会基于由 `datetime.isoformat()` 所输出的某一特定格式字符串构建 `datetime` 对象。（由 Paul Ganssle 在 bpo-15873 中贡献。）

`tzinfo` 类现在支持小于一分钟的偏移量。（由 Alexander Belopolsky 在 bpo-5288 中贡献。）

5.12 dbm

`dbm.dumb` 现在支持读取只读文件，并且在其未改变时不再写入索引文件。

5.13 decimal

`decimal` 模块现在使用上下文变量 来储存十进制值上下文。（由 Yury Selivanov 在 bpo-32630 中贡献。）

5.14 dis

`dis()` 函数现在能够反汇编嵌套的代码对象（推导式、生成器表达式和嵌套函数的代码，以及用于构建嵌套类的代码）。反汇编递归的最大深度由新的 *depth* 形参来控制。（由 Serhiy Storchaka 在 bpo-11822 中贡献。）

5.15 distutils

`README.rst` 现在包含在 distutils 的标准 README 列表之中，因而也包含在源码发布之中。（由 Ryan Gonzalez 在 bpo-11913 中贡献。）

5.16 enum

`Enum` 增加了新的 `_ignore_` 类特征属性，该属性允许列出不应当成为枚举成员的特征属性名称。（由 Ethan Furman 在 bpo-31801 中贡献。）

在 Python 3.8 中，尝试在 `Enum` 类中检查非 Enum 对象将引发 `TypeError`（例如 `1 in Color`）；类似地，尝试在 `Flag` 成员中检查非 Flag 对象也将引发 `TypeError`（例如 `1 in Perm.RW`）；目前，两种操作均会返回 `False` 并且已弃用。（由 Ethan Furman 在 bpo-33217 中贡献。）

5.17 functools

`functools.singledispatch()` 现在支持使用类型标注来注册实现。(由 Łukasz Langa 在 bpo-32227 中贡献。)

5.18 gc

新的 `gc.freeze()` 函数允许冻结由垃圾回收器所跟踪的所有对象，并将它们从未来的集合中排除。这可以在 POSIX `fork()` 调用之前使用以令 GC 友好地进行写入时复制或加速收集。新的 `gc.unfreeze()` 函数会反转此操作。此外，`gc.get_freeze_count()` 可被用于获取冻结对象的数量。(由 Li Zekun 在 bpo-31558 中贡献。)

5.19 hmac

`hmac` 现在具有经优化的一次性 `digest()` 函数，其速度比 `HMAC()` 要快三倍。(由 Christian Heimes 在 bpo-32433 中贡献。)

5.20 http.client

`HTTPConnection` 和 `HTTPSConnection` 现在支持新的 *blocksize* 参数以提升上传吞吐量。(由 Nir Soffer 在 bpo-31945 中贡献。)

5.21 http.server

`SimpleHTTPRequestHandler` 现在支持 HTTP `If-Modified-Since` 标头。如果目标文件在该标点指定的时间之后未被修改则服务器会返回 304 响应状态。(由 Pierre Quentel 在 bpo-29654 中贡献。)

`SimpleHTTPRequestHandler` 接受新的 *directory* 参数并增加了新的 `--directory` 命令行参数。通过此形参，服务器可以对服务指定目录，默认情况下它使用当前工作目录。(由 Stéphane Wirtel 和 Julien Palard 在 bpo-28707 中贡献。)

新的 `ThreadingHTTPServer` 类使用线程来处理使用 `ThreadingMixin` 的请求。它会在 `http.server` 附带 `-m` 运行时被使用。(由 Julien Palard 在 bpo-31639 中贡献。)

5.22 idlelib 与 IDLE

多个对自动补全的修正。(由 Louie Lu 在 bpo-15786 中贡献。)

Module Browser (在 File 菜单中，之前称为 Class Browser) 现在会在最高层级函数和类之外显示嵌套的函数和类。(由 Guilherme Polo, Cheryl Sabella 和 Terry Jan Reedy 在 bpo-1612262 中贡献。)

Settings 对话框 (Options 中的 Configure IDLE) 已经被部分重写以改进外观和功能。(由 Cheryl Sabella 和 Terry Jan Reedy 在多个问题项中贡献。)

字体样本现在包括一组非拉丁字符以便用户能更好地查看所选特定字体的效果。(由 Terry Jan Reedy 在 bpo-13802 中贡献。) 样本可以被修改以包括其他字符。(由 Serhiy Storchaka 在 bpo-31860 中贡献。)

之前以扩展形式实现的 IDLE 特性已作为正常特性重新实现。它们的设置已从 Extensions 选项卡移至其他对话框选项卡。(由 Charles Wohlganger 和 Terry Jan Reedy 在 bpo-27099 中实现。)

编辑器代码上下文选项已经过修改。Box 会显示所有上下文行直到最大行数。点击一个上下文行会使编辑器跳转到该行。自定义主题的上下文颜色已添加到 Settings 对话框的 Highlights 选项卡。(由 Cheryl Sabella 和 Terry Jan Reedy 在 bpo-33642, bpo-33768 和 bpo-33679 中贡献。)

在 Windows 上，会有新的 API 调用将 tk 对 DPI 的调整告知 Windows。在 Windows 8.1+ 或 10 上，如果 Python 二进制码的 DPI 兼容属性未改变，并且监视器分辨率大于 96 DPI，这应该会令文本和线条更清晰。否则的话它应该不造成影响。(由 Terry Jan Reedy 在 bpo-33656 中贡献。)

在 3.7.1 中新增:

超过 N 行（默认值为 50）的输出将被折叠为一个按钮。N 可以在 Settings 对话框的 General 页的 PyShell 部分中进行修改。数量较少但是超长的行可以通过在输出上右击来折叠。被折叠的输出可通过双击按钮来展开，或是通过右击按钮来放入剪贴板或是单独的窗口。(由 Tal Einat 在 bpo-1529353 中贡献。)

上述修改已被反向移植到 3.6 维护发行版中。

在 3.7.4 中新增:

在 Run 菜单中增加了"Run Customized" 以使用自定义设置来运行模块。输入的任何命令行参数都会被加入 sys.argv。它们在下次自定义运行时会再次显示在窗体中。用户也可以禁用通常的 Shell 主模块重启。(由 Cheryl Sabella, Terry Jan Reedy 等人在 bpo-5680 和 bpo-37627 中贡献。)

在 3.7.5 中新增:

在 IDLE 编辑器窗口中增加了可选的行序号。窗口打开时默认不带行序号，除非在配置对话框的 General 选项卡中进行设置。已打开窗口中的行序号可以在 Options 菜单中显示和隐藏。(由 Tal Einat 和 Saimadhav Heblikar 在 bpo-17535 中贡献。)

5.23 importlib

引入了 `importlib.abc.ResourceReader` ABC 以支持从包中加载资源。另请参阅*importlib.resources*。(由 Barry Warsaw, Brett Cannon 在 bpo-32248 中贡献。)

如果模块缺少规格描述 `importlib.reload()` 现在会引发 `ModuleNotFoundError`。(由 Garvit Khatri 在 bpo-29851 中贡献。)

如果指定的父模块不是一个包 (即缺少 `__path__` 属性) `importlib.find_spec()` 现在会引发 `ModuleNotFoundError` 而非 `AttributeError`。(由 Milan Oberkirch 在 bpo-30436 中贡献。)

新的 `importlib.source_hash()` 可被用来计算传入源的哈希值。基于哈希值的*.pyc* 文件 会嵌入此函数所返回的值。

5.24 io

新的 `TextIOWrapper.reconfigure()` 方法可用于根据新的设置重新配置文本流。(由 Antoine Pitrou 在 bpo-30526 以及 INADA Naoki 在 bpo-15216 中贡献。)

5.25 ipaddress

methods of `ipaddress.IPv6Network` 和 `ipaddress.IPv4Network` 中新的 `subnet_of()` 以及 `supernet_of()` 方法可用于网络包含测试。(由 Michel Albert 和 Cheryl Sabella 在 bpo-20825 中贡献。)

5.26 itertools

`itertools.islice()` 现在接受 类整数对象作为 start, stop 和 slice 参数。(由 Will Roberts 在 bpo-30537 中贡献。)

5.27 locale

`locale.format_string()` 中新的 *monetary* 参数可用于转换所使用的千位分隔符和分组字符串。(由 Garvit 在 bpo-10379 中贡献。)

现在 `locale.getpreferredencoding()` 函数在 Android 上或是在强制 *UTF-8* 模式 下总是返回 `'UTF-8'`。

5.28 logging

`Logger` 实例现在可以被 pickle。(由 Vinay Sajip 在 bpo-30520 中贡献。)

新的 `StreamHandler.setStream()` 方法可用于在句柄创建之后替换日志流。(由 Vinay Sajip 在 bpo-30522 中创建。)

现在可以在传递给 `logging.config.fileConfig()` 的配置信息中对句柄构造器指定关键字参数。(由 Preston Landers 在 bpo-31080 中贡献。)

5.29 math

新的 `math.remainder()` 函数实现了 IEEE 754 风格的余数运算。(由 Mark Dickinson 在 bpo-29962 中贡献。)

5.30 mimetypes

.bmp 的 MIME type 从 `'image/x-ms-bmp'` 改为 `'image/bmp'`。(由 Nitish Chandra 在 bpo-22589 中贡献。)

5.31 msilib

新的 `Database.Close()` 方法可用于关闭 MSI (last-in, first-out) 数据库。(由 Berker Peksag 在 bpo-20486 中贡献。)

5.32 multiprocessing

新的 `Process.close()` 方法会显式地关闭进程对象并释放与其关联的所有资源。如果底层进程仍在运行则将引发 `ValueError`。(由 Antoine Pitrou 在 bpo-30596 中贡献。)

新的 `Process.kill()` 方法可用于在 Unix 上使用 `SIGKILL` 信号来终止进程。(由 Vitor Pereira 在 bpo-30794 中贡献。)

由 `Process` 所创建的非守护线程现在会在进程退出时被合并。(由 Antoine Pitrou 在 bpo-18966 中贡献。)

5.33 os

`os.fwalk()` 现在接受 `bytes` 类型的 *path* 参数。(由 Serhiy Storchaka 在 bpo-28682 中贡献。)

`os.scandir()` 已获得对 文件描述器的支持。(由 Serhiy Storchaka 在 bpo-25996 中贡献。)

新的 `register_at_fork()` 函数允许注册 Python 回调以便在进程分叉中执行。(由 Antoine Pitrou 在 bpo-16500 中贡献。)

增加了 `os.preadv()` (结合了 `os.readv()` 与 `os.pread()` 的功能) 以及 `os.pwritev()` 函数 (结合了 `os.writev()` 和 `os.pwrite()` 的功能)。(由 Pablo Galindo 在 bpo-31368 中贡献。)

`os.makedirs()` 的 mode 参数不再影响新创建的中间层级目录的文件权限。(由 Serhiy Storchaka 在 bpo-19930 中贡献。)

`os.dup2()` 现在会返回新的文件描述器。之前，返回的总是 `None`。(由 Benjamin Peterson 在 bpo-32441 中贡献。)

在 Solaris 及其派生系统上 `os.stat()` 所返回的结构现在会包含 `st_fstype` 属性。(由 Jesús Cea Avión 在 bpo-32659 中贡献。)

5.34 pathlib

在 POSIX 类系统上新的 `Path.is_mount()` 方法现在可用于确定一个路径是否为挂载点。(由 Cooper Ry Lees 在 bpo-30897 中贡献。)

5.35 pdb

`pdb.set_trace()` 现在接受一个可选的限关键字参数 *header*。如果给出，它会在调试开始之前被打印到控制台。(由 Barry Warsaw 在 bpo-31389 中贡献。)

`pdb` 命令行现在接受 `-m module_name` 作为对脚本文件的替代。(由 Mario Corchero 在 bpo-32206 中贡献。)

5.36 py_compile

`py_compile.compile()` -- 及其扩展形式 `compileall` -- 现在会通过无条件地为基于哈希值的有效性验证创建 `.pyc` 文件来支持 `SOURCE_DATE_EPOCH` 环境变量。这样可以确保当 `.pyc` 文件被主动创建时 可重现的生成。(由 Bernhard M. Wiedemann 在 bpo-29708 中贡献。)

5.37 pydoc

pydoc 服务器现在可以绑定到由新的 `-n` 命令行参数所指定的任意主机名。(由 Feanil Patel 在 bpo-31128 中贡献。)

5.38 queue

新的 `SimpleQueue` 类是一个无界的 FIFO (last-in, first-out) 队列。(由 Antoine Pitrou 在 bpo-14976 中贡献。)

5.39 re

旗标 `re.ASCII`, `re.LOCALE` 和 `re.UNICODE` 可以在组的范围内设置。(由 Serhiy Storchaka 在 bpo-31690 中贡献。)

`re.split()` 现在支持基于匹配一个空字符串的模式例如 `r'\b'`, `'^$'` 或 `(?=-)` 进行拆分。(由 Serhiy Storchaka 在 bpo-25054 中贡献。)

使用 `re.LOCALE` 旗标编译的正则表达式不再依赖于编译时的区域设置。区域设置仅在已编译正则表达式被使用时才被应用。(由 Serhiy Storchaka 在 bpo-30215 中贡献。)

现在如果一个正则表达式包含语义将在未来发生改变的字符集构造，则会引发 `FutureWarning`，例如嵌套集与集合操作等。(由 Serhiy Storchaka 在 bpo-30349 中贡献。)

已编译正则表达式和匹配对象现在可以使用 `copy.copy()` 和 `copy.deepcopy()` 进行拷贝。(由 Serhiy Storchaka 在 bpo-10076 中贡献。)

5.40 signal

`signal.set_wakeup_fd()` 函数新增的 *warn_on_full_buffer* 参数可以指定当唤醒缓冲区溢出时 Python 是否要在 stderr 上打印警告信息。(由 Nathaniel J. Smith 在 bpo-30050 中贡献。)

5.41 socket

新增的 `socket.getblocking()` 方法会在套接字处于阻塞模式时返回 `True`，否则返回 `False`。(由 Yury Selivanov 在 bpo-32373 中贡献。)

新的 `socket.close()` 函数可关闭所传入的套接字文件描述符。应该用此函数来代替 `os.close()` 以获得更好的跨平台兼容性。(由 Christian Heimes 在 bpo-32454 中贡献。)

`socket` 模块现在会公开 `socket.TCP_CONGESTION` (Linux 2.6.13), `socket.TCP_USER_TIMEOUT` (Linux 2.6.37) 以及 `socket.TCP_NOTSENT_LOWAT` (Linux 3.12) 常量。(由 Omar Sandoval 在 bpo-26273 以及 Nathaniel J. Smith 在 bpo-29728 中贡献。)

已加入对 `socket.AF_VSOCK` 套接字的支持以允许在虚拟机及其宿主机之间进行通讯。(由 Cathy Avery 在 bpo-27584 中贡献。)

套接字现在默认会根据文件描述符自动检测所属族、类型和协议。(由 Christian Heimes 在 bpo-28134 中贡献。)

5.42 socketserver

socketserver.ThreadingMixIn.server_close() 现在会等待所有非守护线程完成。socketserver.ForkingMixIn.server_close() 现在会等待所有子进程完成。

为 socketserver.ForkingMixIn 和 socketserver.ThreadingMixIn 类增加了新的 socketserver.ForkingMixIn.block_on_close 类属性。该类属性值设为 False 以保持 3.7 之前的行为。

5.43 sqlite3

现在当下层的 SQLite 库版本为 3.6.11 及以上时 sqlite3.Connection 会开放 backup() 方法。(由 Lele Gaifax 在 bpo-27645 中贡献。)

sqlite3.connect() 的 *database* 参数现在接受任何 path-like object，而不是只接受字符串。(由 Anders Lorentsen 在 bpo-31843 中贡献。)

5.44 ssl

ssl 模块现在使用 OpenSSL 的内置 API 代替 match_hostname() 来检查主机名或 IP 地址。值的验证会在 TLS 握手期间进行。任何证书验证错误包括主机名检查失败现在将引发 SSLCertVerificationError 并使用正确的 TLS Alert 消息中止握手过程。这个新异常包含有额外的信息。主机名验证可通过 SSLContext.hostname_checks_common_name 进行自定义。(由 Christian Heimes 在 bpo-31399 中贡献。)

注解： 改进的主机名检测需要有兼容 OpenSSL 1.0.2 或 1.1 的 *libssl* 实现。因此，OpenSSL 0.9.8 和 1.0.1 不再被支持（请参阅平台支持的移除 了解详情）。目前 ssl 模块主要兼容 LibreSSL 2.7.2 及更高版本。

---

ssl 模块不再以 SNI TLS 扩展发送 IP 地址。(由 Christian Heimes 在 bpo-32185 中贡献。)

match_hostname() 不再支持部分通配符例如 www*.example.org。(由 Mandeep Singh 在 bpo-23033 以及 Christian Heimes 在 bpo-31399 中贡献。)

ssl 模块默认的加密套件选择现在是使用黑名单方式而非硬编码的白名单。Python 不会再重新启用已经被 OpenSSL 安全更新所阻止的加密。默认的加密套件选择可以在编译时进行配置。(由 Christian Heimes 在 bpo-31429 中贡献。)

现在已支持包含国际化域名 (IDN) 的服务器证书验证。作为此更改的一部分，SSLSocket.server_hostname 属性现在会以预期的 A 标签形式 ("xn--pythn-mua.org") 而不是以 U 标签形式 ("pythön.org") 存储。(由 Nathaniel J. Smith 与 Christian Heimes 在 bpo-28414 中贡献。)

ssl 模块对 TLS 1.3 和 OpenSSL 1.1.1 具有初步和实验性的支持。在 Python 3.7.0 发布的时刻，OpenSSL 1.1.1 仍在开发中，而 TLS 1.3 尚未最终确定。TLS 1.3 握手和协议行为与 TLS 1.2 及更早的版本略有差异，请参阅 ssl-tlsv1_3。(由 Christian Heimes 在 bpo-32947, bpo-20995, bpo-29136, bpo-30622 以及 bpo-33618 中贡献。)

SSLSocket 和 SSLObject 不再具有公共构造器。直接实例化从未成为有文档和受支持的特性。实际必须通过 SSLContext 的方法 wrap_socket() 和 wrap_bio() 来创建。(由 Christian Heimes 在 bpo-32951 中贡献。)

用于设置最小和最大 TLS 协议版本的 OpenSSL 1.1 API 现已可用，名称分别为 SSLContext.minimum_version 和 SSLContext.maximum_version。受支持的协议由几个新增旗标指定，例如 HAS_TLSv1_1。(由 Christian Heimes 在 bpo-32609 中贡献。)

增加了 SSLContext.post_handshake_auth 以及 ssl.SSLSocket.verify_client_post_handshake() 来启用并初始化 TLS 1.3 握手后验证。(由 Christian Heimes 在 bpo-34670 中贡献。)

5.45 string

`string.Template` 现在允许你有选择地分别修改带大括号的占位符和不带大括号的占位符所对应的正则表达式模式。(由 Barry Warsaw 在 bpo-1198569 中贡献。)

5.46 subprocess

`subprocess.run()` 函数接受新的 *capture_output* 关键字参数。当其为真值时，将会捕获 stdout 和 stderr。这相当于将 `subprocess.PIPE` 作为 *stdout* 和 *stderr* 参数传入。(由 Bo Bayles 在 bpo-32102 中贡献。)

`subprocess.run` 函数和 `subprocess.Popen` 构造器现在接受 *text* 关键字参数作为 *universal_newlines* 的别名。(由 Andrew Clegg 在 bpo-31756 中贡献。)

在 Windows 中当重定向标准句柄时 *close_fds* 的默认值由 `False` 改为 `True`。现在可以在重定向标准句柄时将 *close_fds* 设为真值。参阅 `subprocess.Popen`。这意味着现在 *close_fds* 在所有受支持的平台上默认值均为 `True`。(由 Segev Finer 在 bpo-19764 中贡献。)

在 `subprocess.call()`, `subprocess.run()` 期间或在 `Popen` 上下文管理器中，subprocess 模块现在能更优雅地处理 `KeyboardInterrupt`。它现在会等待一小段时间以便子进程退出，然后再继续处理 `KeyboardInterrupt` 异常。(由 Gregory P. Smith 在 bpo-25942 中贡献。)

5.47 sys

新增 `sys.breakpointhook()` 钩子函数，供内置的 `breakpoint()` 进行调用。(由 Barry Warsaw 在 bpo-31353 中贡献。)

在 Android 中新增的 `sys.getandroidapilevel()` 会返回构建时的 Android API 版本。(由 Victor Stinner 在 bpo-28740 中贡献。)

新的 `sys.get_coroutine_origin_tracking_depth()` 函数会返回当前协程的由新的 `sys.set_coroutine_origin_tracking_depth()` 所设定的原始跟踪深度。`asyncio` 已转换为使用这个新 API 代替已弃用的 `sys.set_coroutine_wrapper()`。(由 Nathaniel J. Smith 在 bpo-32591 中贡献。)

5.48 time

PEP 564 向 `time` 模块增加六个具有纳秒级精度的新函数:

- `time.clock_gettime_ns()`
- `time.clock_settime_ns()`
- `time.monotonic_ns()`
- `time.perf_counter_ns()`
- `time.process_time_ns()`
- `time.time_ns()`

增加了新的时钟标识符:

- `time.CLOCK_BOOTTIME` (Linux): 与 `time.CLOCK_MONOTONIC` 相似，不同点在于它还包括任何系统挂起的时间。
- `time.CLOCK_PROF` (FreeBSD, NetBSD 和 OpenBSD): 高精度的分进程 CPU 计时器。
- `time.CLOCK_UPTIME` (FreeBSD, OpenBSD): 该时间的绝对值是系统运行且未挂起的时间，提供准确的正常运行时间度量。

新的 time.thread_time() 和 time.thread_time_ns() 函数可用于获取每线程的 CPU 时间度量。(由 Antoine Pitrou 在 bpo-32025 中贡献。)

新的 time.pthread_getcpuclockid() 函数会返回特定线程中 CPU 时钟的时钟 ID。

5.49 tkinter

新的 tkinter.ttk.Spinbox 类现已可用。(由 Alan Moore 在 bpo-32585 中贡献。)

5.50 tracemalloc

tracemalloc.Traceback 的行为更接近正规的回溯，会对所有帧按从最旧到最新来排序。Traceback.format() 现在接受负的 *limit*，并会将结果截短至排在第 abs(limit) 位的旧帧。如果要获得旧的行为，请在 Traceback.format() 中使用新的 *most_recent_first* 参数。(由 Jesse Bakker 在 bpo-32121 中贡献。)

5.51 types

新的 WrapperDescriptorType, MethodWrapperType, MethodDescriptorType 和 ClassMethodDescriptorType 类现已可用。(由 Manuel Krebber 和 Guido van Rossum 在 bpo-29377 以及 Serhiy Storchaka 在 bpo-32265 中贡献。)

新的 types.resolve_bases() 函数会以 **PEP 560** 所规定的方式动态解析 MRO 条目。(由 Ivan Levkivskyi 在 bpo-32717 中贡献。)

5.52 unicodedata

内部的 unicodedata 数据库已升级为使用 Unicode 11。(由 Benjamin Peterson 贡献。)

5.53 unittest

新的 -k 命令行选项允许通过名称子字符串或类似于 Unix shell 的模式来筛选测试项。例如，python -m unittest -k foo 将运行 foo_tests.SomeTest.test_something, bar_tests.SomeTest.test_foo，但不会运行 bar_tests.FooTest.test_something。(由 Jonas Haag 在 bpo-32071 中贡献。)

5.54 unittest.mock

现在 sentinel 属性会在它们被 复制或 封存时保存其标识。(由 Serhiy Storchaka 在 bpo-20804 中贡献。)

新的 seal() 函数允许 Mock 对实例进行密封，这将禁止进一步创建属性模拟。密封会以递归方式应用于自身模拟的所有属性。(由 Mario Corchero 在 bpo-30541 中贡献。)

5.55 urllib.parse

`urllib.parse.quote()` 已经从 **RFC 2396** 更新为 **RFC 3986**，将 ~ 添加到默认情况下从未引用的字符集。（由 Christian Theune 和 Ratnadeep Debnath 在 bpo-16285 中贡献。）

5.56 uu

`uu.encode()` 函数现在接受可选的 *backtick* 关键字参数。当其为真时，零会以 `'`'` 而非空格来表示。（由 Xiang Zhang 在 bpo-30103 中贡献。）

5.57 uuid

新的 `UUID.is_safe` 属性会从平台中继有关是否使用多进程安全模式来生成所需 UUID 的信息。（由 Barry Warsaw 在 bpo-22807 中贡献。）

`uuid.getnode()` 现在更倾向于统一管理的 MAC 地址而不是本地管理的 MAC 地址。这样可以更好地保证从 `uuid.uuid1()` 返回的 UUID 的全局唯一性。如果只有本地管理的 MAC 地址可用，则返回首个找到的此类地址。（由 Barry Warsaw 在 bpo-32107 中贡献。）

5.58 warnings

默认警告过滤器的初始化已进行以下更改:

- 通过命令行选项（包括 `-b` 以及新的 CPython 专属的 `-X dev` 选项）启用的警告总是会通过 `sys.warnoptions` 属性被传递给警告机制。
- 通过命令行或环境变量启用的警告过滤器现在具有以下优先顺序:
  - 用于 `-b` (或 `-bb`) 的 `BytesWarning` 过滤器
  - 通过 `-W` 选项指定的任何过滤器
  - 通过 `PYTHONWARNINGS` 环境变量指定的任何过滤器
  - 任何其他 CPython 专属过滤器（例如 `-X dev` 模式中新增的 `default` 过滤器）
  - 由警告机制所定义的任何隐式过滤器
- 在 CPython 调试版本中，现在默认情况下会显示所有警告（隐式过滤器列表为空）

（由 Nick Coghlan 和 Victor Stinner 在 bpo-20361, bpo-32043 以及 bpo-32230 中贡献。）

在单文件脚本和交互式提示符中，默认情况下会再次显示已弃用警告。详情参见*PEP 565: 在 __main__ 中显示 DeprecationWarning*。（由 Nick Coghlan 在 bpo-31975 中贡献。）

5.59 xml

作为对 DTD 和外部实体检索的缓解，在默认情况下 `xml.dom.minidom` 和 `xml.sax` 模块不再处理外部实体。（由 Christian Heimes 在 bpo-17239 中贡献。）

5.60 xml.etree

`find()` 方法中的 ElementPath 描述词现在可以将当前节点文本与 `[. = "text"]` 进行比较，而不仅是子节点中的文本。描述词还允许添加空格以提高可读性。（由 Stefan Behnel 在 bpo-31648 中贡献。）

5.61 xmlrpc.server

`SimpleXMLRPCDispatcher.register_function` 现在可以被用作装饰器。（由 Xiang Zhang 在 bpo-7769 中贡献。）

5.62 zipapp

函数 `create_archive()` 现在接受可选的 *filter* 参数，以允许用户选择哪些文件应被加入归档包。（由 Irmen de Jong 在 bpo-31072 中贡献。）

函数 `create_archive()` 现在接受可选的 *compressed* 参数，以生成压缩归档包。另外也加入了命令行选项 `--compress` 以支持压缩。（由 Zhiming Wang 在 bpo-31638 中贡献。）

5.63 zipfile

`ZipFile` 现在接受新的 *compresslevel* 形参，以控制压缩级别。（由 Bo Bayles 在 bpo-21417 中贡献。）

`ZipFile` 所创建的归档包中的子目录现在会按字母表顺序保存。（由 Bernhard M. Wiedemann 在 bpo-30693 中贡献。）

6 C API 的改变

已实现了用于线程本地存储的新 API。相关概述请参阅*PEP 539: 用于线程局部存储的新 C API*，完整参考文档请查看 thread-specific-storage-api。（由 Masayuki Yamamoto 在 bpo-25658 中贡献。）

新的上下文变量 功能开放了许多 新的 C API。

新的 `PyImport_GetModule()` 函数会返回之前所导入的具有给定名称的模块。（由 Eric Snow 在 bpo-28411 中贡献。）

新的 `Py_RETURN_RICHCOMPARE` 宏可以简化丰富比较函数的编写。（由 Petr Victorin 在 bpo-23699 中贡献。）

新的 `Py_UNREACHABLE` 宏可用于标记不可到达的代码路径。（由 Barry Warsaw 在 bpo-31338 中贡献。）

`tracemalloc` 现在通过新的 `PyTraceMalloc_Track()` 和 `PyTraceMalloc_Untrack()` 函数公开了一个 C API。（由 Victor Stinner 在 bpo-30054 中贡献。）

新的 `import__find__load__start()` 和 `import__find__load__done()` 静态标记可用于跟踪模块导入。（由 Christian Heimes 在 bpo-31574 中贡献。）

结构体 `PyMemberDef`, `PyGetSetDef`, `PyStructSequence_Field`, `PyStructSequence_Desc` 和 `wrapperbase` 的字段 `name` 和 `doc` 现在的类型为 `const char *` 而不是 `char *`。（由 Serhiy Storchaka 在 bpo-28761 中贡献。）

`PyUnicode_AsUTF8AndSize()` 和 `PyUnicode_AsUTF8()` 的结果类型现在是 `const char *` 而非 `char *`。（由 Serhiy Storchaka 在 bpo-28769 中贡献。）

`PyMapping_Keys()`, `PyMapping_Values()` 和 `PyMapping_Items()` 的结果现在肯定是列表，而非可能是列表也可能是元组。（由 Oren Milman 在 bpo-28280 中贡献。）

添加了函数 `PySlice_Unpack()` 和 `PySlice_AdjustIndices()`。(由 Serhiy Storchaka 在 bpo-27867 中贡献。)

`PyOS_AfterFork()` 已弃用，建议改用新的 functions `PyOS_BeforeFork()`, `PyOS_AfterFork_Parent()` 和 `PyOS_AfterFork_Child()`。(由 Antoine Pitrou 在 bpo-16500 中贡献。)

曾经作为公共 API 一部分的 `PyExc_RecursionErrorInst` 单例已被移除，因为它的成员永远不会被清理，可能在解释器的最终化过程中导致段错误。由 Xavier de Gaye 在 bpo-22898 和 bpo-30697 中贡献。

添加 C API 对使用 timezone 的构造器 `PyTimeZone_FromOffset()` 和 `PyTimeZone_FromOffsetAndName()` 的时区的支持，以及通常 `PyDateTime_TimeZone_UTC` 使用 UTC 单例。由 Paul Ganssle 在 bpo-10381 中贡献。

`PyThread_start_new_thread()` 和 `PyThread_get_thread_ident()` 的结果类型以及 `PyThreadState_SetAsyncExc()` 的 *id* 参数类型由 `long` 改为 `unsigned long`。(由 Serhiy Storchaka 在 bpo-6532 中贡献。)

现在 `PyUnicode_AsWideCharString()` 如果第二个参数为 `NULL` 并且 `wchar_t*` 字符串包含空字符则会引发 `ValueError`。(由 Serhiy Storchaka 在 bpo-30708 中贡献。)

对启动顺序以及动态内存分配器管理的更改意味着早已记录在案的，对在调用大多数 C API 函数之前调用 `Py_Initialize()` 的要求的依赖现在变得更加强烈，未遵循此要求可能导致嵌入式应用程序中的段错误。请参阅此文档的*移植到 Python 3.7* 一节以及 C API 文档的 pre-init-safe 一节了解更多细节。

新的 `PyInterpreterState_GetID()` 会返回给定解释器的唯一 ID。(由 Eric Snow 在 bpo-29102 中贡献。)

现在当启用*UTF-8 模式* 时 `Py_DecodeLocale()`, `Py_EncodeLocale()` 会使用 UTF-8 编码。(由 Victor Stinner 在 bpo-29240 中贡献。)

`PyUnicode_DecodeLocaleAndSize()` 和 `PyUnicode_EncodeLocale()` 现在会为 `surrogateescape` 错误句柄使用当前区域编码。(由 Victor Stinner 在 bpo-29240 中贡献。)

`PyUnicode_FindChar()` 的 *start* 和 *end* 形参的行为现在调整为与字符串切片类似。(由 Xiang Zhang 在 bpo-28822 中贡献。)

7 构建的改变

对于 `--without-threads` 构建的支持已被移除。`threading` 模块现在将总是可用。(由 Antoine Pitrou 在 bpo-31370 中贡献。)

在非 OSX UNIX 平台上已不再包含用于构建 `_ctypes` 模块的完整 libffi 副本。现在当在此类平台上构建 `_ctypes` 时需要事先装有 libffi 的副本。(由 Zachary Ware 在 bpo-27979 中贡献。)

Windows 构建过程不再依赖 Subversion 来拉取外部源码，而是改用一段 Python 脚本从 GitHub 下载 zip 文件。如果未在系统中找到 Python 3.6 (通过 `py -3.6`)，则会使用 NuGet 下载一份 32 位的 Python 副本用于此目的。(由 Zachary Ware 在 bpo-30450 中贡献。)

`ssl` 模块需要兼容 OpenSSL 1.0.2 或 1.1 的 libssl。OpenSSL 1.0.1 的生命期已于 2016-12-31 终止且不再受支持。LibreSSL 暂时也不受支持。LibreSSL 发布版直到 2.6.4 版还缺少所需的 OpenSSL 1.0.2 API。

8 性能优化

通过移植更多代码来使用 METH_FASTCALL 的约定，可以显著地减少调用 C 代码中实现的各类标准库的很多方法的开销。(由 Victor Stinner 在 bpo-29300、bpo-29507、bpo-29452 以及 bpo-29286 中贡献。)

通过各种优化方式，使 Python 在 Linux 上的启动时间缩短了 10%，在 macOS 上缩短了 30%。(由 Victor Stinner, INADA Naoki 在 bpo-29585 中，以及 Ivan Levkivskyi 在 bpo-31333 中贡献。)

由于避免创建绑定方法案例的字节码更改，方法调用速度现在加快了 20%。(由 Yury Selivanov 和 INADA Naoki 在 bpo-26110 中贡献。)

对 asyncio 模块里面的一些常用函数做了显著的性能优化。

- asyncio.get_event_loop() 函数已经改用 C 重新实现，使其执行速度加快了 15 倍。(由 Yury Selivanov 在 bpo-32296 中贡献。)
- asyncio.Future 回调管理已经过优化。(由 Yury Selivanov 在 bpo-32348 中贡献。)
- asyncio.gather() 的执行速度现在加快了 15%。(由 Yury Selivanov 在 bpo-32355 中贡献。)
- 当 *delay* 参数为零或负值时 asyncio.sleep() 的执行速度现在加快了 2 倍。(由 Andrew Svetlov 在 bpo-32351 中贡献。)
- asyncio 调试模式的执行开销已获减轻。(由 Antoine Pitrou 在 bpo-31970 中贡献。)

作为*PEP 560 工作* 的结果，typing 的导入时间已减少了 7 倍，许多与类型相关的操作现在会执行得更快。(由 Ivan Levkivskyi 在 bpo-32226 中贡献。)

sorted() 和 list.sort() 已经过优化，在通常情况下执行速度可提升 40-75%。(由 Elliot Gorokhovsky 在 bpo-28685 中贡献。)

dict.copy() 的执行速度现在加快了 5.5 倍。(由 Yury Selivanov 在 bpo-31179 中贡献。)

当 *name* 未找到并且 *obj* 未重载 object.__getattr__() 或 object.__getattribute__() 时 hasattr() 和 getattr() 现在会比原来快大约 4 倍。(由 INADA Naoki 在 bpo-32544 中贡献。)

在字符串中搜索特定的 Unicode 字符（例如乌克兰语字母 “Є”）会比搜索其他字符慢上 25 倍。而现在最坏情况下也只会慢上 3 倍。(由 Serhiy Storchaka 在 bpo-24821 中贡献。)

collections.namedtuple() 工厂对象已经重写实现，使得创建具名元组的速度加快了 4 到 6 倍。(由 Jelle Zijlstra 在 bpo-28638 中贡献，进一步的改进由 INADA Naoki, Serhiy Storchaka 和 Raymond Hettinger 贡献。)

现在 date.fromordinal() 和 date.fromtimestamp() 在通常情况下执行速度可提升 30%。(由 Paul Ganssle 在 bpo-32403 中贡献。)

由于使用了 os.scandir()，现在 os.fwalk() 函数执行速度提升了 2 倍。(由 Serhiy Storchaka 在 bpo-25996 中贡献。)

由于使用了 os.scandir() 函数，shutil.rmtree() 函数的执行速度已经提升了 20--40%。(由 Serhiy Storchaka 在 bpo-28564 中贡献。)

正则表达式忽略大小写的匹配和搜索已获得优化。现在搜索某些模式的速度提升了 20 倍。(由 Serhiy Storchaka 在 bpo-30285 中贡献。)

re.compile() 现在会将 flags 形参转换为 int 对象，如果它是 RegexFlag 的话。它现在会和 Python 3.5 一样快，而比 Python 3.6 快大约 10%，实际速度取决于具体的模式。(由 INADA Naoki 在 bpo-31671 中贡献。)

selectors.EpollSelector, selectors.PollSelector 和 selectors.DevpollSelector 这几个类的 modify() 方法在重负载下可以加快 10% 左右。(由 Giampaolo Rodola' 在 bpo-30014 中贡献。)

常量折叠已经从窥孔优化器迁移至新的 AST 优化器，后者可以以更高的一致性来执行优化。(由 Eugene Toder 和 INADA Naoki 在 bpo-29469 和 bpo-11549 中贡献。)

abc 中的大部分函数和方法已经用 C 重写。这使得创建抽像基类以及调用其 isinstance() 和 issubclass() 的速度加快了 1.5 倍。这也使得 Python 启动耗时减少了 10%。(由 Ivan Levkivskyi 和 INADA Naoki 在 bpo-31333 中贡献。)

在不构造子类时，通过使用快速路径构造器使得 datetime.date 和 datetime.datetime 的替代构造器获得了显著的速度提升。(由 Paul Ganssle 在 bpo-32403 中贡献。)

在特定情况下 array.array 实例的比较速度已获得很大提升。现在当比较存放相同的整数类型的值的数组时会比原来快 10 到 70 倍。(由 Adrian Wielgosik 在 bpo-24700 中贡献。)

在大多数平台上 math.erf() 和 math.erfc() 函数现在使用（更快的）C 库实现。(由 Serhiy Storchaka 在 bpo-26121 中贡献。)

9 其他 CPython 实现的改变

- 跟踪钩子现在可以选择不接收 line 而是选择从解释器接收 opcode 事件，具体做法是在被跟踪的帧上相应地设置新的 f_trace_lines 和 f_trace_opcodes 属性。(由 Nick Coghlan 在 bpo-31344 中贡献。)
- 修复了一些命名空间包模块属性的一致性问题。命名空间模块对象的 __file__ 被设置为 None (原先未设置)，对象的 __spec__.origin 也被设置为 None (之前为字符串 "namespace")。参见 bpo-32305。而且，命名空间模块对象的 __spec__.loader 被设置的值与 __loader__ 相同 (原先前者被设置为 None)。参见 bpo-32303。
- locals() 字典现在以变量定义的词法顺序显示。原先未定义顺序。(由 Raymond Hettinger 在 bpo-32690 中贡献。)
- distutils upload 命令不会再试图将行结束字符 CR 改为 CRLF。这修复了 sdists 的一个以与 CR 等价的字节结束的数据损坏问题。(由 Bo Bayles 在 bpo-32304 中贡献。)

10 已弃用的 Python 行为

在推导式和生成器表达式中的 yield 语句（包括 yield 和 yield from 子句）现在已弃用（最左端的 for 子句中的可迭代对象表达式除外）。这确保了推导式总是立即返回适当类型的容器（而不是有可能返回 generator iterator 对象），这样生成器表达式不会试图将它们的隐式输出与任何来自显式 yield 表达式的输出交错起来。在 Python 3.7 中，这样的表达式会在编译时引发 DeprecationWarning，在 Python 3.8 中则将引发 SyntaxError。(由 Serhiy Storchaka 在 bpo-10544 中贡献。)

从 object.__complex__() 返回一个 complex 的子类的行为已弃用并将在未来的 Python 版本中引发错误。这使得 __complex__() 的行为与 object.__int__() 和 object.__float__() 保持一致。(由 Serhiy Storchaka 在 bpo-28894 中贡献。)

11 已弃用的 Python 模块、函数和方法

11.1 aifc

aifc.openfp() 已弃用并将在 Python 3.9 中被移除。请改用 aifc.open()。(由 Brian Curtin 在 bpo-31985 中贡献。)

11.2 asyncio

对 `asyncio.Lock` 和其他 asyncio 同步原语的 `await` 实例的直接支持已弃用。想要获取并释放同步资源必须使用异步上下文管理器。(由 Andrew Svetlov 在 bpo-32253 中贡献。)

`asyncio.Task.current_task()` 和 `asyncio.Task.all_tasks()` 方法已弃用。(由 Andrew Svetlov 在 bpo-32250 中贡献。)

11.3 collections

在 Python 3.8 中，`collections.abc` 内的抽象基类将不会再通过常规的 `collections` 模块公开。这有助于更清晰地区别具体类与抽象基类。(由 Serhiy Storchaka 在 bpo-25988 中贡献。)

11.4 dbm

`dbm.dumb` 现在支持读取只读文件，且当其未被更改时不会再写入索引文件。现在如果索引文件丢失并在 `'r'` 与 `'w'` 模式下被重新创建，则会发出已弃用警告（在未来的 Python 发布版中将改为错误)。(由 Serhiy Storchaka 在 bpo-28847 中贡献。)

11.5 enum

在 Python 3.8 中，尝试在 `Enum` 类中检查非 Enum 对象将引发 `TypeError` (例如 `1 in Color`)；类似地，尝试在 `Flag` 成员中检查非 Flag 对象也将引发 `TypeError` (例如 `1 in Perm.RW`)；目前，两种操作均会返回 `False`。(由 Ethan Furman 在 bpo-33217 中贡献。)

11.6 gettext

使用非整数值在 `gettext` 中选择复数形式现在已弃用。它从未正确地发挥作用。(由 Serhiy Storchaka 在 bpo-28692 中贡献。)

11.7 importlib

下列方法 `MetaPathFinder.find_module()` (被 `MetaPathFinder.find_spec()` 替代) 和 `PathEntryFinder.find_loader()` (被 `PathEntryFinder.find_spec()` 替代) 都在 Python 3.4 中已弃用，现在会引发 `DeprecationWarning`。(由 Matthias Bussonnier 在 bpo-29576 中贡献)

`importlib.abc.ResourceLoader` ABC 已弃用，推荐改用 `importlib.abc.ResourceReader`。

11.8 locale

`locale.format()` 已弃用，请改用 `locale.format_string()`。(由 Garvit 在 bpo-10379 中贡献。)

11.9 macpath

`macpath` 现在已弃用，将在 Python 3.8 中被移除。(由 Chi Hsuan Yen 在 bpo-9850 中贡献。)

11.10 threading

`dummy_threading` 和 `_dummy_thread` 已弃用。构建禁用线程的 Python 已不再可能。请改用 `threading`。(由 Antoine Pitrou 在 bpo-31370 中贡献。)

11.11 socket

`socket.htons()` 和 `socket.ntohs()` 中的静默参数截断已弃用。在未来的 Python 版本中，如果传入的参数长度大于 16 比特位，将会引发异常。(由 Oren Milman 在 bpo-28332 中贡献。)

11.12 ssl

`ssl.wrap_socket()` 已弃用。请改用 `ssl.SSLContext.wrap_socket()`。(由 Christian Heimes 在 bpo-28124 中贡献。)

11.13 sunau

`sunau.openfp()` 已弃用并将在 Python 3.9 中被移除。请改用 `sunau.open()`。(由 Brian Curtin 在 bpo-31985 中贡献。)

11.14 sys

已弃用 `sys.set_coroutine_wrapper()` 和 `sys.get_coroutine_wrapper()`。

未写入文档的 `sys.callstats()` 函数已弃用并将在未来的 Python 版本中被移除。(由 Victor Stinner 在 bpo-28799 中贡献。)

11.15 wave

`wave.openfp()` 已弃用并将在 Python 3.9 中被移除。请改用 `wave.open()`。(由 Brian Curtin 在 bpo-31985 中贡献。)

12 已弃用的 C API 函数和类型

如果 `Py_LIMITED_API` 未设定或设定为范围在 `0x03050400` 和 `0x03060000` (不含) 之间，或为 `0x03060100` 或更高的值，函数 `PySlice_GetIndicesEx()` 已弃用并被一个宏所替代。(由 Serhiy Storchaka 在 bpo-27867 中贡献。)

`PyOS_AfterFork()` 已弃用。请改用 `PyOS_BeforeFork()`，`PyOS_AfterFork_Parent()` 或 `PyOS_AfterFork_Child()`。(由 Antoine Pitrou 在 bpo-16500 中贡献。)

13 平台支持的移除

- 官方已不再支持 FreeBSD 9 及更旧的版本。
- 为了完整的 Unicode 支持，包括在扩展模块之内，*nix 平台现在至少应当提供 `C.UTF-8` (完整区域), `C.utf8` (完整区域) 或 `UTF-8` (`LC_CTYPE` 专属区域) 中的一个作为基于 `ASCII` 的传统 `C` 区域的替代。
- OpenSSL 0.9.8 和 1.0.1 已不再受支持，这意味着在仍然使用这些版本的旧平台上构建带有 SSL/TLS 支持的 CPython 3.7 时，需要自定义构建选项以链接到更新的 OpenSSL 版本。

  注意，此问题会影响到 Debian 8 (代号“jessie”) 和 Ubuntu 14.04 (代号“Trusty”) 等长期支持 Linux 发行版，因为它们默认仍然使用 OpenSSL 1.0.1。

  Debian 9 (“stretch”) 和 Ubuntu 16.04 (“xenial”) 以及其他最新的长期支持 Linux 发行版 (例如 RHEL/CentOS 7.5, SLES 12-SP3) 都使用 OpenSSL 1.0.2 或更新的版本，因此继续在默认的构建配置中受到支持。

  CPython 自己的 CI 配置文件 提供了一个使用 CPython 测试套件中的 SSL 兼容性测试架构 基于 OpenSSL 1.1.0 进行构建和链接的例子，而不是使用过时的系统所提供的 OpenSSL。

14 API 与特性的移除

下列特性与 API 已从 Python 3.7 中移除:

- `os.stat_float_times()` 函数已被移除。它在 Python 2.3 中被引入用于向下兼容 Python 2.2，并自 Python 3.1 起就已弃用。
- 在 `re.sub()` 的替换模块中由 `'\'` 与一个 ASCII 字母构成的未知转义在 Python 3.5 中已弃用，现在将会引发错误。
- 在 `tarfile.TarFile.add()` 中移除了对 *exclude* 参数的支持。它在 Python 2.7 和 3.2 中已弃用。请改用 *filter* 参数。
- `ntpath` 模块中的 `splitunc()` 函数在 Python 3.1 中已弃用，现在已被移除。请改用 `splitdrive()` 函数。
- `collections.namedtuple()` 不再支持 *verbose* 形参或 `_source` 属性，该属性会显示为具名元组类所生成的源代码。这是加速类创建的设计优化的一部分。(由 Jelle Zijlstra 在 bpo-28638 中贡献，进一步的改进由 INADA Naoki, Serhiy Storchaka 和 Raymond Hettinger 贡献。)
- 函数 `bool()`, `float()`, `list()` 和 `tuple()` 不再接受关键字参数。`int()` 的第一个参数现在只能作为位置参数传入。
- 移除了之前在 Python 2.4 中已弃用的 `plistlib` 模块的类 `Plist`, `Dict` 和 `_InternalDict`。作为函数 `readPlist()` 和 `readPlistFromBytes()` 返回结果的 Dict 值现在为普通 dict。你不能再使用属性访问来获取这些字典的项。
- `asyncio.windows_utils.socketpair()` 函数已被移除。请改用 `socket.socketpair()` 函数，它自 Python 3.5 起就在所有平台上可用。`asyncio.windows_utils.socketpair` 在 Python 3.5 及更新版本上只是 `socket.socketpair` 的别名。
- `asyncio` 不再将 `selectors` 和 `_overlapped` 模块导出为 `asyncio.selectors` 和 `asyncio._overlapped`。请将 `from asyncio import selectors` 替换为 `import selectors`。
- 现在已禁止直接实例化 `ssl.SSLSocket` 和 `ssl.SSLObject` 对象。相应构造器从未写入文档、也从未作为公有构造器进行测试或设计。用户应当使用 `ssl.wrap_socket()` 或 `ssl.SSLContext`。(由 Christian Heimes 在 bpo-32951 中贡献。)
- 未被使用的 `distutils install_misc` 命令已被移除。(由 Eric N. Vander Weele 在 bpo-29218 中贡献。)

15 移除的模块

`fpectl` 模块已被移除。它从未被默认启用，从未在 x86-64 上正确发挥效果，并且它对 Python ABI 的改变会导致 C 扩展的意外损坏。（由 Nathaniel J. Smith 在 bpo-29137 中贡献。）

16 Windows 专属的改变

Python 启动器（py.exe）可以接受 32 位或 64 位标记而 **不必**同时指定一个小版本。因此 `py -3-32` 和 `py -3-64` 与 `py -3.7-32` 均为有效，并且现在还接受 *-m*-64 和 *-m.n*-64 来强制使用 64 位 python 命令，即使是本应使用 32 位的时候。如果指定版本不可用则 py.exe 将报错退出。（由 Steve Barnes 在 bpo-30291 中贡献。）

启动器可以运行 `py -0` 来列出已安装的所有 python，默认版本会以星号标出。运行 `py -0p` 将同时列出相应的路径。如果运行 py 时指定了无法匹配的版本，它将显示 简短形式的可用版本列表。（由 Steve Barnes 在 bpo-30362 中贡献。）

17 移植到 Python 3.7

本节列出了先前描述的更改以及可能需要更改代码的其他错误修正.

17.1 Python 行为的更改

- `async` 和 `await` 现在是保留关键字。使用了这些名称作为标识符的代码现在将引发 `SyntaxError`。（由 Jelle Zijlstra 在 bpo-30406 中贡献。）
- **PEP 479** 在 Python 3.7 中对所有代码启用，在协程和生成器中直接或间接引发的 `StopIteration` 异常会被转换为 `RuntimeError` 异常。（由 Yury Selivanov 在 bpo-32670 中贡献。）
- `object.__aiter__()` 方法不再能被声明为异步的。（由 Yury Selivanov 在 bpo-31709 中贡献。）
- 由于一个疏忽，之前的 Python 版本会错误地接受以下语法:

```
f(1 for x in [1],)

class C(1 for x in [1]):
    pass
```

  现在 Python 3.7 会正确地引发 `SyntaxError`，因为生成器表达式总是必须直接包含于一对括号之内，且前后都不能有逗号，仅在调用时可以忽略重复的括号。（由 Serhiy Storchaka 在 bpo-32012 和 bpo-32023 中贡献。）

- 现在当使用 `-m` 开关时，会将初始工作目录添加到 `sys.path`，而不再是一个空字符串（即在每次导入时动态地指明当前工作目录）。任何会检测该空字符串，或是以其他方式依赖之前行为的的程序将需要进行相应的更新（例如改为还要检测 `os.getcwd()` 或 `os.path.dirname(__main__.__file__)`，具体做法首先要取决于为何要对代码执行空字符串检测）。

17.2 更改的 Python API

- `socketserver.ThreadingMixIn.server_close()` 现在会等待所有非守护线程完成。将新增的 `socketserver.ThreadingMixIn.block_on_close` 类属性设为 `False` 可获得 3.7 之前版本的行为。(由 Victor Stinner 在 bpo-31233 和 bpo-33540 中贡献。)
- `socketserver.ForkingMixIn.server_close()` 现在会等等所有子进程完成。将新增的 `socketserver.ForkingMixIn.block_on_close` 类属性设为 `False` 可获得 3.7 之前版本的行为。(由 Victor Stinner 在 bpo-31151 和 bpo-33540 中贡献。)
- 某些情况下 `locale.localeconv()` 函数现在会临时将 `LC_CTYPE` 区域设置为 `LC_NUMERIC` 的值。(由 Victor Stinner 在 bpo-31900 中贡献。)
- 如果 *path* 为字符串 `pkgutil.walk_packages()` 现在会引发 `ValueError`。之前则是返回一个空列表。(由 Sanyam Khurana 在 bpo-24744 中贡献。)
- `string.Formatter.format()` 的格式字符串参数现在为 仅限位置参数。将其作为关键字参数传入的方式自 Python 3.5 起已弃用。(由 Serhiy Storchaka 在 bpo-29193 中贡献。)
- 类 `http.cookies.Morsel` 的属性 `key`, `value` 和 `coded_value` 现在均为只读。对其赋值的操作自 Python 3.5 起已弃用。要设置它们的值请使用 `set()` 方法。(由 Serhiy Storchaka 在 bpo-29192 中贡献。)
- `os.makedirs()` 的 *mode* 参数不会再影响新建中间层级目录的文件权限位。要设置它们的文件权限位你可以在发起调用 `makedirs()` 之前设置 umask。(由 Serhiy Storchaka 在 bpo-19930 中贡献。)
- `struct.Struct.format` 的类型现在是 `str` 而非 `bytes`。(由 Victor Stinner 在 bpo-21071 中贡献。)
- `parse_multipart()` 现在接受 *encoding* 和 *errors* 参数并返回与 `FieldStorage` 同样的结果：对于非文件字段，与键相关联的值是一个字符串列表，而非字节串。(由 Pierre Quentel 在 bpo-29979 中贡献。)
- 由于 `socket` 中的内部更改，在由旧版 Python 中的 `socket.share` 所创建的套接字上调用 `socket.fromshare()` 已不受支持。
- `BaseException` 的 `repr` 已更改为不包含末尾的逗号。大多数异常都会受此更改影响。(由 Serhiy Storchaka 在 bpo-30399 中贡献。)
- `datetime.timedelta` 的 `repr` 已更改为在输出中包含关键字参数。(由 Utkarsh Upadhyay 在 bpo-30302 中贡献。)
- 因为 `shutil.rmtree()` 现在是使用 `os.scandir()` 函数实现的，用户指定的句柄 *onerror* 现在被调用时如果列目录失败会附带第一个参数 `os.scandir` 而不是 `os.listdir`。
- 未来可能加入在正则表达式中对 Unicode 技术标准 #18 中嵌套集合与集合操作的支持。这会改变现有语法。为了推动这项未来的改变，目前在有歧义的情况下会引发 `FutureWarning`。这包括以字面值 `'['` 开头或包含字面值字符序列 `'--'`, `'&&'`, `'~~'` 以及 `'||'` 的集合。要避免警告，请用反斜杠对其进行转义。(由 Serhiy Storchaka 在 bpo-30349 中贡献。)
- 基于可以匹配空字符串的 正则表达式对字符串进行拆分的结果已被更改。例如基于 `r'\s*'` 的拆分现在不仅会像原先那样拆分空格符，而且会拆分所有非空格字符之前和字符串结尾处的空字符串。通过将模式修改为 `r'\s+'` 可以恢复原先的行为。自 Python 3.5 开始此类模式将会引发 `FutureWarning`。

  对于同时匹配空字符串和非空字符串的模式，在其他情况下搜索所有匹配的结果也可能会被更改。例如在字符串 `'a\n\n'` 中，模式 `r'(?m)^\s*?$'` 将不仅会匹配位置 2 和 3 上的空字符串，还会匹配位置 2--3 上的字符串 `'\n'`。想要只匹配空行，模式应当改写为 `r'(?m)^[^\S\n]*$'`。

  `re.sub()` 现在会替换与前一个的非空匹配相邻的空匹配。例如 `re.sub('x*', '-', 'abxd')` 现在会返回 `'-a-b--d-'` 而不是 `'-a-b-d-'` ('b' 和'd' 之间的第一个减号是替换'x'，而第二个减号则是替换'x' 和'd' 之间的空字符串)。

  (由 Serhiy Storchaka 在 bpo-25054 和 bpo-32308 中贡献。)
- `re.escape()` 更改为只转义正则表达式特殊字符，而不转义 ASCII 字母、数字和 `'_'` 以外的所有字符。(由 Serhiy Storchaka 在 bpo-29995 中贡献。)

- tracemalloc.Traceback 帧现在是按从最旧到最新排序，以便与 traceback 更为一致。（由 Jesse Bakker 在 bpo-32121 中贡献。）
- 在支持 socket.SOCK_NONBLOCK 或 socket.SOCK_CLOEXEC 标志位的操作系统上，socket.type 不再应用它们。因此，像 if sock.type == socket.SOCK_STREAM 之类的检测会在所有平台上按预期的方式工作。（由 Yury Selivanov 在 bpo-32331 中贡献。）
- 在 Windows 上当重定向标准句柄时，subprocess.Popen 的 *close_fds* 参数的默认值从 False 更改为 True。如果你以前依赖于在使用带有标准 io 重定向的 subprocess.Popen 时所继承的句柄，则必须传入 close_fds=False 以保留原先的行为，或是使用 STARTUPINFO.lpAttributeList。
- importlib.machinery.PathFinder.invalidate_caches() -- 此方法隐式地影响 importlib.invalidate_caches() -- 现在会删除 sys.path_importer_cache 中被设为 None 的条目。（由 Brett Cannon 在 bpo-33169 中贡献。）
- 在 asyncio 中，loop.sock_recv()，loop.sock_sendall()，loop.sock_accept()，loop.getaddrinfo()，loop.getnameinfo() 已被更改为正确的协程方法以与培训五日文档相匹配。之前，这些方法会返回 asyncio.Future 实例。（由 Yury Selivanov 在 bpo-32327 中贡献。）
- asyncio.Server.sockets 现在会返回服务器套接字列表的副本，而不是直接地返回它。（由 Yury Selivanov 在 bpo-32662 中贡献。）
- Struct.format 现在是一个 str 实例而非 bytes 实例。（由 Victor Stinner 在 bpo-21071 中贡献。）
- 现在可以通过将 required=True 传给 ArgumentParser.add_subparsers() 使得 argparse 子解析器成为必需的。（由 Anthony Sottile 在 bpo-26510 中贡献。）
- ast.literal_eval() 现在更为严格。任意地加减数字已不再被允许。（由 Serhiy Storchaka 在 bpo-31778 中贡献。）
- 当一个日期超出 0001-01-01 到 9999-12-31 范围时 Calendar.itermonthdates 现在将始终如一地引发异常，以便支持不能容忍此类异常的应用程序，可以使用新增的 Calendar.itermonthdays3 和 Calendar.itermonthdays4。这些新方法返回元组，并且其不受 datetime.date 所支持的范围限制。（由 Alexander Belopolsky 在 bpo-28292 中贡献。）
- collections.ChainMap 现在会保留底层映射的顺序。（由 Raymond Hettinger 在 bpo-32792 中贡献。）
- 如果在解释器关闭期间被调用，concurrent.futures.ThreadPoolExecutor 和 concurrent.futures.ProcessPoolExecutor 的 submit() 方法现在会引发 RuntimeError。（由 Mark Nemec 在 bpo-33097 中贡献。）
- configparser.ConfigParser 构造器现在使用 read_dict() 来处理默认值，以使其行为与解析器的其余部分保持致。在默认字典中的非字符串键和值现在会被隐式地转换为字符串。（由 James Tocknell 在 bpo-23835 中贡献。）
- 一些未写入文档的内部导入已被移除。一个例子是 os.errno 已不再可用；应改为直接使用 import errno。请注意此类未写入文档的内部导入可能未经通知地随时被移除，甚至是在微版本号发行版中移除。

17.3 C API 中的改变

函数 PySlice_GetIndicesEx() 被认为对于大小可变的序列来说并不安全。如果切片索引不是 int 的实例，而是实现了 __index__() 方法的对象，则序列可以在其长度被传给 PySlice_GetIndicesEx() 之后调整大小。这可能导致返回超出序列长度的索引号。为了避免可能的问题，请使用新增的函数 PySlice_Unpack() 和 PySlice_AdjustIndices()。（由 Serhiy Storchaka 在 bpo-27867 中贡献。）

17.4 CPython 字节码的改变

新增了两个操作码: `LOAD_METHOD` 和 `CALL_METHOD`。(由 Yury Selivanov 和 INADA Naoki 在 bpo-26110 中贡献。)

`STORE_ANNOTATION` 操作码已被移除。(由 Mark Shannon 在 bpo-32550 中贡献。)

17.5 Windows 专属的改变

用于重载 `sys.path` 的文件现在命名为 `<python-executable>._pth` 而不是 `'sys.path'`。请参阅 finding_modules 了解更多信息。(由 Steve Dower 在 bpo-28137 中贡献。)

17.6 其他 CPython 实现的改变

为了准备在未来对公开的 CPython 运行时初始化 API 进行潜在更改(请参阅 **PEP 432** 获取最初但略为过时的文稿),CPython 内部的启动和配置管理逻辑已经过大幅重构。虽然这些更新旨在对嵌入式应用程序和常规的 CPython CLI 用户都完全透明,但它们在这里被提及是因为重构会改变解释器启动期间许多操作的内部顺序,因此可能提示出原先隐藏的缺陷,这可能存在于嵌入式应用程序中,或是在 CPython 自身内部。(最初由 Nick Coghlan 和 Eric Snow 作为 bpo-22257 的一部分贡献,并由 Nick, Eric 和 Victor Stinner 在一系列其他问题报告中进一步更新)。已知会受到影响的一些细节:

- `PySys_AddWarnOptionUnicode()` 目前对嵌入式应用程序不可用,因为在调用 *Py_Initialize* 之前需要创建 Unicode 对象。请改用 `PySys_AddWarnOption()`。
- 嵌入式应用程序通过 `PySys_AddWarnOption()` 所添加的警告过滤器现在应该以更高的一致性优先于由解释器所设置的默认过滤器

由于默认警告过滤器的配置方式发生了变化,将 `Py_BytesWarningFlag` 设置为大于一的值不再足以在发出 `BytesWarning` 消息的同时将其转换为异常。而是改为必须设置旗标(以便首先发出警告),以及添加显式的 `error::BytesWarning` 警告过滤器来将其转换为异常。

由于编译器处理文档字符串的方式发生了变化,一个仅由文档字符串构成的函数体中隐式的 `return None` 现在被标记为在与文档字符串相同的行,而不是在函数的标题行。

当前异常状态已从帧对象移到协程对象。这会简化解释器并修正由于在进入或退出生成器时具有交换异常状态而导致的一些模糊错误。(由 Mark Shannon 在 bpo-25612 中贡献。)

18 Python 3.7.1 中的重要变化

从 3.7.1 开始, `Py_Initialize()` 现在始终会读取并遵循与 `Py_Main()` 相同的环境设置 (在更早的 Python 版本中,它会遵循一个错误定义的环境变量子集,而在 Python 3.7.0 中则会由于 bpo-34247 而完全不读取它们)。如果不想要此行为,请在调用 `Py_Initialize()` 之前将 `Py_IgnoreEnvironmentFlag` 设为 1。

在 3.7.1 中,上下文变量的 C API 已 获得更新以使用 `PyObject` 指针。另请参阅 bpo-34762。

在默认情况下 `xml.dom.minidom` 和 `xml.sax` 模块将不再处理外部实体。另请参阅 bpo-17239。

在 3.7.1 中,当提供不带末尾新行的输入时 `tokenize` 模块现在会隐式地添加 `NEWLINE` 形符。此行为现在已与 C 词法分析器的内部行为相匹配。(由 Ammar Askar 在 bpo-33899 中贡献。)

19 Python 3.7.2 中的重要变化

在 3.7.2 中，Windows 下的 `venv` 不再复制原来的二进制文件，而是改为创建名为 `python.exe` 和 `pythonw.exe` 的重定向脚本。这解决了一个长期存在的问题，即所有虚拟环境在每次 Python 升级后都必须进行升级或是重新创建。然而，要注意此发布版仍然要求重新创建虚拟环境以获得新的脚本。

20 Python 3.7.6 中的重要变化

出于重要的安全性考量，`asyncio.loop.create_datagram_endpoint()` 的 *reuse_address* 形参不再被支持。这是由 UDP 中的套接字选项 `SO_REUSEADDR` 的行为导致的。更多细节请参阅 `loop.create_datagram_endpoint()` 的文档。(由 Kyle Stanley, Antoine Pitrou 和 Yury Selivanov 在 bpo-37228 中贡献。。)

索引

非字母

P

R

S