What's New in Python

发布 *3.9.1*

A. M. Kuchling

十二月 16, 2020

Python Software Foundation
Email: docs@python.org

Contents

发布版本 3.9.1

日期 十二月 16, 2020

编者 Łukasz Langa

1 摘要 - 发布重点

新的语法特性：

- **PEP 584**，为 `dict` 增加合并运算符；
- **PEP 585**，标准多项集中的类型标注泛型。
- **PEP 614**，放宽对装饰器的语法限制。

新的内置特性：

- **PEP 616**，移除前缀和后缀的字符串方法。

标准库中的新特性:

- **PEP 593**，灵活的函数和变量标注；
- 添加了 `os.pidfd_open()` 以允许不带竞争和信号的进程管理。

解释器的改进:

- **PEP 573**，从 C 扩展类型的方法快速访问模块状态；
- **PEP 617**，CPython 现在使用基于 PEG 的新解析器；
- 许多 Python 内置类型（range, tuple, set, frozenset, list, dict）现在通过使用 **PEP 590** vectorcall 获得了加速；
- 垃圾回收不会因恢复的对象而阻塞；
- 许多 Python 模块 (`_abc`, `audioop`, `_bz2`, `_codecs`, `_contextvars`, `_crypt`, `_functools`, `_json`, `_locale`, `math`, `operator`, `resource`, `time`, `_weakref`) 现在会使用 PEP 489 所定义的多阶段初始化；
- 许多标准库模块 (`audioop`, `ast`, `grp`, `_hashlib`, `pwd`, `_posixsubprocess`, `random`, `select`, `struct`, `termios`, `zlib`) 现在会使用 PEP 384 所定义的稳定 ABI。

新的库模块：

- **PEP 615**，IANA 时区数据库现在已包含于标准库的 `zoneinfo` 模块；
- 图的拓扑排序实现现在已由新的 `graphlib` 模块提供。

发布进程的变化:

- **PEP 602**，CPython 采用年度发布周期。

2 你应该在你的代码中检查 DeprecationWarning.

在 Python 2.7 仍受支持的时候，有许多旧功能在 Python 3 中被保留以维持对 Python 2.7 的向下兼容。随着对 Python 2 支持的结束，这些向下兼容层已经被移除或即将被移除。它们中的大部分都持续多年引发 `DeprecationWarning` 警告。例如，使用 `collections.Mapping` 而不是 `collections.abc.Mapping` 自 2012 年发布的 Python 3.3 开始即会引发 `DeprecationWarning`。

请使用 `-W default` 命令行选项测试你的应用程序来查看 `DeprecationWarning` 和 `PendingDeprecationWarning`，甚至可以使用 `-W error` 将它们视为错误。可以使用 警告过滤器来忽略来自第三方代码的警告。

Python 3.9 是最后一个提供这些 Python 2 向下兼容层的版本，以给予 Python 项目维护者更多时间来组织移除 Python 2 支持并添加 Python 3.9 支持。

`collections` 模块中 抽象基类的别名，例如 `collections.abc.Mapping` 的别名 `collections.Mapping` 会为向下兼容最后保留一个发行版。它们将在 Python 3.10 中被移除。

更一般地说，请尝试在 Python 开发模式下运行你的测试，这可以帮助你的代码兼容以后的 Python 版本。

Note: a number of pre-existing deprecations were removed in this version of Python as well. Consult the 移除 section.

3 新的特性

3.1 字典合并与更新运算符

合并 (|) 与更新 (|=) 运算符已被加入内置的 dict 类。它们为现有的 dict.update 和 {**d1, **d2} 字典合并方法提供了补充。

示例:

```
>>> x = {"key1": "value1 from x", "key2": "value2 from x"}
>>> y = {"key2": "value2 from y", "key3": "value3 from y"}
>>> x | y
{'key1': 'value1 from x', 'key2': 'value2 from y', 'key3': 'value3 from y'}
>>> y | x
{'key2': 'value2 from x', 'key3': 'value3 from y', 'key1': 'value1 from x'}
```

请参阅 **PEP 584** 了解详情。(由 Brandt Bucher 在 bpo-36144 中贡献。)

3.2 新增用于移除前缀和后缀的字符串方法

增加了 str.removeprefix(prefix) 和 str.removesuffix(suffix) 用于方便地从字符串移除不需要的前缀或后缀。也增加了 bytes, bytearray 以及 collections.UserString 的对应方法。请参阅 **PEP 616** 了解详情。(由 Dennis Sweeney 在 bpo-39939 中贡献。)

3.3 标准多项集中的类型标注泛型

在类型标注中现在你可以使用内置多项集类型例如 list 和 dict 作为通用类型而不必从 typing 导入对应的大写形式类型名 (例如 List 和 Dict)。标准库中的其他一些类型现在同样也是通用的，例如 queue.Queue。

示例:

```
def greet_all(names: list[str]) -> None:
    for name in names:
        print("Hello", name)
```

请参阅 **PEP 585** 了解详情。(由 Guido van Rossum, Ethan Smith 和 Batuhan Taşkaya 在 bpo-39481 中贡献。)

3.4 新的解析器

Python 3.9 使用于基于 PEG 的新解析器替代 LL(1)。新解析器的性能与旧解析器大致相当，但 PEG 在设计新语言特性时的形式化比 LL(1) 更灵活。我们将在 Python 3.10 及之后版本中开始使用这种灵活性。

ast 模块会使用新解析器并会生成与旧解析器一致的 AST。

在 Python 3.10 中，旧解析器将被移除，依赖于它的所有功能也将被移除(主要是 parser 模块，它早已被弃用)。只有在 Python 3.9 中，你可以使用命令行开关 (-X oldparser) 或环境变量 (PYTHONOLDPARSER=1) 切换回 LL(1) 解析器。

请参阅 **PEP 617** 了解详情。(由 Guido van Rossum, Pablo Galindo 和 Lysandros Nikolaou 在 bpo-40334 中贡献。)

4 其他语言特性修改

- `__import__()` 现在会引发 `ImportError` 而不是 `ValueError`，后者曾经会在相对导入超出其最高层级包时发生。(由 Ngalim Siregar 在 bpo-37444 中贡献。)
- Python 现在会获取命令行中指定的脚本文件名 (例如: `python3 script.py`) 的绝对路径: `__main__` 模块的 `__file__` 属性将是一个绝对路径，而不是相对路径。现在此路径在当前目录通过 `os.chdir()` 被改变后仍将保持有效。作为附带效果，回溯信息也将在此情况下为 `__main__` 模块帧显示绝对路径。(由 Victor Stinner 在 bpo-20443 中贡献。)
- 在 Python 开发模式以及调试编译版本中，现在会针对字符串编码和解码操作检查 *encoding* 和 *errors* 参数。例如: `open()`, `str.encode()` 和 `bytes.decode()`。

  默认设置下，为保证性能，*errors* 参数只会在第一次发生编码/解码错误时被检查，并且对于空字符串 *encoding* 参数有时会被忽略。(由 Victor Stinner 在 bpo-37388 中贡献。)
- `"".replace("", s, n)` 对于所有非零的 `n` 都将返回 `s` 而不是空字符串。现在此方法会与 `"".replace("", s)` 保持一致。对于 `bytes` 和 `bytearray` 对象也有类似的修改。(由 Serhiy Storchaka 在 bpo-28029 中贡献。)
- 任何有效的表达式现在都可被用作 decorator。在之前版本中，相关语法则更为严格。请参阅 **PEP 614** 了解详情。(由 Brandt Bucher 在 bpo-39702 中贡献。)
- 改进了 `typing` 模块的帮助信息。现在将为所有特殊形式和特殊通用别名 (例如 `Union` 和 `List`) 显示文档字符串。使用 `help()` 时传入通用别名例如 `List[int]` 将显示对应实体类型 (这里对应的是 `list`) 的帮助信息。(由 Serhiy Storchaka 在 bpo-40257 中贡献。)
- `aclose()` / `asend()` / `athrow()` 的并行运行现在已被禁止，且 `ag_running` 现在会反映异步生成器的实际运行状态。(由 Yury Selivanov 在 bpo-30773 中贡献。)
- 调用 `__iter__` 方法时发生的非预期错误不会再被 `in` 运算符以及 `operator` 的 `contains()`, `indexOf()` 和 `countOf()` 中的 `TypeError` 所掩盖。(由 Serhiy Storchaka 在 bpo-40824 中贡献。)

5 新增模块

5.1 zoneinfo

`zoneinfo` 模块为标准库引入了 IANA 时区数据库。它添加了 `zoneinfo.ZoneInfo`，这是一个基于系统时区数据的实体 `datetime.tzinfo` 实现。

示例:

```
>>> from zoneinfo import ZoneInfo
>>> from datetime import datetime, timedelta

>>> # Daylight saving time
>>> dt = datetime(2020, 10, 31, 12, tzinfo=ZoneInfo("America/Los_Angeles"))
>>> print(dt)
2020-10-31 12:00:00-07:00
>>> dt.tzname()
'PDT'

>>> # Standard time
>>> dt += timedelta(days=7)
>>> print(dt)
2020-11-07 12:00:00-08:00
>>> print(dt.tzname())
PST
```

作为不包含 IANA 数据库的平台的一个回退数据源，还以第一方软件包的形式发布了 `tzdata` 模块 -- 通过 PyPI 发行并由 CPython 核心团队维护。

参见:

PEP 615 -- 在标准库中支持 IANA 时区数据库 PEP 由 Paul Ganssle 撰写并实现

5.2 graphlib

添加了新的 `graphlib` 模块，其中包含 `graphlib.TopologicalSorter` 类来提供图的拓扑排序功能。(由 Pablo Galindo, Tim Peters 和 Larry Hastings 在 bpo-17005 中贡献。)

6 改进的模块

6.1 ast

将 *indent* 选项添加到 `dump()`，这允许它产生多行缩进的输出。(由 Serhiy Storchaka 在 bpo-37995 中贡献。)

添加了 `ast.unparse()` 作为 `ast` 模块中的一个函数，它可被用来反解析 `ast.AST` 对象并产生相应的代码字符串，当它被解析时将会产生一个等价的 `ast.AST` 对象。(由 Pablo Galindo 和 Batuhan Taskaya 在 bpo-38870 中贡献。)

为 AST 节点添加了文档字符串，其中包含 ASDL 签名，可被用来构造对应的节点。(由 Batuhan Taskaya 在 bpo-39638 中贡献。)

6.2 asyncio

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。。)

添加了新的 coroutine `shutdown_default_executor()`，它可为等待 `ThreadPoolExecutor` 结束关闭的默认执行器安排关闭日程操作。此外，`asyncio.run()` 已被更新以使用新的 coroutine。(由 Kyle Stanley 在 bpo-34037 中贡献。)

添加了 `asyncio.PidfdChildWatcher`，这是一个 Linux 专属的子监视器实现，它负责轮询进程的文件描述符。(bpo-38692)

添加了新的 coroutine `asyncio.to_thread()`。它主要被用于在单独线程中运行 IO 密集型函数以避免阻塞事件循环，实质上就相当于是 `run_in_executor()` 的高层级版本，可直接接受关键字参数。(由 Kyle Stanley 和 Yury Selivanov 在 bpo-32309 中贡献。)

当由于超时而取消任务时，`asyncio.wait_for()` 现在将会等待直到也在 *timeout* 值 <= 0 的情况下完成取消。就像 timeout 值为正数时一样。(由 Elvis Pranskevichus 在 bpo-32751 中贡献。)

当附带 `ssl.SSLSocket` 套接字调用不兼容的方法时 `asyncio` 现在会引发 `TyperError`。(由 Ido Michael 在 bpo-37404 中贡献。)

6.3 compileall

为重复的 `.pyc` 文件添加了使用硬软件的可能性: *hardlink_dupes* 形参以及 --hardlink-dupes 命令行选项。(由 Lumír 'Frenzy' Balhar 在 bpo-40495 中贡献。)

新增了一些用于在结果 `.pyc` 文件中操纵路径的选项: *stripdir*, *prependdir*, *limit_sl_dest* 形参以及 -s, -p, -e 命令行选项。并使得为优化等级多次指定选项成为可能。(由 Lumír 'Frenzy' Balhar 在 bpo-38112 中贡献。)

6.4 concurrent.futures

将新的 *cancel_futures* 形参添加到 `concurrent.futures.Executor.shutdown()`，可以取消尚未开始运行的所有挂起的 Future，而不必等待它们完成运行再关闭执行器。(由 Kyle Stanley 在 bpo-39349 中贡献。)

从 `ThreadPoolExecutor` 和 `ProcessPoolExecutor` 中移除了守护线程。这改善与与子解释器的兼容性及它们在关闭进程时的可预测性。(由 Kyle Stanley 在 bpo-39812 中贡献。)

现在 `ProcessPoolExecutor` 中的工作进程仅会在没有可重用的空闲工作进程时按需产生。这优化了启动开销并减少了由空闲工作进程导致的 CPU 时间损失。(由 Kyle Stanley 在 bpo-39207 中贡献。)

6.5 curses

增加了 `curses.get_escdelay()`, `curses.set_escdelay()`, `curses.get_tabsize()` 以及 `curses.set_tabsize()` 函数。(由 Anthony Sottile 在 bpo-38312 中贡献。)

6.6 datetime

`datetime.date` 的 `isocalendar()` 以及 `datetime.datetime` 的 `isocalendar()` 等方法现在将返回 `namedtuple()` 而不是 `tuple`。(由 Dong-hee Na 在 bpo-24416 中贡献。)

6.7 distutils

`upload` 命令现在会创建 SHA2-256 和 Blake2b-256 哈希摘要。它会在禁用 MD5 摘要的平台上跳过 MD5。(由 Christian Heimes 在 bpo-40698 中贡献。)

6.8 fcntl

增加了 `F_OFD_GETLK`, `F_OFD_SETLK` 和 `F_OFD_SETLKW` 等常量。(由 Dong-hee Na 在 bpo-38602 中贡献。)

6.9 ftplib

现在 `FTP` 和 `FTP_TLS` 当它们的构造器所给定的超时参数为零以防止创建非阻塞套接字时会引发 `ValueError`。(由 Dong-hee Na 在 bpo-39259 中贡献。)

6.10 gc

当垃圾回收器进行某些复活对象的收集时（在终结器被执行之后这些对象可以在隔离周期之外被访问），不会阻止对所有仍然无法访问的对象的收集。(由 Pablo Galindo 和 Tim Peters 在 bpo-38379 中贡献。)

增加了一个新的函数 `gc.is_finalized()` 用来检测一个对象是否已被垃圾回收器所终结。(由 Pablo Galindo 在 bpo-39322 中贡献。)

6.11 hashlib

hashlib 模块现在会在可能的情况下使用 OpenSSL 中的 SHA3 哈希和 SHAKE XOF。(由 Christian Heimes 在 bpo-37630 中贡献。)

内置的哈希模块现在可通过 `./configure --without-builtin-hashlib-hashes` 禁用或通过 `./configure --with-builtin-hashlib-hashes=sha3,blake2` 这样的形式有选择地启用以强制使用基于 OpenSSL 的实现。(由 Christian Heimes 在 bpo-40479 中贡献)

6.12 http

添加 HTTP 状态码 `103 EARLY_HINTS`, `418 IM_A_TEAPOT` 和 `425 TOO_EARLY` 到 `http.HTTPStatus`。(由 Dong-hee Na 在 bpo-39509 以及 Ross Rhodes 在 bpo-39507 中贡献。)

6.13 IDLE 与 idlelib

添加了切换光标闪烁停止的选项。(由 Zackery Spytz 在 bpo-4603 中贡献。)

Esc 键现在会关闭 IDLE 补全提示窗口。(由 Johnny Najera 在 bpo-38944 中贡献。)

添加关键字到模块名称补全列表。(由 Terry J. Reedy 在 bpo-37765 中贡献。)

上述修改已被反向移植到 3.8 维护发行版中。

6.14 imaplib

现在 `IMAP4` 和 `IMAP4_SSL` 的构造器具有可选的 *timeout* 形参。并且，现在 `open()` 方法也具有可选的 *timeout* 形参提供同样的修改。`IMAP4_SSL` 和 `IMAP4_stream` 中被重载的方法也应用了这个修改。(由 Dong-hee Na 在 bpo-38615 中贡献。)

增加了 `imaplib.IMAP4.unselect()`。`imaplib.IMAP4.unselect()` 会释放关联到选定邮箱的服务器资源并将服务器返回到已认证状态。此命令会执行与 `imaplib.IMAP4.close()` 相同的动作，区别在于它不会从当前选定邮箱中永久性地移除消息。(由 Dong-hee Na 在 bpo-40375 中贡献。)

6.15 importlib

为提升与 import 语句的一致性，现在 `importlib.util.resolve_name()` 对于无效的相对导入尝试会引发 `ImportError` 而不是 `ValueError`。(由 Ngalim Siregar 在 bpo-37444 中贡献。)

发布不可变模块对象的导入加载器除了发布单独模块以外现在也可以发布不可变包。(由 Dino Viehland 在 bpo-39336 中贡献。)

添加了带有对包数据中子目录支持的 `importlib.resources.files()` 函数，与 `importlib_resources` 1.5 版的反向端口相匹配。(由 Jason R. Coombs 在 bpo-39791 中贡献。)

来自 `importlib_metadata` 1.6.1 版的已更新 `importlib.metadata`。

6.16 inspect

`inspect.BoundArguments.arguments` 已从 `OrderedDict` 改为常规字典。(由 Inada Naoki 在 bpo-36350 和 bpo-39775 中贡献。)

6.17 ipaddress

`ipaddress` 现在支持 IPv6 作用域地址（即带有 `%<scope_id>` 前缀的 IPv6 地址)。

IPv6 作用域地址可使用 `ipaddress.IPv6Address` 来解析。作用域的区 ID 如果存在，可通过 `scope_id` 属性来获取。(由 Oleksandr Pavliuk 在 bpo-34788 中贡献。)

6.18 math

对 `math.gcd()` 函数进行了扩展以处理多个参数。在之前版本中，它只支持两个参数。（由 Serhiy Storchaka 在 bpo-39648 中贡献。)

增加了 `math.lcm()`: 返回指定参数的最小公倍数。(由 Mark Dickinson, Ananthakrishnan 和 Serhiy Storchaka 在 bpo-39479 和 bpo-39648 中贡献。)

增加了 `math.nextafter()`: 返回从 *x* 往 *y* 方向的下一个浮点数值。(由 Victor Stinner 在 bpo-39288 中贡献。)

增加了 `math.ulp()`: 返回一个浮点数的最小有效比特位。(由 Victor Stinner 在 bpo-39310 中贡献。)

6.19 multiprocessing

`multiprocessing.SimpleQueue` 类新增了 `close()` 方法用来显式地关闭队列。(由 Victor Stinner 在 bpo-30966 中贡献。)

6.20 nntplib

现在 `NNTP` 和 `NNTP_SSL` 当它们的构造器所给定的超时参数为零以防止创建非阻塞套接字时会引发 `ValueError`。(由 Dong-hee Na 在 bpo-39259 中贡献。)

6.21 os

增加了 `CLD_KILLED` 和 `CLD_STOPPED` 作为 `si_code`。(由 Dong-hee Na 在 bpo-38493 中贡献。)

对外公开了 Linux 专属的 `os.pidfd_open()` (bpo-38692) 和 `os.P_PIDFD` (bpo-38713) 用于文件描述符的进程管理。

现在 `os.unsetenv()` 函数在 Windows 上也已可用。(由 Victor Stinner 在 bpo-39413 中贡献。)

现在 `os.putenv()` 和 `os.unsetenv()` 函数将总是可用。(由 Victor Stinner 在 bpo-39395 中贡献。)

增加了 `os.waitstatus_to_exitcode()` 函数：将等待状态转换为退出码。（由 Victor Stinner 在 bpo-40094 中贡献。)

6.22 pathlib

增加了 `pathlib.Path.readlink()`，其行为类似于 `os.readlink()`。（由 Girts Folkmanis 在 bpo-30618 中贡献。）

6.23 pdb

在 Windows 上 `Pdb` 现在支持 `~/.pdbrc`。（由 Tim Hopper 和 Dan Lidral-Porter 在 bpo-20523 中贡献。）

6.24 poplib

现在 `POP3` 和 `POP3_SSL` 当它们的构造器所给定的超时参数为零以防止创建非阻塞套接字时会引发 `ValueError`。（由 Dong-hee Na 在 bpo-39259 中贡献。）

6.25 pprint

现在 `pprint` 能美化打印 `types.SimpleNamespace`。（由 Carl Bordum Hansen 在 bpo-37376 中贡献。）

6.26 pydoc

文档字符串的显示现在不仅针对类、函数、方法等，也针对任何具有自己的 `__doc__` 属性的对象。（由 Serhiy Storchaka 在 bpo-40257 中贡献。）

6.27 random

增加了新的 `random.Random.randbytes` 方法：生成随机字节串。（由 Victor Stinner 在 bpo-40286 中贡献。）

6.28 signal

对外公开了 Linux 专属的 `signal.pidfd_send_signal()` 用于向使用文件描述符而非 pid 的进程发送信号。(bpo-38712)

6.29 smtplib

现在 `SMTP` 和 `SMTP_SSL` 当它们的构造器所给定的超时参数为零以防止创建非阻塞套接字时会引发 `ValueError`。（由 Dong-hee Na 在 bpo-39259 中贡献。）

现在 `LMTP` 构造器具有可选的 *timeout* 形参。（由 Dong-hee Na 在 bpo-39329 中贡献。）

6.30 socket

`socket` 模块现在会在 Linux 4.1 或更高版本上导出 `CAN_RAW_JOIN_FILTERS` 常量。（由 Stefan Tatschner 和 Zackery Spytz 在 bpo-25780 中贡献。）

现在 socket 模块会在支持的平台上支持 `CAN_J1939` 协议。（由 Karl Ding 在 bpo-40291 上贡献。）

现在 socket 模块具有 `socket.send_fds()` 和 `socket.recv.fds()` 方法。（由 Joannah Nanjekye, Shinya Okano 和 Victor Stinner 在 bpo-28724 中贡献。）

6.31 time

在 AIX 上，现在 `thread_time()` 是使用具有纳秒级精度的 `thread_cputime()` 实现，而不再是只有 10 毫秒精度的 `clock_gettime(CLOCK_THREAD_CPUTIME_ID)`。(由 Batuhan Taskaya 在 bpo-40192 中贡献。)

6.32 sys

增加了新的 `sys.platlibdir` 属性：平台专属库目录的名称。它被用于构建标准库的路径以及已安装扩展模块的路径。它在大多数平台上等于 `"lib"`。在 Fedora 和 SuSE 上，它等于 64 位平台上的 `"lib64"`。(由 Jan Matějek, Matěj Cepl, Charalampos Stratakis 和 Victor Stinner 在 bpo-1294959 中贡献。)

之前的版本中，`sys.stderr` 在非交互模式时是带块缓冲的。现在 `stderr` 默认总是为行缓冲的。(由 Jendrik Seipp 在 bpo-13601 中贡献。)

6.33 tracemalloc

增加了 `tracemalloc.reset_peak()` 用于将跟踪的内存块峰值大小设为当前大小，以测量特定代码段的峰值。(由 Huon Wilson 在 bpo-40630 中贡献。)

6.34 typing

PEP 593 引入了一种 `typing.Annotated` 类型以使用上下文专属的元数据来装饰现有类型，并将新的 `include_extras` 形参添加到 `typing.get_type_hints()` 以在运行时访问元数据。(由 Till Varoquaux 和 Konstantin Kashin 贡献。)

6.35 unicodedata

Unicode 数据库已更新到 13.0.0 版。(bpo-39926)。

6.36 venv

由 `venv` 所提供的激活脚本现在总是会使用 `__VENV_PROMPT__` 设置的值来一致地指明它们的自定义提示符。在之前版本中某些脚本会无条件地使用 `__VENV_PROMPT__`，而另一些脚本只在其恰好被设置时（这是默认情况）才会使用，还有的脚本会改用 `__VENV_NAME__`。(由 Brett Cannon 在 bpo-37663 中贡献。)

6.37 xml

当把 `xml.etree.ElementTree` 序列化为 XML 文件时属性内部的空白字符现在将被保留。不同的行结束符不会再被正规化为"n"。这是对于如何解读 XML 规范 2.11 节的相关讨论的最终结果。(由 Mefistotelis 在 bpo-39011 中贡献。)

7 性能优化

- 优化了在推导式中为临时变量赋值的惯用方式。现在推导式中的 `for y in [expr]` 会与简单赋值语句 `y = expr` 一样快速。例如:

  ```
  sums = [s for s in [0] for x in data for s in [s + x]]
  ```

  不同于 `:=` 运算符，这个惯用方式不会使变量泄露到外部作用域中。

  (由 Serhiy Storchaka 在 bpo-32856 中贡献。)

- 优化了多线程应用中的信号处理。如果一个线程不是获得信号的主线程，字节码求值循环不会在每条字节码指令上被打断以检查无法被处理的挂起信号。只有主解释器的主线程能够处理信号。

  在之前版本中，字节码求值循环会在每条指令上被打断直到主线程处理了信号。(由 Victor Stinner 在 bpo-40010 上贡献。)

- 在 FreeBSD 上使用 `closefrom()` 优化了 `subprocess` 模块。(由 Ed Maste, Conrad Meyer, Kyle Evans, Kubilay Kocak 和 Victor Stinner 在 bpo-38061 中贡献。)

- `PyLong_FromDouble()` 对于匹配 `long` 的值执行速度现在加快了 1.87 倍。(由 Sergey Fedoseev d bpo-37986 中贡献。)

- 多个 Python 内置类型 (`range`, `tuple`, `set`, `frozenset`, `list`, `dict`) 现在通过使用 **PEP 590** 向量调用协议得到加速。(由 Dong-hee Na, Mark Shannon, Jeroen Demeyer 和 Petr Viktorin 在 bpo-37207 中贡献。)

- 当另一集合远大于基础集合的情况下优化了 `difference_update()` 的性能。(由 Evgeny Kapun 提议，由 Michele Orrù 在 bpo-8425 中贡献代码。)

- Python 的小对象分配器 (`obmalloc.c`) 现在允许（至多）一个空位可用于立即重用，而不必将其返回给 OS。这可以防止简单循环中的多余消耗，在每次迭代中可以创建和销毁全新的空位。(由 Tim Peters 在 bpo-37257 中贡献。)

- 浮点数运算中的 floor division 现在会有更好的性能。并且此运算的 `ZeroDivisionError` 的消息也已更新。(由 Dong-hee Na 在 bpo-39434 中贡献。)

- 使用 UTF-8 和 ascii 编解码器解码短 ASCII 字符串现在会加快大约 15%。(由 Inada Naoki 在 bpo-37348 中贡献。)

以下是对从 Python 3.4 到 Python 3.9 的提升提升情况的总结:

```
Python version                       3.4     3.5     3.6     3.7     3.8     3.9
--------------                       ---     ---     ---     ---     ---     ---

Variable and attribute read access:
    read_local                       7.1     7.1     5.4     5.1     3.9     3.9
    read_nonlocal                    7.1     8.1     5.8     5.4     4.4     4.5
    read_global                     15.5    19.0    14.3    13.6     7.6     7.8
    read_builtin                    21.1    21.6    18.5    19.0     7.5     7.8
    read_classvar_from_class        25.6    26.5    20.7    19.5    18.4    17.9
    read_classvar_from_instance     22.8    23.5    18.8    17.1    16.4    16.9
    read_instancevar                32.4    33.1    28.0    26.3    25.4    25.3
    read_instancevar_slots          27.8    31.3    20.8    20.8    20.2    20.5
    read_namedtuple                 73.8    57.5    45.0    46.8    18.4    18.7
    read_boundmethod                37.6    37.9    29.6    26.9    27.7    41.1

Variable and attribute write access:
    write_local                      8.7     9.3     5.5     5.3     4.3     4.3
    write_nonlocal                  10.5    11.1     5.6     5.5     4.7     4.8
    write_global                    19.7    21.2    18.0    18.0    15.8    16.7
    write_classvar                  92.9    96.0   104.6   102.1    39.2    39.8
    write_instancevar               44.6    45.8    40.0    38.9    35.5    37.4
    write_instancevar_slots         35.6    36.1    27.3    26.6    25.7    25.8
```

(下页继续)

(续上页)

```
Data structure read access:
    read_list                   24.2    24.5    20.8    20.8    19.0    19.5
    read_deque                  24.7    25.5    20.2    20.6    19.8    20.2
    read_dict                   24.3    25.7    22.3    23.0    21.0    22.4
    read_strdict                22.6    24.3    19.5    21.2    18.9    21.5

Data structure write access:
    write_list                  27.1    28.5    22.5    21.6    20.0    20.0
    write_deque                 28.7    30.1    22.7    21.8    23.5    21.7
    write_dict                  31.4    33.3    29.3    29.2    24.7    25.4
    write_strdict               28.4    29.9    27.5    25.2    23.1    24.5

Stack (or queue) operations:
    list_append_pop             93.4   112.7    75.4    74.2    50.8    50.6
    deque_append_pop            43.5    57.0    49.4    49.2    42.5    44.2
    deque_append_popleft        43.7    57.3    49.7    49.7    42.8    46.4

Timing loop:
    loop_overhead                0.5     0.6     0.4     0.3     0.3     0.3
```

以上结果是由以下变量访问基准测试脚本所生成的: `Tools/scripts/var_access_benchmark.py`。该基准测试脚本以纳秒为单位显示时间。基准测试数据是在一块 Intel® Core™ i7-4960HQ 处理器 运行从 python.org 获取的 macOS 64 位编译版本所得到的。

8 弃用

- distutils 的 `bdist_msi` 命令现在已被弃用，请改用 `bdist_wheel` (wheel 包)。(由 Hugo van Kemenade 在 bpo-39586 中贡献。)
- 目前 `math.factorial()` 接受具有非负整数值的 `float` 实例 (如 `5.0`)。对于非整数和负浮点数它会引发 `ValueError`。此行为现在已被弃用。在未来的 Python 版本中对所有浮点数都将引发 `TypeError`。(由 Serhiy Storchaka 在 bpo-37315 中贡献。)
- `parser` 和 `symbol` 模块已被弃用并将在未来的 Python 版本中移除。对于大多数用例，用户都可以使用 `ast` 模块来控制抽象语法树 (AST) 的生成和编译阶段。
- 公有 C API 函数 `PyParser_SimpleParseStringFlags()`, `PyParser_SimpleParseStringFlagsFilename()`, `PyParser_SimpleParseFileFlags()` 和 `PyNode_Compile()` 已被弃用并将在 Python 3.10 版与旧解析器一起被移除。
- 在布尔运算中使用 `NotImplemented` 已被弃用，因为它几乎必定是不正确的富比较运算符实现的结果。它将在未来的 Python 版本中引发 `TypeError`。(由 Josh Rosenberg 在 bpo-35712 中贡献。)
- `random` 模块目前接受任何可哈希类型作为可能的种子值。不幸的是，某些这样的类型并不保证具有确定性的哈希值。在 Python 3.9 之后，该模块将限定其种子值为 `None`, `int`, `float`, `str`, `bytes` 以及 `bytearray`。
- 打开 `GzipFile` 文件用于写入而不指定 *mode* 参数的特性已被弃用。在未来的 Python 版本中将总是默认打开用于读取。在打开文件用于写入时请指定 *mode* 参数以静默相关警告信息。(由 Serhiy Storchaka 在 bpo-28286 中贡献。)
- 弃用了 `_tkinter.TkappType` 的 `split()` 方法而改用 `splitlist()` 方法，此方法具有更稳定且可预测的行为。(由 Serhiy Storchaka 在 bpo-38371 中贡献。)
- 将协程对象显式传递给 `asyncio.wait()` 的做法已被弃用并且将在 3.11 版中被移除。(由 Yury Selivanov 和 Kyle Stanley 在 bpo-34790 中贡献。)
- binhex4 和 hexbin4 标准现已被弃用。`binhex` 模块和下列 `binascii` 函数现已被弃用:
  - `b2a_hqx()`, `a2b_hqx()`
  - `rlecode_hqx()`, `rledecode_hqx()`

（由 Victor Stinner 在 bpo-39353 中贡献。）

- `ast` 类 `slice`, `Index` 和 `ExtSlice` 被视为已弃用并将在未来的 Python 版本中被移除。应当使用 `value` 本身而不再是 `Index(value)`。应当使用 `Tuple(slices, Load())` 而不再是 `ExtSlice(slices)`。（由 Serhiy Storchaka 在 bpo-34822 中贡献。）
- `ast` 类 `Suite`, `Param`, `AugLoad` 和 `AugStore` 被视为已弃用并将在未来的 Python 版本中被移除。它们不会被解析器所生成且不会被 Python 3 中的代码生成器所接受。（由 Batuhan Taskaya 在 bpo-39639 和 bpo-39969 中以及 Serhiy Storchaka 在 bpo-39988 中贡献。）
- `PyEval_InitThreads()` 和 `PyEval_ThreadsInitialized()` 函数现已被弃用并将在 Python 3.11 中被移除。调用 `PyEval_InitThreads()` 现在没有任何效果。自 Python 3.7 起 GIL 会由 `Py_Initialize()` 初始化。（由 Victor Stinner 在 bpo-39877 中贡献。）
- 传入 `None` 作为 `shlex.split()` 函数的第一个参数的做法已被弃用。（由 Zackery Spytz 在 bpo-33262 中贡献。）
- `smtpd.MailmanProxy()` 现在已被弃用，因为它在没有外部模块 `mailman` 的情况下无法使用。（由 Samuel Colvin 在 bpo-35800 中贡献。）
- 现在 `lib2to3` 模块将发出 `PendingDeprecationWarning`。Python 3.9 已切换到 PEG 解析器 (参见 **PEP 617**)，Python 3.10 可以会包含 lib2to3 的 LL(1) 解析器所不能解析的新语法。`lib2to3` 模块可能会在未来的 Python 版本中被移出标准库。请考虑使用第三方替换例如 LibCST 或 parso。（由 Carl Meyer 在 bpo-40360 中贡献。）
- `random.shuffle()` 的 *random* 形参已被弃用。（由 Raymond Hettinger 在 bpo-40465 中贡献。）

9 移除

- `unittest.mock.__version__` 上的错误版本已经被移除。
- `nntplib.NNTP`: `xpath()` 和 `xgtitle()` 方法已被移除。这些方法自 Python 3.3 起已被弃用。一般来说，这些扩展都不再为 NNTP 服务管理员所支持或启用。对于 `xgtitle()`，请改用 `nntplib.NNTP.descriptions()` 或 `nntplib.NNTP.description()`。（由 Dong-hee Na 在 bpo-39366 中贡献。）
- `array.array`: `tostring()` 和 `fromstring()` 方法已被移除。它们分别是 `tobytes()` 和 `frombytes()` 的别名，自 Python 3.2 起已被弃用。（由 Victor Stinner 在 bpo-38916 中贡献。）
- 未写入文档的 `sys.callstats()` 函数已被移除。自 Python 3.7 起它就已被弃用并且总是会返回 `None`。它需要一个特殊的构建选项 `CALL_PROFILE` 而该选项在 Python 3.7 中已被移除。（由 Victor Stinner 在 bpo-37414 中贡献。）
- `sys.getcheckinterval()` 和 `sys.setcheckinterval()` 函数已被移除。它们自 Python 3.2 起已被弃用。请改用 `sys.getswitchinterval()` 和 `sys.setswitchinterval()`。（由 Victor Stinner 在 bpo-37392 中贡献。）
- C 函数 `PyImport_Cleanup()` 已被移除。它原本的文档为: " 清空模块表。仅限内部使用。"（由 Victor Stinner 在 bpo-36710 中贡献。）
- `_dummy_thread` 和 `dummy_threading` 模块已被移除。这些模块自 Python 3.7 起已被弃用，它们需要线程支持。（由 Victor Stinner 在 bpo-37312 中贡献。）
- `aifc.open()` 的别名 `aifc.openfp()`，`sunau.open()` 的别名 `sunau.openfp()`，以及 `wave.open()` 的别名 `wave.openfp()` 已被移除。它们自 Python 3.7 起已被弃用。（由 Victor Stinner 在 bpo-37320 中贡献。）
- `threading.Thread` 的 `isAlive()` 方法已被移除。它自 Python 3.8 起已被弃用。请改用 `is_alive()`。（由 Dong-hee Na 在 bpo-37804 中贡献。）
- `ElementTree` 模块中 `ElementTree` 和 `Element` 等类的 `getchildren()` 和 `getiterator()` 方法已被移除。它们在 Python 3.2 中已被弃用。请使用 `iter(x)` 或 `list(x)` 替代 `x.getchildren()` 并用 `x.iter()` 或 `list(x.iter())` 替代 `x.getiterator()`。（由 Serhiy Storchaka 在 bpo-36543 中贡献。）

- 旧的 `plistlib` API 已被移除，它自 Python 3.4 起已被弃用。请使用 `load()`, `loads()`, `dump()` 和 `dumps()` 等函数。此外，*use_builtin_types* 形参已被移除而总是会使用 `bytes` 对象。(由 Jon Janzen 在 bpo-36409 中贡献。)
- C 函数 `PyGen_NeedsFinalizing` 已被移除。它未被写入文档、未经测试，且自 **PEP 442** 实现之后未在 CPython 的任何地方被使用。由 Joannah Nanjekye 提供补丁。(由 Joannah Nanjekye 在 bpo-15088 中贡献。)
- 自 Python 3.1 起被弃用的别名 `base64.encodestring()` 和 `base64.decodestring()` 已被移除：请改用 `base64.encodebytes()` 和 `base64.decodebytes()`。(由 Victor Stinner 在 bpo-39351 中贡献。)
- `fractions.gcd()` 函数已被移除，它自 Python 3.5 起被弃用 (bpo-22486)：请改用 `math.gcd()`。(由 Victor Stinner 在 bpo-39350 中贡献。)
- `bz2.BZ2File` 的 *buffering* 形参已被移除。它自 Python 3.0 起即被忽略，使用它将会引发 `DeprecationWarning`。请传入一个打开文件对象来控制文件的打开方式。(由 Victor Stinner 在 bpo-39357 中贡献。)
- `json.loads()` 的 *encoding* 形参已被移除。它在 Python 3.1 中已被弃用和忽略；自 Python 3.8 起使用它将会引发 `DeprecationWarning`。(由 Inada Naoki 在 bpo-39377 中贡献。)
- `with (await asyncio.lock):` 和 `with (yield from asyncio.lock):` 语句已不再受支持，请改用 `async with lock`。`asyncio.Condition` 和 `asyncio.Semaphore` 也同样如此。(由 Andrew Svetlov 在 bpo-34793 中贡献。)
- `sys.getcounts()` 函数，`-X showalloccount` 命令行选项以及 C 结构体 `PyConfig` 的 `show_alloc_count` 字段已被移除。它们需要使用定义了 `COUNT_ALLOCS` 宏的特殊 Python 编译版本。(由 Victor Stinner 在 bpo-39489 中贡献。)
- `typing.NamedTuple` 类的 `_field_types` 属性已被移除。它自 Python 3.8 起已被弃用。请改用 `__annotations__` 属性。(由 Serhiy Storchaka 在 bpo-40182 中贡献。)
- `symtable.SymbolTable.has_exec()` 方法已被移除。它自 2006 年起已被弃用，当被调用时仅会返回 `False`。(由 Batuhan Taskaya 在 bpo-40208 中贡献。)
- `asyncio.Task.current_task()` 和 `asyncio.Task.all_tasks()` 已被移除。它们自 Python 3.7 起已被弃用，你可以改用 `asyncio.current_task()` 和 `asyncio.all_tasks()`。(由 Rémi Lapeyre 在 bpo-40967 中贡献。)
- `html.parser.HTMLParser` 类的 `unescape()` 方法已被移除 (它自 Python 3.4 起已被弃用)。应当使用 `html.unescape()` 来将字符引用转换为对应的 unicode 字符。

10 移植到 Python 3.9

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.

10.1 Python API 的变化

- `__import__()` 和 `importlib.util.resolve_name()` 现在会引发 `ImportError` 取代之前所引发的 `ValueError`。捕获特定异常类型并同时支持 Python 3.9 和更早版本的调用者将需要使用 `except (ImportError, ValueError):` 来同时捕获两者。
- `venv` 激活脚本不再将 `__VENV_PROMPT__` 被设为 `""` 的情况作为特例处理。
- `select.epoll.unregister()` 方法不会再忽略 `EBADF` 错误。(由 Victor Stinner 在 bpo-39239 中贡献。)
- `bz2.BZ2File` 的 *compresslevel* 形参已成为仅限关键字形参，因为 *buffering* 形参已被移除。(由 Victor Stinner 在 bpo-39357 中贡献。)

- 简化了 AST 的抽取操作。简单索引将以它们的值来代表，扩展切片将以元组形式来代表。`Index(value)` 将返回 `value` 本身，`ExtSlice(slices)` 将返回 `Tuple(slices, Load())`。(由 Serhiy Storchaka 在 bpo-34822 中贡献。)
- 当使用了 `-E` 或 `-I` 命令行参数时 `importlib` 模块现在会忽略 `PYTHONCASEOK` 环境变量。
- *encoding* 形参已作为仅限关键字形参被添加到 `ftplib.FTP` 和 `ftplib.FTP_TLS` 类，并且默认编码格式由 Latin-1 改为 UTF-8 以遵循 **RFC 2640**。
- `asyncio.loop.shutdown_default_executor()` 已被添加到 `AbstractEventLoop`，这意味着继承自它的替代事件循环应当定义此方法。(由 Kyle Stanley 在 bpo-34037 中贡献。)
- 更新了 `__future__` 模块中未来特性旗标的常量值以防止与编译器旗标相冲突。在之前版本中 `PyCF_ALLOW_TOP_LEVEL_AWAIT` 会与 `CO_FUTURE_DIVISION` 发生冲突。(由 Batuhan Taskaya 在 bpo-39562 中贡献。)
- `array('u')` 现在使用 `wchar_t` 作为 C 类型而不是 `Py_UNICODE`。这个改变不会影响其行为，因为自 Python 3.3 起 `Py_UNICODE` 是 `wchar_t` 的别名。(由 Inada Naoki 在 bpo-34538 中贡献。)
- 现在 `logging.getLogger()` API 当传入名称 `'root'` 时将返回根日志记录器，而在之前它则返回一个名为 `'root'` 的非根日志记录器。这可能会影响到用户代码明确希望使用一个名为 `'root'` 的非根日志记录器，或在某个名为 `'root.py'` 的最高层级模块中使用 `logging.getLogger(__name__)` 来实例化日志记录器的情况。(由 Vinay Sajip 在 bpo-37742 中贡献。)
- 现在 `PurePath` 的拆分处理当传入 `str` 或 `PurePath` 的实例以外的对象时会返回 `NotImplemented` 而不是引发 `TypeError`。这将允许创建不继承自上述类型的兼容类。(由 Roger Aiudi 在 bpo-34775 中贡献。)

10.2 C API 的变化

- 堆分配类型的实例（例如使用 `PyType_FromSpec()` 和类似 API 创建的实例）自 Python 3.8 起会带有一个对其类型对象的引用。正如 Python 3.8 的"C API 中的改变" 部分所述，对于大部分情况来说，这应当不会有任何副作用，但对于具有自定义 `tp_traverse` 函数的类型来说，则要确保所有堆分配类型的自定义 `tp_traverse` 函数可访问对象的类型。

  示例:

```
int
foo_traverse(foo_struct *self, visitproc visit, void *arg) {
// Rest of the traverse function
#if PY_VERSION_HEX >= 0x03090000
    // This was not needed before Python 3.9 (Python issue 35810 and␣
↪40217)
    Py_VISIT(Py_TYPE(self));
#endif
}
```

  如果你的遍历函数委托给了基类（或其他类）的 `tp_traverse`，则要确保 `Py_TYPE(self)` 只被访问一次。请注意应当只有堆类型可访问 `tp_traverse` 中的类型。

  举例来说，如果你的 `tp_traverse` 函数包括以下内容:

```
base->tp_traverse(self, visit, arg)
```

  则要添加:

```
#if PY_VERSION_HEX >= 0x03090000
    // This was not needed before Python 3.9 (Python issue 35810 and␣
↪40217)
    if (base->tp_flags & Py_TPFLAGS_HEAPTYPE) {
        // a heap type's tp_traverse already visited Py_TYPE(self)
    } else {
        Py_VISIT(Py_TYPE(self));
```

(下页继续)

(续上页)

```
    }
#else
```

(参阅 bpo-35810 和 bpo-40217 了解更多信息。)

- PyEval_CallObject, PyEval_CallFunction, PyEval_CallMethod 和 PyEval_CallObjectWithKeywords 函数已被弃用。请改用 PyObject_Call() 及其变化形式。(详情参见 bpo-29548。)

10.3 CPython 字节码的改变

- 添加了 LOAD_ASSERTION_ERROR 操作码用于处理 assert 语句。在之前的版本中，如果 AssertionError 异常被屏蔽则 assert 语句将不能正常运作。(由 Zackery Spytz 在 bpo-34880 中贡献。)
- COMPARE_OP 操作码已被拆分为四个单独指令:
  - COMPARE_OP 用于富比较
  - IS_OP 用于'is' 和'is not' 检测
  - CONTAINS_OP 用于'in' 和'not in' 检测
  - JUMP_IF_NOT_EXC_MATCH 用于检查'try-except' 语句中的异常。

  (由 Mark Shannon 在 bpo-39156 中贡献。)

11 构建的改变

- 将 --with-platlibdir 选项添加到 configure 脚本：平台专属库目录的名称，保存在新的 sys.platlibdir 属性中。请参阅 sys.platlibdir 属性了解详情。(由 Jan Matějek, Matěj Cepl, Charalampos Stratakis 和 Victor Stinner 在 bpo-1294959 中贡献。)
- COUNT_ALLOCS 特殊构建宏已被移除。(由 Victor Stinner 在 bpo-39489 中贡献。)
- 在非 Windows 平台上，现在需要用 setenv() 和 unsetenv() 函数来构建 Python。(由 Victor Stinner 在 bpo-39395 中贡献。)
- 在非 Windows 平台上，创建 bdist_wininst 安装器现在已不受官方支持。(详情参见 bpo-10945。)
- 当在 macOS 上用源代码编译 Python 时，_tkinter 现在会链接到非系统的 Tcl 和 Tk 框架，如果它们被安装到 /Library/Frameworks 的话，就如在较旧的 macOS 发行版上的情况一样。如果通过使用 --enable-universalsdk= 或 -isysroot 明确地配置了 macOS SDK，则只会搜索 SDK 本身。默认行为仍然可以通过 --with-tcltk-includes 和 --with-tcltk-libs 来覆盖。(由 Ned Deily 在 bpo-34956 中贡献。)
- Python 现在可以针对 Windows 10 ARM64 进行编译。(由 Steve Dower 在 bpo-33125 中贡献。)
- 现在当使用 --pgo 时一些单独的测试会被跳过。这些测试显著增加了 PGO 任务的时间并且可能无助于提升最终可执行文件的优化程度。这样能使任务加速大约 15 倍。运行完整的单元测试是很慢的。这个改变可能导致优化程序稍差的构建，因为将被执行的代码分支不够多。如果你愿意等待更缓慢的构建，则可以使用 ./configure [..] PROFILE_TASK="-m test --pgo-extended" 来恢复旧版本的行为。我们不保证哪个 PGO 任务集能产生更快的构建。关心此问题的用户应当自行运行相关基准测试，因为结果可能取决于具体环境、工作负载以及编译工具链。(请参阅 bpo-36044 和 bpo-37707 了解详情。)

12 C API 的改变

12.1 新的特性

- **PEP 573**: 添加了 `PyType_FromModuleAndSpec()` 用于通过类来关联一个模块; `PyType_GetModule()` 和 `PyType_GetModuleState()` 用于获取模块及其状态; 以及 `PyCMethod` 和 `METH_METHOD` 用于允许一个方法访问其定义所在的类。(由 Marcel Plch 和 Petr Viktorin 在 bpo-38787 中贡献。)
- 增加了 `PyFrame_GetCode()` 函数: 获取帧代码。增加了 `PyFrame_GetBack()` 函数: 获取帧的下一个外部帧。(由 Victor Stinner 在 bpo-40421 中贡献。)
- 将 `PyFrame_GetLineNumber()` 添加到受限的 C API。(由 Victor Stinner 在 bpo-40421 中贡献。)
- 增加了 `PyThreadState_GetInterpreter()` 和 `PyInterpreterState_Get()` 函数用于获取解释器。增加了 `PyThreadState_GetFrame()` 函数用于获取 Python 线程状态的当前帧。增加了 `PyThreadState_GetID()` 函数: 获取 Python 线程状态的唯一标识符。(由 Victor Stinner 在 bpo-39947 中贡献。)
- 将新的公有 `PyObject_CallNoArgs()` 函数添加到 C API，该函数可不带任何参数调用一个 Python 可调用对象。它是不带参数调用 Python 可调用对象最有效率的方式。(由 Victor Stinner 在 bpo-37194 中贡献。)
- 受限 C API 中的改变（如果定义了 `Py_LIMITED_API` 宏）:
  - 提供 `Py_EnterRecursiveCall()` 和 `Py_LeaveRecursiveCall()` 作为常规函数用于受限 API。在之前版本中是使用宏定义，但这些宏不能与无法访问 `PyThreadState.recursion_depth` 字段的受限 C API 一同编译（该结构体在受限 C API 中是不透明的）。
  - `PyObject_INIT()` 和 `PyObject_INIT_VAR()` 已成为常规“不透明”函数以隐藏实现细节。

  (由 Victor Stinner 在 bpo-38644 和 bpo-39542 中贡献。)
- 增加了 `PyModule_AddType()` 函数以协助将类型加入到模块中。(由 Dong-hee Na 在 bpo-40024 中贡献。)
- 将 `PyObject_GC_IsTracked()` 和 `PyObject_GC_IsFinalized()` 函数添加到公有 API 以允许分别查询 Python 对象当前是正在被追踪还是已经被垃圾回收器所终结。(由 Pablo Galindo Salgado 在 bpo-40241 中贡献。)
- 增加了 `_PyObject_FunctionStr()` 以获取函数类对象的用户友好的表示形式。(由 Jeroen Demeyer 在 bpo-37645 中修正。)
- 增加了 `PyObject_CallOneArg()` 用于调用具有一个位置参数的对象（由 Jeroen Demeyer 在 bpo-37483 中修正。)

12.2 移植到 Python 3.9

- `PyInterpreterState.eval_frame` (**PEP 523**) 现在需要有新的强制性形参 *tstate* (`PyThreadState*`)。(由 Victor Stinner 在 bpo-38500 中贡献。)
- 扩展模块: `PyModuleDef` 的 `m_traverse`, `m_clear` 和 `m_free` 等函数在模块状态被请求但尚未被分配时将不会再被调用。这种情况出现在模块被创建之后且模块被执行 (`Py_mod_exec` 函数) 之前的时刻。更准确地说，这些函数在 `m_size` 大于 0 并且模块状态（即 `PyModule_GetState()` 的返回值）为 `NULL` 时将不会被调用。

  没有模块状态的扩展模块 (`m_size <= 0`) 不会受到影响。
- 现在如果 `Py_AddPendingCall()` 是在子解释器内部被调用，该函数会被排入子解释器的调用日程，而不是由主解释器调用。每个子解释器现在都拥有它们自己的调用日程列表。(由 Victor Stinner 在 bpo-39984 中贡献。)

- 当 -E 选项被使用 (如果 PyConfig.use_environment 设为 0) 时将不再使用 Windows 注册表来初始化 sys.path。这会影响在 Windows 上嵌入 Python 的操作。(由 Zackery Spytz 在 bpo-8901 中贡献。)
- 全局变量 PyStructSequence_UnnamedField 现在为常量并且指向一个字符串常量。(由 Serhiy Storchaka 在 bpo-38650 中贡献。)
- 现在 PyGC_Head 结构是不透明的。它只在内部 C API (pycore_gc.h) 中定义。(由 Victor Stinner 在 bpo-40241 中贡献。)
- Py_UNICODE_COPY, Py_UNICODE_FILL, PyUnicode_WSTR_LENGTH, PyUnicode_FromUnicode(), PyUnicode_AsUnicode(), _PyUnicode_AsUnicode 以及 PyUnicode_AsUnicodeAndSize() 在 C 中被标记为已弃用。它们自 Python 3.3 起就已被 **PEP 393** 弃用。(由 Inada Naoki 在 bpo-36346 中贡献。)
- Py_FatalError() 函数会被一个自动记录当前函数名称的宏所替代，除非已定义了 Py_LIMITED_API 宏。(由 Victor Stinner 在 bpo-39882 中贡献。)
- vectorcall 协议现在要求调用者只传入字符串作为键名。(请参阅 bpo-37540 了解详情。)
- 多个宏和函数的实现细节现在已被隐藏:
  - PyObject_IS_GC() 宏已被转换为函数。
  - PyObject_NEW() 宏已成为 PyObject_New() 宏的别名，而 PyObject_NEW_VAR() 宏已成为 PyObject_NewVar() 宏的别名。它们将不再直接访问 PyTypeObject.tp_basicsize 成员。
  - PyType_HasFeature() 现在总是会调用 PyType_GetFlags()。在之前版本中，当受限的 C API 未被使用时它会直接访问 PyTypeObject.tp_flags 成员。
  - PyObject_GET_WEAKREFS_LISTPTR() 宏已被转换为函数：该宏会直接访问 PyTypeObject.tp_weaklistoffset 成员。
  - PyObject_CheckBuffer() 宏已被转换为函数：该宏会直接访问 PyTypeObject.tp_as_buffer 成员。
  - 现在 PyIndex_Check() 总是被声明为不透明函数以隐藏实现细节；PyIndex_Check() 宏已被移除。该宏会直接访问 PyTypeObject.tp_as_number 成员。

  (详情请参阅 bpo-40170。)

12.3 移除

- pyfpe.h 的 PyFPE_START_PROTECT() 和 PyFPE_END_PROTECT() 宏已从受限的 C API 中被排除。(由 Victor Stinner 在 bpo-38835 中贡献。)
- PyTypeObject 的 tp_print 空位已被移除。它在 Python 2.7 及之前的版本中被用来将对象打印到文件。自 Python 3.0 起，它已被忽略并且不再使用。(由 Jeroen Demeyer 在 bpo-36974 中贡献。)
- 受限 C API 中的改变 (如果定义了 Py_LIMITED_API 宏)：
  - 以下函数已从受限 C API 中排除:
    * PyThreadState_DeleteCurrent() (由 Joannah Nanjekye 在 bpo-37878 中贡献。)
    * _Py_CheckRecursionLimit
    * _Py_NewReference()
    * _Py_ForgetReference()
    * _PyTraceMalloc_NewReference()
    * _Py_GetRefTotal()
    * 在受限 C API 中从未使用的垃圾箱机制。
    * PyTrash_UNWIND_LEVEL

    * `Py_TRASHCAN_BEGIN_CONDITION`
    * `Py_TRASHCAN_BEGIN`
    * `Py_TRASHCAN_END`
    * `Py_TRASHCAN_SAFE_BEGIN`
    * `Py_TRASHCAN_SAFE_END`
  - 已将下列函数和定义移至内部 C API:
    * `_PyDebug_PrintTotalRefs()`
    * `_Py_PrintReferences()`
    * `_Py_PrintReferenceAddresses()`
    * `_Py_tracemalloc_config`
    * `_Py_AddToAllObjects()` (`Py_TRACE_REFS` 构建专属)

  (由 Victor Stinner 在 bpo-38644 和 bpo-39542 中贡献。)

- 移除了 `_PyRuntime.getframe` 钩子并移除了 `_PyThreadState_GetFrame` 宏，该宏是 `_PyRuntime.getframe` 的一个别名。它们仅由内部 C API 对外公开。同样地移除了 `PyThreadFrameGetter` 类型。(由 Victor Stinner 在 bpo-39946 中贡献。)
- 从 C API 移除了下列函数。请显式地调用 `PyGC_Collect()` 来清空所有自由列表。(由 Inada Naoki 和 Victor Stinner 在 bpo-37340, bpo-38896 和 bpo-40428 中贡献。)
  - `PyAsyncGen_ClearFreeLists()`
  - `PyContext_ClearFreeList()`
  - `PyDict_ClearFreeList()`
  - `PyFloat_ClearFreeList()`
  - `PyFrame_ClearFreeList()`
  - `PyList_ClearFreeList()`
  - `PyMethod_ClearFreeList()` 和 `PyCFunction_ClearFreeList()`: 绑定方法对象的自由列表已被移除。
  - `PySet_ClearFreeList()`: 集合自由列表已在 Python 3.4 中被移除。
  - `PyTuple_ClearFreeList()`
  - `PyUnicode_ClearFreeList()`: Unicode 自由列表已在 Python 3.3 中被移除。
- 移除了 `_PyUnicode_ClearStaticStrings()` 函数。(由 Victor Stinner 在 bpo-39465 中贡献。)
- 移除了 `Py_UNICODE_MATCH`。它已被 **PEP 393** 所弃用，并自 Python 3.3 起不再可用。可以改用 `PyUnicode_Tailmatch()` 函数。(由 Inada Naoki 在 bpo-36346 中贡献。)
- 清除了已定义但未实现的接口的头文件。被移除了公共 API 符号有: `_PyBytes_InsertThousandsGroupingLocale`, `_PyBytes_InsertThousandsGrouping`, `_Py_InitializeFromArgs`, `_Py_InitializeFromWideArgs`, `_PyFloat_Repr`, `_PyFloat_Digits`, `_PyFloat_DigitsInit`, `PyFrame_ExtendStack`, `_PyAIterWrapper_Type`, `PyNullImporter_Type`, `PyCmpWrapper_Type`, `PySortWrapper_Type`, `PyNoArgsFunction`。 (由 Pablo Galindo Salgado 在 bpo-39372 中贡献。)

13 Notable changes in Python 3.9.1

13.1 typing

The behavior of `typing.Literal` was changed to conform with **PEP 586** and to match the behavior of static type checkers specified in the PEP.

1. `Literal` now de-duplicates parameters.
2. Equality comparisons between `Literal` objects are now order independent.
3. `Literal` comparisons now respect types. For example, `Literal[0] == Literal[False]` previously evaluated to `True`. It is now `False`. To support this change, the internally used type cache now supports differentiating types.
4. `Literal` objects will now raise a `TypeError` exception during equality comparisons if one of their parameters are not immutable. Note that declaring `Literal` with mutable parameters will not throw an error:

```
>>> from typing import Literal
>>> Literal[{0}]
>>> Literal[{0}] == Literal[{False}]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'set'
```

(Contributed by Yurii Karabas in bpo-42345.)

13.2 macOS 11.0 (Big Sur) and Apple Silicon Mac support

As of 3.9.1, Python now fully supports building and running on macOS 11.0 (Big Sur) and on Apple Silicon Macs (based on the `ARM64` architecture). A new universal build variant, `universal2`, is now available to natively support both `ARM64` and `Intel 64` in one set of executables. Binaries can also now be built on current versions of macOS to be deployed on a range of older macOS versions (tested to 10.9) while making some newer OS functions and options conditionally available based on the operating system version in use at runtime (”weaklinking”).

(Contributed by Ronald Oussoren and Lawrence D’Anna in bpo-41100.)

索引

非字母

P

R