
What's New in Python

Release 3.9.21

A. M. Kuchling

dezembro 09, 2024

Python Software Foundation
Email: docs@python.org

Sumário

1	Resumo – Destaques da versão	3
2	Você deve verificar DeprecationWarning no seu código	4
3	Novas funcionalidades	4
3.1	Operadores de mesclagem & atualização de dicionário	4
3.2	Novos métodos de strings para remover prefixos e sufixos	4
3.3	Type hinting genéricos nas coleções padrão	5
3.4	Novo analisador sintático	5
4	Outras mudanças na linguagem	5
5	Novos módulos	6
5.1	zoneinfo	6
5.2	graphlib	7
6	Módulos melhorados	7
6.1	ast	7
6.2	asyncio	7
6.3	compileall	7
6.4	concurrent.futures	8
6.5	curses	8
6.6	datetime	8
6.7	distutils	8
6.8	fcntl	8
6.9	ftplib	8
6.10	gc	8
6.11	hashlib	9
6.12	http	9
6.13	IDLE e idlelib	9
6.14	imaplib	9
6.15	importlib	10
6.16	inspect	10
6.17	ipaddress	10
6.18	math	10
6.19	multiprocessing	10
6.20	nntplib	11
6.21	os	11

6.22	pathlib	11
6.23	pdb	11
6.24	poplib	11
6.25	pprint	11
6.26	pydoc	12
6.27	random	12
6.28	signal	12
6.29	smtplib	12
6.30	socket	12
6.31	time	12
6.32	sys	12
6.33	tempfile	13
6.34	tracemalloc	13
6.35	typing	13
6.36	unicodedata	13
6.37	venv	13
6.38	xml	13
7	Otimizações	13
8	Descontinuados	15
9	Removidos	16
10	Portando para Python 3.9	17
10.1	Alterações na API Python	17
10.2	Alterações na API C	18
10.3	Alterações de bytecode do CPython	19
11	Mudanças na construção	19
12	Alterações na API C	20
12.1	Novas funcionalidades	20
12.2	Portando para Python 3.9	21
12.3	Removidos	22
13	Alterações notáveis no Python 3.9.1	23
13.1	typing	23
13.2	Suporte a macOS 11.0 (Big Sur) e Apple Silicon Mac	24
14	Alterações notáveis no Python 3.9.2	24
14.1	collections.abc	24
14.2	urllib.parse	24
15	Alterações notáveis no Python 3.9.3	24
16	Alterações notáveis no Python 3.9.5	24
16.1	urllib.parse	24
17	Recursos de segurança notáveis no 3.9.14	25
18	Notable Changes in 3.9.17	25
18.1	tarfile	25
19	Notable changes in 3.9.20	25
19.1	ipaddress	25
19.2	email	25
Índice		26

Versão 3.9.21

Data dezembro 09, 2024

Editor Łukasz Langa

Esse artigo explica os novos recursos no Python 3.9, comparado ao 3.8. O Python 3.9 foi lançado em 5 de outubro de 2020.

Para detalhes completos, veja o changelog.

Ver também:

PEP 596 - Python 3.9 Release Schedule

1 Resumo – Destaques da versão

Novos recursos de sintaxe:

- **PEP 584**, operadores de união adicionados ao `dict`;
- **PEP 585**, type hinting genéricos em coleções comuns;
- **PEP 614**, restrições de gramática relaxadas em decoradores.

Novos recursos embutidos:

- **PEP 616**, métodos de strings para remover prefixos e sufixos.

Novos recursos na biblioteca padrão:

- **PEP 593**, anotações flexíveis de função e variável;
- `os.pidfd_open()` adicionada para permitidos gerenciamento de processo sem corridas e sinais.

Melhorias no interpretador:

- **PEP 573**, acesso rápido ao estado do módulo a partir de métodos de tipos de extensão C;
- **PEP 617**, CPython agora usa um novo analisador sintático com base em GASE;
- vários tipos embutidos do Python (`range`, `tuple`, `set`, `frozenset`, `list`, `dict`) estão agora mais rápidos usando `vectorcall` da **PEP 590**;
- coleta de lixo não causa bloqueio em objetos ressuscitados;
- vários módulos do Python (`_abc`, `audioop`, `_bz2`, `_codecs`, `_contextvars`, `_crypt`, `_functools`, `_json`, `_locale`, `math`, `operator`, `resource`, `time`, `_weakref`) agora usam inicialização multifásica conforme definido pela PEP 489;
- vários módulos da biblioteca padrão (`audioop`, `ast`, `grp`, `_hashlib`, `pwd`, `_posixsubprocess`, `random`, `select`, `struct`, `termios`, `zlib`) estão agora usando a ABI estável definida pela PEP 384.

Novos módulos de biblioteca:

- **PEP 615**, o banco de dados de fusos horários da IANA é agora apresentado na biblioteca padrão no módulo `zoneinfo`;
- uma implementação de um tipo topológico de um grafo agora é fornecida no novo módulo `graphlib`.

Alterações no processo de lançamento:

- **PEP 602**, CPython adota um ciclo de lançamento anual.

2 Você deve verificar DeprecationWarning no seu código

Quando o Python 2.7 ainda tinha suporte, muitas funcionalidades do Python 3 foram mantidas para compatibilidade com versões anteriores do Python 2.7. Com o fim do suporte ao Python 2, essas camadas de compatibilidade com versões anteriores foram removidas ou serão removidas em breve. A maioria deles emitia um aviso de `DeprecationWarning` por vários anos. Por exemplo, usar `collections.Mapping` em vez de `collections.abc.Mapping` emite uma `DeprecationWarning` desde Python 3.3, lançado em 2012.

Teste sua aplicação com a opção de linha de comando `-W default` para ver `DeprecationWarning` e `PendingDeprecationWarning`, ou mesmo com `-W error` para tratá-los como erros. Filtro de avisos pode ser usado para ignorar avisos de código de terceiros.

Python 3.9 é a última versão que fornece essas camadas de compatibilidade com versões anteriores do Python 2, para dar mais tempo aos mantenedores de projetos Python para organizar a remoção do suporte Python 2 e adicionar suporte para Python 3.9.

Aliases para classes bases abstratas no módulo `collections`, como o alias `collections.Mapping` para `collections.abc.Mapping`, são mantidos para uma última versão para compatibilidade reversa. Eles serão removidos do Python 3.10.

De maneira mais geral, experimente executar seus testes no Modo de Desenvolvimento do Python, que ajuda a preparar seu código para torná-lo compatível com a próxima versão do Python.

Observação: vários itens descontinuados pré-existentes também foram removidos nesta versão do Python. Consulte a seção [Removidos](#).

3 Novas funcionalidades

3.1 Operadores de mesclagem & atualização de dicionário

Operadores de mesclagem (`|`) e atualização (`|=`) foram adicionados à classe embutida `dict`. Eles complementam o métodos `dict.update` e `{**d1, **d2}` existentes de mesclagem de dicionários.

Exemplo:

```
>>> 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'}
```

Veja [PEP 584](#) para uma descrição completa. (Contribuição de Brandt Bucher em [bpo-36144](#).)

3.2 Novos métodos de strings para remover prefixos e sufixos

`str.removeprefix(prefix)` e `str.removesuffix(suffix)` foram adicionados para remover facilmente um prefixo ou sufixo desnecessário de uma string. Os métodos correspondentes `bytes`, `bytearray` e `collections.UserString` também foram adicionados. Veja [PEP 616](#) para uma descrição completa. (Contribuição de Dennis Sweeney em [bpo-39939](#).)

3.3 Type hinting genéricos nas coleções padrão

Agora, nas anotações de tipo, você pode usar tipos de coleção embutidos como `list` e `dict` como tipos genéricos, em vez de importar os tipos de letras maiúsculas correspondentes (por exemplo, `List` ou `Dict`) de `typing`. Agora, outros tipos na biblioteca padrão também são genéricos, por exemplo, `queue.Queue`.

Exemplo:

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

Consulte [PEP 585](#) para mais detalhes. (Contribuição de Guido van Rossum, Ethan Smith e Batuhan Taşkaya em [bpo-39481](#).)

3.4 Novo analisador sintático

O Python 3.9 usa um novo analisador sintático, baseado em [GASE](#) em vez de [LL\(1\)](#). O desempenho do novo analisador sintático é aproximadamente comparável ao do analisador sintático antigo, mas o formalismo do GASE é mais flexível que do LL(1) quando se trata de projetar novos recursos de linguagem. Começaremos a usar essa flexibilidade no Python 3.10 e posterior.

O módulo `ast` usa o novo analisador e produz o mesmo AST que o analisador antigo.

No Python 3.10, o analisador antigo será excluído e todas as funcionalidades que dependem dele (principalmente o módulo `parser`, que foi descontinuado a muito tempo). No Python 3.9 *apenas*, você pode voltar ao analisador LL(1) usando uma opção de linha de comando (`-X oldparser`) ou uma variável de ambiente (`PYTHONOLDPARSER=1`).

Consulte [PEP 617](#) para mais detalhes. (Contribuição de Guido van Rossum, Pablo Galindo e Lysandros Nikolaou em [bpo-40334](#).)

4 Outras mudanças na linguagem

- `__import__()` agora levanta `ImportError` em vez de `ValueError`, o que é usado para ocorrer quando uma importação relativa passou de seu um pacote de nível superior. (Contribuição de Ngalim Siregar em [bpo-37444](#).)
- Agora, o Python obtém o caminho absoluto do nome do arquivo do script especificado na linha de comando (p.ex., `python3 script.py`): o atributo `__file__` do módulo `__main__` tornou-se um caminho absoluto, não um caminho relativo. Esses caminhos agora permanecem válidos depois que o diretório atual é alterado por `os.chdir()`. Como efeito colateral, o `traceback` também exibe o caminho absoluto para quadros de módulos de `__main__` neste caso. (Contribuição de Victor Stinner em [bpo-20443](#).)
- No Modo de Desenvolvimento do Python e na compilação de depuração, os argumentos `encoding` e `errors` agora são verificados quanto às operações de codificação e decodificação de strings. Exemplos: `open()`, `str.encode()` e `bytes.decode()`.

Por padrão, para obter o melhor desempenho, o argumento `errors` é verificado apenas no primeiro erro de codificação/decodificação, e o argumento `encoding` às vezes é ignorado para strings vazias. (Contribuição de Victor Stinner em [bpo-37388](#).)

- `"".replace("", s, n)` agora retorna `s` em vez de uma string vazia para todos `n` não zero. Isso é agora consistente com `"".replace("", s)`. Há alterações similares para objetos `bytes` e `bytearray`. (Contribuição de Serhiy Storchaka em [bpo-28029](#).)
- Qualquer expressão válida agora pode ser usada como decorador. Anteriormente, a gramática era muito mais restritiva. Veja [PEP 614](#) para detalhes. (Contribuição de Brandt Bucher em [bpo-39702](#).)

- Aprimorada a ajuda para o módulo `digitando`. Agora, as linhas de documento são mostradas para todos os formulários especiais e aliases genéricos especiais (como `Union` e `List`). Usando `help()` com alias genérico como `List[int]` mostrará a ajuda para o tipo concreto correspondente (neste caso, `list`). (Contribuição de Serhiy Storchaka em [bpo-40257](#).)
- A execução paralela de `aclose()` / `asend()` / `athrow()` é agora proibida e `ag_running` agora reflete o status atual de execução do gerador `async`. (Contribuição de Yury Selivanov em [bpo-30773](#).)
- Erros inesperados ao chamar o método `__iter__` não são mais mascarados por `TypeError` no operador `in` e nas funções `contains()`, `indexOf()` e `countOf()` do módulo `operator`. (Contribuição de Serhiy Storchaka em [bpo-40824](#).)
- Expressões lambda sem parênteses não podem mais ser a parte da expressão em uma cláusula `if` em compreensões e expressões geradoras. Veja [bpo-41848](#) e [bpo-43755](#) para detalhes.

5 Novos módulos

5.1 zoneinfo

O módulo `zoneinfo` oferece suporte ao banco de dados de fuso horário da IANA para a biblioteca padrão. Ele adiciona `zoneinfo.ZoneInfo`, uma implementação concreta `datetime.tzinfo`, apoiada pelos dados do fuso horário do sistema.

Exemplo:

```
>>> 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
```

Como fonte alternativa de dados para plataformas que não entregam o banco de dados da IANA, o módulo `tzdata` foi lançado como um pacote inicial – distribuído via PyPI e mantido pela equipe principal do CPython.

Ver também:

PEP 615 – Suporte ao banco de dados de fuso horário da IANA na biblioteca padrão PEP escrita e implementada por Paul Ganssle

5.2 graphlib

Um novo módulo, `graphlib`, foi adicionado contendo a classe `graphlib.TopologicalSorter` para oferecer funcionalidade de realizar classificação topológica de grafos. (Contribuição de Pablo Galindo, Tim Peters e Larry Hastings em [bpo-17005](#).)

6 Módulos melhorados

6.1 ast

Adicionada a opção `indent` a `dump()`, que permite produzir uma saída indentada com várias linhas. (Contribuição de Serhiy Storchaka em [bpo-37995](#).)

Adicionada `ast.unparse()` como uma função no módulo `ast` que pode ser usado para remover um objeto `ast.AST` e produzir uma string com código que produziria um objeto `ast.AST` equivalente quando analisado. (Contribuição de Pablo Galindo e Batuhan Taskaya em [bpo-38870](#).)

Adicionadas docstrings aos nós AST que contêm a assinatura ASDL usada para construir esse nó. (Contribuição de Batuhan Taskaya em [bpo-39638](#).)

6.2 asyncio

Devido a significativas preocupações de segurança, o parâmetro `reuse_address` de `asyncio.loop.create_datagram_endpoint()` não é mais suportado. Isso ocorre devido ao comportamento da opção de soquete `SO_REUSEADDR` no UDP. Para mais detalhes, consulte a documentação de `loop.create_datagram_endpoint()`. (Contribuição de Kyle Stanley, Antoine Pitrou e Yury Selivanov em [bpo-37228](#).)

Adicionada uma nova corrotina `shutdown_default_executor()` que agenda um desligamento para o executor padrão que espera na `ThreadPoolExecutor` para concluir o fechamento. Além disso, `asyncio.run()` foi atualizada para usar o novo corrotina. (Contribuição de Kyle Stanley em [bpo-34037](#).)

Adicionada `asyncio.PidfdChildWatcher`, uma implementação de monitorador de filhos específica do Linux que controla descritores de arquivo de processo. ([bpo-38692](#))

Adicionada uma nova corrotina `asyncio.to_thread()`. É usada principalmente para executar funções vinculadas à E/S em uma thread separada para evitar o bloqueio do loop de eventos e funciona essencialmente como uma versão de alto nível de `run_in_executor()` que pode receber argumentos nomeados diretamente. (Contribuição de Kyle Stanley e Yury Selivanov em [bpo-32309](#).)

Ao cancelar a tarefa devido a um tempo limite, `asyncio.wait_for()` irá agora esperar até que o cancelamento seja concluído também no caso quando `timeout` é ≤ 0 , como acontece com tempos limite positivos. (Contribuição de Elvis Pranskevichus em [bpo-32751](#).)

`asyncio` agora levanta `TypeError` ao chamar métodos incompatíveis com um soquete `ssl.SSLSocket`. (Contribuição de Ido Michael em [bpo-37404](#).)

6.3 compileall

Adicionada nova possibilidade de usar hardlinks para arquivos `.pyc` duplicados: parâmetro `hardlink_dupes` e opção de linha de comando `-hardlink-dupes`. (Contribuição de Lumír 'Frenzy' Balhar em [bpo-40495](#).)

Adicionadas novas opções para manipulação de caminho nos arquivos `.pyc` resultantes: parâmetros `stripdir`, `prependdir`, `limit_sl_dest` e opções de linha de comando `-s`, `-p`, `-e`. Foi adicionada a possibilidade de especificar a opção para um nível de otimização várias vezes. (Contribuição de Lumír 'Frenzy' Balhar em [bpo-38112](#).)

6.4 concurrent.futures

Adicionado um novo parâmetro `cancel_futures` em `concurrent.futures.Executor.shutdown()` que cancela todos os futuros pendentes que ainda não começaram a ser executados, em vez de esperar que eles sejam concluídos antes de desligar o executor. (Contribuição de Kyle Stanley em [bpo-39349](#).)

Removidos as threads do daemon de `ThreadPoolExecutor` e `ProcessPoolExecutor`. Isso melhora a compatibilidade com subinterpretadores e a previsibilidade em seus processos de desligamento. (Contribuição de Kyle Stanley em [bpo-39812](#).)

Workers em `ProcessPoolExecutor` agora são gerados sob demanda, apenas quando não há workers ociosos disponíveis para reutilização. Isso otimiza a sobrecarga da inicialização e reduz a quantidade de tempo perdido da CPU para os workers ociosos. (Contribuição de Kyle Stanley em [bpo-39207](#).)

6.5 curses

Adicionadas as funções `curses.get_escdelay()`, `curses.set_escdelay()`, `curses.get_tabsize()` e `curses.set_tabsize()`. (Contribuição de Anthony Sottile em [bpo-38312](#).)

6.6 datetime

Os métodos `isocalendar()` de `datetime.date` e `isocalendar()` de `datetime.datetime` agora retornam uma `namedtuple()` em vez de uma `tuple`. (Contribuição de Dong-hee Na em [bpo-24416](#).)

6.7 distutils

O comando **upload** agora cria digests de hash SHA2-256 e Blake2b-256. Ele pula o MD5 em plataformas que bloqueiam digest do MD5. (Contribuição de Christian Heimes em [bpo-40698](#).)

6.8 fcntl

Adicionadas as constantes `F_OFD_GETLK`, `F_OFD_SETLK` e `F_OFD_SETLKW`. (Contribuição de Dong-hee Na em [bpo-38602](#).)

6.9 ftplib

FTP e FTP_TLS agora levantam a `ValueError` se o tempo limite especificado para o construtor for zero para impedir a criação de um soquete não-bloqueante. (Contribuição de Dong-hee Na em [bpo-39259](#).)

6.10 gc

Quando o coletor de lixo faz uma coleção na qual alguns objetos ressuscitam (são alcançáveis fora dos ciclos isolados após a execução dos finalizadores), não bloqueia a coleção de todos os objetos que ainda estão inacessíveis. (Contribuição de Pablo Galindo e Tim Peters em [bpo-38379](#).)

Adicionada uma nova função `gc.is_finalized()` para verificar se um objeto foi finalizado pelo coletor de lixo. (Contribuição de Pablo Galindo em [bpo-39322](#).)

6.11 hashlib

O módulo `hashlib` pode agora usar hashes SHA3 e SHAKE XOF de OpenSSL quando disponível. (Contribuição de Christian Heimes em [bpo-37630](#).)

Os módulos `hash` integrados agora podem ser desativados com `./configure --without-builtin-hashlib-hashes` ou ativados seletivamente com por exemplo `./configure --with-builtin-hashlib-hashes=sha3,blake2` para forçar o uso da implementação baseada em OpenSSL. (Contribuição de Christian Heimes em [bpo-40479](#))

6.12 http

Códigos de status HTTP 103 `EARLY_HINTS`, 418 `IM_A_TEAPOT` e 425 `TOO_EARLY` foram adicionados a `http.HTTPStatus`. (Contribuição de Dong-hee Na em [bpo-39509](#) e Ross Rhodes em [bpo-39507](#).)

6.13 IDLE e idlelib

Adicionada a opção para ativar e desativar o piscar do cursor. (Contribuição de Zackery Spytz na [bpo-4603](#).)

A tecla `Escape` agora fecha as janelas de conclusão IDLE. (Contribuição de Johnny Najera em [bpo-38944](#).)

Adicionada argumentos nomeados à lista de completamento de nome de módulo. (Contribuição de Terry J. Reedy em [bpo-37765](#).)

Novo nas versões de manutenção 3.9.

Faz o IDLE invocar `sys.excepthook()` (quando iniciado sem `-n`). Ganchos de usuário eram ignorados anteriormente. (Contribuição de Ken Hilton em [bpo-43008](#).)

As alterações acima foram portadas para versões de manutenção do 3.8.

Reorganiza a caixa de diálogo de configurações. Divide a aba `General` nas abas `Windows` e `Shell/Ed`. Move as fontes de ajuda, que estendem o menu `Help`, para a aba `Extensions`. Abre espaço para novas opções e encurta a caixa de diálogo. O último faz com que o diálogo se ajuste melhor a telas pequenas. (Contribuição de Terry Jan Reedy em [bpo-40468](#).) Move a configuração do espaço de recuo da aba `Font` para a nova aba `Windows`. (Contribuição de Mark Roseman e Terry Jan Reedy em [bpo-33962](#).)

Aplica realce de sintaxe em arquivos `.pyi`. (Contribuição de Alex Waygood e Terry Jan Reedy em [bpo-45447](#).)

6.14 imaplib

IMAP4 e IMAP4_SSL agora possuem o parâmetro opcional `timeout` para seus construtores. Também, o método `open()` agora possui um parâmetro opcional `timeout` com esta alteração. Os métodos substituídos de IMAP4_SSL e IMAP4_stream foram aplicados nesta alteração. (Contribuição de Dong-hee Na em [bpo-38615](#).)

`imaplib.IMAP4.unselect()` foi adicionado. `imaplib.IMAP4.unselect()` libera os recursos do servidor associados à caixa de correio selecionada e retorna o servidor ao estado autenticado. Este comando executa as mesmas ações que `imaplib.IMAP4.close()`, exceto que nenhuma mensagem é removida permanentemente da caixa de correio selecionada no momento. (Contribuição de Dong-hee Na em [bpo-40375](#).)

6.15 importlib

Para melhorar a consistência com as instruções de importação, `importlib.util.resolve_name()` agora levanta `ImportError` em vez de `ValueError` para tentativas de importação relativas inválidas. (Contribuição de Ngalim Siregar em [bpo-37444](#).)

Carregadores de importação que publicam objetos de módulo imutáveis agora podem publicar pacotes imutáveis além de módulos individuais. (Contribuição de Dino Viehland em [bpo-39336](#).)

Adicionada a função `importlib.resources.files()` com suporte a subdiretórios nos dados do pacote, correspondendo ao backport na versão 1.5 do `importlib_resources`. (Contribuição de Jason R. Coombs em [bpo-39791](#).)

Renovado `importlib.metadata` de `importlib_metadata` versão 1.6.1.

6.16 inspect

`inspect.BoundArguments.arguments` foi alterado de `OrderedDict` para `dict` regular. (Contribuição de Inada Naoki em [bpo-36350](#) e [bpo-39775](#).)

6.17 ipaddress

`ipaddress` agora suporta endereços IPv6 com escopo definido (endereço IPv6 com sufixo `%<scope_id>`).

Os endereços IPv6 com escopo definido podem ser analisados usando `ipaddress.IPv6Address`. Se presente, o ID da zona de escopo está disponível através do atributo `scope_id`. (Contribuição de Oleksandr Pavliuk em [bpo-34788](#).)

A partir do Python 3.9.5, o módulo `ipaddress` não aceita mais zeros à esquerda em strings de endereço IPv4. (Contribuição de Christian Heimes em [bpo-36384](#).)

6.18 math

Expandida a função `math.gcd()` para manipular vários argumentos. Anteriormente, tinha suporte apenas a dois argumentos. (Contribuição de Serhiy Storchaka em [bpo-39648](#).)

Adicionada `math.lcm()`: retorna o menor múltiplo comum dos argumentos especificados. (Contribuição de Mark Dickinson, Ananthakrishnan e Serhiy Storchaka em [bpo-39479](#) e `issue: 39648`.)

Adicionada `math.nextafter()`: retorna o próximo valor de ponto flutuante após `x` em direção a `y`. (Contribuição de Victor Stinner em [bpo-39288](#).)

Adicionada `math.ulp()`: retorna o valor do bit menos significativo de um ponto flutuante. (Contribuição de Victor Stinner em [bpo-39310](#).)

6.19 multiprocessing

A classe `multiprocessing.SimpleQueue` tem um novo método `close()` para fechar explicitamente a fila. (Contribuição de Victor Stinner em [bpo-30966](#).)

6.20 nntplib

NNTP e NNTP_SSL agora levantam uma exceção `ValueError` se o tempo limite fornecido ao construtor é zero, para prevenir a criação de um socket não-bloqueante. (Contribuição de Dong-hee Na em [bpo-39259](#).)

6.21 os

Adicionadas as constantes `CLD_KILLED` e `CLD_STOPPED` para `si_code`. (Contribuição de Dong-hee Na em [bpo-38493](#).)

Exposta `os.pidfd_open()` específica do Linux ([bpo-38692](#)) e `os.P_PIDFD` ([bpo-38713](#)) para gerenciamento de processos com descritores de arquivo.

A função `os.unsetenv()` agora também está disponível no Windows. (Contribuição de Victor Stinner em [bpo-39413](#).)

As funções `os.putenv()` e `os.unsetenv()` estão agora sempre disponíveis. (Contribuição de Victor Stinner em [bpo-39395](#).)

Adicionada a função `os.waitstatus_to_exitcode()`: converte um status de espera em um código de saída. (Contribuição de Victor Stinner em [bpo-40094](#).)

As of 3.9.20, `os.mkdir()` and `os.makedirs()` on Windows now support passing a *mode* value of `0o700` to apply access control to the new directory. This implicitly affects `tempfile.mkdtemp()` and is a mitigation for CVE-2024-4030. Other values for *mode* continue to be ignored. (Contributed by Steve Dower in [gh-118486](#).)

6.22 pathlib

Adicionado `pathlib.Path.readlink()`, que age de maneira semelhante a `os.readlink()`. (Contribuição de Girts Folkmanis em [issue: 30618](#))

6.23 pdb

No Windows, agora `Pdb` tem suporte a `~/ .pdbrc`. (Contribuição de Tim Hopper e Dan Lidral-Porter em [bpo-20523](#).)

6.24 poplib

POP3 e POP3_SSL agora levantam a `ValueError` se o tempo limite especificado para o construtor for zero para impedir a criação de um soquete sem bloqueio. (Contribuição de Dong-hee Na em [bpo-39259](#).)

6.25 pprint

`pprint` agora pode imprimir `types.SimpleNamespace`. (Contribuição de Carl Bordum Hansen em [bpo-37376](#).)

6.26 pydoc

A string de documentação agora é mostrada não apenas para classe, função, método etc, mas para qualquer objeto que tenha seu próprio atributo `__doc__`. (Contribuição de Serhiy Storchaka em [bpo-40257](#).)

6.27 random

Adicionado um novo método `random.Random.randbytes`: gera bytes aleatórios. (Contribuição de Victor Stinner em [bpo-40286](#).)

6.28 signal

Exposta a `signal.pidfd_send_signal()` específica do Linux para enviar sinais a um processo usando um descritor de arquivo em vez de um pid. ([bpo-38712](#))

6.29 smtplib

SMTP e SMTP_SSL agora levantam uma `ValueError` se o tempo limite especificado para seu construtor for zero para impedir a criação de um soquete sem bloqueio. (Contribuição de Dong-hee Na em [bpo-39259](#).)

O construtor de LMTP agora tem um parâmetro opcional `timeout`. (Contribuição de Dong-hee Na em [bpo-39329](#).)

6.30 socket

O módulo `socket` agora exporta a constante `CAN_RAW_JOIN_FILTERS` no Linux 4.1 e superior. (Contribuição de Stefan Tatschner e Zackery Spytz em [bpo-25780](#).)

O módulo `socket` agora suporta o protocolo `CAN_J1939` nas plataformas que o suportam. (Contribuição de Karl Ding em [bpo-40291](#).)

O módulo `socket` agora tem as funções `socket.send_fds()` e `socket.recv_fds()`. (Contribuição de Joannah Nanjeyke, Shinya Okano e Victor Stinner em [bpo-28724](#).)

6.31 time

No AIX, `thread_time()` agora é implementado com `thread_cputime()` que possui resolução de nanossegundos, em vez de `clock_gettime(CLOCK_THREAD_CPUTIME_ID)`, que tem uma resolução de 10 ms. (Contribuição de Batuhan Taskaya em [bpo-40192](#).)

6.32 sys

Adicionado um novo atributo `sys.platlibdir`: nome do diretório da biblioteca específica da plataforma. É usado para construir o caminho da biblioteca padrão e os caminhos dos módulos de extensão instalados. É igual a `"lib"` na maioria das plataformas. No Fedora e SuSE, é igual a `"lib64"` nas plataformas de 64 bits. (Contribuição de Jan Matějček, Matěj Cepl, Charalampos Stratakis e Victor Stinner em [bpo-1294959](#).)

Anteriormente, `sys.stderr` era armazenado em buffer quando não interativo. Agora, `stderr` assume como padrão sempre o buffer de linha. (Contribuição de Jendrik Seipp em [bpo-13601](#).)

6.33 tempfile

As of 3.9.20 on Windows, the default mode `0o700` used by `tempfile.mkdtemp()` now limits access to the new directory due to changes to `os.mkdir()`. This is a mitigation for CVE-2024-4030. (Contributed by Steve Dower in [gh-118486](#).)

6.34 tracemalloc

Adicionada `tracemalloc.reset_peak()` para definir o tamanho do pico dos blocos de memória rastreados para o tamanho atual, para medir o pico de partes específicas do código. (Contribuição de Huon Wilson em [bpo-40630](#).)

6.35 typing

PEP 593 introduziu um tipo `typing.Annotated` para decorar tipos existentes com metadados específicos de contexto e o novo parâmetro `include_extras` para `typing.get_type_hints()` para acessar os metadados em tempo de execução. (Contribuição de Till Varoquaux e Konstantin Kashin.)

6.36 unicodedata

O banco de dados Unicode foi atualizado para a versão 13.0.0. ([bpo-39926](#)).

6.37 venv

Os scripts de ativação fornecidos por `venv` agora especificam sua personalização de prompt de forma consistente, sempre usando o valor especificado por `__VENV_PROMPT__`. Anteriormente, alguns scripts usavam incondicionalmente `__VENV_PROMPT__`, outros apenas se fosse definido (que era o caso padrão) e um usava `__VENV_NAME__`. (Contribuição de Brett Cannon em [bpo-37663](#).)

6.38 xml

Os caracteres de espaço em branco nos atributos agora são preservados ao serializar `xml.etree.ElementTree` para o arquivo XML. EOLNs não são mais normalizados para “n”. Este é o resultado da discussão sobre como interpretar a seção 2.11 da especificação XML. (Contribuição de Mefistotelis em [bpo-39011](#).)

7 Otimizações

- Otimizado o idioma para atribuição de uma variável temporária nas compreensões. Agora `for y in [expr]` nas compreensões é tão rápido quanto uma atribuição simples `y = expr`. Por exemplo:

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

Ao contrário do operador `:=`, este idioma não vaza uma variável para o escopo externo.

(Contribuição de Serhiy Storchaka em [bpo-32856](#).)

- Otimizada a manipulação de sinais em aplicações multithread. Se uma thread diferente da thread principal obtiver um sinal, o loop de avaliação do bytecode não será mais interrompido a cada instrução de bytecode para verificar se há sinais pendentes que não podem ser manipulados. Somente o segmento principal do interpretador principal pode manipular sinais.

Anteriormente, o loop de avaliação do bytecode era interrompido em cada instrução até que a thread principal lidasse com os sinais. (Contribuição de Victor Stinner em [bpo-40010](#).)

- Otimizado o módulo `subprocess` no FreeBSD usando `closefrom()`. (Contribuição de Ed Maste, Conrad Meyer, Kyle Evans, Kubilay Kocak e Victor Stinner em [bpo-38061](#).)
- `PyLong_FromDouble()` está agora até 1.87x mais rápido para valores que cabem em um `long`. (Contribuição de Sergey Fedoseev em [bpo-37986](#).)
- Vários componentes embutidos do Python (`range`, `tuple`, `set`, `frozenset`, `list`, `dict`) agora são acelerados usando o protocolo `vectorcall` da [PEP 590](#). (Contribuição de Dong-hee Na, Mark Shannon, Jeroen Demeyer e Petr Viktorin em [bpo-37207](#).)
- Otimizada a `difference_update()` para o caso em que o outro conjunto é muito maior do que o conjunto base. (Sugestão de Evgeny Kapun com contribuição de código de Michele Orrù em [bpo-8425](#).)
- O alocador de pequenos objetos do Python (`obmalloc.c`) agora permite (não mais que) uma arena vazia para permanecer disponível para reutilização imediata, sem retorná-la ao sistema operacional. Isso evita o desperdiçar loops simples, onde uma arena pode ser criada e destruída novamente em cada iteração. (Contribuição de Tim Peters em [bpo-37257](#).)
- divisão pelo piso de operação flutuante agora tem um melhor desempenho. Além disso, a mensagem de `ZeroDivisionError` para esta operação foi atualizada. (Contribuição de Dong-hee Na em [bpo-39434](#).)
- A decodificação de strings curtas ASCII com codecs UTF-8 e `ascii` é agora cerca de 15% mais rápida. (Contribuição de Inada Naoki em [bpo-37348](#).)

Aqui está um resumo das melhorias de desempenho do Python 3.4 ao Python 3.9:

Python version	3.4	3.5	3.6	3.7	3.8	3.9
-----	---	---	---	---	---	---
Variable and attribute read access:						
<code>read_local</code>	7.1	7.1	5.4	5.1	3.9	3.9
<code>read_nonlocal</code>	7.1	8.1	5.8	5.4	4.4	4.5
<code>read_global</code>	15.5	19.0	14.3	13.6	7.6	7.8
<code>read_builtin</code>	21.1	21.6	18.5	19.0	7.5	7.8
<code>read_classvar_from_class</code>	25.6	26.5	20.7	19.5	18.4	17.9
<code>read_classvar_from_instance</code>	22.8	23.5	18.8	17.1	16.4	16.9
<code>read_instancevar</code>	32.4	33.1	28.0	26.3	25.4	25.3
<code>read_instancevar_slots</code>	27.8	31.3	20.8	20.8	20.2	20.5
<code>read_namedtuple</code>	73.8	57.5	45.0	46.8	18.4	18.7
<code>read_boundmethod</code>	37.6	37.9	29.6	26.9	27.7	41.1
Variable and attribute write access:						
<code>write_local</code>	8.7	9.3	5.5	5.3	4.3	4.3
<code>write_nonlocal</code>	10.5	11.1	5.6	5.5	4.7	4.8
<code>write_global</code>	19.7	21.2	18.0	18.0	15.8	16.7
<code>write_classvar</code>	92.9	96.0	104.6	102.1	39.2	39.8
<code>write_instancevar</code>	44.6	45.8	40.0	38.9	35.5	37.4
<code>write_instancevar_slots</code>	35.6	36.1	27.3	26.6	25.7	25.8
Data structure read access:						
<code>read_list</code>	24.2	24.5	20.8	20.8	19.0	19.5
<code>read_deque</code>	24.7	25.5	20.2	20.6	19.8	20.2
<code>read_dict</code>	24.3	25.7	22.3	23.0	21.0	22.4
<code>read_strdict</code>	22.6	24.3	19.5	21.2	18.9	21.5
Data structure write access:						
<code>write_list</code>	27.1	28.5	22.5	21.6	20.0	20.0
<code>write_deque</code>	28.7	30.1	22.7	21.8	23.5	21.7
<code>write_dict</code>	31.4	33.3	29.3	29.2	24.7	25.4
<code>write_strdict</code>	28.4	29.9	27.5	25.2	23.1	24.5
Stack (or queue) operations:						
<code>list_append_pop</code>	93.4	112.7	75.4	74.2	50.8	50.6
<code>deque_append_pop</code>	43.5	57.0	49.4	49.2	42.5	44.2

(continua na próxima página)

(continuação da página anterior)

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

Estes resultados foram gerados a partir do script de benchmark de acesso variável em: `Tools/scripts/var_access_benchmark.py`. O script de benchmark exibe tempos em nanossegundos. Os benchmarks foram medidos em um processador Intel® Core™ i7-4960HQ executando as compilações do macOS de 64 bits encontradas em python.org.

8 Descontinuados

- O comando `bdist_msi` do `distutils` está agora descontinuado, use `bdist_wheel` (pacotes `wheel`). (Contribuição de Hugo van Kemenade em [bpo-39586](#).)
- Atualmente `math.factorial()` aceita instâncias da `float` com valores inteiros não negativos (como `5.0`). Ela levanta uma `ValueError` para flutuadores não integrais e negativos. Agora está descontinuado. Nas versões futuras do Python, será levantada uma `TypeError` para todos os flutuadores. (Contribuição de Serhiy Storchaka em [bpo-37315](#).)
- Os módulos `parser` e `symbol` foram descontinuados e serão removidos em versões futuras do Python. Para a maioria dos casos de uso, os usuários podem aproveitar o estágio de geração e compilação da Árvore de Sintaxe Abstrata (AST), usando o módulo `ast`.
- As funções públicas de API C `PyParser_SimpleParseStringFlags()`, `PyParser_SimpleParseStringFlagsFilename()`, `PyParser_SimpleParseFileFlags()` e `PyNode_Compile()` foram descontinuados e serão removidos no Python 3.10 junto com o analisador sintático antigo.
- O uso de `NotImplemented` em um contexto booleano foi descontinuado, pois é quase exclusivamente o resultado de implementações incorretas de comparadores avançados. Isso será tornado em uma `TypeError` em uma versão futura do Python. (Contribuição de Josh Rosenberg em [bpo-35712](#).)
- O módulo `random` atualmente aceita qualquer tipo hashável como um possível valor inicial. Infelizmente, alguns desses tipos não têm garantia de ter um valor de hash determinístico. Após o Python 3.9, o módulo restringirá suas sementes a `None`, `int`, `float`, `str`, `bytes` e `bytearray`.
- A abertura do arquivo `GzipFile` para escrita sem especificar o argumento `mode` foi descontinuado. Nas versões futuras do Python, ele sempre será aberto para leitura por padrão. Especifique o argumento `mode` para abri-lo para escrever e silenciar um aviso. (Contribuição de Serhiy Storchaka em [bpo-28286](#).)
- Descontinuado o método `split()` de `_tkinter.TkappType` a favor do método `splitlist()` que possui um comportamento mais consistente e previsível. (Contribuição de Serhiy Storchaka em [bpo-38371](#).)
- A passagem explícita dos objetos da corrotina para `asyncio.wait()` foi descontinuada e será removida na versão 3.11. (Contribuição de Yury Selivanov e Kyle Stanley em [bpo-34790](#).)
- Os padrões `binhex4` e `hexbin4` agora estão obsoletos. O módulo `binhex` e as seguintes funções `binascii` foram descontinuadas:
 - `b2a_hqx()`, `a2b_hqx()`
 - `rlecode_hqx()`, `rledecode_hqx()`(Contribuição de Victor Stinner em [bpo-39353](#).)
- As classes `slice`, `Index` e `ExtSlice` de `ast` são consideradas descontinuadas e serão removidas nas versões futuras do Python. O próprio `value` deve ser usado em vez de `Index(value)`. `Tuple(slices, Load())` deve ser usada em vez de `ExtSlice(slices)`. (Contribuição de Serhiy Storchaka em [bpo-34822](#).)

- As classes `Suite`, `Param`, `AugLoad` e `AugStore` de `ast` são consideradas descontinuadas e serão removidas nas versões futuras do Python. Elas não foram geradas pelo analisador sintático e não foram aceitos pelo gerador de código no Python 3. (Contribuição de Batuhan Taskaya em [bpo-39639](#) e [bpo-39969](#) e Serhiy Storchaka em [bpo-39988](#).)
- As funções `PyEval_InitThreads()` e `PyEval_ThreadsInitialized()` agora estão descontinuadas e serão removidas no Python 3.11. Chamar `PyEval_InitThreads()` agora não faz nada. O GIL é inicializado por `Py_Initialize()` desde o Python 3.7. (Contribuição de Victor Stinner em [bpo-39877](#).)
- Passar `None` como o primeiro argumento para a função `shlex.split()` foi descontinuada. (Contribuição de Zackery Spytz em [bpo-33262](#).)
- `smtpd.MailmanProxy()` está agora descontinuado, pois está inutilizável sem um módulo externo, `mailman`. (Contribuição de Samuel Colvin em [bpo-35800](#).)
- O módulo `lib2to3` agora emite um `PendingDeprecationWarning`. Python 3.9 mudou para um analisador sintático GASE (veja [PEP 617](#)) e Python 3.10 pode incluir uma nova sintaxe de linguagem que não é analisável pelo analisador sintático LL(1) de `lib2to3`. O módulo `lib2to3` pode ser removido da biblioteca padrão em uma versão futura do Python. Considere alternativas de terceiros, como `LibCST` ou `parso`. (Contribuição de Carl Meyer em [bpo-40360](#).)
- Adicionado o parâmetro `random` da `random.shuffle()` foi descontinuado. (Contribuição de Raymond Hettinger em [bpo-40465](#))

9 Removidos

- A versão errônea em `unittest.mock.__version__` foi removida.
- `nntplib.NNTP`: os métodos `xpath()` e `xgtitle()` foram removidos. Esses métodos foram descontinuados desde Python 3.3. Geralmente, essas extensões não são suportadas ou não são habilitadas pelos administradores do servidor NNTP. Para `xgtitle()`, use `nntplib.NNTP.descriptions()` ou `nntplib.NNTP.description()`. (Contribuição de Dong-hee Na em [bpo-39366](#).)
- `array.array`: os métodos `tostring()` e `fromstring()` foram removidos. Eles eram aliases para `tobytes()` e `frombytes()`, descontinuado desde Python 3.2. (Contribuição de Victor Stinner em [bpo-38916](#).)
- A função `sys.callstats()` não documentada foi removida. Desde o Python 3.7, ela foi descontinuada e sempre retornou `None`. É necessária uma opção de compilação especial `CALL_PROFILE` que já foi removida no Python 3.7. (Contribuição de Victor Stinner em [bpo-37414](#).)
- As funções `sys.getcheckinterval()` e `sys.setcheckinterval()` foram removidas. Elas foram descontinuadas desde o Python 3.2. Use `sys.getswitchinterval()` e `sys.setswitchinterval()` em seu lugar. (Contribuição de Victor Stinner em [bpo-37392](#).)
- A função `C PyImport_Cleanup()` foi removida. Foi documentada como: “Esvazia a tabela do módulo. Apenas para uso interno.” (Contribuição de Victor Stinner em [bpo-36710](#).)
- Os módulos `_dummy_thread` e `dummy_threading` foram removidos. Esses módulos foram descontinuados desde o Python 3.7, que requer suporte a `threading`. (Contribuição de Victor Stinner em [bpo-37312](#).)
- `aifc.openfp()` alias para `aifc.open()`, `sunau.openfp()` alias para `sunau.open()` e `wave.openfp()` alias para `wave.open()` foi removido. Eles foram descontinuados desde o Python 3.7. (Contribuição de Victor Stinner em [bpo-37320](#).)
- O método `isAlive()` de `threading.Thread` foi removido. Ele foi descontinuado desde o Python 3.8. Use `is_alive()` ao invés. (Contribuição de Dong-hee Na em [bpo-37804](#).)
- Os métodos `getchildren()` e `getiterator()` das classes `ElementTree` e `Element` no módulo `ElementTree` foram removidos. Eles foram descontinuados no Python 3.2. Use `iter(x)` ou `list(x)` em vez de `x.getchildren()` e `x.iter()` ou `list(x.iter())` em vez de `x.getiterator()`. (Contribuição Serhiy Storchaka no [bpo-36543](#).)

- A API antiga `plistlib` foi removida. Ela tinha sido descontinuada desde o Python 3.4. Use as funções `load()`, `loads()`, `dump()` e `dumps()`. Além disso, o parâmetro `use_builtin_types` foi removido, os objetos padrão `bytes` são sempre usados em seu lugar. (Contribuição de Jon Janzen em [bpo-36409](#).)
- A função `C PyGen_NeedsFinalizing` foi removida. Ela não foi documentada, testada ou usada em qualquer lugar dentro do CPython após a implementação de [PEP 442](#). Patch de Joanna Nanjey. (Contribuição de Joanna Nanjey em [bpo-15088](#))
- `base64.encodestring()` e `base64.decodestring()`, aliases descontinuados desde Python 3.1, foram removidos: use `base64.encodebytes()` e `base64.decodebytes()`. (Contribuição de Victor Stinner em [bpo-39351](#).)
- A função `fractions.gcd()` foi removida, foi descontinuado desde Python 3.5 ([bpo-22486](#)): use `math.gcd()` em seu lugar. (Contribuição de Victor Stinner em [bpo-39350](#).)
- O parâmetro `buffering` de `bz2.BZ2File` foi removido. Desde Python 3.0, ele era ignorado e ao usá-lo emitia uma `DeprecationWarning`. Passe um objeto arquivo aberto para controlar como o arquivo é aberto. (Contribuição de Victor Stinner em [bpo-39357](#).)
- O parâmetro `encoding` de `json.loads()` foi removido. A partir do Python 3.1, ele foi descontinuado e ignorado; usá-lo emitia uma `DeprecationWarning` desde Python 3.8. (Contribuição de Inada Naoki em [bpo-39377](#))
- As instruções `with (await asyncio.lock):` e `with (yield from asyncio.lock):` não são mais suportadas, use `async with lock`. O mesmo é se aplica para `asyncio.Condition` e `asyncio.Semaphore`. (Contribuição de Andrew Svetlov em [bpo-34793](#).)
- A função `sys.getcounts()`, a opção de linha de comando `-X showalloccount` e o campo `show_alloc_count` da estrutura `C PyConfig` foram removidos. Eles exigiram uma construção especial do Python definindo a macro `COUNT_ALLOCS`. (Contribuição de Victor Stinner em [bpo-39489](#).)
- O atributo `_field_types` da classe `typing.NamedTuple` foi removido. Ele foi descontinuado desde o Python 3.8. Use o atributo `__annotations__`. (Contribuição de Serhiy Storchaka em [bpo-40182](#).)
- O método `symtable.SymbolTable.has_exec()` foi removido. Ele foi descontinuado desde 2006, e retornando apenas `False` quando é chamado. (Contribuição de Batuhan Taskaya em [bpo-40208](#))
- Os métodos `asyncio.Task.current_task()` e `asyncio.Task.all_tasks()` foram removidos. Eles foram descontinuados desde o Python 3.7 e você pode usar `asyncio.current_task()` e `asyncio.all_tasks()` em seu lugar. (Contribuição de Rémi Lapeyre em [bpo-40967](#))
- O método `unescape()` na classe `html.parser.HTMLParser` foi removido (foi descontinuado desde o Python 3.4). `html.unescape()` deve ser usada para converter referências de caracteres para os caracteres Unicode correspondentes.

10 Portando para Python 3.9

Esta seção lista as alterações descritas anteriormente e outras correções que podem exigir alterações no seu código.

10.1 Alterações na API Python

- `__import__()` e `importlib.util.resolve_name()` agora levanta `ImportError` onde anteriormente levantava `ValueError`. Os chamadores capturando o tipo de exceção específico e suportando Python 3.9 e versões anteriores precisarão capturar ambas usando `except (ImportError, ValueError):`.
- Os scripts de ativação `venv` não são mais casos especiais quando `__VENV_PROMPT__` está definido como `""`.
- O método `select.epoll.unregister()` não ignora mais o erro `EBADF`. (Contribuição de Victor Stinner em [bpo-39239](#).)

- O parâmetro `compresslevel` de `bz2.BZ2File` tornou-se um parâmetro somente-nomeado, uma vez que o parâmetro `buffering` foi removido. (Contribuição de Victor Stinner em [bpo-39357](#).)
- AST simplificada para assinatura. Índices simples serão representados por seus valores, fatias estendidas serão representadas como tuplas. `Index(value)` retornará um `value` em si, `ExtSlice(slices)` retornará `Tuple(slices, Load())`. (Contribuição de Serhiy Storchaka em [bpo-34822](#).)
- O módulo `importlib` agora ignora a variável de ambiente `PYTHONCASEOK` quando as opções de linha de comando `-E` ou `-I` estão sendo usadas.
- O parâmetro `encoding` foi adicionado às classes `ftplib.FTP` e `ftplib.FTP_TLS` como um parâmetro somente-nomeado, e a codificação padrão foi alterada de `Latin-1` para `UTF-8` para siga [RFC 2640](#).
- `asyncio.loop.shutdown_default_executor()` foi adicionado a `AbstractEventLoop`, significando que laços de eventos alternativos que herdam dele devem ter este método definido. (Contribuição de Kyle Stanley em [bpo-34037](#).)
- Os valores constantes de futuras sinalizações no módulo `__future__` são atualizados para evitar colisão com sinalizações do compilador. Anteriormente, `PyCF_ALLOW_TOP_LEVEL_AWAIT` estava em conflito com `CO_FUTURE_DIVISION`. (Contribuição de Batuhan Taskaya em [bpo-39562](#).)
- `array('u')` agora usa `wchar_t` como tipo C em vez de `Py_UNICODE`. Esta mudança não afeta seu comportamento porque `Py_UNICODE` é um apelido de `wchar_t` desde Python 3.3. (Contribuição de Inada Naoki em [bpo-34538](#).)
- A API de `logging.getLogger()` agora retorna o logger raiz quando passado o nome `'root'`, enquanto anteriormente retornava um logger não-raiz chamado `'root'`. Isso pode afetar casos onde o código do usuário deseja explicitamente um logger não-root chamado `'root'`, ou instancia um logger usando `logging.getLogger(__name__)` em algum módulo de nível superior chamado `'root.py'`. (Contribuição de Vinay Sajip em [bpo-37742](#).)
- O tratamento de divisão de `PurePath` agora retorna `NotImplemented` ao invés de levantar um `TypeError` quando passado algo diferente de uma instância de `str` ou `PurePath`. Isso permite a criação de classes compatíveis que não herdam desses tipos mencionados. (Contribuição de Roger Aiudi em [bpo-34775](#).)
- A partir do Python 3.9.5, o módulo `ipaddress` não aceita mais zeros à esquerda em strings de endereço IPv4. Os zeros à esquerda são ambíguos e interpretados como notação octal por algumas bibliotecas. Por exemplo, a função legada `socket.inet_aton()` trata os zeros à esquerda como notação octal. A implementação glibc de `inet_pton()` moderna não aceita nenhum zeros à esquerda. (Contribuição de Christian Heimes em [bpo-36384](#).)
- `codecs.lookup()` agora normaliza o nome da codificação da mesma forma que `encodings.normalize_encoding()`, exceto que `codecs.lookup()` também converte o nome para minúsculas. Por exemplo, `"latex+latin1"` nome de codificação agora está normalizado para `"latex_latin1"`. (Contribuição de Jordon Xu em [bpo-37751](#).)

10.2 Alterações na API C

- Instâncias de tipos alocados por heap (como aquelas criadas com `PyType_FromSpec()` e APIs semelhantes) mantêm uma referência para seu objeto de tipo desde o Python 3.8. Conforme indicado nas “Mudanças na API C” do Python 3.8, para a grande maioria dos casos, não deve haver nenhum efeito colateral, mas para os tipos que têm uma função personalizada `tp_traverse`, certifique-se de que todas as funções `tp_traverse` personalizadas de tipos alocados por heap visitam o tipo do objeto.

Exemplo:

```
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)
```

(continua na próxima página)

(continuação da página anterior)

```
Py_VISIT(Py_TYPE(self));  
#endif  
}
```

Se sua função transversal delega para `tp_traverse` de sua classe base (ou outro tipo), certifique-se de que `Py_TYPE(self)` seja visitado apenas uma vez. Observe que apenas os tipos de heap devem visitar o tipo em `tp_traverse`.

Por exemplo, se sua função `tp_traverse` inclui:

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

adicione:

```
#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));  
    }  
#else
```

(Veja [bpo-35810](#) e [bpo-40217](#) para mais informações.)

- As funções `PyEval_CallObject`, `PyEval_CallFunction`, `PyEval_CallMethod` e `PyEval_CallObjectWithKeywords` foram descontinuadas. Use `PyObject_Call()` e suas variantes. (Veja mais detalhes em [bpo-29548](#).)

10.3 Alterações de bytecode do CPython

- O opcode `LOAD_ASSERTION_ERROR` foi adicionado para tratar a instrução `assert`. Anteriormente, a instrução `assert` não funcionava corretamente se a exceção `AssertionError` estivesse sendo ocultada. (Contribuição de Zackery Spytz em [bpo-34880](#).)
- O código de operação `COMPARE_OP` foi dividido em quatro instruções distintas:
 - `COMPARE_OP` para comparações ricas
 - `IS_OP` para testes “is” e “is not”
 - `CONTAINS_OP` para testes “in” e “not in”
 - `JUMP_IF_NOT_EXC_MATCH` para verificar exceções em instruções “try-except”.

(Contribuição de Mark Shannon em [bpo-39156](#).)

11 Mudanças na construção

- Adicionada a opção `--with-platlibdir` ao script `configure`: nome do diretório de biblioteca específico da plataforma, armazenado no novo atributo `sys.platlibdir`. Veja o atributo `sys.platlibdir` para mais informações. (Contribuição de Jan Matějček, Matěj Cepl, Charalampos Stratakis e Victor Stinner em [bpo-1294959](#).)
- A macro especial de construção `COUNT_ALLOCS` foi removida. (Contribuição de Victor Stinner em [bpo-39489](#).)
- Em plataformas não Windows, as funções `setenv()` e `unsetenv()` agora são necessárias para construir o Python. (Contribuição de Victor Stinner em [bpo-39395](#).)

- Em plataformas não-Windows, a criação de instaladores `bdist_wininst` não possui mais suporte oficial. (Veja [bpo-10945](#) para mais detalhes.)
- Ao compilar Python no macOS a partir do código-fonte, `_tkinter` agora se vincula a frameworks Tcl e Tk não pertencentes ao sistema se eles estiverem instalados em `/Library/Frameworks`, como era o caso em versões mais antigas do macOS. Se um macOS SDK for explicitamente configurado, usando `--enable-universalsdk=` ou `-isysroot`, apenas o próprio SDK será pesquisado. O comportamento padrão ainda pode ser substituído com `--with-tcltk-includes` e `--with-tcltk-libs`. (Contribuição de Ned Deily em [bpo-34956](#).)
- Python pode agora ser construído para Windows 10 ARM64. (Contribuição de Steve Dower em [bpo-33125](#).)
- Alguns testes individuais agora são ignorados quando `--pgo` é usada. Os testes em questão aumentaram significativamente o tempo da tarefa de PGO e provavelmente não ajudaram a melhorar a otimização do executável final. Isso acelera a tarefa por um fator de cerca de 15x. A execução de todo o conjunto de testes de unidade é lenta. Esta mudança pode resultar em uma construção um pouco menos otimizada, uma vez que nem tantos ramos de código serão executados. Se você estiver disposto a esperar por uma construção muito mais lenta, o comportamento antigo pode ser restaurado usando `./configure [...] PROFILE_TASK="-m test --pgo-extended"`. Não oferecemos garantia de qual conjunto de tarefas de PGO produz uma construção mais rápida. Os usuários que se importam devem executar seus próprios benchmarks relevantes, pois os resultados podem depender do ambiente, da carga de trabalho e da cadeia de ferramentas do compilador. (Veja [bpo-36044](#) e [bpo-37707](#) para mais detalhes.)

12 Alterações na API C

12.1 Novas funcionalidades

- **PEP 573:** Adicionada `PyType_FromModuleAndSpec()` para associar um módulo a uma classe; `PyType_GetModule()` e `PyType_GetModuleState()` para obter o módulo e seu estado; e `PyCMethod` e `METH_METHOD` para permitir que um método acesse a classe na qual ela foi definida. (Contribuição de Marcel Plch e Petr Viktorin em [bpo-38787](#).)
- Adicionada a função `PyFrame_GetCode()`: obtém um código de quadro. Adiciona a função `PyFrame_GetBack()`: obtém o próximo quadro externo. (Contribuição de Victor Stinner em [bpo-40421](#).)
- Adicionada `PyFrame_GetLineNumber()` à API C limitada. (Contribuição de Victor Stinner em [bpo-40421](#).)
- Adicionada as funções `PyThreadState_GetInterpreter()` e `PyInterpreterState_Get()` para obter o interpretador. Adiciona a função `PyThreadState_GetFrame()` para obter o quadro atual de um estado de thread do Python. Adiciona a função `PyThreadState_GetID()`: obtém o identificador único de um estado de thread do Python. (Contribuição de Victor Stinner em [bpo-39947](#).)
- Adicionada uma nova função pública `PyObject_CallNoArgs()` à API C, que chama um objeto chamável Python sem nenhum argumento. É a maneira mais eficiente de chamar um objeto chamável Python sem nenhum argumento. (Contribuição de Victor Stinner em [bpo-37194](#).)
- Alterações na API C limitada (se a macro `Py_LIMITED_API` estiver definida):
 - Fornece `Py_EnterRecursiveCall()` e `Py_LeaveRecursiveCall()` como funções regulares para a API limitada. Anteriormente, eram definidas como macros, mas essas macros não compilavam com a API C limitada, que não pode acessar o campo `PyThreadState.recursion_depth` (a estrutura não é visível na API C limitada).
 - `PyObject_INIT()` e `PyObject_INIT_VAR()` se uma função regular não visível para ocultar os detalhes da implementação.
 (Contribuição de Victor Stinner em [bpo-38644](#) e [bpo-39542](#).)
- A função `PyModule_AddType()` é adicionado para ajudar a adicionar um tipo a um módulo. (Contribuição de Dong-hee Na em [bpo-40024](#).)

- Adicionadas as funções `PyObject_GC_IsTracked()` e `PyObject_GC_IsFinalized()` para a API pública para permitir a consulta se os objetos Python estão sendo rastreados ou já foram finalizados pelo coletor de lixo, respectivamente. (Contribuição de Pablo Galindo em [bpo-40241](#).)
- Adicionada a `_PyObject_FunctionStr()` para obter uma representação amigável para o usuário de strings de um objeto função ou similar. (Patch por Jeroen Demeyer em [bpo-37645](#).)
- Adicionada `PyObject_CallOneArg()` para chamar um objeto com um argumento posicional (Patch de Jeroen Demeyer em [bpo-37483](#).)

12.2 Portando para Python 3.9

- `PyInterpreterState.eval_frame` (**PEP 523**) agora exige um novo parâmetro *tstate* obrigatório (`PyThreadState*`). (Contribuição de Victor Stinner em [bpo-38500](#).)
- Módulos de extensão: As funções `m_traverse`, `m_clear` e `m_free` de `PyModuleDef` não são mais chamados se o estado do módulo foi solicitado, mas ainda não está alocado. Este é o caso imediatamente após o módulo ser criado e antes de o módulo ser executado (função `Py_mod_exec`). Mais precisamente, estas funções são chamados se `m_size` for maior que 0 e o estado do módulo (como retornado por `PyModule_GetState()`) for `NULL`.

Módulos de extensão sem um estado de módulo (`m_size <= 0`) não são afetados.

- Se `Py_AddPendingCall()` for chamada em um subinterpretador, a função agora está programada para ser chamada a partir do subinterpretador, em vez de ser chamada a partir do interpretador principal. Cada subinterpretador agora tem sua própria lista de chamadas programadas. (Contribuição de Victor Stinner em [bpo-39984](#).)
- O registro do Windows não é mais usado para inicializar `sys.path` quando a opção `-E` é usada (se `PyConfig.use_environment` estiver definido como 0). Isso é significativo ao incorporar Python no Windows. (Contribuição de Zackery Spytz na [bpo-8901](#).)
- A variável global `PyStructSequence_UnnamedField` agora é uma constante e se refere a uma string constante. (Contribuição de Serhiy Storchaka em [bpo-38650](#).)
- A estrutura `PyGC_Head` agora não é mais visível. Ela só é definida na API C interna (`pycore_gc.h`). (Contribuição de Victor Stinner em [bpo-40241](#).)
- `Py_UNICODE_COPY`, `Py_UNICODE_FILL`, `PyUnicode_WSTR_LENGTH`, `PyUnicode_FromUnicode()`, `PyUnicode_AsUnicode()`, `_PyUnicode_AsUnicode` e `PyUnicode_AsUnicodeAndSize()` estão marcadas como descontinuadas no C. Elas foram descontinuadas pela **PEP 393** desde Python 3.3. (Contribuição de Inada Naoki em [bpo-36346](#).)
- A função `Py_FatalError()` é substituída por uma macro que registra automaticamente o nome da função atual, a menos que a macro `Py_LIMITED_API` seja definida. (Contribuição de Victor Stinner em [bpo-39882](#).)
- O protocolo vectorcall agora exige que o chamador passe apenas strings como nomes de argumentos. (Veja [bpo-37540](#) para mais informações.)
- Os detalhes de implementação de uma série de macros e funções agora estão ocultos:
 - A macro `PyObject_IS_GC()` foi convertida para uma função.
 - A macro `PyObject_NEW()` se torna um alias para a macro `PyObject_New()` e a macro `PyObject_NEW_VAR()` se torna um alias para a macro `PyObject_NewVar()`. Elas não acessam mais diretamente o membro `PyTypeObject.tp_basicsize`.
 - A macro `PyObject_GET_WEAKREFS_LISTPTR()` foi convertida para uma função: a macro acessava diretamente o membro `PyTypeObject.tp_weaklistoffset`.
 - A macro `PyObject_CheckBuffer()` foi convertida para uma função: a macro acessava diretamente o membro `PyTypeObject.tp_as_buffer`.

- `PyIndex_Check()` está agora declarada como uma função “opaca” para ocultar detalhes da implementação: removida a macro `PyIndex_Check()`. A macro acessava diretamente o membro `PyObject.tp_as_number`.

(Veja [bpo-40170](#) para mais detalhes.)

12.3 Removidos

- Excluídas as macros `PyFPE_START_PROTECT()` e `PyFPE_END_PROTECT()` de `pyfpe.h` da API C limitada. (Contribuição por Victor Stinner em [bpo-38835](#).)
- O slot `tp_print` de `PyObject` foi removido. Ele era usado para imprimir objetos em arquivos no Python 2.7 e anteriores. Desde Python 3.0, ele era ignorado e não utilizado. (Contribuição de Jeroen Demeyer em [bpo-36974](#).)
- Alterações na API C limitada (se a macro `Py_LIMITED_API` estiver definida):

- Excluídas as seguintes funções da API C limitada:

- * `PyThreadState_DeleteCurrent()` (Contribuição de Joannah Nanjeyke em [bpo-37878](#).)
- * `_Py_CheckRecursionLimit`
- * `_Py_NewReference()`
- * `_Py_ForgetReference()`
- * `_PyTraceMalloc_NewReference()`
- * `_Py_GetRefTotal()`
- * O mecanismo trashcan que nunca funcionou na API C limitada.
- * `PyTrash_UNWIND_LEVEL`
- * `Py_TRASHCAN_BEGIN_CONDITION`
- * `Py_TRASHCAN_BEGIN`
- * `Py_TRASHCAN_END`
- * `Py_TRASHCAN_SAFE_BEGIN`
- * `Py_TRASHCAN_SAFE_END`

- Movidas as seguintes funções e definições para a API C interna:

- * `_PyDebug_PrintTotalRefs()`
- * `_Py_PrintReferences()`
- * `_Py_PrintReferenceAddresses()`
- * `_Py_tracemalloc_config`
- * `_Py_AddToAllObjects()` (específico para construção de `Py_TRACE_REFS`)

(Contribuição de Victor Stinner em [bpo-38644](#) e [bpo-39542](#).)

- Removido o gancho `_PyRuntime.getframe` e removida a macro `_PyThreadState_GetFrame` que era um alias para `_PyRuntime.getframe`. Eles eram os únicos expostos pela API C interna. Removido também o tipo `PyThreadFrameGetter`. (Contribuição de Victor Stinner em [bpo-39946](#).)
 - Removidas as funções a seguir da API C. Chame `PyGC_Collect()` explicitamente para limpar todas as listas livres. (Contribuição de Inada Naoki e Victor Stinner em [bpo-37340](#), [bpo-38896](#) e [bpo-40428](#).)
- `PyAsyncGen_ClearFreeLists()`
 - `PyContext_ClearFreeList()`
 - `PyDict_ClearFreeList()`

- `PyFloat_ClearFreeList()`
- `PyFrame_ClearFreeList()`
- `PyList_ClearFreeList()`
- `PyMethod_ClearFreeList()` e `PyCFunction_ClearFreeList()`: as listas livres de objetos de método vinculado foram removidas.
- `PySet_ClearFreeList()`: a definição de lista livres foi removida no Python 3.4.
- `PyTuple_ClearFreeList()`
- `PyUnicode_ClearFreeList()`: a lista livre de Unicode foi removida no Python 3.3.
- Removida a função `_PyUnicode_ClearStaticStrings()`. (Contribuição de Victor Stinner em [bpo-39465](#).)
- Removido `Py_UNICODE_MATCH`. Ele foi descontinuado por [PEP 393](#) e quebrado desde o Python 3.3. A função `PyUnicode_Tailmatch()` pode ser usada em seu lugar. (Contribuição de Inada Naoki em [bpo-36346](#).)
- Limpados os arquivos de cabeçalho de interfaces definidas, mas sem implementação. Os símbolos públicos da API sendo removidos são: `_PyBytes_InsertThousandsGroupingLocale`, `_PyBytes_InsertThousandsGrouping`, `_Py_InitializeFromArgs`, `_Py_InitializeFromWideArgs`, `_PyFloat_Repr`, `_PyFloat_Digits`, `_PyFloat_DigitsInit`, `PyFrame_ExtendStack`, `_PyAlterWrapper_Type`, `PyNullImporter_Type`, `PyCmpWrapper_Type`, `PySortWrapper_Type`, `PyNoArgsFunction`. (Contribuição de Pablo Galindo Salgado em [bpo-39372](#).)

13 Alterações notáveis no Python 3.9.1

13.1 typing

O comportamento de `typing.Literal` foi alterado para ficar em conformidade com a [PEP 586](#) e para corresponder ao comportamento de verificadores de tipo estático especificados na PEP.

1. `Literal` agora elimina a duplicação de parâmetros.
2. Comparações de igualdade entre objetos `Literal` agora são independentes da ordem.
3. Comparações de `Literal` agora respeitam os tipos. Por exemplo, `Literal[0] == Literal[False]` avaliava anteriormente como `True`. Agora é `False`. Para oferecer suporte a essa mudança, o cache de tipo usado internamente agora oferece suporte a tipos de diferenciação.
4. Objetos `Literal` agora irão levantar uma exceção `TypeError` durante as comparações de igualdade se algum de seus parâmetros não for hashável. Observe que declarar `Literal` com parâmetros mutáveis não gerará um erro:

```
>>> 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'
```

(Contribuição de Yuri Karabas em [bpo-42345](#).)

13.2 Suporte a macOS 11.0 (Big Sur) e Apple Silicon Mac

A partir do 3.9.1, o Python agora oferece suporte total à construção e execução no macOS 11.0 (Big Sur) e no Apple Silicon Macs (baseado na arquitetura ARM64). Uma nova variante de compilação universal, `universal2`, está agora disponível para suportar nativamente ambos ARM64 e Intel 64 em um conjunto de executáveis. Os binários também podem ser construídos em versões atuais do macOS para serem implantados em uma variedade de versões mais antigas do macOS (testado para 10.9), enquanto algumas funções e opções do sistema operacional mais recentes estão disponíveis condicionalmente com base na versão do sistema operacional em uso no tempo de execução (“ligação fraca”)

(Contribuição de Ronald Oussoren e Lawrence D’Anna em [bpo-41100](#).)

14 Alterações notáveis no Python 3.9.2

14.1 `collections.abc`

`collections.abc.Callable` genérica agora nivela os parâmetros de tipo, semelhante ao que `typing.Callable` faz atualmente. Isso significa que `collections.abc.Callable[[int, str], str]` terá `__args__` de `(int, str, str)`; anteriormente era `([int, str], str)`. Para permitir esta mudança, `types.GenericAlias` agora pode ter uma subclasse, e uma subclasse será retornada ao subscrever o tipo `Collections.abc.Callable`. O código que acessa os argumentos via `typing.get_args()` ou `__args__` precisa levar em conta esta mudança. Um `DeprecationWarning` pode ser emitido para formas inválidas de parametrização `Collections.abc.Callable` que pode ter passado silenciosamente no Python 3.9.1. Esta `DeprecationWarning` se tornará uma `TypeError` no Python 3.10. (Contribuição de Ken Jin em [bpo-42195](#).)

14.2 `urllib.parse`

Versões anteriores do Python permitiam o uso de `;` e `&` como separadores de parâmetros de consulta nas funções `urllib.parse.parse_qs()` e `urllib.parse.parse_qsl()`. Devido a questões de segurança, e em conformidade com as recomendações mais recentes do W3C, isso foi alterado para permitir apenas uma única chave separadora, com `&` como padrão. Esta mudança também afeta as funções `cgi.parse()` e `cgi.parse_multipart()` já que elas usam as funções afetadas internamente. Para obter mais detalhes, consulte sua respectiva documentação. (Contribuição de Adam Goldschmidt, Senthil Kumaran e Ken Jin em [bpo-42967](#).)

15 Alterações notáveis no Python 3.9.3

Uma correção de segurança altera o comportamento `ftplib.FTP` para não confiar no endereço IPv4 enviado do servidor remoto ao configurar um canal de dados passivo. Em vez disso, reutilizamos o endereço IP do servidor ftp. Para códigos incomuns que requerem o comportamento antigo, defina um atributo `trust_server_pasv_ipv4_address` em sua instância de FTP para `True`. (Veja [bpo-43285](#))

16 Alterações notáveis no Python 3.9.5

16.1 `urllib.parse`

A presença de caracteres de nova linha ou tab em partes de um URL permite algumas formas de ataques. Seguindo a especificação WHATWG que atualiza [RFC 3986](#), nova linha ASCII `\n`, `\r` e os caracteres de tabulação `\t` são retirados da URL pelo analisador sintático em `urllib.parse` impedindo tais ataques. Os caracteres de remoção são controlados por uma nova variável de nível de módulo `urllib.parse._UNSAFE_URL_BYTES_TO_REMOVE`. (Veja [bpo-43882](#))

17 Recursos de segurança notáveis no 3.9.14

Converter entre `int` e `str` em bases diferentes de 2 (binário), 4, 8 (octal), 16 (hexadecimal) ou 32 como base 10 (decimal) agora levanta uma exceção `ValueError` se o número de dígitos em forma de string estiver acima de um limite para evitar possíveis ataques de negação de serviço devido à complexidade algorítmica. Esta é uma mitigação para [CVE-2020-10735](#). Este limite pode ser configurado ou desabilitado por variável de ambiente, sinalizador de linha de comando ou APIs de `sys`. Veja a documentação de limitação de comprimento de conversão de string inteira. O limite padrão é de 4300 dígitos em forma de string.

18 Notable Changes in 3.9.17

18.1 tarfile

- Os métodos de extração em `tarfile` e `shutil.unpack_archive()`, têm um novo argumento *filter* que permite limitar recursos do tar que podem ser surpreendentes ou perigosos, como criar arquivos fora do diretório de destino. Veja `tarfile-extraction-filter` para detalhes. No Python 3.12, usar sem o argumento *filter* mostrará um `DeprecationWarning`. No Python 3.14, o padrão mudará para `'data'`. (Contribuição de Petr Viktorin em [PEP 706](#).)

19 Notable changes in 3.9.20

19.1 ipaddress

- Corrigido o comportamento de `is_global` e `is_private` em `IPv4Address`, `IPv6Address`, `IPv4Network` e `IPv6Network`.

19.2 email

- Cabeçalhos com novas linhas embutidas agora são colocados entre aspas na saída.
O `generator` agora se recusará a serializar (escrever) cabeçalhos que são dobrados ou delimitados incorretamente, de modo que eles seriam analisados como vários cabeçalhos ou unidos a dados adjacentes. Se você precisar desativar esse recurso de segurança, defina `verify_generated_headers`. (Contribuição de Bas Bloemsaat e Petr Viktorin em [gh-121650](#).)
- `email.utils.getaddresses()` e `email.utils.parseaddr()` agora retornam tuplas de 2 elementos (`'', ''`) em mais situações onde endereços de e-mail inválidos são encontrados, em vez de valores potencialmente imprecisos. Um parâmetro *strict* opcional foi adicionado a essas duas funções: use `strict=False` para obter o comportamento antigo, aceitando entradas malformadas. `getattr(email.utils, 'supports_strict_parsing', False)` pode ser usado para verificar se o parâmetro *strict* está disponível. (Contribuição de Thomas Dwyer e Victor Stinner para [gh-102988](#) para melhorar a correção de [CVE-2023-27043](#).)

Índice

P

Propostas Estendidas Python

PEP 393, [21](#), [23](#)

PEP 442, [17](#)

PEP 523, [21](#)

PEP 573, [3](#), [20](#)

PEP 584, [3](#), [4](#)

PEP 585, [3](#), [5](#)

PEP 586, [23](#)

PEP 590, [3](#), [14](#)

PEP 593, [3](#), [13](#)

PEP 596, [3](#)

PEP 602, [3](#)

PEP 614, [3](#), [5](#)

PEP 615, [3](#), [6](#)

PEP 616, [3](#), [4](#)

PEP 617, [3](#), [5](#), [16](#)

PEP 706, [25](#)

PYTHONCASEOK, [18](#)

R

RFC

RFC 2640, [18](#)

RFC 3986, [24](#)

V

váriavel de ambiente

PYTHONCASEOK, [18](#)