
What's New in Python

Versión 3.14.0a0

A. M. Kuchling

septiembre 06, 2024

Python Software Foundation
Email: docs@python.org

Índice general

1	Summary – Release highlights	2
2	New Features	2
3	Other Language Changes	2
4	New Modules	3
5	Improved Modules	3
5.1	ast	3
5.2	ctypes	3
5.3	dis	3
5.4	fractions	4
5.5	http	4
5.6	json	4
5.7	operator	4
5.8	os	4
5.9	pathlib	4
5.10	pdb	5
5.11	pickle	5
5.12	symtable	5
6	Optimizations	5
6.1	asyncio	5
7	Deprecated	5
7.1	Pending Removal in Python 3.15	6
7.2	Pending Removal in Python 3.16	6
7.3	Pending Removal in Future Versions	7
8	Removed	9
8.1	argparse	9
8.2	ast	9
8.3	asyncio	10
8.4	collections.abc	10

8.5	email	10
8.6	importlib	10
8.7	itertools	11
8.8	pathlib	11
8.9	pty	11
8.10	sqlite3	11
8.11	typing	11
8.12	urllib	11
8.13	Others	11
9	Porting to Python 3.14	12
9.1	Changes in the Python API	12
10	Build Changes	12
11	C API Changes	12
11.1	New Features	12
11.2	Porting to Python 3.14	14
11.3	Deprecated	14
11.4	Removed	15
	Índice	16

Editor

TBD

This article explains the new features in Python 3.14, compared to 3.13.

For full details, see the changelog.

Nota

Prerelease users should be aware that this document is currently in draft form. It will be updated substantially as Python 3.14 moves towards release, so it's worth checking back even after reading earlier versions.

1 Summary – Release highlights

2 New Features

3 Other Language Changes

- Incorrect usage of `await` and asynchronous comprehensions is now detected even if the code is optimized away by the `-O` command line option. For example, `python -O -c 'assert await 1'` now produces a `SyntaxError`. (Contributed by Jelle Zijlstra in [gh-121637](#).)
- Writes to `__debug__` are now detected even if the code is optimized away by the `-O` command line option. For example, `python -O -c 'assert (__debug__ := 1)'` now produces a `SyntaxError`. (Contributed by Irit Katriel in [gh-122245](#).)

- Added class methods `float.from_number()` and `complex.from_number()` to convert a number to `float` or `complex` type correspondingly. They raise an error if the argument is a string. (Contributed by Serhiy Storchaka in [gh-84978](#).)

4 New Modules

- None yet.

5 Improved Modules

5.1 ast

- Added `ast.compare()` for comparing two ASTs. (Contributed by Batuhan Taskaya and Jeremy Hylton in [bpo-15987](#).)
- Add support for `copy.replace()` for AST nodes.
(Contributed by Bénédict Tran in [gh-121141](#).)

5.2 ctypes

- The layout of bit fields in `Structure` and `Union` now matches platform defaults (GCC/Clang or MVSC) more closely. In particular, fields no longer overlap. (Contributed by Matthias Görgens in [gh-97702](#).)
- The `Structure._layout_` class attribute can now be set to help match a non-default ABI. (Contributed by Petr Viktorin in [gh-97702](#).)

5.3 dis

- Added support for rendering full source location information of `instructions`, rather than only the line number. This feature is added to the following interfaces via the `show_positions` keyword argument:
 - `dis.Bytecode`,
 - `dis.dis()`, `dis.distb()`, and
 - `dis.disassemble()`.

This feature is also exposed via `dis --show-positions`.

(Contributed by Bénédict Tran in [gh-123165](#).)

5.4 fractions

Added support for converting any objects that have the `as_integer_ratio()` method to a `Fraction`. (Contributed by Serhiy Storchaka in [gh-82017](#).)

5.5 http

Directory lists and error pages generated by the `http.server` module allow the browser to apply its default dark mode. (Contributed by Yorik Hansen in [gh-123430](#).)

5.6 json

Add notes for JSON serialization errors that allow to identify the source of the error. (Contributed by Serhiy Storchaka in [gh-122163](#).)

Enable `json` module to work as a script using the `-m` switch: `python -m json`. See the JSON command-line interface documentation. (Contributed by Trey Hunner in [gh-122873](#).)

5.7 operator

- Two new functions `operator.is_none` and `operator.is_not_none` have been added, such that `operator.is_none(obj)` is equivalent to `obj is None` and `operator.is_not_none(obj)` is equivalent to `obj is not None`. (Contributed by Raymond Hettinger and Nico Maxis in [gh-115808](#).)

5.8 os

- Added the `os.environ.refresh()` method to update `os.environ` with changes to the environment made by `os.putenv()`, by `os.unsetenv()`, or made outside Python in the same process. (Contributed by Victor Stinner in [gh-120057](#).)

5.9 pathlib

- Add methods to `pathlib.Path` to recursively copy or move files and directories:
 - `copy()` copies a file or directory tree to a destination.
 - `copy_into()` copies *into* a destination directory.
 - `move()` moves a file or directory tree to a destination.
 - `move_into()` moves *into* a destination directory.

(Contributed by Barney Gale in [gh-73991](#).)

5.10 pdb

- **Hard-coded breakpoints** (`breakpoint()` and `pdb.set_trace()`) now reuse the most recent `Pdb` instance that calls `set_trace()`, instead of creating a new one each time. As a result, all the instance specific data like `display` and `commands` are preserved across hard-coded breakpoints. (Contributed by Tian Gao in [gh-121450](#).)

5.11 pickle

- Set the default protocol version on the `pickle` module to 5. For more details, please see [pickle protocols](#).

5.12 symtable

- Expose the following `symtable.Symbol` methods:
 - `is_free_class()`
 - `is_comp_iter()`
 - `is_comp_cell()`

(Contributed by Bénédict Tran in [gh-120029](#).)

6 Optimizations

6.1 asincio

- `asyncio` now uses double linked list implementation for native tasks which speeds up execution by 10% on standard pyperformance benchmarks and reduces memory usage. (Contributed by Kumar Aditya in [gh-107803](#).)

7 Deprecated

- **builtins:** Passing a complex number as the *real* or *imag* argument in the `complex()` constructor is now deprecated; it should only be passed as a single positional argument. (Contributed by Serhiy Storchaka in [gh-109218](#).)
- **os:** Soft deprecate `os.popen()` and `os.spawn*` functions. They should no longer be used to write new code. The `subprocess` module is recommended instead. (Contributed by Victor Stinner in [gh-120743](#).)
- **symtable:** Deprecate `symtable.Class.get_methods()` due to the lack of interest. (Contributed by Bénédict Tran in [gh-119698](#).)

7.1 Pending Removal in Python 3.15

- `http.server.CGIHTTPRequestHandler` will be removed along with its related `--cgi` flag to `python -m http.server`. It was obsolete and rarely used. No direct replacement exists. *Anything* is better than CGI to interface a web server with a request handler.
- `locale: locale.getdefaultlocale()` was deprecated in Python 3.11 and originally planned for removal in Python 3.13 ([gh-90817](#)), but removal has been postponed to Python 3.15. Use `locale.setlocale()`, `locale.getencoding()` and `locale.getlocale()` instead. (Contributed by Hugo van Kemenade in [gh-111187](#).)
- `pathlib: pathlib.PurePath.is_reserved()` is deprecated and scheduled for removal in Python 3.15. Use `os.path.isreserved()` to detect reserved paths on Windows.
- `platform: java_ver()` is deprecated and will be removed in 3.15. It was largely untested, had a confusing API, and was only useful for Jython support. (Contributed by Nikita Sobolev in [gh-116349](#).)
- `threading: Passing any arguments to threading.RLock() is now deprecated. C version allows any numbers of args and kwargs, but they are just ignored. Python version does not allow any arguments. All arguments will be removed from threading.RLock() in Python 3.15. (Contributed by Nikita Sobolev in gh-102029.)`
- `typing.NamedTuple`:
 - The undocumented keyword argument syntax for creating `NamedTuple` classes (`NT = NamedTuple("NT", x=int)`) is deprecated, and will be disallowed in 3.15. Use the class-based syntax or the functional syntax instead.
 - When using the functional syntax to create a `NamedTuple` class, failing to pass a value to the *fields* parameter (`NT = NamedTuple("NT")`) is deprecated. Passing `None` to the *fields* parameter (`NT = NamedTuple("NT", None)`) is also deprecated. Both will be disallowed in Python 3.15. To create a `NamedTuple` class with 0 fields, use `class NT(NamedTuple): pass` or `NT = NamedTuple("NT", [])`.
- `typing.TypedDict`: When using the functional syntax to create a `TypedDict` class, failing to pass a value to the *fields* parameter (`TD = TypedDict("TD")`) is deprecated. Passing `None` to the *fields* parameter (`TD = TypedDict("TD", None)`) is also deprecated. Both will be disallowed in Python 3.15. To create a `TypedDict` class with 0 fields, use `class TD(TypedDict): pass` or `TD = TypedDict("TD", {})`.
- `wave`: Deprecate the `getmark()`, `setmark()` and `getmarkers()` methods of the `wave.Wave_read` and `wave.Wave_write` classes. They will be removed in Python 3.15. (Contributed by Victor Stinner in [gh-105096](#).)

7.2 Pending Removal in Python 3.16

- `array: array.array 'u' type (wchar_t): use the 'w' type instead (Py_UCS4).`
- `builtins: ~bool, bitwise inversion on bool.`
- `symtable: Deprecate symtable.Class.get_methods() due to the lack of interest. (Contributed by Bénédict Tran in gh-119698.)`
- `shutil: Deprecate shutil.ExecError, which hasn't been raised by any shutil function since Python 3.4. It's now an alias for RuntimeError.`

7.3 Pending Removal in Future Versions

The following APIs will be removed in the future, although there is currently no date scheduled for their removal.

- `argparse`: Nesting argument groups and nesting mutually exclusive groups are deprecated.
- `array`'s 'u' format code ([gh-57281](#))
- `builtins`:
 - `bool(NotImplemented)`.
 - `Generators`: `throw(type, exc, tb)` and `athrow(type, exc, tb)` signature is deprecated: use `throw(exc)` and `athrow(exc)` instead, the single argument signature.
 - Currently Python accepts numeric literals immediately followed by keywords, for example `0in x, 1or x, 0if 1else 2`. It allows confusing and ambiguous expressions like `[0x1for x in y]` (which can be interpreted as `[0x1 for x in y]` or `[0x1f or x in y]`). A syntax warning is raised if the numeric literal is immediately followed by one of keywords `and`, `else`, `for`, `if`, `in`, `is` and `or`. In a future release it will be changed to a syntax error. ([gh-87999](#))
 - Support for `__index__()` and `__int__()` method returning non-int type: these methods will be required to return an instance of a strict subclass of `int`.
 - Support for `__float__()` method returning a strict subclass of `float`: these methods will be required to return an instance of `float`.
 - Support for `__complex__()` method returning a strict subclass of `complex`: these methods will be required to return an instance of `complex`.
 - Delegation of `int()` to `__trunc__()` method.
 - Passing a complex number as the *real* or *imag* argument in the `complex()` constructor is now deprecated; it should only be passed as a single positional argument. (Contributed by Serhiy Storchaka in [gh-109218](#).)
- `calendar`: `calendar.January` and `calendar.February` constants are deprecated and replaced by `calendar.JANUARY` and `calendar.FEBRUARY`. (Contributed by Prince Roshan in [gh-103636](#).)
- `codeobject.co_lnotab`: use the `codeobject.co_lines()` method instead.
- `datetime`:
 - `utcnow()`: use `datetime.datetime.now(tz=datetime.UTC)`.
 - `utcfromtimestamp()`: use `datetime.datetime.fromtimestamp(timestamp, tz=datetime.UTC)`.
- `gettext`: Plural value must be an integer.
- `importlib`:
 - `load_module()` method: use `exec_module()` instead.
 - `cache_from_source()` *debug_override* parameter is deprecated: use the *optimization* parameter instead.
- `importlib.metadata`:
 - `EntryPoints` tuple interface.
 - Implicit `None` on return values.
- `logging`: the `warn()` method has been deprecated since Python 3.3, use `warning()` instead.
- `mailbox`: Use of `StringIO` input and text mode is deprecated, use `BytesIO` and binary mode instead.
- `os`: Calling `os.register_at_fork()` in multi-threaded process.

- `pydoc.ErrorDuringImport`: A tuple value for `exc_info` parameter is deprecated, use an exception instance.
- `re`: More strict rules are now applied for numerical group references and group names in regular expressions. Only sequence of ASCII digits is now accepted as a numerical reference. The group name in bytes patterns and replacement strings can now only contain ASCII letters and digits and underscore. (Contributed by Serhiy Storchaka in [gh-91760](#).)
- `sre_compile`, `sre_constants` and `sre_parse` modules.
- `shutil.rmtree()`'s `onerror` parameter is deprecated in Python 3.12; use the `onexc` parameter instead.
- `ssl` options and protocols:
 - `ssl.SSLContext` without protocol argument is deprecated.
 - `ssl.SSLContext.set_npn_protocols()` and `selected_npn_protocol()` are deprecated: use ALPN instead.
 - `ssl.OP_NO_SSL*` options
 - `ssl.OP_NO_TLS*` options
 - `ssl.PROTOCOL_SSLv3`
 - `ssl.PROTOCOL_TLS`
 - `ssl.PROTOCOL_TLSv1`
 - `ssl.PROTOCOL_TLSv1_1`
 - `ssl.PROTOCOL_TLSv1_2`
 - `ssl.TLSVersion.SSLv3`
 - `ssl.TLSVersion.TLSv1`
 - `ssl.TLSVersion.TLSv1_1`
- `sysconfig.is_python_build()` `check_home` parameter is deprecated and ignored.
- `threading` methods:
 - `threading.Condition.notifyAll()`: use `notify_all()`.
 - `threading.Event.isSet()`: use `is_set()`.
 - `threading.Thread.isDaemon()`, `threading.Thread.setDaemon()`: use `threading.Thread.daemon` attribute.
 - `threading.Thread.getName()`, `threading.Thread.setName()`: use `threading.Thread.name` attribute.
 - `threading.currentThread()`: use `threading.current_thread()`.
 - `threading.activeCount()`: use `threading.active_count()`.
- `typing.Text` ([gh-92332](#)).
- `unittest.IsolatedAsyncioTestCase`: it is deprecated to return a value that is not `None` from a test case.
- `urllib.parse` deprecated functions: `urlparse()` instead
 - `splitattr()`
 - `splithost()`
 - `splitnport()`

- `splitpasswd()`
 - `splitport()`
 - `splitquery()`
 - `splittag()`
 - `splitttype()`
 - `splituser()`
 - `splitvalue()`
 - `to_bytes()`
- `urllib.request: URLOpener` and `FancyURLOpener` style of invoking requests is deprecated. Use newer `urlopen()` functions and methods.
 - `wsgiref: SimpleHandler.stdout.write()` should not do partial writes.
 - `xml.etree.ElementTree: Testing the truth value of an Element` is deprecated. In a future release it will always return `True`. Prefer explicit `len(elem)` or `elem is not None` tests instead.
 - `zipimport.zipimporter.load_module()` is deprecated: use `exec_module()` instead.

8 Removed

8.1 argparse

- Remove the *type*, *choices*, and *metavar* parameters of `argparse.BooleanOptionalAction`. They were deprecated since 3.12.

8.2 ast

- Remove the following classes. They were all deprecated since Python 3.8, and have emitted deprecation warnings since Python 3.12:

- `ast.Num`
- `ast.Str`
- `ast.Bytes`
- `ast.NameConstant`
- `ast.Ellipsis`

Use `ast.Constant` instead. As a consequence of these removals, user-defined `visit_Num`, `visit_Str`, `visit_Bytes`, `visit_NameConstant` and `visit_Ellipsis` methods on custom `ast.NodeVisitor` subclasses will no longer be called when the `NodeVisitor` subclass is visiting an AST. Define a `visit_Constant` method instead.

Also, remove the following deprecated properties on `ast.Constant`, which were present for compatibility with the now-removed AST classes:

- `ast.Constant.n`
- `ast.Constant.s`

Use `ast.Constant.value` instead.

(Contributed by Alex Waygood in [gh-119562](#).)

8.3 asyncio

- Remove the following classes and functions. They were all deprecated and emitted deprecation warnings since Python 3.12:

- `asyncio.AbstractChildWatcher`
- `asyncio.SafeChildWatcher`
- `asyncio.MultiLoopChildWatcher`
- `asyncio.FastChildWatcher`
- `asyncio.ThreadedChildWatcher`
- `asyncio.PidfdChildWatcher`
- `asyncio.AbstractEventLoopPolicy.get_child_watcher()`
- `asyncio.AbstractEventLoopPolicy.set_child_watcher()`
- `asyncio.get_child_watcher()`
- `asyncio.set_child_watcher()`

(Contributed by Kumar Aditya in [gh-120804](#).)

8.4 collections.abc

- Remove `collections.abc.ByteString`. It had previously raised a `DeprecationWarning` since Python 3.12.

8.5 email

- Remove the `isdst` parameter from `email.utils.localtime()`. (Contributed by Hugo van Kemenade in [gh-118798](#).)

8.6 importlib

- Remove deprecated `importlib.abc` classes:

- `importlib.abc.ResourceReader`
- `importlib.abc.Traversable`
- `importlib.abc.TraversableResources`

Use `importlib.resources.abc` classes instead:

- `importlib.resources.abc.Traversable`
- `importlib.resources.abc.TraversableResources`

(Contributed by Jason R. Coombs and Hugo van Kemenade in [gh-93963](#).)

8.7 itertools

- Remove `itertools` support for `copy`, `deepcopy`, and `pickle` operations. These had previously raised a `DeprecationWarning` since Python 3.12. (Contributed by Raymond Hettinger in [gh-101588](#).)

8.8 pathlib

- Remove support for passing additional keyword arguments to `pathlib.Path`. In previous versions, any such arguments are ignored.
- Remove support for passing additional positional arguments to `pathlib.PurePath.relative_to()` and `is_relative_to()`. In previous versions, any such arguments are joined onto *other*.

8.9 pty

- Remove deprecated `pty.master_open()` and `pty.slave_open()`. They had previously raised a `DeprecationWarning` since Python 3.12. Use `pty.openpty()` instead. (Contributed by Nikita Sobolev in [gh-118824](#).)

8.10 sqlite3

- Remove `version` and `version_info` from `sqlite3`. (Contributed by Hugo van Kemenade in [gh-118924](#).)
- Disallow using a sequence of parameters with named placeholders. This had previously raised a `DeprecationWarning` since Python 3.12; it will now raise a `sqlite3.ProgrammingError`. (Contributed by Erlend E. Aasland in [gh-118928](#) and [gh-101693](#).)

8.11 typing

- Remove `typing.ByteString`. It had previously raised a `DeprecationWarning` since Python 3.12.

8.12 urllib

- Remove deprecated `Quoter` class from `urllib.parse`. It had previously raised a `DeprecationWarning` since Python 3.11. (Contributed by Nikita Sobolev in [gh-118827](#).)

8.13 Others

- Using `NotImplemented` in a boolean context will now raise a `TypeError`. It had previously raised a `DeprecationWarning` since Python 3.9. (Contributed by Jelle Zijlstra in [gh-118767](#).)
- The `int()` built-in no longer delegates to `__trunc__()`. Classes that want to support conversion to integer must implement either `__int__()` or `__index__()`. (Contributed by Mark Dickinson in [gh-119743](#).)

9 Porting to Python 3.14

This section lists previously described changes and other bugfixes that may require changes to your code.

9.1 Changes in the Python API

- `functools.partial` is now a method descriptor. Wrap it in `staticmethod()` if you want to preserve the old behavior. (Contributed by Serhiy Storchaka and Dominykas Grigonis in [gh-121027](#).)

10 Build Changes

11 C API Changes

11.1 New Features

- Add `PyLong_GetSign()` function to get the sign of `int` objects. (Contributed by Sergey B Kirpichev in [gh-116560](#).)
- Add a new `PyUnicodeWriter` API to create a Python `str` object:
 - `PyUnicodeWriter_Create()`.
 - `PyUnicodeWriter_Discard()`.
 - `PyUnicodeWriter_Finish()`.
 - `PyUnicodeWriter_WriteChar()`.
 - `PyUnicodeWriter_WriteUTF8()`.
 - `PyUnicodeWriter_WriteUCS4()`.
 - `PyUnicodeWriter_WriteWideChar()`.
 - `PyUnicodeWriter_WriteStr()`.
 - `PyUnicodeWriter_WriteRepr()`.
 - `PyUnicodeWriter_WriteSubstring()`.
 - `PyUnicodeWriter_Format()`.
 - `PyUnicodeWriter_DecodeUTF8Stateful()`.

(Contributed by Victor Stinner in [gh-119182](#).)

- Add `PyIter_NextItem()` to replace `PyIter_Next()`, which has an ambiguous return value. (Contributed by Irit Katriel and Erlend Aasland in [gh-105201](#).)
- `Py_Finalize()` now deletes all interned strings. This is backwards incompatible to any C-Extension that holds onto an interned string after a call to `Py_Finalize()` and is then reused after a call to `Py_Initialize()`. Any issues arising from this behavior will normally result in crashes during the execution of the subsequent call to `Py_Initialize()` from accessing uninitialized memory. To fix, use an address sanitizer to identify any use-after-free coming from an interned string and deallocate it during module shutdown. (Contributed by Eddie Elizondo in [gh-113601](#).)
- Add new functions to convert C `<stdint.h>` numbers from/to Python `int`:

- PyLong_FromInt32()
- PyLong_FromInt64()
- PyLong_FromUInt32()
- PyLong_FromUInt64()
- PyLong_AsInt32()
- PyLong_AsInt64()
- PyLong_AsUInt32()
- PyLong_AsUInt64()

(Contributed by Victor Stinner in [gh-120389](#).)

- Add `PyBytes_Join(sep, iterable)` function, similar to `sep.join(iterable)` in Python. (Contributed by Victor Stinner in [gh-121645](#).)
- Add `Py_HashBuffer()` to compute and return the hash value of a buffer. (Contributed by Antoine Pitrou and Victor Stinner in [gh-122854](#).)
- Add functions to get and set the current runtime Python configuration (**PEP 741**):
 - `PyConfig_Get()`
 - `PyConfig_GetInt()`
 - `PyConfig_Set()`
 - `PyConfig_Names()`

(Contributed by Victor Stinner in [gh-107954](#).)

- Add functions to configure the Python initialization (**PEP 741**):
 - `PyInitConfig_Create()`
 - `PyInitConfig_Free()`
 - `PyInitConfig_GetError()`
 - `PyInitConfig_GetExitCode()`
 - `PyInitConfig_HasOption()`
 - `PyInitConfig_GetInt()`
 - `PyInitConfig_GetStr()`
 - `PyInitConfig_GetStrList()`
 - `PyInitConfig_FreeStrList()`
 - `PyInitConfig_SetInt()`
 - `PyInitConfig_SetStr()`
 - `PyInitConfig_SetStrList()`
 - `PyInitConfig_AddModule()`
 - `Py_InitializeFromInitConfig()`

(Contributed by Victor Stinner in [gh-107954](#).)

11.2 Porting to Python 3.14

- In the limited C API 3.14 and newer, `Py_TYPE()` is now implemented as an opaque function call to hide implementation details. (Contributed by Victor Stinner in [gh-120600](#).)

11.3 Deprecated

- Macros `Py_IS_NAN`, `Py_IS_INFINITY` and `Py_IS_FINITE` are soft deprecated, use instead `isnan`, `isinf` and `isfinite` available from `math.h` since C99. (Contributed by Sergey B Kirpichev in [gh-119613](#).)
- `asyncio.iscoroutinefunction()` is deprecated and will be removed in Python 3.16, use `inspect.iscoroutinefunction()` instead. (Contributed by Jiahao Li and Kumar Aditya in [gh-122875](#).)

Pending Removal in Python 3.15

- The bundled copy of `libmpdecimal`.
- `PyImport_ImportModuleNoBlock()`: use `PyImport_ImportModule()` instead.
- `PyWeakref_GET_OBJECT()`: use `PyWeakref_GetRef()` instead.
- `PyWeakref_GetObject()`: use `PyWeakref_GetRef()` instead.
- `Py_UNICODE_WIDE` type: use `wchar_t` instead.
- `Py_UNICODE` type: use `wchar_t` instead.
- Python initialization functions:
 - `PySys_ResetWarnOptions()`: clear `sys.warnoptions` and `warnings.filters` instead.
 - `Py_GetExecPrefix()`: get `sys.exec_prefix` instead.
 - `Py_GetPath()`: get `sys.path` instead.
 - `Py_GetPrefix()`: get `sys.prefix` instead.
 - `Py_GetProgramFullPath()`: get `sys.executable` instead.
 - `Py_GetProgramName()`: get `sys.executable` instead.
 - `Py_GetPythonHome()`: get `PyConfig.home` or the `PYTHONHOME` environment variable instead.

Pending Removal in Future Versions

The following APIs are deprecated and will be removed, although there is currently no date scheduled for their removal.

- `Py_TPFLAGS_HAVE_FINALIZE`: unneeded since Python 3.8.
- `PyErr_Fetch()`: use `PyErr_GetRaisedException()` instead.
- `PyErr_NormalizeException()`: use `PyErr_GetRaisedException()` instead.
- `PyErr_Restore()`: use `PyErr_SetRaisedException()` instead.
- `PyModule_GetFilename()`: use `PyModule_GetFilenameObject()` instead.
- `PyOS_AfterFork()`: use `PyOS_AfterFork_Child()` instead.
- `PySlice_GetIndicesEx()`: use `PySlice_Unpack()` and `PySlice_AdjustIndices()` instead.
- `PyUnicode_AsDecodedObject()`: use `PyCodec_Decode()` instead.

- `PyUnicode_AsDecodedUnicode()`: use `PyCodec_Decode()` instead.
- `PyUnicode_AsEncodedObject()`: use `PyCodec_Encode()` instead.
- `PyUnicode_AsEncodedUnicode()`: use `PyCodec_Encode()` instead.
- `PyUnicode_READY()`: **unneeded since Python 3.12**
- `PyErr_Display()`: use `PyErr_DisplayException()` instead.
- `_PyErr_ChainExceptions()`: use `_PyErr_ChainExceptions1` instead.
- `PyBytesObject.ob_shash` member: call `PyObject_Hash()` instead.
- `PyDictObject.ma_version_tag` member.
- **Thread Local Storage (TLS) API:**
 - `PyThread_create_key()`: use `PyThread_tss_alloc()` instead.
 - `PyThread_delete_key()`: use `PyThread_tss_free()` instead.
 - `PyThread_set_key_value()`: use `PyThread_tss_set()` instead.
 - `PyThread_get_key_value()`: use `PyThread_tss_get()` instead.
 - `PyThread_delete_key_value()`: use `PyThread_tss_delete()` instead.
 - `PyThread_ReInitTLS()`: **unneeded since Python 3.7.**

11.4 Removed

- Creating `immutable` types with mutable bases was deprecated since 3.12 and now raises a `TypeError`.

Índice

P

Python Enhancement Proposals

PEP 741, [13](#)

PYTHONHOME, [14](#)

V

variables de entorno

PYTHONHOME, [14](#)