What's New in Python

Release 3.12.0a3

A. M. Kuchling

January 10, 2023

Python Software Foundation
Email: docs@python.org

Contents

Release 3.12.0a3

Date January 10, 2023

This article explains the new features in Python 3.12, compared to 3.11.

For full details, see the changelog.

Note: Prerelease users should be aware that this document is currently in draft form. It will be updated substantially as Python 3.12 moves towards release, so it's worth checking back even after reading earlier versions.

---

1 Summary – Release highlights

Important deprecations, removals or restrictions:

- **PEP 623**, Remove wstr from Unicode
- **PEP 632**, Remove the `distutils` package.

2 Improved Error Messages

- Modules from the standard library are now potentially suggested as part of the error messages displayed by the interpreter when a `NameError` is raised to the top level. Contributed by Pablo Galindo in gh-98254.

```
>>> sys.version_info
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'sys' is not defined. Did you forget to import 'sys'?
```

- Improve the error suggestion for `NameError` exceptions for instances. Now if a `NameError` is raised in a method and the instance has an attribute that's exactly equal to the name in the exception, the suggestion will include `self.<NAME>` instead of the closest match in the method scope. Contributed by Pablo Galindo in gh-99139.

```
>>> class A:
...     def __init__(self):
...         self.blech = 1
...
...     def foo(self):
...         somethin = blech
```

```
>>> A().foo()
  File "<stdin>", line 1
    somethin = blech
               ^^^^^
NameError: name 'blech' is not defined. Did you mean: 'self.blech'?
```

- Improve the `SyntaxError` error message when the user types `import x from y` instead of `from y import x`. Contributed by Pablo Galindo in gh-98931.

```
>>> import a.y.z from b.y.z
  File "<stdin>", line 1
    import a.y.z from b.y.z
    ^^^^^^^^^^^^^^^^^^^^^^
SyntaxError: Did you mean to use 'from ... import ...' instead?
```

- `ImportError` exceptions raised from failed `from <module> import <name>` statements now include suggestions for the value of `<name>` based on the available names in `<module>`. Contributed by Pablo Galindo in gh-91058.

```
>>> from collections import chainmap
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: cannot import name 'chainmap' from 'collections'. Did you mean:
↪'ChainMap'?
```

3 New Features

- Add perf_profiling through the new environment variable `PYTHONPERFSUPPORT`, the new command-line option `-X perf`, as well as the new `sys.activate_stack_trampoline()`, `sys.deactivate_stack_trampoline()`, and `sys.is_stack_trampoline_active()` APIs. (Design by Pablo Galindo. Contributed by Pablo Galindo and Christian Heimes with contributions from Gregory P. Smith [Google] and Mark Shannon in gh-96123.)

4 Other Language Changes

- `types.MappingProxyType` instances are now hashable if the underlying mapping is hashable. (Contributed by Serhiy Storchaka in gh-87995.)
- `memoryview` now supports the half-float type (the "e" format code). (Contributed by Dong-hee Na and Antoine Pitrou in gh-90751.)
- The parser now raises `SyntaxError` when parsing source code containing null bytes. (Contributed by Pablo Galindo in gh-96670.)
- `ast.parse()` now raises `SyntaxError` instead of `ValueError` when parsing source code containing null bytes. (Contributed by Pablo Galindo in gh-96670.)

- The Garbage Collector now runs only on the eval breaker mechanism of the Python bytecode evaluation loop instead on object allocations. The GC can also run when `PyErr_CheckSignals()` is called so C extensions that need to run for a long time without executing any Python code also have a chance to execute the GC periodically. (Contributed by Pablo Galindo in gh-97922.)
- A backslash-character pair that is not a valid escape sequence now generates a `SyntaxWarning`, instead of `DeprecationWarning`. For example, `re.compile("\d+\.\d+")` now emits a `SyntaxWarning` (`"\d"` is an invalid escape sequence), use raw strings for regular expression: `re.compile(r"\d+\.\d+")`. In a future Python version, `SyntaxError` will eventually be raised, instead of `SyntaxWarning`. (Contributed by Victor Stinner in gh-98401.)
- Octal escapes with value larger than `0o377` (ex: `"\477"`), deprecated in Python 3.11, now produce a `SyntaxWarning`, instead of `DeprecationWarning`. In a future Python version they will be eventually a `SyntaxError`. (Contributed by Victor Stinner in gh-98401.)
- All builtin and extension callables expecting boolean parameters now accept arguments of any type instead of just `bool` and `int`. (Contributed by Serhiy Storchaka in gh-60203.)
- Variables used in the target part of comprehensions that are not stored to can now be used in assignment expressions (`:=`). For example, in `[(b := 1) for a, b.prop in some_iter]`, the assignment to `b` is now allowed. Note that assigning to variables stored to in the target part of comprehensions (like `a`) is still disallowed, as per **PEP 572**. (Contributed by Nikita Sobolev in gh-100581.)

5 New Modules

- None yet.

6 Improved Modules

6.1 array

- The `array.array` class now supports subscripting, making it a generic type. (Contributed by Jelle Zijlstra in gh-98658.)

6.2 asyncio

- On Linux, `asyncio` uses `PidfdChildWatcher` by default if `os.pidfd_open()` is available and functional instead of `ThreadedChildWatcher`. (Contributed by Kumar Aditya in gh-98024.)
- The child watcher classes `MultiLoopChildWatcher`, `FastChildWatcher`, `AbstractChildWatcher` and `SafeChildWatcher` are deprecated and will be removed in Python 3.14. It is recommended to not manually configure a child watcher as the event loop now uses the best available child watcher for each platform (`PidfdChildWatcher` if supported and `ThreadedChildWatcher` otherwise). (Contributed by Kumar Aditya in gh-94597.)
- `asyncio.set_child_watcher()`, `asyncio.get_child_watcher()`, `asyncio.AbstractEventLoopPolicy.set_child_watcher()` and `asyncio.AbstractEventLoopPolicy.get_child_watcher()` are deprecated and will be removed in Python 3.14. (Contributed by Kumar Aditya in gh-94597.)
- Add *loop_factory* parameter to `asyncio.run()` to allow specifying a custom event loop factory. (Contributed by Kumar Aditya in gh-99388.)

- Add C implementation of `asyncio.current_task()` for 4x-6x speedup. (Contributed by Itamar Ostricher and Pranav Thulasiram Bhat in gh-100344.)

6.3 inspect

- Add `inspect.markcoroutinefunction()` to mark sync functions that return a coroutine for use with `iscoroutinefunction()`. (Contributed Carlton Gibson in gh-99247.)

6.4 pathlib

- Add `walk()` for walking the directory trees and generating all file or directory names within them, similar to `os.walk()`. (Contributed by Stanislav Zmiev in gh-90385.)
- Add *walk_up* optional parameter to `pathlib.PurePath.relative_to()` to allow the insertion of `..` entries in the result; this behavior is more consistent with `os.path.relpath()`. (Contributed by Domenico Ragusa in bpo-40358.)
- Add `pathlib.Path.is_junction()` as a proxy to `os.path.isjunction()`. (Contributed by Charles Machalow in gh-99547.)

6.5 dis

- Pseudo instruction opcodes (which are used by the compiler but do not appear in executable bytecode) are now exposed in the `dis` module. `HAVE_ARGUMENT` is still relevant to real opcodes, but it is not useful for pseudo instructions. Use the new `hasarg` collection instead. (Contributed by Irit Katriel in gh-94216.)

6.6 math

- Added `math.sumprod()` for computing a sum of products. (Contributed by Raymond Hettinger in gh-100485.)

6.7 os

- Add `os.PIDFD_NONBLOCK` to open a file descriptor for a process with `os.pidfd_open()` in non-blocking mode. (Contributed by Kumar Aditya in gh-93312.)
- Add `os.path.isjunction()` to check if a given path is a junction. (Contributed by Charles Machalow in gh-99547.)
- `os.DirEntry` now includes an `os.DirEntry.is_junction()` method to check if the entry is a junction. (Contributed by Charles Machalow in gh-99547.)

6.8 shutil

- `shutil.make_archive()` now passes the *root_dir* argument to custom archivers which support it. In this case it no longer temporarily changes the current working directory of the process to *root_dir* to perform archiving. (Contributed by Serhiy Storchaka in gh-74696.)

6.9 sqlite3

- Add a command-line interface. (Contributed by Erlend E. Aasland in gh-77617.)
- Add the `autocommit` attribute to `Connection` and the *autocommit* parameter to `connect()` to control **PEP 249**-compliant transaction handling. (Contributed by Erlend E. Aasland in gh-83638.)

6.10 threading

- Add `threading.settrace_all_threads()` and `threading.setprofile_all_threads()` that allow to set tracing and profiling functions in all running threads in addition to the calling one. (Contributed by Pablo Galindo in gh-93503.)

6.11 unicodedata

- The Unicode database has been updated to version 15.0.0. (Contributed by Benjamin Peterson in gh-96734).

6.12 tempfile

The `tempfile.NamedTemporaryFile` function has a new optional parameter *delete_on_close* (Contributed by Evgeny Zorin in gh-58451.)

6.13 sys

- Add `sys.activate_stack_trampoline()` and `sys.deactivate_stack_trampoline()` for activating and deactivating stack profiler trampolines, and `sys.is_stack_trampoline_active()` for querying if stack profiler trampolines are active. (Contributed by Pablo Galindo and Christian Heimes with contributions from Gregory P. Smith [Google] and Mark Shannon in gh-96123.)

7 Optimizations

- Removed `wstr` and `wstr_length` members from Unicode objects. It reduces object size by 8 or 16 bytes on 64bit platform. (**PEP 623**) (Contributed by Inada Naoki in gh-92536.)
- Added experimental support for using the BOLT binary optimizer in the build process, which improves performance by 1-5%. (Contributed by Kevin Modzelewski in gh-90536.)
- Speed up the regular expression substitution (functions `re.sub()` and `re.subn()` and corresponding `re.Pattern` methods) for replacement strings containing group references by 2–3 times. (Contributed by Serhiy Storchaka in gh-91524.)

8 CPython bytecode changes

- Removed the `LOAD_METHOD` instruction. It has been merged into `LOAD_ATTR`. `LOAD_ATTR` will now behave like the old `LOAD_METHOD` instruction if the low bit of its oparg is set. (Contributed by Ken Jin in gh-93429.)

9 Demos and Tools

- Remove the `Tools/demo/` directory which contained old demo scripts. A copy can be found in the old-demos project. (Contributed by Victor Stinner in gh-97681.)
- Remove outdated example scripts of the `Tools/scripts/` directory. A copy can be found in the old-demos project. (Contributed by Victor Stinner in gh-97669.)

10 Deprecated

- `typing.Hashable` and `typing.Sized` aliases for `collections.abc.Hashable` and `collections.abc.Sized`. (gh-94309.)
- The `sqlite3` default adapters and converters are now deprecated. Instead, use the sqlite3-adapter-converter-recipes and tailor them to your needs. (Contributed by Erlend E. Aasland in gh-90016.)
- The 3-arg signatures (type, value, traceback) of `throw()`, `throw()` and `athrow()` are deprecated and may be removed in a future version of Python. Use the single-arg versions of these functions instead. (Contributed by Ofey Chan in gh-89874.)
- `DeprecationWarning` is now raised when `__package__` on a module differs from `__spec__.parent` (previously it was `ImportWarning`). (Contributed by Brett Cannon in gh-65961.)

10.1 Pending Removal in Python 3.13

The following modules and APIs have been deprecated in earlier Python releases, and will be removed in Python 3.13.

Modules (see **PEP 594**):

- `aifc`
- `audioop`
- `cgi`
- `cgitb`
- `chunk`
- `crypt`
- `imghdr`
- `mailcap`
- `msilib`
- `nis`
- `nntplib`
- `ossaudiodev`

- `pipes`
- `sndhdr`
- `spwd`
- `sunau`
- `telnetlib`
- `uu`
- `xdrlib`

APIs:

- `configparser.LegacyInterpolation` (gh-90765)
- `locale.getdefaultlocale()` (gh-90817)
- `turtle.RawTurtle.settiltangle()` (gh-50096)
- `unittest.findTestCases()` (gh-50096)
- `unittest.makeSuite()` (gh-50096)
- `unittest.getTestCaseNames()` (gh-50096)
- `webbrowser.MacOSX` (gh-86421)

11 Pending Removal in Python 3.14

- Deprecated the following `importlib.abc` classes, scheduled for removal in Python 3.14:
  - `importlib.abc.ResourceReader`
  - `importlib.abc.Traversable`
  - `importlib.abc.TraversableResources`

  Use `importlib.resources.abc` classes instead:
  - `importlib.resources.abc.TraversableResources`
  - `importlib.resources.abc.Traversable`
  - `importlib.resources.abc.TraversableResources`

  (Contributed by Jason R. Coombs and Hugo van Kemenade in gh-93963.)
- Creating `immutable types` with mutable bases using the C API.
- `__package__` and `__cached__` will cease to be set or taken into consideration by the import system (gh-97879).

11.1 Pending Removal in Future Versions

The following APIs were deprecated in earlier Python versions and will be removed, although there is currently no date scheduled for their removal.

- `typing.Text` (gh-92332)
- 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)

12 Removed

- Remove the `distutils` package. It was deprecated in Python 3.10 by **PEP 632** "Deprecate distutils module". For projects still using `distutils` and cannot be updated to something else, the `setuptools` project can be installed: it still provides `distutils`. (Contributed by Victor Stinner in gh-92584.)
- Removed many old deprecated `unittest` features:
  - A number of `TestCase` method aliases:

Deprecated alias	Method Name	Deprecated in
`failUnless`	`assertTrue()`	3.1
`failIf`	`assertFalse()`	3.1
`failUnlessEqual`	`assertEqual()`	3.1
`failIfEqual`	`assertNotEqual()`	3.1
`failUnlessAlmostEqual`	`assertAlmostEqual()`	3.1
`failIfAlmostEqual`	`assertNotAlmostEqual()`	3.1
`failUnlessRaises`	`assertRaises()`	3.1
`assert_`	`assertTrue()`	3.2
`assertEquals`	`assertEqual()`	3.2
`assertNotEquals`	`assertNotEqual()`	3.2
`assertAlmostEquals`	`assertAlmostEqual()`	3.2
`assertNotAlmostEquals`	`assertNotAlmostEqual()`	3.2
`assertRegexpMatches`	`assertRegex()`	3.2
`assertRaisesRegexp`	`assertRaisesRegex()`	3.2
`assertNotRegexpMatches`	`assertNotRegex()`	3.5

    You can use https://github.com/isidentical/teyit to automatically modernise your unit tests.

  - Undocumented and broken `TestCase` method `assertDictContainsSubset` (deprecated in Python 3.2).
  - Undocumented `TestLoader.loadTestsFromModule` parameter *use_load_tests* (deprecated and ignored since Python 3.2).
  - An alias of the `TextTestResult` class: `_TextTestResult` (deprecated in Python 3.2).

  (Contributed by Serhiy Storchaka in bpo-45162.)
- Several names deprecated in the `configparser` way back in 3.2 have been removed per gh-89336:
  - `configparser.ParsingError` no longer has a `filename` attribute or argument. Use the `source` attribute and argument instead.

  - `configparser` no longer has a `SafeConfigParser` class. Use the shorter `ConfigParser` name instead.
  - `configparser.ConfigParser` no longer has a `readfp` method. Use `read_file()` instead.
- The following undocumented `sqlite3` features, deprecated in Python 3.10, are now removed:
  - `sqlite3.enable_shared_cache()`
  - `sqlite3.OptimizedUnicode`

  If a shared cache must be used, open the database in URI mode using the `cache=shared` query parameter.

  The `sqlite3.OptimizedUnicode` text factory has been an alias for `str` since Python 3.3. Code that previously set the text factory to `OptimizedUnicode` can either use `str` explicitly, or rely on the default value which is also `str`.

  (Contributed by Erlend E. Aasland in gh-92548.)
- `smtpd` has been removed according to the schedule in **PEP 594**, having been deprecated in Python 3.4.7 and 3.5.4. Use aiosmtpd PyPI module or any other `asyncio`-based server instead. (Contributed by Oleg Iarygin in gh-93243.)
- `asynchat` and `asyncore` have been removed according to the schedule in **PEP 594**, having been deprecated in Python 3.6. Use `asyncio` instead. (Contributed by Nikita Sobolev in gh-96580.)
- Remove `io.OpenWrapper` and `_pyio.OpenWrapper`, deprecated in Python 3.10: just use `open()` instead. The `open()` (`io.open()`) function is a built-in function. Since Python 3.10, `_pyio.open()` is also a static method. (Contributed by Victor Stinner in gh-94169.)
- Remove the `ssl.RAND_pseudo_bytes()` function, deprecated in Python 3.6: use `os.urandom()` or `ssl.RAND_bytes()` instead. (Contributed by Victor Stinner in gh-94199.)
- `gzip`: Remove the `filename` attribute of `gzip.GzipFile`, deprecated since Python 2.6, use the `name` attribute instead. In write mode, the `filename` attribute added `'.gz'` file extension if it was not present. (Contributed by Victor Stinner in gh-94196.)
- Remove the `ssl.match_hostname()` function. The `ssl.match_hostname()` was deprecated in Python 3.7. OpenSSL performs hostname matching since Python 3.7, Python no longer uses the `ssl.match_hostname()` function. (Contributed by Victor Stinner in gh-94199.)
- Remove the `locale.format()` function, deprecated in Python 3.7: use `locale.format_string()` instead. (Contributed by Victor Stinner in gh-94226.)
- `hashlib`: Remove the pure Python implementation of `hashlib.pbkdf2_hmac()`, deprecated in Python 3.10. Python 3.10 and newer requires OpenSSL 1.1.1 (**PEP 644**): this OpenSSL version provides a C implementation of `pbkdf2_hmac()` which is faster. (Contributed by Victor Stinner in gh-94199.)
- `xml.etree`: Remove the `ElementTree.Element.copy()` method of the pure Python implementation, deprecated in Python 3.10, use the `copy.copy()` function instead. The C implementation of `xml.etree` has no `copy()` method, only a `__copy__()` method. (Contributed by Victor Stinner in gh-94383.)
- `zipimport`: Remove `find_loader()` and `find_module()` methods, deprecated in Python 3.10: use the `find_spec()` method instead. See **PEP 451** for the rationale. (Contributed by Victor Stinner in gh-94379.)
- Remove the `ssl.wrap_socket()` function, deprecated in Python 3.7: instead, create a `ssl.SSLContext` object and call its `ssl.SSLContext.wrap_socket` method. Any package that still uses `ssl.wrap_socket()` is broken and insecure. The function neither sends a SNI TLS extension nor validates server hostname. Code is subject to CWE-295: Improper Certificate Validation. (Contributed by Victor Stinner in gh-94199.)
- Many previously deprecated cleanups in `importlib` have now been completed:
  - References to, and support for `module_repr()` has been eradicated.

- `importlib.util.set_package` has been removed. (Contributed by Brett Cannon in gh-65961.)
- Removed the `suspicious` rule from the documentation Makefile, and removed `Doc/tools/rstlint.py`, both in favor of sphinx-lint. (Contributed by Julien Palard in gh-98179.)
- Remove the *keyfile* and *certfile* parameters from the `ftplib`, `imaplib`, `poplib` and `smtplib` modules, and the *key_file*, *cert_file* and *check_hostname* parameters from the `http.client` module, all deprecated since Python 3.6. Use the *context* parameter (*ssl_context* in `imaplib`) instead. (Contributed by Victor Stinner in gh-94172.)
- `ftplib`: Remove the `FTP_TLS.ssl_version` class attribute: use the *context* parameter instead. (Contributed by Victor Stinner in gh-94172.)

13 Porting to Python 3.12

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

13.1 Changes in the Python API

- 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.)
- Removed randrange() functionality deprecated since Python 3.10. Formerly, randrange(10.0) losslessly converted to randrange(10). Now, it raises a TypeError. Also, the exception raised for non-integral values such as randrange(10.5) or randrange('10') has been changed from ValueError to TypeError. This also prevents bugs where `randrange(1e25)` would silently select from a larger range than `randrange(10**25)`. (Originally suggested by Serhiy Storchaka gh-86388.)
- `argparse.ArgumentParser` changed encoding and error handler for reading arguments from file (e.g. `fromfile_prefix_chars` option) from default text encoding (e.g. `locale.getpreferredencoding(False)`) to filesystem encoding and error handler. Argument files should be encoded in UTF-8 instead of ANSI Codepage on Windows.
- Removed the `asyncore`-based `smtpd` module deprecated in Python 3.4.7 and 3.5.4. A recommended replacement is the `asyncio`-based aiosmtpd PyPI module.
- `shlex.split()`: Passing `None` for *s* argument now raises an exception, rather than reading `sys.stdin`. The feature was deprecated in Python 3.9. (Contributed by Victor Stinner in gh-94352.)
- The `os` module no longer accepts bytes-like paths, like `bytearray` and `memoryview` types: only the exact `bytes` type is accepted for bytes strings. (Contributed by Victor Stinner in gh-98393.)
- `syslog.openlog()` and `syslog.closelog()` now fail if used in subinterpreters. `syslog.syslog()` may still be used in subinterpreters, but now only if `syslog.openlog()` has already been called in the main interpreter. These new restrictions do not apply to the main interpreter, so only a very small set of users might be affected. This change helps with interpreter isolation. Furthermore, `syslog` is a wrapper around process-global resources, which are best managed from the main interpreter. (Contributed by Dong-hee Na in gh-99127.)
- `asyncio.get_event_loop()` and many other `asyncio` functions like `ensure_future()`, `shield()` or `gather()`, and also the `get_event_loop()` method of `BaseDefaultEventLoopPolicy` now raise a `RuntimeError` if called when there is no running event loop and the current event loop was not set. Previously they implicitly created and set a new current event loop. `DeprecationWarning` is no longer emitted if there is no running event loop but the current event loop is set in the policy. (Contributed by Serhiy Storchaka in gh-93453.)

14 Build Changes

- Python no longer uses `setup.py` to build shared C extension modules. Build parameters like headers and libraries are detected in `configure` script. Extensions are built by `Makefile`. Most extensions use `pkg-config` and fall back to manual detection. (Contributed by Christian Heimes in gh-93939.)
- `va_start()` with two parameters, like `va_start(args, format)`, is now required to build Python. `va_start()` is no longer called with a single parameter. (Contributed by Kumar Aditya in gh-93207.)
- CPython now uses the ThinLTO option as the default link time optimization policy if the Clang compiler accepts the flag. (Contributed by Dong-hee Na in gh-89536.)
- Add `COMPILEALL_OPTS` variable in Makefile to override `compileall` options (default: `-j0`) in `make install`. Also merged the 3 `compileall` commands into a single command to build .pyc files for all optimization levels (0, 1, 2) at once. (Contributed by Victor Stinner in gh-99289.)

15 C API Changes

15.1 New Features

- Added the new limited C API function `PyType_FromMetaclass()`, which generalizes the existing `PyType_FromModuleAndSpec()` using an additional metaclass argument. (Contributed by Wenzel Jakob in gh-93012.)
- API for creating objects that can be called using the vectorcall protocol was added to the Limited API:
  - `Py_TPFLAGS_HAVE_VECTORCALL`
  - `PyVectorcall_NARGS()`
  - `PyVectorcall_Call()`
  - `vectorcallfunc`

  The `Py_TPFLAGS_HAVE_VECTORCALL` flag is now removed from a class when the class's `__call__()` method is reassigned. This makes vectorcall safe to use with mutable types (i.e. heap types without the `immutable` flag). Mutable types that do not override `tp_call` now inherit the `Py_TPFLAGS_HAVE_VECTORCALL` flag. (Contributed by Petr Viktorin in gh-93274.)

  The `Py_TPFLAGS_MANAGED_DICT` and `Py_TPFLAGS_MANAGED_WEAKREF` flags have been added. This allows extensions classes to support object `__dict__` and weakrefs with less bookkeeping, using less memory and with faster access.
- API for performing calls using the vectorcall protocol was added to the Limited API:
  - `PyObject_Vectorcall()`
  - `PyObject_VectorcallMethod()`
  - `PY_VECTORCALL_ARGUMENTS_OFFSET`

  This means that both the incoming and outgoing ends of the vector call protocol are now available in the Limited API. (Contributed by Wenzel Jakob in gh-98586.)
- Added two new public functions, `PyEval_SetProfileAllThreads()` and `PyEval_SetTraceAllThreads()`, that allow to set tracing and profiling functions in all running threads in addition to the calling one. (Contributed by Pablo Galindo in gh-93503.)
- Added new function `PyFunction_SetVectorcall()` to the C API which sets the vectorcall field of a given `PyFunctionObject`. (Contributed by Andrew Frost in gh-92257.)

- The C API now permits registering callbacks via `PyDict_AddWatcher()`, `PyDict_AddWatch()` and related APIs to be called whenever a dictionary is modified. This is intended for use by optimizing interpreters, JIT compilers, or debuggers. (Contributed by Carl Meyer in gh-91052.)
- Added `PyType_AddWatcher()` and `PyType_Watch()` API to register callbacks to receive notification on changes to a type. (Contributed by Carl Meyer in gh-91051.)
- Added `PyCode_AddWatcher()` and `PyCode_ClearWatcher()` APIs to register callbacks to receive notification on creation and destruction of code objects. (Contributed by Itamar Ostricher in gh-91054.)
- Add `PyFrame_GetVar()` and `PyFrame_GetVarString()` functions to get a frame variable by its name. (Contributed by Victor Stinner in gh-91248.)

15.2 Porting to Python 3.12

- Legacy Unicode APIs based on `Py_UNICODE*` representation has been removed. Please migrate to APIs based on UTF-8 or `wchar_t*`.
- Argument parsing functions like `PyArg_ParseTuple()` doesn't support `Py_UNICODE*` based format (e.g. `u`, `Z`) anymore. Please migrate to other formats for Unicode like `s`, `z`, `es`, and `U`.
- `tp_weaklist` for all static builtin types is always `NULL`. This is an internal-only field on `PyTypeObject` but we're pointing out the change in case someone happens to be accessing the field directly anyway. To avoid breakage, consider using the existing public C-API instead, or, if necessary, the (internal-only) `_PyObject_GET_WEAKREFS_LISTPTR()` macro.
- This internal-only `PyTypeObject.tp_subclasses` may now not be a valid object pointer. Its type was changed to `void*` to reflect this. We mention this in case someone happens to be accessing the internal-only field directly.

  To get a list of subclasses, call the Python method `__subclasses__()` (using `PyObject_CallMethod()`, for example).
- An unrecognized format character in `PyUnicode_FromFormat()` and `PyUnicode_FromFormatV()` now sets a `SystemError`. In previous versions it caused all the rest of the format string to be copied as-is to the result string, and any extra arguments discarded. (Contributed by Serhiy Storchaka in gh-95781.)
- Fixed wrong sign placement in `PyUnicode_FromFormat()` and `PyUnicode_FromFormatV()`. (Contributed by Philip Georgi in gh-95504.)
- Extension classes wanting to add a `__dict__` or weak reference slot should use `Py_TPFLAGS_MANAGED_DICT` and `Py_TPFLAGS_MANAGED_WEAKREF` instead of `tp_dictoffset` and `tp_weaklistoffset`, respectively. The use of `tp_dictoffset` and `tp_weaklistoffset` is still supported, but does not fully support multiple inheritance (gh-95589), and performance may be worse. Classes declaring `Py_TPFLAGS_MANAGED_DICT` should call `_PyObject_VisitManagedDict()` and `_PyObject_ClearManagedDict()` to traverse and clear their instance's dictionaries. To clear weakrefs, call `PyObject_ClearWeakRefs()`, as before.
- The `PyUnicode_FSDecoder()` function no longer accepts bytes-like paths, like `bytearray` and `memoryview` types: only the exact `bytes` type is accepted for bytes strings. (Contributed by Victor Stinner in gh-98393.)
- The `Py_CLEAR`, `Py_SETREF` and `Py_XSETREF` macros now only evaluate their arguments once. If an argument has side effects, these side effects are no longer duplicated. (Contributed by Victor Stinner in gh-98724.)

15.3 Deprecated

- Deprecate global configuration variable:
  - `Py_DebugFlag`: use `PyConfig.parser_debug`
  - `Py_VerboseFlag`: use `PyConfig.verbose`
  - `Py_QuietFlag`: use `PyConfig.quiet`
  - `Py_InteractiveFlag`: use `PyConfig.interactive`
  - `Py_InspectFlag`: use `PyConfig.inspect`
  - `Py_OptimizeFlag`: use `PyConfig.optimization_level`
  - `Py_NoSiteFlag`: use `PyConfig.site_import`
  - `Py_BytesWarningFlag`: use `PyConfig.bytes_warning`
  - `Py_FrozenFlag`: use `PyConfig.pathconfig_warnings`
  - `Py_IgnoreEnvironmentFlag`: use `PyConfig.use_environment`
  - `Py_DontWriteBytecodeFlag`: use `PyConfig.write_bytecode`
  - `Py_NoUserSiteDirectory`: use `PyConfig.user_site_directory`
  - `Py_UnbufferedStdioFlag`: use `PyConfig.buffered_stdio`
  - `Py_HashRandomizationFlag`: use `PyConfig.use_hash_seed` and `PyConfig.hash_seed`
  - `Py_IsolatedFlag`: use `PyConfig.isolated`
  - `Py_LegacyWindowsFSEncodingFlag`: use `PyConfig.legacy_windows_fs_encoding`
  - `Py_LegacyWindowsStdioFlag`: use `PyConfig.legacy_windows_stdio`
  - `Py_FileSystemDefaultEncoding`: use `PyConfig.filesystem_encoding`
  - `Py_FileSystemDefaultEncodeErrors`: use `PyConfig.filesystem_errors`
  - `Py_UTF8Mode`: use `PyPreConfig.utf8_mode` (see `Py_PreInitialize()`)

  The `Py_InitializeFromConfig()` API should be used with `PyConfig` instead. (Contributed by Victor Stinner in gh-77782.)
- Creating `immutable types` with mutable bases is deprecated and will be disabled in Python 3.14.
- The `structmember.h` header is deprecated, though it continues to be available and there are no plans to remove it.

  Its contents are now available just by including `Python.h`, with a `Py` prefix added if it was missing:
  - `PyMemberDef`, `PyMember_GetOne()` and `PyMember_SetOne()`
  - Type macros like `Py_T_INT`, `Py_T_DOUBLE`, etc. (previously `T_INT`, `T_DOUBLE`, etc.)
  - The flags `Py_READONLY` (previously `READONLY`) and `Py_AUDIT_READ` (previously all uppercase)

  Several items are not exposed from `Python.h`:
  - `T_OBJECT` (use `Py_T_OBJECT_EX`)
  - `T_NONE` (previously undocumented, and pretty quirky)
  - The macro `WRITE_RESTRICTED` which does nothing.
  - The macros `RESTRICTED` and `READ_RESTRICTED`, equivalents of `Py_AUDIT_READ`.

  - In some configurations, <stddef.h> is not included from Python.h. It should be included manually when using offsetof().

  The deprecated header continues to provide its original contents under the original names. Your old code can stay unchanged, unless the extra include and non-namespaced macros bother you greatly.

  (Contributed in gh-47146 by Petr Viktorin, based on earlier work by Alexander Belopolsky and Matthias Braun.)

15.4 Removed

- Remove the token.h header file. There was never any public tokenizer C API. The token.h header file was only designed to be used by Python internals. (Contributed by Victor Stinner in gh-92651.)
- Legacy Unicode APIs has been removed. See **PEP 623** for detail.
  - PyUnicode_WCHAR_KIND
  - PyUnicode_AS_UNICODE()
  - PyUnicode_AsUnicode()
  - PyUnicode_AsUnicodeAndSize()
  - PyUnicode_AS_DATA()
  - PyUnicode_FromUnicode()
  - PyUnicode_GET_SIZE()
  - PyUnicode_GetSize()
  - PyUnicode_GET_DATA_SIZE()
- Remove the PyUnicode_InternImmortal() function and the SSTATE_INTERNED_IMMORTAL macro. (Contributed by Victor Stinner in gh-85858.)
- Remove Jython compatibility hacks from several stdlib modules and tests. (Contributed by Nikita Sobolev in gh-99482.)
- Remove _use_broken_old_ctypes_structure_semantics_ flag from ctypes module. (Contributed by Nikita Sobolev in gh-99285.)

Index

E

P