What's New in Python

Release 3.13.0a0

A. M. Kuchling

October 12, 2023

Python Software Foundation
Email: docs@python.org

Contents

Editor
TBD

This article explains the new features in Python 3.13, compared to 3.12.

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 3.13 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

- Allow the *count* argument of `str.replace()` to be a keyword. (Contributed by Hugo van Kemenade in gh-106487.)
- Compiler now strip indents from docstrings. This will reduce the size of bytecode cache (e.g. `.pyc` file). For example, cache file size for `sqlalchemy.orm.session` in SQLAlchemy 2.0 is reduced by about 5%. This change will affect tools using docstrings, like `doctest`. (Contributed by Inada Naoki in gh-81283.)
- The `compile()` built-in can now accept a new flag, `ast.PyCF_OPTIMIZED_AST`, which is similar to `ast.PyCF_ONLY_AST` except that the returned `AST` is optimized according to the value of the `optimize` argument. (Contributed by Irit Katriel in gh-108113).
- `multiprocessing`, `concurrent.futures`, `compileall`: Replace `os.cpu_count()` with `os.process_cpu_count()` to select the default number of worker threads and processes. Get the CPU affinity if supported. (Contributed by Victor Stinner in gh-109649.)
- `os.path.realpath()` now resolves MS-DOS style file names even if the file is not accessible. (Contributed by Moonsik Park in gh-82367.)

4 New Modules

- None yet.

5 Improved Modules

5.1 ast

- `ast.parse()` now accepts an optional argument `optimize` which is passed on to the `compile()` built-in. This makes it possible to obtain an optimized `AST`. (Contributed by Irit Katriel in gh-108113).

5.2 array

- Add `'w'` type code (`Py_UCS4`) that can be used for Unicode strings. It can be used instead of `'u'` type code, which is deprecated. (Contributed by Inada Naoki in gh-80480.)

5.3 copy

- Add `copy.replace()` function which allows to create a modified copy of an object, which is especially useful for immutable objects. It supports named tuples created with the factory function `collections.namedtuple()`, `dataclass` instances, various `datetime` objects, `Signature` objects, `Parameter` objects, code object, and any user classes which define the `__replace__()` method. (Contributed by Serhiy Storchaka in gh-108751.)

5.4 dbm

- Add `dbm.gnu.gdbm.clear()` and `dbm.ndbm.ndbm.clear()` methods that remove all items from the database. (Contributed by Donghee Na in gh-107122.)

5.5 doctest

- The `doctest.DocTestRunner.run()` method now counts the number of skipped tests. Add `doctest.DocTestRunner.skips` and `doctest.TestResults.skipped` attributes. (Contributed by Victor Stinner in gh-108794.)

5.6 io

The `io.IOBase` finalizer now logs the `close()` method errors with `sys.unraisablehook`. Previously, errors were ignored silently by default, and only logged in Python Development Mode or on Python built on debug mode. (Contributed by Victor Stinner in gh-62948.)

5.7 opcode

- Move `opcode.ENABLE_SPECIALIZATION` to `_opcode.ENABLE_SPECIALIZATION`. This field was added in 3.12, it was never documented and is not intended for external usage. (Contributed by Irit Katriel in gh-105481.)
- Removed `opcode.is_pseudo`, `opcode.MIN_PSEUDO_OPCODE` and `opcode.MAX_PSEUDO_OPCODE`, which were added in 3.12, were never documented or exposed through `dis`, and were not intended to be used externally.

5.8 os

- Add `os.process_cpu_count()` function to get the number of logical CPUs usable by the calling thread of the current process. (Contributed by Victor Stinner in gh-109649.)
- Add a low level interface for Linux's timer notification file descriptors via `os.timerfd_create()`, `os.timerfd_settime()`, `os.timerfd_settime_ns()`, `os.timerfd_gettime()`, and `os.timerfd_gettime_ns()`, `os.TFD_NONBLOCK`, `os.TFD_CLOEXEC`, `os.TFD_TIMER_ABSTIME`, and `os.TFD_TIMER_CANCEL_ON_SET` (Contributed by Masaru Tsuchiyama in gh-108277.)
- `os.cpu_count()` and `os.process_cpu_count()` can be overridden through the new environment variable `PYTHON_CPU_COUNT` or the new command-line option `-X cpu_count`. This option is useful for users who need to limit CPU resources of a container system without having to modify the container (application code). (Contributed by Donghee Na in gh-109595)

5.9 pathlib

- Add `pathlib.UnsupportedOperation`, which is raised instead of `NotImplementedError` when a path operation isn't supported. (Contributed by Barney Gale in gh-89812.)
- Add `pathlib.Path.from_uri()`, a new constructor to create a `pathlib.Path` object from a 'file' URI (`file:/`). (Contributed by Barney Gale in gh-107465.)
- Add support for recursive wildcards in `pathlib.PurePath.match()`. (Contributed by Barney Gale in gh-73435.)
- Add *follow_symlinks* keyword-only argument to `pathlib.Path.glob()`, `rglob()`, `is_file()`, and `is_dir()`. (Contributed by Barney Gale in gh-77609 and gh-105793.)

5.10 pdb

- Add ability to move between chained exceptions during post mortem debugging in `pm()` using the new `exceptions [exc_number]` command for Pdb. (Contributed by Matthias Bussonnier in gh-106676.)
- Expressions/Statements whose prefix is a pdb command are now correctly identified and executed. (Contributed by Tian Gao in gh-108464.)

5.11 sqlite3

- A `ResourceWarning` is now emitted if a `sqlite3.Connection` object is not `closed` explicitly. (Contributed by Erlend E. Aasland in gh-105539.)

5.12 tkinter

- Add `tkinter` widget methods: `tk_busy_hold()`, `tk_busy_configure()`, `tk_busy_cget()`, `tk_busy_forget()`, `tk_busy_current()`, and `tk_busy_status()`. (Contributed by Miguel, klappnase and Serhiy Storchaka in gh-72684.)

5.13 traceback

- Add *show_group* parameter to `traceback.TracebackException.format_exception_only()` to format the nested exceptions of a `BaseExceptionGroup` instance, recursively. (Contributed by Irit Katriel in gh-105292.)

5.14 typing

- Add `typing.get_protocol_members()` to return the set of members defining a `typing.Protocol`. Add `typing.is_protocol()` to check whether a class is a `typing.Protocol`. (Contributed by Jelle Zijlstra in gh-104873.)

5.15 venv

- Add support for adding source control management (SCM) ignore files to a virtual environment's directory. By default, Git is supported. This is implemented as opt-in via the API which can be extended to support other SCMs (`venv.EnvBuilder` and `venv.create()`), and opt-out via the CLI (using `--without-scm-ignore-files`). (Contributed by Brett Cannon in gh-108125.)

6 Optimizations

- `textwrap.indent()` is now ~30% faster than before for large input. (Contributed by Inada Naoki in gh-107369.)

7 Deprecated

- `array`: `array`'s `'u'` format code, deprecated in docs since Python 3.3, emits `DeprecationWarning` since 3.13 and will be removed in Python 3.16. Use the `'w'` format code instead. (contributed by Hugo van Kemenade in gh-80480)
- `ctypes`: Deprecate undocumented `ctypes.SetPointerType()` and `ctypes.ARRAY()` functions. Replace `ctypes.ARRAY(item_type, size)` with `item_type * size`. (Contributed by Victor Stinner in gh-105733.)
- `getopt` and `optparse` modules: They are now soft deprecated: the `argparse` should be used for new projects. Previously, the `optparse` module was already deprecated, its removal was not scheduled, and no warnings was emitted: so there is no change in practice. (Contributed by Victor Stinner in gh-106535.)

- `http.server`: `http.server.CGIHTTPRequestHandler` now emits a `DeprecationWarning` as it will be removed in 3.15. Process based CGI http servers have been out of favor for a very long time. This code was outdated, unmaintained, and rarely used. It has a high potential for both security and functionality bugs. This includes removal of the `--cgi` flag to the `python -m http.server` command line in 3.15.
- `typing`:
  - Creating a `typing.NamedTuple` class using keyword arguments to denote the fields (`NT = NamedTuple("NT", x=int, y=int)`) is deprecated, and will be disallowed in Python 3.15. Use the class-based syntax or the functional syntax instead. (Contributed by Alex Waygood in gh-105566.)
  - When using the functional syntax to create a `typing.NamedTuple` class or a `typing.TypedDict` class, failing to pass a value to the 'fields' parameter (`NT = NamedTuple("NT")` or `TD = TypedDict("TD")`) is deprecated. Passing `None` to the 'fields' parameter (`NT = NamedTuple("NT", None)` or `TD = TypedDict("TD", 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", [])`. To create a TypedDict class with 0 fields, use `class TD(TypedDict): pass` or `TD = TypedDict("TD", {})`. (Contributed by Alex Waygood in gh-105566 and gh-105570.)
  - `typing.no_type_check_decorator()` is deprecated, and scheduled for removal in Python 3.15. After eight years in the `typing` module, it has yet to be supported by any major type checkers. (Contributed by Alex Waygood in gh-106309.)
  - `typing.AnyStr` is deprecated. In Python 3.16, it will be removed from `typing.__all__`, and a `DeprecationWarning` will be emitted when it is imported or accessed. It will be removed entirely in Python 3.18. Use the new type parameter syntax instead. (Contributed by Michael The in gh-107116.)
- `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.)
- Passing more than one positional argument to `sqlite3.connect()` and the `sqlite3.Connection` constructor is deprecated. The remaining parameters will become keyword-only in Python 3.15.

  Deprecate passing name, number of arguments, and the callable as keyword arguments, for the following `sqlite3.Connection` APIs:

  - `create_function()`
  - `create_aggregate()`

  Deprecate passing the callback callable by keyword for the following `sqlite3.Connection` APIs:

  - `set_authorizer()`
  - `set_progress_handler()`
  - `set_trace_callback()`

  The affected parameters will become positional-only in Python 3.15.

  (Contributed by Erlend E. Aasland in gh-107948 and gh-108278.)
- The `dis.HAVE_ARGUMENT` separator is deprecated. Check membership in `hasarg` instead. (Contributed by Irit Katriel in gh-109319.)
- Deprecate non-standard format specifier "N" for `decimal.Decimal`. It was not documented and only supported in the C implementation. (Contributed by Serhiy Storchaka in gh-89902.)

7.1 Pending Removal in Python 3.14

- `argparse`: The *type*, *choices*, and *metavar* parameters of `argparse.BooleanOptionalAction` are deprecated and will be removed in 3.14. (Contributed by Nikita Sobolev in gh-92248.)
- `ast`: The following features have been deprecated in documentation since Python 3.8, now cause a `DeprecationWarning` to be emitted at runtime when they are accessed or used, and will be removed in Python 3.14:
  - `ast.Num`
  - `ast.Str`
  - `ast.Bytes`
  - `ast.NameConstant`
  - `ast.Ellipsis`

  Use `ast.Constant` instead. (Contributed by Serhiy Storchaka in gh-90953.)
- `collections.abc`: Deprecated `ByteString`. Prefer `Sequence` or `Buffer`. For use in typing, prefer a union, like `bytes | bytearray`, or `collections.abc.Buffer`. (Contributed by Shantanu Jain in gh-91896.)
- `email`: Deprecated the *isdst* parameter in `email.utils.localtime()`. (Contributed by Alan Williams in gh-72346.)
- `importlib`: `__package__` and `__cached__` will cease to be set or taken into consideration by the import system (gh-97879).
- `importlib.abc` deprecated 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.)
- `itertools` had undocumented, inefficient, historically buggy, and inconsistent support for copy, deepcopy, and pickle operations. This will be removed in 3.14 for a significant reduction in code volume and maintenance burden. (Contributed by Raymond Hettinger in gh-101588.)
- `multiprocessing`: The default start method will change to a safer one on Linux, BSDs, and other non-macOS POSIX platforms where `'fork'` is currently the default (gh-84559). Adding a runtime warning about this was deemed too disruptive as the majority of code is not expected to care. Use the `get_context()` or `set_start_method()` APIs to explicitly specify when your code *requires* `'fork'`. See multiprocessing-start-methods.
- `pathlib`: `is_relative_to()`, `relative_to()`: passing additional arguments is deprecated.
- `pkgutil.find_loader()` and `pkgutil.get_loader()` now raise `DeprecationWarning`; use `importlib.util.find_spec()` instead. (Contributed by Nikita Sobolev in gh-97850.)
- `pty`:
  - `master_open()`: use `pty.openpty()`.
  - `slave_open()`: use `pty.openpty()`.
- `shutil.rmtree()` *onerror* parameter is deprecated in 3.12, and will be removed in 3.14: use the *onexc* parameter instead.

- `sqlite3`:
  - `version` and `version_info`.
  - `execute()` and `executemany()` if named placeholders are used and *parameters* is a sequence instead of a `dict`.
  - date and datetime adapter, date and timestamp converter: see the `sqlite3` documentation for suggested replacement recipes.
- `types.CodeType`: Accessing `co_lnotab` was deprecated in **PEP 626** since 3.10 and was planned to be removed in 3.12, but it only got a proper `DeprecationWarning` in 3.12. May be removed in 3.14. (Contributed by Nikita Sobolev in gh-101866.)
- `typing`: `ByteString`, deprecated since Python 3.9, now causes a `DeprecationWarning` to be emitted when it is used.
- `urllib.parse.Quoter` is deprecated: it was not intended to be a public API. (Contributed by Gregory P. Smith in gh-88168.)
- `xml.etree.ElementTree`: Testing the truth value of an `Element` is deprecated and will raise an exception in Python 3.14.

7.2 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.
- `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.)
- 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.)

7.3 Pending Removal in Python 3.16

- `array.array 'u'` type (`wchar_t`): use the `'w'` type instead (`Py_UCS4`).

7.4 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.

- `argparse`: Nesting argument groups and nesting mutually exclusive groups are deprecated.
- `builtins`:
  - `~bool`, bitwise inversion on bool.
  - `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.
- `calendar`: `calendar.January` and `calendar.February` constants are deprecated and replaced by `calendar.JANUARY` and `calendar.FEBRUARY`. (Contributed by Prince Roshan in gh-103636.)
- `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.
- `importlib.resources`: First parameter to files is renamed to 'anchor'.
- `importlib.resources` deprecated methods:
  - `contents()`
  - `is_resource()`

  - `open_binary()`
  - `open_text()`
  - `path()`
  - `read_binary()`
  - `read_text()`

  Use `files()` instead. Refer to importlib-resources: Migrating from Legacy for migration advice.

- `locale.getdefaultlocale()`: use `locale.setlocale()`, `locale.getencoding()` and `locale.getlocale()` instead (gh-90817)
- `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.)
- `ssl` options and protocols:
  - `ssl.SSLContext` without protocol argument is deprecated.
  - `ssl.SSLContext`: `set_npn_protocols()` and `~ssl.SSLContext.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`
- `sre_compile`, `sre_constants` and `sre_parse` modules.
- `types.CodeType.co_lnotab`: use the `co_lines` attribute instead.
- `typing.Text` (gh-92332).
- `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()`.
- `unittest.IsolatedAsyncioTestCase`: it is deprecated to return a value that is not None from a test case.
- `urllib.request`: `URLopener` and `FancyURLopener` style of invoking requests is deprecated. Use newer `urlopen()` functions and methods.
- `urllib.parse.to_bytes()`.
- `urllib.parse` deprecated functions: `urlparse()` instead
  - `splitattr()`
  - `splithost()`
  - `splitnport()`
  - `splitpasswd()`
  - `splitport()`
  - `splitquery()`
  - `splittag()`
  - `splittype()`
  - `splituser()`
  - `splitvalue()`
- `wsgiref`: `SimpleHandler.stdout.write()` should not do partial writes.
- `zipimport.zipimporter.load_module()` is deprecated: use `exec_module()` instead.

8 Removed

- **PEP 594**: Remove the `telnetlib` module, deprecated in Python 3.11: use the projects telnetlib3 or Exscript instead. (Contributed by Victor Stinner in gh-104773.)
- Remove the `2to3` program and the `lib2to3` module, deprecated in Python 3.11. (Contributed by Victor Stinner in gh-104780.)
- Namespaces `typing.io` and `typing.re`, deprecated in Python 3.8, are now removed. The items in those namespaces can be imported directly from `typing`. (Contributed by Sebastian Rittau in gh-92871.)
- Remove the untested and undocumented `webbrowser` `MacOSX` class, deprecated in Python 3.11. Use the `MacOSXOSAScript` class (introduced in Python 3.2) instead. (Contributed by Hugo van Kemenade in gh-104804.)
- Remove support for using `pathlib.Path` objects as context managers. This functionality was deprecated and made a no-op in Python 3.9.
- Remove the undocumented `configparser.LegacyInterpolation` class, deprecated in the docstring since Python 3.2, and with a deprecation warning since Python 3.11. (Contributed by Hugo van Kemenade in gh-104886.)
- Remove the `turtle.RawTurtle.settiltangle()` method, deprecated in docs since Python 3.1 and with a deprecation warning since Python 3.11. (Contributed by Hugo van Kemenade in gh-104876.)
- Removed the following `unittest` functions, deprecated in Python 3.11:
  - `unittest.findTestCases()`
  - `unittest.makeSuite()`
  - `unittest.getTestCaseNames()`

  Use `TestLoader` methods instead:

- `unittest.TestLoader.loadTestsFromModule()`
- `unittest.TestLoader.loadTestsFromTestCase()`
- `unittest.TestLoader.getTestCaseNames()`

(Contributed by Hugo van Kemenade in gh-104835.)

- **PEP 594**: Remove the `cgi` and `cgitb` modules, deprecated in Python 3.11.
  - `cgi.FieldStorage` can typically be replaced with `urllib.parse.parse_qsl()` for `GET` and `HEAD` requests, and the `email.message` module or multipart PyPI project for `POST` and `PUT`.
  - `cgi.parse()` can be replaced by calling `urllib.parse.parse_qs()` directly on the desired query string, except for `multipart/form-data` input, which can be handled as described for `cgi.parse_multipart()`.
  - `cgi.parse_multipart()` can be replaced with the functionality in the `email` package (e.g. `email.message.EmailMessage` and `email.message.Message`) which implements the same MIME RFCs, or with the multipart PyPI project.
  - `cgi.parse_header()` can be replaced with the functionality in the `email` package, which implements the same MIME RFCs. For example, with `email.message.EmailMessage`:

```
from email.message import EmailMessage
msg = EmailMessage()
msg['content-type'] = 'application/json; charset="utf8"'
main, params = msg.get_content_type(), msg['content-type'].params
```

(Contributed by Victor Stinner in gh-104773.)

- **PEP 594**: Remove the `sndhdr` module, deprecated in Python 3.11: use the projects filetype, puremagic, or python-magic instead. (Contributed by Victor Stinner in gh-104773.)
- **PEP 594**: Remove the `pipes` module, deprecated in Python 3.11: use the `subprocess` module instead. (Contributed by Victor Stinner in gh-104773.)
- **PEP 594**: Remove the `ossaudiodev` module, deprecated in Python 3.11: use the pygame project for audio playback. (Contributed by Victor Stinner in gh-104780.)
- **PEP 594**: Remove the `sunau` module, deprecated in Python 3.11. (Contributed by Victor Stinner in gh-104773.)
- **PEP 594**: Remove the `mailcap` module, deprecated in Python 3.11. The `mimetypes` module provides an alternative. (Contributed by Victor Stinner in gh-104773.)
- **PEP 594**: Remove the `spwd` module, deprecated in Python 3.11: the python-pam project can be used instead. (Contributed by Victor Stinner in gh-104773.)
- **PEP 594**: Remove the `nntplib` module, deprecated in Python 3.11: the PyPI nntplib project can be used instead. (Contributed by Victor Stinner in gh-104773.)
- **PEP 594**: Remove the `nis` module, deprecated in Python 3.11. (Contributed by Victor Stinner in gh-104773.)
- **PEP 594**: Remove the `xdrlib` module, deprecated in Python 3.11. (Contributed by Victor Stinner in gh-104773.)
- **PEP 594**: Remove the `msilib` module, deprecated in Python 3.11. (Contributed by Zachary Ware in gh-104773.)
- **PEP 594**: Remove the `crypt` module and its private `_crypt` extension, deprecated in Python 3.11. The `hashlib` module is a potential replacement for certain use cases. Otherwise, the following PyPI projects can be used:
  - bcrypt: Modern password hashing for your software and your servers.
  - passlib: Comprehensive password hashing framework supporting over 30 schemes.
  - argon2-cffi: The secure Argon2 password hashing algorithm.

  - legacycrypt: Wrapper to the POSIX crypt library call and associated functionality.

  (Contributed by Victor Stinner in gh-104773.)

- **PEP 594**: Remove the `uu` module, deprecated in Python 3.11: the `base64` module is a modern alternative. (Contributed by Victor Stinner in gh-104773.)
- **PEP 594**: Remove the `aifc` module, deprecated in Python 3.11. (Contributed by Victor Stinner in gh-104773.)
- **PEP 594**: Remove the `audioop` module, deprecated in Python 3.11. (Contributed by Victor Stinner in gh-104773.)
- **PEP 594**: Remove the `chunk` module, deprecated in Python 3.11. (Contributed by Victor Stinner in gh-104773.)
- Remove support for the keyword-argument method of creating `typing.TypedDict` types, deprecated in Python 3.11. (Contributed by Tomas Roun in gh-104786.)
- **PEP 594**: Remove the `imghdr` module, deprecated in Python 3.11: use the projects filetype, puremagic, or python-magic instead. (Contributed by Victor Stinner in gh-104773.)
- Remove the untested and undocumented `unittest.TestProgram.usageExit()` method, deprecated in Python 3.11. (Contributed by Hugo van Kemenade in gh-104992.)
- Remove the `tkinter.tix` module, deprecated in Python 3.6. The third-party Tix library which the module wrapped is unmaintained. (Contributed by Zachary Ware in gh-75552.)
- Remove the old trashcan macros `Py_TRASHCAN_SAFE_BEGIN` and `Py_TRASHCAN_SAFE_END`. They should be replaced by the new macros `Py_TRASHCAN_BEGIN` and `Py_TRASHCAN_END`. The new macros were added in Python 3.8 and the old macros were deprecated in Python 3.11. (Contributed by Irit Katriel in gh-105111.)
- Remove `locale.resetlocale()` function deprecated in Python 3.11: use `locale.setlocale(locale.LC_ALL, "")` instead. (Contributed by Victor Stinner in gh-104783.)
- `logging`: Remove undocumented and untested `Logger.warn()` and `LoggerAdapter.warn()` methods and `logging.warn()` function. Deprecated since Python 3.3, they were aliases to the `logging.Logger.warning()` method, `logging.LoggerAdapter.warning()` method and `logging.warning()` function. (Contributed by Victor Stinner in gh-105376.)
- Remove *cafile*, *capath* and *cadefault* parameters of the `urllib.request.urlopen()` function, deprecated in Python 3.6: use the *context* parameter instead. Please use `ssl.SSLContext.load_cert_chain()` instead, or let `ssl.create_default_context()` select the system's trusted CA certificates for you. (Contributed by Victor Stinner in gh-105382.)
- Remove deprecated `webbrowser.MacOSXOSAScript._name` attribute. Use `webbrowser.MacOSXOSAScript.name` attribute instead. (Contributed by Nikita Sobolev in gh-105546.)
- Remove undocumented, never working, and deprecated `re.template` function and `re.TEMPLATE` flag (and `re.T` alias). (Contributed by Serhiy Storchaka and Nikita Sobolev in gh-105687.)

9 Porting to Python 3.13

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

- The old trashcan macros `Py_TRASHCAN_SAFE_BEGIN` and `Py_TRASHCAN_SAFE_END` were removed. They should be replaced by the new macros `Py_TRASHCAN_BEGIN` and `Py_TRASHCAN_END`.

  A tp_dealloc function that has the old macros, such as:

```
static void
mytype_dealloc(mytype *p)
{
    PyObject_GC_UnTrack(p);
```

(continues on next page)

(continued from previous page)

```
    Py_TRASHCAN_SAFE_BEGIN(p);
    ...
    Py_TRASHCAN_SAFE_END
}
```

should migrate to the new macros as follows:

```
static void
mytype_dealloc(mytype *p)
{
    PyObject_GC_UnTrack(p);
    Py_TRASHCAN_BEGIN(p, mytype_dealloc)
    ...
    Py_TRASHCAN_END
}
```

Note that `Py_TRASHCAN_BEGIN` has a second argument which should be the deallocation function it is in.

10 Build Changes

- Autoconf 2.71 and aclocal 1.16.4 is now required to regenerate the `configure` script. (Contributed by Christian Heimes in gh-89886.)
- SQLite 3.15.2 or newer is required to build the `sqlite3` extension module. (Contributed by Erlend Aasland in gh-105875.)
- Python built with `configure --with-trace-refs` (tracing references) is now ABI compatible with Python release build and debug build. (Contributed by Victor Stinner in gh-108634.)
- Building CPython now requires a compiler with support for the C11 atomic library, GCC built-in atomic functions, or MSVC interlocked intrinsics.
- The `_stat` C extension is now built with the limited C API. (Contributed by Victor Stinner in gh-85283.)

11 C API Changes

11.1 New Features

- You no longer have to define the `PY_SSIZE_T_CLEAN` macro before including `Python.h` when using `#` formats in format codes. APIs accepting the format codes always use `Py_ssize_t` for `#` formats. (Contributed by Inada Naoki in gh-104922.)
- Add `PyImport_AddModuleRef()`: similar to `PyImport_AddModule()`, but return a strong reference instead of a borrowed reference. (Contributed by Victor Stinner in gh-105922.)
- Add `PyWeakref_GetRef()` function: similar to `PyWeakref_GetObject()` but returns a strong reference, or `NULL` if the referent is no longer live. (Contributed by Victor Stinner in gh-105927.)
- Add `PyObject_GetOptionalAttr()` and `PyObject_GetOptionalAttrString()`, variants of `PyObject_GetAttr()` and `PyObject_GetAttrString()` which don't raise `AttributeError` if the attribute is not found. These variants are more convenient and faster if the missing attribute should not be treated as a failure. (Contributed by Serhiy Storchaka in gh-106521.)
- Add `PyMapping_GetOptionalItem()` and `PyMapping_GetOptionalItemString()`: variants of `PyObject_GetItem()` and `PyMapping_GetItemString()` which don't raise `KeyError` if the key is not found. These variants are more convenient and faster if the missing key should not be treated as a failure. (Contributed by Serhiy Storchaka in gh-106307.)
- Add fixed variants of functions which silently ignore errors:

  - `PyObject_HasAttrWithError()` replaces `PyObject_HasAttr()`.
  - `PyObject_HasAttrStringWithError()` replaces `PyObject_HasAttrString()`.
  - `PyMapping_HasKeyWithError()` replaces `PyMapping_HasKey()`.
  - `PyMapping_HasKeyStringWithError()` replaces `PyMapping_HasKeyString()`.

  New functions return not only `1` for true and `0` for false, but also `-1` for error.

  (Contributed by Serhiy Storchaka in gh-108511.)

- If Python is built in debug mode or `with assertions`, `PyTuple_SET_ITEM()` and `PyList_SET_ITEM()` now check the index argument with an assertion. If the assertion fails, make sure that the size is set before. (Contributed by Victor Stinner in gh-106168.)
- Add `PyModule_Add()` function: similar to `PyModule_AddObjectRef()` and `PyModule_AddObject()` but always steals a reference to the value. (Contributed by Serhiy Storchaka in gh-86493.)
- Added `PyDict_GetItemRef()` and `PyDict_GetItemStringRef()` functions: similar to `PyDict_GetItemWithError()` but returning a strong reference instead of a borrowed reference. Moreover, these functions return -1 on error and so checking `PyErr_Occurred()` is not needed. (Contributed by Victor Stinner in gh-106004.)
- Added `PyDict_ContainsString()` function: same as `PyDict_Contains()`, but *key* is specified as a `const char*` UTF-8 encoded bytes string, rather than a `PyObject*`. (Contributed by Victor Stinner in gh-108314.)
- Add `Py_IsFinalizing()` function: check if the main Python interpreter is shutting down. (Contributed by Victor Stinner in gh-108014.)
- Add `PyLong_AsInt()` function: similar to `PyLong_AsLong()`, but store the result in a C `int` instead of a C `long`. Previously, it was known as the private function `_PyLong_AsInt()` (with an underscore prefix). (Contributed by Victor Stinner in gh-108014.)
- Python built with `configure --with-trace-refs` (tracing references) now supports the Limited API. (Contributed by Victor Stinner in gh-108634.)
- Add `PyObject_VisitManagedDict()` and `PyObject_ClearManagedDict()` functions which must be called by the traverse and clear functions of a type using `Py_TPFLAGS_MANAGED_DICT` flag. The pythoncapi-compat project can be used to get these functions on Python 3.11 and 3.12. (Contributed by Victor Stinner in gh-107073.)
- Add `PyUnicode_EqualToUTF8AndSize()` and `PyUnicode_EqualToUTF8()` functions: compare Unicode object with a `const char*` UTF-8 encoded string and return true (`1`) if they are equal, or false (`0`) otherwise. These functions do not raise exceptions. (Contributed by Serhiy Storchaka in gh-110289.)
- Add `PyThreadState_GetUnchecked()` function: similar to `PyThreadState_Get()`, but don't kill the process with a fatal error if it is NULL. The caller is responsible to check if the result is NULL. Previously, the function was private and known as `_PyThreadState_UncheckedGet()`. (Contributed by Victor Stinner in gh-108867.)
- Add `PySys_AuditTuple()` function: similar to `PySys_Audit()`, but pass event arguments as a Python `tuple` object. (Contributed by Victor Stinner in gh-85283.)

11.2 Porting to Python 3.13

- `Python.h` no longer includes the `<ieeefp.h>` standard header. It was included for the `finite()` function which is now provided by the `<math.h>` header. It should now be included explicitly if needed. Remove also the `HAVE_IEEEFP_H` macro. (Contributed by Victor Stinner in gh-108765.)
- `Python.h` no longer includes the `<unistd.h>` standard header file. If needed, it should now be included explicitly. For example, it provides the functions: `read()`, `write()`, `close()`, `isatty()`, `lseek()`, `getpid()`, `getcwd()`, `sysconf()` and `getpagesize()`. As a consequence, `_POSIX_SEMAPHORES` and `_POSIX_THREADS` macros are no longer defined by `Python.h`. The `HAVE_UNISTD_H` and `HAVE_PTHREAD_H` macros defined by `Python.h` can be used to decide if `<unistd.h>` and `<pthread.h>` header files can be included. (Contributed by Victor Stinner in gh-108765.)
- `Python.h` no longer includes these standard header files: `<time.h>`, `<sys/select.h>` and `<sys/time.h>`. If needed, they should now be included explicitly. For example, `<time.h>` provides the `clock()` and `gmtime()` functions, `<sys/select.h>` provides the `select()` function, and `<sys/time.h>` provides the `futimes()`, `gettimeofday()` and `setitimer()` functions. (Contributed by Victor Stinner in gh-108765.)
- `Python.h` no longer includes the `<ctype.h>` standard header file. If needed, it should now be included explicitly. For example, it provides `isalpha()` and `tolower()` functions which are locale dependent. Python provides locale independent functions, like `Py_ISALPHA()` and `Py_TOLOWER()`. (Contributed by Victor Stinner in gh-108765.)
- If the `Py_LIMITED_API` macro is defined, `Py_BUILD_CORE`, `Py_BUILD_CORE_BUILTIN` and `Py_BUILD_CORE_MODULE` macros are now undefined by `<Python.h>`. (Contributed by Victor Stinner in gh-85283.)

11.3 Deprecated

- Passing optional arguments *maxsplit*, *count* and *flags* in module-level functions `re.split()`, `re.sub()` and `re.subn()` as positional arguments is now deprecated. In future Python versions these parameters will be keyword-only. (Contributed by Serhiy Storchaka in gh-56166.)
- Deprecate the old `Py_UNICODE` and `PY_UNICODE_TYPE` types: use directly the `wchar_t` type instead. Since Python 3.3, `Py_UNICODE` and `PY_UNICODE_TYPE` are just aliases to `wchar_t`. (Contributed by Victor Stinner in gh-105156.)
- Deprecate old 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 `PYTHONHOME` environment variable instead.

  Functions scheduled for removal in Python 3.15. (Contributed by Victor Stinner in gh-105145.)
- Deprecate the `PyImport_ImportModuleNoBlock()` function which is just an alias to `PyImport_ImportModule()` since Python 3.3. Scheduled for removal in Python 3.15. (Contributed by Victor Stinner in gh-105396.)
- Deprecate the `PyWeakref_GetObject()` and `PyWeakref_GET_OBJECT()` functions, which return a borrowed reference: use the new `PyWeakref_GetRef()` function instead, it returns a strong reference. The pythoncapi-compat project can be used to get `PyWeakref_GetRef()` on Python 3.12 and older. (Contributed by Victor Stinner in gh-105927.)

11.4 Removed

- Remove many APIs (functions, macros, variables) with names prefixed by `_Py` or `_PY` (considered as private API). If your project is affected by one of these removals and you consider that the removed API should remain available, please open a new issue to request a public C API and add `cc @vstinner` to the issue to notify Victor Stinner. (Contributed by Victor Stinner in gh-106320.)
- Remove functions deprecated in Python 3.9.
  - `PyEval_CallObject()`, `PyEval_CallObjectWithKeywords()`: use `PyObject_CallNoArgs()` or `PyObject_Call()` instead. Warning: `PyObject_Call()` positional arguments must be a `tuple` and must not be *NULL*, keyword arguments must be a `dict` or *NULL*, whereas removed functions checked arguments type and accepted *NULL* positional and keyword arguments. To replace `PyEval_CallObjectWithKeywords(func, NULL, kwargs)` with `PyObject_Call()`, pass an empty tuple as positional arguments using `PyTuple_New(0)`.
  - `PyEval_CallFunction()`: use `PyObject_CallFunction()` instead.
  - `PyEval_CallMethod()`: use `PyObject_CallMethod()` instead.
  - `PyCFunction_Call()`: use `PyObject_Call()` instead.

  (Contributed by Victor Stinner in gh-105107.)
- Remove old buffer protocols deprecated in Python 3.0. Use bufferobjects instead.
  - `PyObject_CheckReadBuffer()`: Use `PyObject_CheckBuffer()` to test if the object supports the buffer protocol. Note that `PyObject_CheckBuffer()` doesn't guarantee that `PyObject_GetBuffer()` will succeed. To test if the object is actually readable, see the next example of `PyObject_GetBuffer()`.
  - `PyObject_AsCharBuffer()`, `PyObject_AsReadBuffer()`: `PyObject_GetBuffer()` and `PyBuffer_Release()` instead:

```
Py_buffer view;
if (PyObject_GetBuffer(obj, &view, PyBUF_SIMPLE) < 0) {
    return NULL;
}
// Use `view.buf` and `view.len` to read from the buffer.
// You may need to cast buf as `(const char*)view.buf`.
PyBuffer_Release(&view);
```

  - `PyObject_AsWriteBuffer()`: Use `PyObject_GetBuffer()` and `PyBuffer_Release()` instead:

```
Py_buffer view;
if (PyObject_GetBuffer(obj, &view, PyBUF_WRITABLE) < 0) {
    return NULL;
}
// Use `view.buf` and `view.len` to write to the buffer.
PyBuffer_Release(&view);
```

  (Contributed by Inada Naoki in gh-85275.)
- Remove the following old functions to configure the Python initialization, deprecated in Python 3.11:
  - `PySys_AddWarnOptionUnicode()`: use `PyConfig.warnoptions` instead.
  - `PySys_AddWarnOption()`: use `PyConfig.warnoptions` instead.
  - `PySys_AddXOption()`: use `PyConfig.xoptions` instead.
  - `PySys_HasWarnOptions()`: use `PyConfig.xoptions` instead.
  - `PySys_SetArgvEx()`: set `PyConfig.argv` instead.
  - `PySys_SetArgv()`: set `PyConfig.argv` instead.

  - `PySys_SetPath()`: set `PyConfig.module_search_paths` instead.
  - `Py_SetPath()`: set `PyConfig.module_search_paths` instead.
  - `Py_SetProgramName()`: set `PyConfig.program_name` instead.
  - `Py_SetPythonHome()`: set `PyConfig.home` instead.
  - `Py_SetStandardStreamEncoding()`: set `PyConfig.stdio_encoding` instead, and set also maybe `PyConfig.legacy_windows_stdio` (on Windows).
  - `_Py_SetProgramFullPath()`: set `PyConfig.executable` instead.

  Use the new `PyConfig` API of the Python Initialization Configuration instead (**PEP 587**), added to Python 3.8. (Contributed by Victor Stinner in gh-105145.)

- Remove `PyEval_InitThreads()` and `PyEval_ThreadsInitialized()` functions, deprecated in Python 3.9. Since Python 3.7, `Py_Initialize()` always creates the GIL: calling `PyEval_InitThreads()` did nothing and `PyEval_ThreadsInitialized()` always returned non-zero. (Contributed by Victor Stinner in gh-105182.)
- Remove `PyEval_AcquireLock()` and `PyEval_ReleaseLock()` functions, deprecated in Python 3.2. They didn't update the current thread state. They can be replaced with:
  - `PyEval_SaveThread()` and `PyEval_RestoreThread()`;
  - low-level `PyEval_AcquireThread()` and `PyEval_RestoreThread()`;
  - or `PyGILState_Ensure()` and `PyGILState_Release()`.

  (Contributed by Victor Stinner in gh-105182.)
- Remove the old aliases to functions calling functions which were kept for backward compatibility with Python 3.8 provisional API:
  - `_PyObject_CallMethodNoArgs()`: use `PyObject_CallMethodNoArgs()`
  - `_PyObject_CallMethodOneArg()`: use `PyObject_CallMethodOneArg()`
  - `_PyObject_CallOneArg()`: use `PyObject_CallOneArg()`
  - `_PyObject_FastCallDict()`: use `PyObject_VectorcallDict()`
  - `_PyObject_Vectorcall()`: use `PyObject_Vectorcall()`
  - `_PyObject_VectorcallMethod()`: use `PyObject_VectorcallMethod()`
  - `_PyVectorcall_Function()`: use `PyVectorcall_Function()`

  Just remove the underscore prefix to update your code. (Contributed by Victor Stinner in gh-106084.)
- Remove private `_PyObject_FastCall()` function: use `PyObject_Vectorcall()` which is available since Python 3.8 (**PEP 590**). (Contributed by Victor Stinner in gh-106023.)
- Remove `cpython/pytime.h` header file: it only contained private functions. (Contributed by Victor Stinner in gh-106316.)
- Remove `_PyInterpreterState_Get()` alias to `PyInterpreterState_Get()` which was kept for backward compatibility with Python 3.8. The pythoncapi-compat project can be used to get `PyInterpreterState_Get()` on Python 3.8 and older. (Contributed by Victor Stinner in gh-106320.)
- The `PyModule_AddObject()` function is now soft deprecated: `PyModule_Add()` or `PyModule_AddObjectRef()` functions should be used instead. (Contributed by Serhiy Storchaka in gh-86493.)

11.5 Pending Removal in Python 3.14

- Creating immutable types (`Py_TPFLAGS_IMMUTABLETYPE`) with mutable bases using the C API.
- Global configuration variables:
  - `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 `PyPreConfig.legacy_windows_fs_encoding`
  - `Py_LegacyWindowsStdioFlag`: use `PyConfig.legacy_windows_stdio`
  - `Py_FileSystemDefaultEncoding`: use `PyConfig.filesystem_encoding`
  - `Py_HasFileSystemDefaultEncoding`: 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.

11.6 Pending Removal in Python 3.15

- `PyImport_ImportModuleNoBlock()`: use `PyImport_ImportModule()`.
- `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 `PYTHONHOME` environment variable instead.

11.7 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.

- `Py_TPFLAGS_HAVE_FINALIZE`: no needed since Python 3.8.
- `PyErr_Fetch()`: use `PyErr_GetRaisedException()`.
- `PyErr_NormalizeException()`: use `PyErr_GetRaisedException()`.
- `PyErr_Restore()`: use `PyErr_SetRaisedException()`.
- `PyModule_GetFilename()`: use `PyModule_GetFilenameObject()`.
- `PyOS_AfterFork()`: use `PyOS_AfterFork_Child()`.
- `PySlice_GetIndicesEx()`.
- `PyUnicode_AsDecodedObject()`.
- `PyUnicode_AsDecodedUnicode()`.
- `PyUnicode_AsEncodedObject()`.
- `PyUnicode_AsEncodedUnicode()`.
- `PyUnicode_READY()`: not needed since Python 3.12.
- `_PyErr_ChainExceptions()`.
- `PyBytesObject.ob_shash` member: call `PyObject_Hash()` instead.
- `PyDictObject.ma_version_tag` member.
- TLS API:
  - `PyThread_create_key()`: use `PyThread_tss_alloc()`.
  - `PyThread_delete_key()`: use `PyThread_tss_free()`.
  - `PyThread_set_key_value()`: use `PyThread_tss_set()`.
  - `PyThread_get_key_value()`: use `PyThread_tss_get()`.
  - `PyThread_delete_key_value()`: use `PyThread_tss_delete()`.
  - `PyThread_ReInitTLS()`: no longer needed.
- Remove undocumented `PY_TIMEOUT_MAX` constant from the limited C API. (Contributed by Victor Stinner in gh-110014.)

Index

E

P