What's New in Python

發Ⓕ *3.13.2*

A. M. Kuchling

3 月 03, 2025

Contents

编辑者
Adam Turner 和 Thomas Wouters

本文介紹了 Python 3.13 與 3.12 相比多了哪些新功能。Python 3.13 已於 2024 年 10 月 7 日發布。完整詳情請見 changelog。

> → **也參考**
>
> **PEP 719** -- Python 3.13 發[F]時程

1 發布重點摘要

Python 3.13 is the latest stable release of the Python programming language, with a mix of changes to the language, the implementation and the standard library. The biggest changes include a new *interactive interpreter*, experimental support for running in a *free-threaded mode* (**PEP 703**), and a *Just-In-Time compiler* (**PEP 744**).

Error messages continue to improve, with tracebacks now highlighted in color by default. The `locals()` builtin now has *defined semantics* for changing the returned mapping, and type parameters now support default values.

The library changes contain removal of deprecated APIs and modules, as well as the usual improvements in user-friendliness and correctness. Several legacy standard library modules have now *been removed* following their deprecation in Python 3.11 (**PEP 594**).

This article doesn't attempt to provide a complete specification of all new features, but instead gives a convenient overview. For full details refer to the documentation, such as the Library Reference and Language Reference. To understand the complete implementation and design rationale for a change, refer to the PEP for a particular new feature; but note that PEPs usually are not kept up-to-date once a feature has been fully implemented. See *Porting to Python 3.13* for guidance on upgrading from earlier versions of Python.

直譯器改進：

- A greatly improved *interactive interpreter* and *improved error messages*.

- **PEP 667**: The `locals()` builtin now has *defined semantics* when mutating the returned mapping. Python debuggers and similar tools may now more reliably update local variables in optimized scopes even during concurrent code execution.

- **PEP 703**: CPython 3.13 has experimental support for running with the global interpreter lock disabled. See *Free-threaded CPython* for more details.

- **PEP 744**: A basic *JIT compiler* was added. It is currently disabled by default (though we may turn it on later). Performance improvements are modest -- we expect to improve this over the next few releases.

- Color support in the new *interactive interpreter*, as well as in *tracebacks* and *doctest* output. This can be disabled through the `PYTHON_COLORS` and `NO_COLOR` environment variables.

Python 資料模型改進：

- `__static_attributes__` stores the names of attributes accessed through `self.X` in any function in a class body.

- `__firstlineno__` records the first line number of a class definition.

標准函式庫中的顯著改進

- Add a new `PythonFinalizationError` exception, raised when an operation is blocked during finalization.

- The `argparse` module now supports deprecating command-line options, positional arguments, and subcommands.

- The new functions `base64.z85encode()` and `base64.z85decode()` support encoding and decoding Z85 data.

- The `copy` module now has a `copy.replace()` function, with support for many builtin types and any class defining the `__replace__()` method.

- 新的 `dbm.sqlite3` 模組現在是預設的 `dbm` 後端。

- The `os` module has a suite of new functions for working with Linux's timer notification file descriptors.

- 現在 `random` 模組有一個命令列介面。

安全性改進：

- `ssl.create_default_context()` sets `ssl.VERIFY_X509_PARTIAL_CHAIN` and `ssl.VERIFY_X509_STRICT` as default flags.

C API 改進：

- The `Py_mod_gil` slot is now used to indicate that an extension module supports running with the GIL disabled.

- The PyTime C API has been added, providing access to system clocks.

- `PyMutex` is a new lightweight mutex that occupies a single byte.

- There is a new suite of functions for generating **PEP 669** monitoring events in the C API.

新增型Ｆ特性：

- **PEP 696**: Type parameters (`typing.TypeVar`, `typing.ParamSpec`, and `typing.TypeVarTuple`) now support defaults.

- **PEP 702**: The new `warnings.deprecated()` decorator adds support for marking deprecations in the type system and at runtime.

- **PEP 705**: `typing.ReadOnly` can be used to mark an item of a `typing.TypedDict` as read-only for type checkers.

- **PEP 742**: `typing.TypeIs` provides more intuitive type narrowing behavior, as an alternative to `typing.TypeGuard`.

平台支援：

- **PEP 730**: Apple's iOS is now an *officially supported platform*, at **tier 3**.

- **PEP 738**: Android is now an *officially supported platform*, at **tier 3**.

- `wasm32-wasi` 現在作Ｆ **tier 2** 平台支援。

- `wasm32-emscripten` 不再是官方支援的平台。

重要的移除：

- *PEP 594*: 剩下的 19 個"dead batteries"（遺留標准函式庫模組）已自標准函式庫中移除：`aifc`、`audioop`、`cgi`、`cgitb`、`chunk`、`crypt`、`imghdr`、`mailcap`、`msilib`、`nis`、`nntplib`、`ossaudiodev`、`pipes`、`sndhdr`、`spwd`、`sunau`、`telnetlib`、`uu` 和 `xdrlib`.

- Remove the **2to3** tool and `lib2to3` module (deprecated in Python 3.11).

- 移除 `tkinter.tix` 模組（在 Python 3.6 中已停用）。

- 移除 `locale.resetlocale()` 函式。

- 移除 `typing.io` 和 `typing.re` 命名空間。

- Remove chained `classmethod` descriptors.

Release schedule changes:

PEP 602 ("Annual Release Cycle for Python") has been updated to extend the full support ('bugfix') period for new releases to two years. This updated policy means that:

- Python 3.9--3.12 have one and a half years of full support, followed by three and a half years of security fixes.

- Python 3.13 and later have two years of full support, followed by three years of security fixes.

2 新增功能

2.1 A better interactive interpreter

Python now uses a new interactive shell by default, based on code from the PyPy project. When the user starts the REPL from an interactive terminal, the following new features are now supported:

- Multiline editing with history preservation.

- Direct support for REPL-specific commands like `help`, `exit`, and `quit`, without the need to call them as functions.

- Prompts and tracebacks with color enabled by default.

- Interactive help browsing using `F1` with a separate command history.

- History browsing using `F2` that skips output as well as the »> and ... prompts.

- "Paste mode" with `F3` that makes pasting larger blocks of code easier (press `F3` again to return to the regular prompt).

To disable the new interactive shell, set the `PYTHON_BASIC_REPL` environment variable. For more on interactive mode, see tut-interac.

(Contributed by Pablo Galindo Salgado, Łukasz Langa, and Lysandros Nikolaou in gh-111201 based on code from the PyPy project. Windows support contributed by Dino Viehland and Anthony Shaw.)

2.2 Improved error messages

- The interpreter now uses color by default when displaying tracebacks in the terminal. This feature can be controlled via the new `PYTHON_COLORS` environment variable as well as the canonical `NO_COLOR` and `FORCE_COLOR` environment variables. (Contributed by Pablo Galindo Salgado in gh-112730.)

- A common mistake is to write a script with the same name as a standard library module. When this results in errors, we now display a more helpful error message:

```
$ python random.py
Traceback (most recent call last):
  File "/home/me/random.py", line 1, in <module>
    import random
  File "/home/me/random.py", line 3, in <module>
    print(random.randint(5))
          ^^^^^^^^^^^^^^
AttributeError: module 'random' has no attribute 'randint' (consider renaming '/home/me/
```

```
→random.py' since it has the same name as the standard library module named 'random'␣
→and prevents importing that standard library module)
```

Similarly, if a script has the same name as a third-party module that it attempts to import and this results in errors, we also display a more helpful error message:

```
$ python numpy.py
Traceback (most recent call last):
  File "/home/me/numpy.py", line 1, in <module>
    import numpy as np
  File "/home/me/numpy.py", line 3, in <module>
    np.array([1, 2, 3])
    ^^^^^^^^
AttributeError: module 'numpy' has no attribute 'array' (consider renaming '/home/me/
→numpy.py' if it has the same name as a library you intended to
```

（由 Shantanu Jain 在 gh-95754 中貢獻。）

- The error message now tries to suggest the correct keyword argument when an incorrect keyword argument is passed to a function.

```
>>> "Better error messages!".split(max_split=1)
Traceback (most recent call last):
  File "<python-input-0>", line 1, in <module>
    "Better error messages!".split(max_split=1)
    ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^
TypeError: split() got an unexpected keyword argument 'max_split'. Did you mean
→'maxsplit'?
```

（由 Pablo Galindo Salgado 和 Shantanu Jain 於 gh-107944 中貢獻。）

2.3 Free-threaded CPython

CPython now has experimental support for running in a free-threaded mode, with the global interpreter lock (GIL) disabled. This is an experimental feature and therefore is not enabled by default. The free-threaded mode requires a different executable, usually called `python3.13t` or `python3.13t.exe`. Pre-built binaries marked as *free-threaded* can be installed as part of the official Windows and macOS installers, or CPython can be built from source with the `--disable-gil` option.

Free-threaded execution allows for full utilization of the available processing power by running threads in parallel on available CPU cores. While not all software will benefit from this automatically, programs designed with threading in mind will run faster on multi-core hardware. **The free-threaded mode is experimental** and work is ongoing to improve it: expect some bugs and a substantial single-threaded performance hit. Free-threaded builds of CPython support optionally running with the GIL enabled at runtime using the environment variable `PYTHON_GIL` or the command-line option `-X gil=1`.

To check if the current interpreter supports free-threading, `python -VV` and `sys.version` contain "experimental free-threading build". The new `sys._is_gil_enabled()` function can be used to check whether the GIL is actually disabled in the running process.

C-API extension modules need to be built specifically for the free-threaded build. Extensions that support running with the GIL disabled should use the `Py_mod_gil` slot. Extensions using single-phase init should use `PyUnstable_Module_SetGIL()` to indicate whether they support running with the GIL disabled. Importing C extensions that don't use these mechanisms will cause the GIL to be enabled, unless the GIL was explicitly disabled with the `PYTHON_GIL` environment variable or the `-X gil=0` option. pip 24.1 or newer is required to install packages with C extensions in the free-threaded build.

This work was made possible thanks to many individuals and organizations, including the large community of contributors to Python and third-party projects to test and enable free-threading support. Notable contributors include: Sam Gross, Ken Jin, Donghee Na, Itamar Oren, Matt Page, Brett Simmers, Dino Viehland, Carl Meyer, Nathan

Goldbaum, Ralf Gommers, Lysandros Nikolaou, and many others. Many of these contributors are employed by Meta, which has provided significant engineering resources to support this project.

> ↪ **也參考**
>
> **PEP 703** "Making the Global Interpreter Lock Optional in CPython" contains rationale and information surrounding this work.
>
> Porting Extension Modules to Support Free-Threading: A community-maintained porting guide for extension authors.

2.4 An experimental just-in-time (JIT) compiler

When CPython is configured and built using the `--enable-experimental-jit` option, a just-in-time (JIT) compiler is added which may speed up some Python programs. On Windows, use `PCbuild/build.bat --experimental-jit` to enable the JIT or `--experimental-jit-interpreter` to enable the Tier 2 interpreter. Build requirements and further supporting information are contained at `Tools/jit/README.md`.

The `--enable-experimental-jit` option takes these (optional) values, defaulting to `yes` if `--enable-experimental-jit` is present without the optional value.

- `no`: Disable the entire Tier 2 and JIT pipeline.

- `yes`: Enable the JIT. To disable the JIT at runtime, pass the environment variable `PYTHON_JIT=0`.

- `yes-off`: Build the JIT but disable it by default. To enable the JIT at runtime, pass the environment variable `PYTHON_JIT=1`.

- `interpreter`: Enable the Tier 2 interpreter but disable the JIT. The interpreter can be disabled by running with `PYTHON_JIT=0`.

The internal architecture is roughly as follows:

- We start with specialized *Tier 1 bytecode*. See What's new in 3.11 for details.

- When the Tier 1 bytecode gets hot enough, it gets translated to a new purely internal intermediate representation (IR), called the *Tier 2 IR*, and sometimes referred to as micro-ops ("uops").

- The Tier 2 IR uses the same stack-based virtual machine as Tier 1, but the instruction format is better suited to translation to machine code.

- We have several optimization passes for Tier 2 IR, which are applied before it is interpreted or translated to machine code.

- There is a Tier 2 interpreter, but it is mostly intended for debugging the earlier stages of the optimization pipeline. The Tier 2 interpreter can be enabled by configuring Python with `--enable-experimental-jit=interpreter`.

- When the JIT is enabled, the optimized Tier 2 IR is translated to machine code, which is then executed.

- The machine code translation process uses a technique called *copy-and-patch*. It has no runtime dependencies, but there is a new build-time dependency on LLVM.

> ↪ **也參考**
>
> **PEP 744**

(JIT by Brandt Bucher, inspired by a paper by Haoran Xu and Fredrik Kjolstad. Tier 2 IR by Mark Shannon and Guido van Rossum. Tier 2 optimizer by Ken Jin.)

2.5 Defined mutation semantics for `locals()`

Historically, the expected result of mutating the return value of `locals()` has been left to individual Python implementations to define. Starting from Python 3.13, **PEP 667** standardises the historical behavior of CPython for most code execution scopes, but changes optimized scopes (functions, generators, coroutines, comprehensions, and generator expressions) to explicitly return independent snapshots of the currently assigned local variables, including locally referenced nonlocal variables captured in closures.

This change to the semantics of `locals()` in optimized scopes also affects the default behavior of code execution functions that implicitly target `locals()` if no explicit namespace is provided (such as `exec()` and `eval()`). In previous versions, whether or not changes could be accessed by calling `locals()` after calling the code execution function was implementation-dependent. In CPython specifically, such code would typically appear to work as desired, but could sometimes fail in optimized scopes based on other code (including debuggers and code execution tracing tools) potentially resetting the shared snapshot in that scope. Now, the code will always run against an independent snapshot of the local variables in optimized scopes, and hence the changes will never be visible in subsequent calls to `locals()`. To access the changes made in these cases, an explicit namespace reference must now be passed to the relevant function. Alternatively, it may make sense to update affected code to use a higher level code execution API that returns the resulting code execution namespace (e.g. `runpy.run_path()` when executing Python files from disk).

To ensure debuggers and similar tools can reliably update local variables in scopes affected by this change, `FrameType.f_locals` now returns a write-through proxy to the frame's local and locally referenced nonlocal variables in these scopes, rather than returning an inconsistently updated shared `dict` instance with undefined runtime semantics.

See **PEP 667** for more details, including related C API changes and deprecations. Porting notes are also provided below for the affected *Python APIs* and *C APIs*.

(PEP and implementation contributed by Mark Shannon and Tian Gao in gh-74929. Documentation updates provided by Guido van Rossum and Alyssa Coghlan.)

2.6 Support for mobile platforms

PEP 730: iOS is now a **PEP 11** supported platform, with the `arm64-apple-ios` and `arm64-apple-ios-simulator` targets at tier 3 (iPhone and iPad devices released after 2013 and the Xcode iOS simulator running on Apple silicon hardware, respectively). `x86_64-apple-ios-simulator` (the Xcode iOS simulator running on older `x86_64` hardware) is not a tier 3 supported platform, but will have best-effort support. (PEP written and implementation contributed by Russell Keith-Magee in gh-114099.)

PEP 738: Android is now a **PEP 11** supported platform, with the `aarch64-linux-android` and `x86_64-linux-android` targets at tier 3. The 32-bit targets `arm-linux-androideabi` and `i686-linux-android` are not tier 3 supported platforms, but will have best-effort support. (PEP written and implementation contributed by Malcolm Smith in gh-116622.)

> ↪ **也參考**
>
> **PEP 730**、**PEP 738**

3 其他語言更動

- The compiler now strips common leading whitespace from every line in a docstring. This reduces the size of the bytecode cache (such as `.pyc` files), with reductions in file size of around 5%, for example in `sqlalchemy.orm.session` from SQLAlchemy 2.0. This change affects tools that use docstrings, such as `doctest`.

```
>>> def spam():
...     """
...         This is a docstring with
...          leading whitespace.
...
```

(繼續下一頁)

```
...          It even has multiple paragraphs!
...       """
...
>>> spam.__doc__
'\nThis is a docstring with\n  leading whitespace.\n\nIt even has multiple paragraphs!\n
↪'
```

（由 Inada Naoki 在 gh-81283 中貢獻。）

- Annotation scopes within class scopes can now contain lambdas and comprehensions. Comprehensions that are located within class scopes are not inlined into their parent scope.

```
class C[T]:
    type Alias = lambda: T
```

（由 Jelle Zijlstra 在 gh-109118 和 gh-118160 中貢獻。）

- Future statements are no longer triggered by relative imports of the __future__ module, meaning that statements of the form `from .__future__ import ...` are now simply standard relative imports, with no special features activated. (Contributed by Jeremiah Gabriel Pascual in gh-118216.)

- `global` declarations are now permitted in `except` blocks when that global is used in the `else` block. Previously this raised an erroneous `SyntaxError`. (Contributed by Irit Katriel in gh-111123.)

- Add `PYTHON_FROZEN_MODULES`, a new environment variable that determines whether frozen modules are ignored by the import machinery, equivalent to the `-X frozen_modules` command-line option. (Contributed by Yilei Yang in gh-111374.)

- Add support for the perf profiler working without frame pointers through the new environment variable `PYTHON_PERF_JIT_SUPPORT` and command-line option `-X perf_jit`. (Contributed by Pablo Galindo in gh-118518.)

- The location of a `.python_history` file can be changed via the new `PYTHON_HISTORY` environment variable. (Contributed by Levi Sabah, Zackery Spytz and Hugo van Kemenade in gh-73965.)

- Classes have a new `__static_attributes__` attribute. This is populated by the compiler with a tuple of the class's attribute names which are assigned through `self.<name>` from any function in its body. (Contributed by Irit Katriel in gh-115775.)

- The compiler now creates a `__firstlineno__` attribute on classes with the line number of the first line of the class definition. (Contributed by Serhiy Storchaka in gh-118465.)

- The `exec()` and `eval()` builtins now accept the *globals* and *locals* arguments as keywords. (Contributed by Raphael Gaschignard in gh-105879)

- The `compile()` builtin now accepts 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).

- Add a `__name__` attribute on `property` objects. (Contributed by Eugene Toder in gh-101860.)

- Add `PythonFinalizationError`, a new exception derived from `RuntimeError` and used to signal when operations are blocked during finalization. The following callables now raise `PythonFinalizationError`, instead of `RuntimeError`:

  - `_thread.start_new_thread()`

  - `os.fork()`

  - `os.forkpty()`

  - `subprocess.Popen`

  （由 Victor Stinner 在 gh-114570 中貢獻。）

- 允許 `str.replace()` 的 *count* 引數囸關鍵字。（由 Hugo van Kemenade 在 gh-106487 中貢獻。）

- Many functions now emit a warning if a boolean value is passed as a file descriptor argument. This can help catch some errors earlier. (Contributed by Serhiy Storchaka in gh-82626.)

- Added `name` and `mode` attributes for compressed and archived file-like objects in the `bz2`, `lzma`, `tarfile`, and `zipfile` modules. (Contributed by Serhiy Storchaka in gh-115961.)

4 新增模組

- `dbm.sqlite3`: An SQLite backend for `dbm`. (Contributed by Raymond Hettinger and Erlend E. Aasland in gh-100414.)

5 改進的模組

5.1 argparse

- Add the *deprecated* parameter to the `add_argument()` and `add_parser()` methods, to enable deprecating command-line options, positional arguments, and subcommands. (Contributed by Serhiy Storchaka in gh-83648.)

5.2 array

- Add the `'w'` type code (`Py_UCS4`) for Unicode characters. It should be used instead of the deprecated `'u'` type code. (Contributed by Inada Naoki in gh-80480.)

- Register `array.array` as a `MutableSequence` by implementing the `clear()` method. (Contributed by Mike Zimin in gh-114894.)

5.3 ast

- The constructors of node types in the `ast` module are now stricter in the arguments they accept, with more intuitive behavior when arguments are omitted.

  If an optional field on an AST node is not included as an argument when constructing an instance, the field will now be set to `None`. Similarly, if a list field is omitted, that field will now be set to an empty list, and if an `expr_context` field is omitted, it defaults to `Load()`. (Previously, in all cases, the attribute would be missing on the newly constructed AST node instance.)

  In all other cases, where a required argument is omitted, the node constructor will emit a `DeprecationWarning`. This will raise an exception in Python 3.15. Similarly, passing a keyword argument to the constructor that does not map to a field on the AST node is now deprecated, and will raise an exception in Python 3.15.

  These changes do not apply to user-defined subclasses of `ast.AST` unless the class opts in to the new behavior by defining the `AST._field_types` mapping.

  (Contributed by Jelle Zijlstra in gh-105858, gh-117486, and gh-118851.)

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

5.4 asyncio

- `asyncio.as_completed()` now returns an object that is both an asynchronous iterator and a plain iterator of awaitables. The awaitables yielded by asynchronous iteration include original task or future objects that were passed in, making it easier to associate results with the tasks being completed. (Contributed by Justin Arthur in gh-77714.)

- `asyncio.loop.create_unix_server()` will now automatically remove the Unix socket when the server is closed. (Contributed by Pierre Ossman in gh-111246.)

- `DatagramTransport.sendto()` will now send zero-length datagrams if called with an empty bytes object. The transport flow control also now accounts for the datagram header when calculating the buffer size. (Contributed by Jamie Phan in gh-115199.)

- Add `Queue.shutdown` and `QueueShutDown` to manage queue termination. (Contributed by Laurie Opperman and Yves Duprat in gh-104228.)

- Add the `Server.close_clients()` and `Server.abort_clients()` methods, which more forcefully close an asyncio server. (Contributed by Pierre Ossman in gh-113538.)

- Accept a tuple of separators in `StreamReader.readuntil()`, stopping when any one of them is encountered. (Contributed by Bruce Merry in gh-81322.)

- Improve the behavior of `TaskGroup` when an external cancellation collides with an internal cancellation. For example, when two task groups are nested and both experience an exception in a child task simultaneously, it was possible that the outer task group would hang, because its internal cancellation was swallowed by the inner task group.

  In the case where a task group is cancelled externally and also must raise an `ExceptionGroup`, it will now call the parent task's `cancel()` method. This ensures that a `CancelledError` will be raised at the next `await`, so the cancellation is not lost.

  An added benefit of these changes is that task groups now preserve the cancellation count (`cancelling()`).

  In order to handle some corner cases, `uncancel()` may now reset the undocumented `_must_cancel` flag when the cancellation count reaches zero.

  (Inspired by an issue reported by Arthur Tacca in gh-116720.)

- When `TaskGroup.create_task()` is called on an inactive `TaskGroup`, the given coroutine will be closed (which prevents a `RuntimeWarning` about the given coroutine being never awaited). (Contributed by Arthur Tacca and Jason Zhang in gh-115957.)

5.5 base64

- Add `z85encode()` and `z85decode()` functions for encoding `bytes` as Z85 data and decoding Z85-encoded data to `bytes`. (Contributed by Matan Perelman in gh-75299.)

5.6 compileall

- The default number of worker threads and processes is now selected using `os.process_cpu_count()` instead of `os.cpu_count()`. (Contributed by Victor Stinner in gh-109649.)

5.7 concurrent.futures

- The default number of worker threads and processes is now selected using `os.process_cpu_count()` instead of `os.cpu_count()`. (Contributed by Victor Stinner in gh-109649.)

5.8 configparser

- `ConfigParser` now has support for unnamed sections, which allows for top-level key-value pairs. This can be enabled with the new *allow_unnamed_section* parameter. (Contributed by Pedro Sousa Lacerda in gh-66449.)

5.9 copy

- The new `replace()` function and the `replace protocol` make creating modified copies of objects much simpler. This is especially useful when working with immutable objects. The following types support the `replace()` function and implement the replace protocol:

  - `collections.namedtuple()`

  - `dataclasses.dataclass`

  - `datetime.datetime`、`datetime.date`、`datetime.time`

11

– `inspect.Signature`、`inspect.Parameter`

– `types.SimpleNamespace`

– 程式碼物件

Any user-defined class can also support `copy.replace()` by defining the `__replace__()` method. (Contributed by Serhiy Storchaka in gh-108751.)

5.10 ctypes

- As a consequence of necessary internal refactoring, initialization of internal metaclasses now happens in `__init__` rather than in `__new__`. This affects projects that subclass these internal metaclasses to provide custom initialization. Generally:

  – Custom logic that was done in `__new__` after calling `super().__new__` should be moved to `__init__`.

  – To create a class, call the metaclass, not only the metaclass's `__new__` method.

  See gh-124520 for discussion and links to changes in some affected projects.

- `ctypes.Structure` objects have a new `_align_` attribute which allows the alignment of the structure being packed to/from memory to be specified explicitly. (Contributed by Matt Sanderson in gh-112433)

5.11 dbm

- Add `dbm.sqlite3`, a new module which implements an SQLite backend, and make it the default `dbm` backend. (Contributed by Raymond Hettinger and Erlend E. Aasland in gh-100414.)

- Allow removing all items from the database through the new `gdbm.clear()` and `ndbm.clear()` methods. (Contributed by Donghee Na in gh-107122.)

5.12 dis

- Change the output of `dis` module functions to show logical labels for jump targets and exception handlers, rather than offsets. The offsets can be added with the new `-O` command-line option or the *show_offsets* argument. (Contributed by Irit Katriel in gh-112137.)

- `get_instructions()` no longer represents cache entries as separate instructions. Instead, it returns them as part of the `Instruction`, in the new *cache_info* field. The *show_caches* argument to `get_instructions()` is deprecated and no longer has any effect. (Contributed by Irit Katriel in gh-112962.)

5.13 doctest

- `doctest` output is now colored by default. This can be controlled via the new `PYTHON_COLORS` environment variable as well as the canonical `NO_COLOR` and `FORCE_COLOR` environment variables. See also using-on-controlling-color. (Contributed by Hugo van Kemenade in gh-117225.)

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

5.14 email

- Headers with embedded newlines are now quoted on output. The `generator` will now refuse to serialize (write) headers that are improperly folded or delimited, such that they would be parsed as multiple headers or joined with adjacent data. If you need to turn this safety feature off, set `verify_generated_headers`. (Contributed by Bas Bloemsaat and Petr Viktorin in gh-121650.)

- `getaddresses()` and `parseaddr()` now return `('', '')` pairs in more situations where invalid email addresses are encountered instead of potentially inaccurate values. The two functions have a new optional *strict* parameter (default `True`). To get the old behavior (accepting malformed input), use `strict=False`. `getattr(email.utils, 'supports_strict_parsing', False)` can be used to check if the *strict*

parameter is available. (Contributed by Thomas Dwyer and Victor Stinner for gh-102988 to improve the **CVE 2023-27043** fix.)

5.15 enum

- 公開 `EnumDict` 以更好地支援 `EnumType` 的子類⒡。

5.16 fractions

- `Fraction` objects now support the standard format specification mini-language rules for fill, alignment, sign handling, minimum width, and grouping. (Contributed by Mark Dickinson in gh-111320.)

5.17 glob

- Add `translate()`, a function to convert a path specification with shell-style wildcards to a regular expression. (Contributed by Barney Gale in gh-72904.)

5.18 importlib

- The following functions in `importlib.resources` now allow accessing a directory (or tree) of resources, using multiple positional arguments (the *encoding* and *errors* arguments in the text-reading functions are now keyword-only):

  - `is_resource()`

  - `open_binary()`

  - `open_text()`

  - `path()`

  - `read_binary()`

  - `read_text()`

  These functions are no longer deprecated and are not scheduled for removal. (Contributed by Petr Viktorin in gh-116608.)

- `contents()` remains deprecated in favor of the fully-featured `Traversable` API. However, there is now no plan to remove it. (Contributed by Petr Viktorin in gh-116608.)

5.19 io

- The `IOBase` finalizer now logs any errors raised by the `close()` method with `sys.unraisablehook`. Previously, errors were ignored silently by default, and only logged in Python Development Mode or when using a Python debug build. (Contributed by Victor Stinner in gh-62948.)

5.20 ipaddress

- Add the `IPv4Address.ipv6_mapped` property, which returns the IPv4-mapped IPv6 address. (Contributed by Charles Machalow in gh-109466.)

- Fix `is_global` and `is_private` behavior in `IPv4Address`, `IPv6Address`, `IPv4Network`, and `IPv6Network`. (Contributed by Jakub Stasiak in gh-113171.)

5.21 itertools

- `batched()` has a new *strict* parameter, which raises a `ValueError` if the final batch is shorter than the specified batch size. (Contributed by Raymond Hettinger in gh-113202.)

5.22 marshal

- Add the *allow_code* parameter in module functions. Passing `allow_code=False` prevents serialization and de-serialization of code objects which are incompatible between Python versions. (Contributed by Serhiy Storchaka in gh-113626.)

5.23 math

- The new function `fma()` performs fused multiply-add operations. This computes `x * y + z` with only a single round, and so avoids any intermediate loss of precision. It wraps the `fma()` function provided by C99, and follows the specification of the IEEE 754 "fusedMultiplyAdd" operation for special cases. (Contributed by Mark Dickinson and Victor Stinner in gh-73468.)

5.24 mimetypes

- Add the `guess_file_type()` function to guess a MIME type from a filesystem path. Using paths with `guess_type()` is now soft deprecated. (Contributed by Serhiy Storchaka in gh-66543.)

5.25 mmap

- `mmap` is now protected from crashing on Windows when the mapped memory is inaccessible due to file system errors or access violations. (Contributed by Jannis Weigend in gh-118209.)

- `mmap` has a new `seekable()` method that can be used when a seekable file-like object is required. The `seek()` method now returns the new absolute position. (Contributed by Donghee Na and Sylvie Liberman in gh-111835.)

- The new UNIX-only *trackfd* parameter for `mmap` controls file descriptor duplication; if false, the file descriptor specified by *fileno* will not be duplicated. (Contributed by Zackery Spytz and Petr Viktorin in gh-78502.)

5.26 multiprocessing

- The default number of worker threads and processes is now selected using `os.process_cpu_count()` instead of `os.cpu_count()`. (Contributed by Victor Stinner in gh-109649.)

5.27 os

- Add `process_cpu_count()` function to get the number of logical CPU cores usable by the calling thread of the current process. (Contributed by Victor Stinner in gh-109649.)

- `cpu_count()` and `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 application code or the container itself. (Contributed by Donghee Na in gh-109595.)

- Add a low level interface to Linux's *timer file descriptors* via `timerfd_create()`, `timerfd_settime()`, `timerfd_settime_ns()`, `timerfd_gettime()`, `timerfd_gettime_ns()`, `TFD_NONBLOCK`, `TFD_CLOEXEC`, `TFD_TIMER_ABSTIME`, and `TFD_TIMER_CANCEL_ON_SET` (Contributed by Masaru Tsuchiyama in gh-108277.)

- `lchmod()` and the *follow_symlinks* argument of `chmod()` are both now available on Windows. Note that the default value of *follow_symlinks* in `lchmod()` is `False` on Windows. (Contributed by Serhiy Storchaka in gh-59616.)

- `fchmod()` and support for file descriptors in `chmod()` are both now available on Windows. (Contributed by Serhiy Storchaka in gh-113191.)

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

- `posix_spawn()` now accepts `None` for the *env* argument, which makes the newly spawned process use the current process environment. (Contributed by Jakub Kulik in gh-113119.)

- `posix_spawn()` can now use the `POSIX_SPAWN_CLOSEFROM` attribute in the *file_actions* parameter on platforms that support `posix_spawn_file_actions_addclosefrom_np()`. (Contributed by Jakub Kulik in gh-113117.)

5.28 os.path

- Add `isreserved()` to check if a path is reserved on the current system. This function is only available on Windows. (Contributed by Barney Gale in gh-88569.)

- On Windows, `isabs()` no longer considers paths starting with exactly one slash (\ or /) to be absolute. (Contributed by Barney Gale and Jon Foster in gh-44626.)

- `realpath()` now resolves MS-DOS style file names even if the file is not accessible. (Contributed by Moonsik Park in gh-82367.)

5.29 pathlib

- Add `UnsupportedOperation`, which is raised instead of `NotImplementedError` when a path operation isn't supported. (Contributed by Barney Gale in gh-89812.)

- Add a new constructor for creating `Path` objects from 'file' URIs (`file:///`), `Path.from_uri()`. (Contributed by Barney Gale in gh-107465.)

- Add `PurePath.full_match()` for matching paths with shell-style wildcards, including the recursive wildcard "`**`". (Contributed by Barney Gale in gh-73435.)

- Add the `PurePath.parser` class attribute to store the implementation of `os.path` used for low-level path parsing and joining. This will be either `posixpath` or `ntpath`.

- Add *recurse_symlinks* keyword-only argument to `Path.glob()` and `rglob()`. (Contributed by Barney Gale in gh-77609.)

- `Path.glob()` and `rglob()` now return files and directories when given a pattern that ends with "`**`". Previously, only directories were returned. (Contributed by Barney Gale in gh-70303.)

- Add the *follow_symlinks* keyword-only argument to `Path.is_file`, `Path.is_dir`, `Path.owner()`, and `Path.group()`. (Contributed by Barney Gale in gh-105793 and Kamil Turek in gh-107962.)

5.30 pdb

- `breakpoint()` and `set_trace()` now enter the debugger immediately rather than on the next line of code to be executed. This change prevents the debugger from breaking outside of the context when `breakpoint()` is positioned at the end of the context. (Contributed by Tian Gao in gh-118579.)

- `sys.path[0]` is no longer replaced by the directory of the script being debugged when `sys.flags.safe_path` is set. (Contributed by Tian Gao and Christian Walther in gh-111762.)

- `zipapp` is now supported as a debugging target. (Contributed by Tian Gao in gh-118501.)

- 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 and statements whose prefix is a pdb command are now correctly identified and executed. (Contributed by Tian Gao in gh-108464.)

5.31 queue

- Add `Queue.shutdown` and `ShutDown` to manage queue termination. (Contributed by Laurie Opperman and Yves Duprat in gh-104750.)

5.32 random

- Add a command-line interface. (Contributed by Hugo van Kemenade in gh-118131.)

5.33 re

- Rename `re.error` to `PatternError` for improved clarity. `re.error` is kept for backward compatibility.

5.34 shutil

- Support the *dir_fd* and *follow_symlinks* keyword arguments in `chown()`. (Contributed by Berker Peksag and Tahia K in gh-62308)

5.35 site

- `.pth` files are now decoded using UTF-8 first, and then with the locale encoding if UTF-8 decoding fails. (Contributed by Inada Naoki in gh-117802.)

5.36 sqlite3

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

- Add the *filter* keyword-only parameter to `Connection.iterdump()` for filtering database objects to dump. (Contributed by Mariusz Felisiak in gh-91602.)

5.37 ssl

- The `create_default_context()` API now includes `VERIFY_X509_PARTIAL_CHAIN` and `VERIFY_X509_STRICT` in its default flags.

> ℹ️ 備註
>
> `VERIFY_X509_STRICT` may reject pre-**RFC 5280** or malformed certificates that the underlying OpenSSL implementation might otherwise accept. Whilst disabling this is not recommended, you can do so using:
>
> ```python
> import ssl
>
> ctx = ssl.create_default_context()
> ctx.verify_flags &= ~ssl.VERIFY_X509_STRICT
> ```

（由 William Woodruff 在 gh-112389 中貢獻。）

5.38 statistics

- Add `kde()` for kernel density estimation. This makes it possible to estimate a continuous probability density function from a fixed number of discrete samples. (Contributed by Raymond Hettinger in gh-115863.)

- Add `kde_random()` for sampling from an estimated probability density function created by `kde()`. (Contributed by Raymond Hettinger in gh-115863.)

5.39 subprocess

- The `subprocess` module now uses the `posix_spawn()` function in more situations.

  Notably, when *close_fds* is `True` (the default), `posix_spawn()` will be used when the C library provides `posix_spawn_file_actions_addclosefrom_np()`, which includes recent versions of Linux, FreeBSD, and Solaris. On Linux, this should perform similarly to the existing Linux `vfork()` based code.

  A private control knob `subprocess._USE_POSIX_SPAWN` can be set to `False` if you need to force `subprocess` to never use `posix_spawn()`. Please report your reason and platform details in the issue

tracker if you set this so that we can improve our API selection logic for everyone. (Contributed by Jakub Kulik in gh-113117.)

5.40 sys

- Add the `_is_interned()` function to test if a string was interned. This function is not guaranteed to exist in all implementations of Python. (Contributed by Serhiy Storchaka in gh-78573.)

5.41 tempfile

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

5.42 time

- On Windows, `monotonic()` now uses the `QueryPerformanceCounter()` clock for a resolution of 1 microsecond, instead of the `GetTickCount64()` clock which has a resolution of 15.6 milliseconds. (Contributed by Victor Stinner in gh-88494.)

- On Windows, `time()` now uses the `GetSystemTimePreciseAsFileTime()` clock for a resolution of 1 microsecond, instead of the `GetSystemTimeAsFileTime()` clock which has a resolution of 15.6 milliseconds. (Contributed by Victor Stinner in gh-63207.)

5.43 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.)

- The `tkinter` widget method `wm_attributes()` now accepts the attribute name without the minus prefix to get window attributes, for example `w.wm_attributes('alpha')` and allows specifying attributes and values to set as keyword arguments, for example `w.wm_attributes(alpha=0.5)`. (Contributed by Serhiy Storchaka in gh-43457.)

- `wm_attributes()` can now return attributes as a `dict`, by using the new optional keyword-only parameter *return_python_dict*. (Contributed by Serhiy Storchaka in gh-43457.)

- `Text.count()` can now return a simple `int` when the new optional keyword-only parameter *return_ints* is used. Otherwise, the single count is returned as a 1-tuple or `None`. (Contributed by Serhiy Storchaka in gh-97928.)

- Support the "vsapi" element type in the `element_create()` method of `tkinter.ttk.Style`. (Contributed by Serhiy Storchaka in gh-68166.)

- Add the `after_info()` method for Tkinter widgets. (Contributed by Cheryl Sabella in gh-77020.)

- Add a new `copy_replace()` method to `PhotoImage` to copy a region from one image to another, possibly with pixel zooming, subsampling, or both. (Contributed by Serhiy Storchaka in gh-118225.)

- Add *from_coords* parameter to the `PhotoImage` methods `copy()`, `zoom()` and `subsample()`. Add *zoom* and *subsample* parameters to the `PhotoImage` method `copy()`. (Contributed by Serhiy Storchaka in gh-118225.)

- Add the `PhotoImage` methods `read()` to read an image from a file and `data()` to get the image data. Add *background* and *grayscale* parameters to the `write()` method. (Contributed by Serhiy Storchaka in gh-118271.)

5.44 traceback

- Add the `exc_type_str` attribute to `TracebackException`, which holds a string display of the *exc_type*. Deprecate the `exc_type` attribute, which holds the type object itself. Add parameter *save_exc_type* (default `True`) to indicate whether `exc_type` should be saved. (Contributed by Irit Katriel in gh-112332.)

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

5.45 types

- `SimpleNamespace` can now take a single positional argument to initialise the namespace's arguments. This argument must either be a mapping or an iterable of key-value pairs. (Contributed by Serhiy Storchaka in gh-108191.)

5.46 typing

- **PEP 705**: Add `ReadOnly`, a special typing construct to mark a `TypedDict` item as read-only for type checkers.

- **PEP 742**: Add `TypeIs`, a typing construct that can be used to instruct a type checker how to narrow a type.

- Add `NoDefault`, a sentinel object used to represent the defaults of some parameters in the `typing` module. (Contributed by Jelle Zijlstra in gh-116126.)

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

- Add `is_protocol()` to check whether a class is a `Protocol`. (Contributed by Jelle Zijlstra in gh-104873.)

- `ClassVar` can now be nested in `Final`, and vice versa. (Contributed by Mehdi Drissi in gh-89547.)

5.47 unicodedata

- Update the Unicode database to version 15.1.0. (Contributed by James Gerity in gh-109559.)

5.48 venv

- Add support for creating source control management (SCM) ignore files in 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 (`EnvBuilder` and `create()`), and opt-out via the CLI, using `--without-scm-ignore-files`. (Contributed by Brett Cannon in gh-108125.)

5.49 warnings

- **PEP 702**: The new `warnings.deprecated()` decorator provides a way to communicate deprecations to a static type checker and to warn on usage of deprecated classes and functions. A `DeprecationWarning` may also be emitted when a decorated function or class is used at runtime. (Contributed by Jelle Zijlstra in gh-104003.)

5.50 xml

- Allow controlling Expat >=2.6.0 reparse deferral (**CVE 2023-52425**) by adding five new methods:

    - `xml.etree.ElementTree.XMLParser.flush()`

    - `xml.etree.ElementTree.XMLPullParser.flush()`

    - `xml.parsers.expat.xmlparser.GetReparseDeferralEnabled()`

    - `xml.parsers.expat.xmlparser.SetReparseDeferralEnabled()`

    - `xml.sax.expatreader.ExpatParser.flush()`

（由 Sebastian Pipping 在 gh-115623 中貢獻。）

- Add the `close()` method for the iterator returned by `iterparse()` for explicit cleanup. (Contributed by Serhiy Storchaka in gh-69893.)

5.51 zipimport

- Add support for ZIP64 format files. Everybody loves huge data, right? (Contributed by Tim Hatch in gh-94146.)

6 最佳化

- Several standard library modules have had their import times significantly improved. For example, the import time of the `typing` module has been reduced by around a third by removing dependencies on `re` and `contextlib`. Other modules to enjoy import-time speedups include `email.utils`, `enum`, `functools`, `importlib.metadata`, and `threading`. (Contributed by Alex Waygood, Shantanu Jain, Adam Turner, Daniel Hollas, and others in gh-109653.)

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

- The `subprocess` module now uses the `posix_spawn()` function in more situations, including when *close_fds* is `True` (the default) on many modern platforms. This should provide a notable performance increase when launching processes on FreeBSD and Solaris. See the *subprocess* section above for details. (Contributed by Jakub Kulik in gh-113117.)

7 被移除的模組和 API

7.1 PEP 594：從標準函式庫中移除「用完的電池」

PEP 594 提議從標準函式庫中移除因Ｆ歷史、過時或不安全狀態而被稱Ｆ「用完的電池 (dead batteries)」的 19 個模組。以下所有模組在 Python 3.11 中已被Ｆ用，現在已被移除：

- `aifc`

  - standard-aifc：PyPI 上的 `aifc` 函式庫重新發布版。

- `audioop`

  - audioop-lts：PyPI 上的 `audioop-lts` 函式庫。

- `chunk`

  - standard-chunk：PyPI 上的 `chunk` 函式庫重新發布版。

- `cgi` 和 `cgitb`

  - `cgi.FieldStorage`，在於 "GET" 和 `HEAD` 請求中通常可以被 `urllib.parse.parse_qsl()` 取代，而在於 `POST` 和 `PUT` 請求中可以被 `email.message` 模組或 multipart 函式庫取代。

  - `cgi.parse()` 能Ｆ以在所需的查詢字串上直接呼叫 `urllib.parse.parse_qs()` 來取代，除非輸入Ｆ `multipart/form-data`，而這種情Ｆ則應該如下所述地將 `cgi.parse_multipart()` 給替Ｆ掉。

  - `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`:

    ```python
    from email.message import EmailMessage

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

- – `cgi.parse_multipart()` 可以被 `email` 函式庫中實作相同 MIME RFCs 的功能取代，或者可以被 multipart 函式庫取代，例如 `email.message.EmailMessage` 和 `email.message.Message` 類ⅎ。
  - – standard-cgi 和 standard-cgitb：PyPI 上的 `cgi` 和 `cgitb` 函式庫重新發布版。

- `crypt` and the private `_crypt` extension. The `hashlib` module may be an appropriate replacement when simply hashing a value is required. Otherwise, various third-party libraries on PyPI are available:
  - – bcrypt：適用於你的軟體與伺服器的現代密碼雜ⅎ演算法。
  - – passlib：支援超過 30 種方案的廣泛密碼雜ⅎ框架。
  - – argon2-cffi：安全的 Argon2 密碼雜ⅎ演算法。
  - – legacycrypt：對 POSIX crypt 函式庫呼叫及相關功能的 `ctypes` 包裝。
  - – crypt_r：對 `crypt` 模組的分支 (fork)，對 *crypt_r(3)* 函式庫呼叫及相關功能的包裝。
  - – standard-crypt 和 deprecated-crypt-alternative：PyPI 上的 `crypt` 函式庫重新發布版和 `_crypt` 函式庫的重新實作。

- `imghdr`：請改用 filetype、puremagic 或 python-magic 函式庫。例如 `puremagic.what()` 函式可以取代 `imghdr` 過去所支援所有文件格式的 `imghdr.what()` 函式。
  - – standard-imghdr：PyPI 上的 `imghdr` 函式庫重新發布版。

- `mailcap`：請改用 `mimetypes` 模組。
  - – standard-mailcap：PyPI 上的 `mailcap` 函式庫重新發布版。

- `msilib`

- `nis`

- `nntplib`：請改用 PyPI 上的 pynntp 函式庫。
  - – standard-nntplib：PyPI 上的 `nntplib` 函式庫重新發布版。

- `ossaudiodev`：音訊播放方面，請改用 PyPI 上的 pygame 函式庫。

- `pipes`：請改用 `subprocess` 模組。使用 `shlex.quote()` 來取代未以文件紀ⅎ的 `pipes.quote` 函式。
  - – standard-pipes：PyPI 上的 `pipes` 函式庫重新發布版。

- `sndhdr`：請改用 filetype、puremagic 或 python-magic 函式庫。
  - – standard-sndhdr：PyPI 上的 `sndhdr` 函式庫重新發布版。

- `spwd`：請改用 PyPI 上的 python-pam 函式庫。

- `sunau`
  - – standard-sunau：PyPI 上的 `sunau` 函式庫重新發布版。

- `telnetlib`：請改用 PyPI 上的 telnetlib3 或 Exscript 函式庫。
  - – standard-telnetlib：PyPI 上的 `telnetlib` 函式庫重新發布版。

- `uu`：請改用 `base64` 模組來作ⅎ當代的替代方案。
  - – standard-uu：PyPI 上的 `uu` 函式庫重新發布版。

- `xdrlib`
  - – standard-xdrlib：PyPI 上的 `xdrlib` 函式庫重新發布版。

（由 Victor Stinner 和 Zachary Ware 在 gh-104773 和 gh-104780 中貢獻。）

7.2 2to3

- Remove the **2to3** program and the `lib2to3` module, previously deprecated in Python 3.11. (Contributed by Victor Stinner in gh-104780.)

7.3 builtins

- Remove support for chained `classmethod` descriptors (introduced in gh-63272). These can no longer be used to wrap other descriptors, such as `property`. The core design of this feature was flawed and led to several problems. To "pass-through" a `classmethod`, consider using the `__wrapped__` attribute that was added in Python 3.10. (Contributed by Raymond Hettinger in gh-89519.)

- Raise a `RuntimeError` when calling `frame.clear()` on a suspended frame (as has always been the case for an executing frame). (Contributed by Irit Katriel in gh-79932.)

7.4 configparser

- Remove the undocumented `LegacyInterpolation` class, deprecated in the docstring since Python 3.2, and at runtime since Python 3.11. (Contributed by Hugo van Kemenade in gh-104886.)

7.5 importlib.metadata

- Remove deprecated subscript (`__getitem__()`) access for EntryPoint objects. (Contributed by Jason R. Coombs in gh-113175.)

7.6 locale

- Remove the `locale.resetlocale()` function, deprecated in Python 3.11. Use `locale.setlocale(locale.LC_ALL, "")` instead. (Contributed by Victor Stinner in gh-104783.)

7.7 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 use. (Contributed by Irit Katriel in gh-105481.)

- Remove `opcode.is_pseudo()`, `opcode.MIN_PSEUDO_OPCODE`, and `opcode.MAX_PSEUDO_OPCODE`, which were added in Python 3.12, but were neither documented nor exposed through `dis`, and were not intended to be used externally. (Contributed by Irit Katriel in gh-105481.)

7.8 optparse

- This module is no longer considered soft deprecated. While `argparse` remains preferred for new projects that aren't using a third party command line argument processing library, there are aspects of the way `argparse` works that mean the lower level `optparse` module may provide a better foundation for *writing* argument processing libraries, and for implementing command line applications which adhere more strictly than `argparse` does to various Unix command line processing conventions that originate in the behaviour of the C `getopt()` function . (Contributed by Alyssa Coghlan and Serhiy Storchaka in gh-126180.)

7.9 pathlib

- Remove the ability to use `Path` objects as context managers. This functionality was deprecated and has had no effect since Python 3.9. (Contributed by Barney Gale in gh-83863.)

7.10 re

- Remove the undocumented, deprecated, and broken `re.template()` function and `re.TEMPLATE` / `re.T` flag. (Contributed by Serhiy Storchaka and Nikita Sobolev in gh-105687.)

7.11 tkinter.tix

- 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.)

7.12 turtle

- Remove the `RawTurtle.settiltangle()` method, deprecated in the documentation since Python 3.1 and at runtime since Python 3.11. (Contributed by Hugo van Kemenade in gh-104876.)

7.13 typing

- Remove the `typing.io` and `typing.re` namespaces, deprecated since Python 3.8. The items in those namespaces can be imported directly from the `typing` module. (Contributed by Sebastian Rittau in gh-92871.)

- Remove the keyword-argument method of creating `TypedDict` types, deprecated in Python 3.11. (Contributed by Tomas Roun in gh-104786.)

7.14 unittest

- 移除以下在 Python 3.11 中已被圧用的 `unittest` 函式：

    - `unittest.findTestCases()`

    - `unittest.makeSuite()`

    - `unittest.getTestCaseNames()`

  Use `TestLoader` methods instead:

    - `loadTestsFromModule()`

    - `loadTestsFromTestCase()`

    - `getTestCaseNames()`

  （由 Hugo van Kemenade 在 gh-104835 中貢獻。）

- Remove the untested and undocumented `TestProgram.usageExit()` method, deprecated in Python 3.11. (Contributed by Hugo van Kemenade in gh-104992.)

7.15 urllib

- Remove the *cafile*, *capath*, and *cadefault* parameters of the `urllib.request.urlopen()` function, deprecated in Python 3.6. Use the *context* parameter instead with an `SSLContext` instance. The `ssl.SSLContext.load_cert_chain()` function can be used to load specific certificates, or let `ssl.create_default_context()` select the operating system's trusted certificate authority (CA) certificates. (Contributed by Victor Stinner in gh-105382.)

7.16 webbrowser

- Remove the untested and undocumented `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 the deprecated `MacOSXOSAScript._name` attribute. Use the `MacOSXOSAScript.name` attribute instead. (Contributed by Nikita Sobolev in gh-105546.)

8 New Deprecations

- User-defined functions:

- Deprecate assignment to a function's `__code__` attribute, where the new code object's type does not match the function's type. The different types are: plain function, generator, async generator, and coroutine. (Contributed by Irit Katriel in [gh-81137](#).)

- `array`:

  - Deprecate the `'u'` format code (`wchar_t`) at runtime. This format code has been deprecated in documentation since Python 3.3, and will be removed in Python 3.16. Use the `'w'` format code (`Py_UCS4`) for Unicode characters instead. (Contributed by Hugo van Kemenade in [gh-80480](#).)

- `ctypes`:

  - Deprecate the undocumented `SetPointerType()` function, to be removed in Python 3.15. (Contributed by Victor Stinner in [gh-105733](#).)

  - Soft-deprecate the `ARRAY()` function in favour of `type * length` multiplication. (Contributed by Victor Stinner in [gh-105733](#).)

- `decimal`:

  - Deprecate the non-standard and undocumented `Decimal` format specifier `'N'`, which is only supported in the `decimal` module's C implementation. (Contributed by Serhiy Storchaka in [gh-89902](#).)

- `dis`:

  - Deprecate the `HAVE_ARGUMENT` separator. Check membership in `hasarg` instead. (Contributed by Irit Katriel in [gh-109319](#).)

- `gettext`:

  - Deprecate non-integer numbers as arguments to functions and methods that consider plural forms in the `gettext` module, even if no translation was found. (Contributed by Serhiy Storchaka in [gh-88434](#).)

- `glob`:

  - Deprecate the undocumented `glob0()` and `glob1()` functions. Use `glob()` and pass a path-like object specifying the root directory to the *root_dir* parameter instead. (Contributed by Barney Gale in [gh-117337](#).)

- `http.server`:

  - Deprecate `CGIHTTPRequestHandler`, to be removed in Python 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. (Contributed by Gregory P. Smith in [gh-109096](#).)

  - Deprecate the `--cgi` flag to the **python -m http.server** command-line interface, to be removed in Python 3.15. (Contributed by Gregory P. Smith in [gh-109096](#).)

- `mimetypes`:

  - Soft-deprecate file path arguments to `guess_type()`, use `guess_file_type()` instead. (Contributed by Serhiy Storchaka in [gh-66543](#).)

- `re`:

  - Deprecate passing the optional *maxsplit*, *count*, or *flags* arguments as positional arguments to the module-level `split()`, `sub()`, and `subn()` functions. These parameters will become keyword-only in a future version of Python. (Contributed by Serhiy Storchaka in [gh-56166](#).)

- `pathlib`:

  - Deprecate `PurePath.is_reserved()`, to be removed in Python 3.15. Use `os.path.isreserved()` to detect reserved paths on Windows. (Contributed by Barney Gale in [gh-88569](#).)

- `platform`:

  - Deprecate `java_ver()`, to be removed in Python 3.15. This function is only useful for Jython support, has a confusing API, and is largely untested. (Contributed by Nikita Sobolev in [gh-116349](#).)

- `pydoc`:

– Deprecate the undocumented `ispackage()` function. (Contributed by Zackery Spytz in gh-64020.)

- `sqlite3`:

  – Deprecate passing more than one positional argument to the `connect()` function and the `Connection` constructor. The remaining parameters will become keyword-only in Python 3.15. (Contributed by Erlend E. Aasland in gh-107948.)

  – Deprecate passing name, number of arguments, and the callable as keyword arguments for `Connection.create_function()` and `Connection.create_aggregate()` These parameters will become positional-only in Python 3.15. (Contributed by Erlend E. Aasland in gh-108278.)

  – Deprecate passing the callback callable by keyword for the `set_authorizer()`, `set_progress_handler()`, and `set_trace_callback()` `Connection` methods. The call-back callables will become positional-only in Python 3.15. (Contributed by Erlend E. Aasland in gh-108278.)

- `sys`:

  – Deprecate the `_enablelegacywindowsfsencoding()` function, to be removed in Python 3.16. Use the `PYTHONLEGACYWINDOWSFSENCODING` environment variable instead. (Contributed by Inada Naoki in gh-73427.)

- `tarfile`:

  – Deprecate the undocumented and unused `TarFile.tarfile` attribute, to be removed in Python 3.16. (Contributed in gh-115256.)

- `traceback`:

  – Deprecate the `TracebackException.exc_type` attribute. Use `TracebackException.exc_type_str` instead. (Contributed by Irit Katriel in gh-112332.)

- `typing`:

  – Deprecate the undocumented keyword argument syntax for creating `NamedTuple` classes (e.g. `Point = NamedTuple("Point", x=int, y=int)`), to be removed in Python 3.15. Use the class-based syntax or the functional syntax instead. (Contributed by Alex Waygood in gh-105566.)

  – Deprecate omitting the *fields* parameter when creating a `NamedTuple` or `typing.TypedDict` class, and deprecate passing `None` to the *fields* parameter of both types. Python 3.15 will require a valid sequence for the *fields* parameter. To create a NamedTuple class with zero fields, use `class NT(NamedTuple): pass` or `NT = NamedTuple("NT", ())`. To create a TypedDict class with zero fields, use `class TD(TypedDict): pass` or `TD = TypedDict("TD", {})`. (Contributed by Alex Waygood in gh-105566 and gh-105570.)

  – Deprecate the `typing.no_type_check_decorator()` decorator function, to be removed in in Python 3.15. After eight years in the `typing` module, it has yet to be supported by any major type checker. (Contributed by Alex Waygood in gh-106309.)

  – Deprecate `typing.AnyStr`. In Python 3.16, it will be removed from `typing.__all__`, and a `DeprecationWarning` will be emitted at runtime 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_read` and `Wave_write` classes, to be removed in Python 3.15. (Contributed by Victor Stinner in gh-105096.)

8.1 Python 3.14 中待移除的項目

- `argparse`: `argparse.BooleanOptionalAction` 的 *type*、*choices* 和 *metavar* 參數已被棄用，將在 3.14 中移除。(由 Nikita Sobolev 於 gh-92248 貢獻。)

- `ast`: 自 Python 3.8 起，下列功能已在文件中被棄用，現在在存取或使用時會於 runtime 發出 `DeprecationWarning`，並將在 Python 3.14 中移除:

- ast.Num

- ast.Str

- ast.Bytes

- ast.NameConstant

- ast.Ellipsis

請改用 ast.Constant。(由 Serhiy Storchaka 於 gh-90953 貢獻。)

- asyncio:

  - 已 F 用 F 將在 Python 3.14 中移除的 child watcher 類 F: MultiLoopChildWatcher、FastChildWatcher、AbstractChildWatcher 和 SafeChildWatcher。(由 Kumar Aditya 於 gh-94597 貢獻。)

  - asyncio.set_child_watcher()、asyncio.get_child_watcher()、asyncio.AbstractEventLoopPolicy.set_child_watcher() 和 asyncio.AbstractEventLoopPolicy.get_child_watcher() 已被 F 用 F 將在 Python 3.14 中移除。(由 Kumar Aditya 於 gh-94597 貢獻。)

  - 預設事件 F 圈策略的 get_event_loop() 方法現在會在 F 有設定目前事件 F 圈且 F 定建立一個時發出 DeprecationWarning。(由 Serhiy Storchaka 和 Guido van Rossum 於 gh-100160 貢獻。)

- collections.abc: 已 F 用 ByteString。請改用 Sequence 或 Buffer。在 typing 中使用時，請改用聯集，如 bytes | bytearray，或 collections.abc.Buffer。(由 Shantanu Jain 於 gh-91896 貢獻。)

- email: 已 F 用 email.utils.localtime() 中的 *isdst* 參數。(由 Alan Williams 於 gh-72346 貢獻。)

- importlib.abc 的已 F 用類 F:

  - importlib.abc.ResourceReader

  - importlib.abc.Traversable

  - importlib.abc.TraversableResources

  請改用 importlib.resources.abc 類 F:

  - importlib.resources.abc.Traversable

  - importlib.resources.abc.TraversableResources

  (由 Jason R. Coombs 和 Hugo van Kemenade 貢獻於 gh-93963。)

- itertools 有不以文件記 F、效率低下、過去常有 bug 且不一致的 copy、deepcopy 和 pickle 操作支援。將在 3.14 中移除以大幅 F 少程式碼量和維護負擔。(由 Raymond Hettinger 於 gh-101588 貢獻。)

- multiprocessing: 預設的 F 動方法將在 Linux、BSD 和其他非 macOS POSIX 平台上更改 F 更安全的方法，目前 'fork' 是預設值 (gh-84559)。對此增加一個 runtime 警告被認 F 太過擾人，因 F 大多數程式碼不會在意。請使用 get_context() 或 set_start_method() API 來明確指定你的程式碼何時需要 'fork'。請參 F multiprocessing-start-methods。

- pathlib: 已 F 用 is_relative_to() 和 relative_to(): 額外引數的傳遞已被 F 用。

- pkgutil: find_loader() 和 get_loader() 現在會引發 DeprecationWarning; 請改用 importlib.util.find_spec()。(由 Nikita Sobolev 於 gh-97850 貢獻。)

- pty:

  - master_open(): 請用 pty.openpty()。

  - slave_open(): 請用 pty.openpty()。

- sqlite3:

  - version 和 version_info。

  - execute() 和 executemany()，如果使用 named placeholders 且 *parameters* 是序列而不是 dict。

- `typing`: 自 Python 3.9 起已被棄用的 `ByteString` 現在在使用時會發出 `DeprecationWarning`。

- `urllib`: `urllib.parse.Quoter` 已被棄用：它並非預期的公開 API。(由 Gregory P. Smith 於 gh-88168 貢獻。)

8.2 Python 3.15 中待移除的項目

- 引入系統 (import system)：

  - 在模組上設定 `__cached__` 而沒有設定 `__spec__.cached` 的做法已被棄用。在 Python 3.15 中，引入系統或標準函式庫將不再設定或考慮 `__cached__`。(gh-97879)

  - 在模組上設定 `__package__` 而沒有設定 `__spec__.parent` 的做法已被棄用。在 Python 3.15 中，引入系統或標準函式庫將不再設定或考慮 `__package__`。(gh-97879)

- `ctypes`:

  - 自 Python 3.13 起，未記錄的 `ctypes.SetPointerType()` 函式已被棄用。

- `http.server`:

  - 過時且很少使用的 `CGIHTTPRequestHandler` 自 Python 3.13 起已被棄用。不存在直接的替代。任何東西都比 CGI 更好地將 Web 伺服器與請求處理程序介接起來。

  - 自 Python 3.13 起，**python -m http.server** 命令列介面的 `--cgi` 旗標已被棄用。

- `importlib`:

  - `load_module()` method：請改用 `exec_module()`。

- `locale`:

  - `getdefaultlocale()` 已在 Python 3.11 中被棄用，原本計劃在 Python 3.13 中移除 (gh-90817)，但被延後至 Python 3.15。請改用 `getlocale()`、`setlocale()` 和 `getencoding()`。(由 Hugo van Kemenade 於 gh-111187 貢獻。)

- `pathlib`:

  - `PurePath.is_reserved()` 已自 Python 3.13 被棄用。請用 `os.path.isreserved()` 來偵測 Windows 上的保留路徑。

- `platform`:

  - 自 Python 3.13 起，`java_ver()` 已被棄用。此函式僅對 Jython 支援有用，具有令人困惑的 API，基本上未經測試。

- `sysconfig`:

  - The *check_home* argument of `sysconfig.is_python_build()` has been deprecated since Python 3.12.

- `threading`:

  - `RLock()` 在 Python 3.15 中將不接受任何引數。自 Python 3.14 起，傳遞任何引數的用法已被棄用，因為 Python 版本不允許任何引數，但 C 版本允許任意數量的位置或關鍵字引數，並忽略每個引數。

- `types`:

  - `types.CodeType`: 自 3.10 起，存取 `co_lnotab` 已在 **PEP 626** 中被棄用，並計劃在 3.12 中移除，但只在 3.12 中於適當時發出 `DeprecationWarning`。可能在 3.15 中移除。(由 Nikita Sobolev 於 gh-101866 貢獻。)

- `typing`:

  - 用於建立 `NamedTuple` 類別的未以文件記錄之關鍵字引數語法 (`Point = NamedTuple("Point", x=int, y=int)`) 已自 Python 3.13 棄用。請改用基於類別的語法或函式語法 (functional syntax)。

  - 自 Python 3.13 起，`typing.no_type_check_decorator()` 裝飾器函式已被棄用。在 `typing` 模組中使用了八年之後，它尚未得到任何主要型別檢查器的支援。

- `wave`:
  - 已□用 `Wave_read` 和 `Wave_write` 類□的 `getmark()`、`setmark()` 和 `getmarkers()` 方法自 Python 3.13 被□用。

8.3 Python 3.16 中待移除的項目

- 引入系統 (import system)：
  - 在模組上設定 `__loader__` 而□有設定 `__spec__.loader` 的做法將於 Python 3.16 被□用。在 Python 3.16 中，引入系統或標準函式庫將不再設定或考慮 `__loader__`。

- `array`：
  - 自 Python 3.3 起，`'u'` 格式碼 (`wchar_t`) 在文件中已被□用，自 Python 3.13 起在 runtime 已被□用。請使用 `'w'` 格式碼 (`Py_UCS4`) 來取代 Unicode 字元。

- `asyncio`：
  - `asyncio.iscoroutinefunction()` 已被□用□將在 Python 3.16 中移除，請改用 `inspect.iscoroutinefunction()`。（由 Jiahao Li 和 Kumar Aditya 於 gh-122875 貢獻。）

- `builtins`：
  - 自 Python 3.12 起，布林型□的位元反轉 `~True` 或 `~False` 已被□用，因□它會□生不預期且不直觀的結果 (`-2` 和 `-1`)。使用 `not x` 代替布林值的邏輯否定。在極少數情□下，你需要對底層的整數進行位元反轉，請明確轉□□ `~int(x)` (`~int(x)`)。

- `shutil`：
  - 自 Python 3.14 起，`ExecError` 例外已被□用。自 Python 3.4 以來，它尚未被 `shutil` 中的任何函式使用，現在是 `RuntimeError` 的□名。

- `symtable`：
  - 自 Python 3.14 起，`Class.get_methods` 方法已被□用。

- `sys`：
  - 自 Python 3.13 起，`_enablelegacywindowsfsencoding()` 函式已被□用。請改用 `PYTHONLEGACYWINDOWSFSENCODING` 環境變數。

- `tarfile`：
  - 自 Python 3.13 起，未以文件記□和未被使用的 `TarFile.tarfile` 屬性已被□用。

8.4 未來版本中的待移除項目

以下 API 將在未來被移除，雖然目前尚未安排移除日期。

- `argparse`：已□用巢狀引數群組和巢狀互斥群組。

- `array` 的 `'u'` 格式碼 (gh-57281)

- `builtins`：
  - `bool(NotImplemented)`。
  - □生器：`throw(type, exc, tb)` 和 `athrow(type, exc, tb)` 簽名已被□用：請改用 `throw(exc)` 和 `athrow(exc)`，□單引數簽名。
  - 目前 Python 接受數值字面值後面立即接關鍵字，例如 `0in x`、`1or x`、`0if 1else 2`。它讓表達式模糊且容易混淆，如 `[0x1for x in y]`（可以解釋□ `[0x1 for x in y]` 或 `[0x1f or x in y]`）。如果數值字面值後立即接 `and`、`else`、`for`、`if`、`in`、`is` 和 `or` 之一的關鍵字，則會引發語法警告。在未來版本中，它將被更改□語法錯誤。(gh-87999)
  - `__index__()` 和 `__int__()` 方法回傳非 int 型□的支援：這些方法將需要回傳 int 的嚴格子類□實例。
  - 回傳 float 嚴格子類□ `__float__()` 方法的支援：這些方法將需要回傳 float 的實例。

- 回傳 complex 嚴格子類□ __complex__() 方法的支援：這些方法將需要回傳 complex 的實例。

- 將 int() 委派給 __trunc__() 方法。

- 在 complex() 建構子中將□數作□ *real* 或 *imag* 引數傳遞現在已被□用；它應該只作□單個位置引數傳遞。(由 Serhiy Storchaka 於 gh-109218 貢獻。)

- calendar：calendar.January 和 calendar.February 常數已被□用□被 calendar.JANUARY 和 calendar.FEBRUARY 取代。(由 Prince Roshan 於 gh-103636 貢獻。)

- codeobject.co_lnotab：請改用 codeobject.co_lines() 方法。

- datetime：

    - utcnow()：請改用 datetime.datetime.now(tz=datetime.UTC)。

    - utcfromtimestamp()：請改用 datetime.datetime.fromtimestamp(timestamp, tz=datetime.UTC)。

- gettext：□數值必須是整數。

- importlib：

    - cache_from_source() *debug_override* 參數已被□用：請改用 *optimization* 參數。

- importlib.metadata：

    - EntryPoints 元組介面。

    - 回傳值上的隱式 None。

- logging：自 Python 3.3 起，warn() 方法已被□用，請改用 warning()。

- mailbox：已□用 StringIO 輸入和文本模式，請改用 BytesIO 和二進位模式。

- os：在多執行緒行程中呼叫 os.register_at_fork()。

- pydoc.ErrorDuringImport：*exc_info* 參數的元組值已被□用，請用例外實例。

- re：現在對正規表示式中的數值群組參照和群組名稱用了更嚴格的規則。現在只有 ASCII 數碼序列被接受作□數值參照。位元組模式和替□字串中的群組名稱現在只能包含 ASCII 字母、數碼和底□。(由 Serhiy Storchaka 於 gh-91760 貢獻。)

- sre_compile、sre_constants 和 sre_parse 模組。

- shutil：rmtree() 的 *onerror* 參數在 Python 3.12 中已被□用；請改用 *onexc* 參數。

- ssl 選項和協定：

    - 不帶協定引數的 ssl.SSLContext 已被□用。

    - ssl.SSLContext: set_npn_protocols() 和 selected_npn_protocol() 已被□用：請改用 ALPN。

    - ssl.OP_NO_SSL* 選項

    - ssl.OP_NO_TLS* 選項

    - 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

- `threading` 方法：
  - `threading.Condition.notifyAll()`：請用 `notify_all()`。
  - `threading.Event.isSet()`：請用 `is_set()`。
  - `threading.Thread.isDaemon()`、`threading.Thread.setDaemon()`：請用 `threading.Thread.daemon` 屬性。
  - `threading.Thread.getName()`、`threading.Thread.setName()`：請用 `threading.Thread.name` 屬性。
  - `threading.currentThread()`：請用 `threading.current_thread()`。
  - `threading.activeCount()`：請用 `threading.active_count()`。
- `typing.Text` (gh-92332)。
- `unittest.IsolatedAsyncioTestCase`：從測試案例中回傳非 `None` 的值已被棄用。
- `urllib.parse` 已棄用函式：請改用 `urlparse()`。
  - `splitattr()`
  - `splithost()`
  - `splitnport()`
  - `splitpasswd()`
  - `splitport()`
  - `splitquery()`
  - `splittag()`
  - `splittype()`
  - `splituser()`
  - `splitvalue()`
  - `to_bytes()`
- `urllib.request`：呼叫請求的 `URLopener` 和 `FancyURLopener` 風格已被棄用。請改用更新的 `urlopen()` 函式和方法。
- `wsgiref`：`SimpleHandler.stdout.write()` 不應該進行部分寫入。
- `xml.etree.ElementTree`：已棄用對 `Element` 的真值測試。在未來版本中，它將始終回傳 `True`。請改用明確的 `len(elem)` 或 `elem is not None` 測試。
- `zipimport.zipimporter.load_module()` 已被棄用：請改用 `exec_module()`。

9 CPython Bytecode Changes

- The oparg of `YIELD_VALUE` is now `1` if the yield is part of a yield-from or await, and `0` otherwise. The oparg of `RESUME` was changed to add a bit indicating if the except-depth is 1, which is needed to optimize closing of generators. (Contributed by Irit Katriel in gh-111354.)

10 C API 變更

10.1 新增功能

- Add the PyMonitoring C API for generating **PEP 669** monitoring events:
  - `PyMonitoringState`
  - `PyMonitoring_FirePyStartEvent()`

- PyMonitoring_FirePyResumeEvent()

- PyMonitoring_FirePyReturnEvent()

- PyMonitoring_FirePyYieldEvent()

- PyMonitoring_FireCallEvent()

- PyMonitoring_FireLineEvent()

- PyMonitoring_FireJumpEvent()

- PyMonitoring_FireBranchEvent()

- PyMonitoring_FireCReturnEvent()

- PyMonitoring_FirePyThrowEvent()

- PyMonitoring_FireRaiseEvent()

- PyMonitoring_FireCRaiseEvent()

- PyMonitoring_FireReraiseEvent()

- PyMonitoring_FireExceptionHandledEvent()

- PyMonitoring_FirePyUnwindEvent()

- PyMonitoring_FireStopIterationEvent()

- PyMonitoring_EnterScope()

- PyMonitoring_ExitScope()

（由 Irit Katriel 在 gh-111997 中貢獻。）

- Add PyMutex, a lightweight mutex that occupies a single byte, and the new PyMutex_Lock() and PyMutex_Unlock() functions. PyMutex_Lock() will release the GIL (if currently held) if the operation needs to block. (Contributed by Sam Gross in gh-108724.)

- Add the PyTime C API to provide access to system clocks:

  - PyTime_t。

  - PyTime_MIN and PyTime_MAX.

  - PyTime_AsSecondsDouble()。

  - PyTime_Monotonic()。

  - PyTime_MonotonicRaw()。

  - PyTime_PerfCounter()。

  - PyTime_PerfCounterRaw()。

  - PyTime_Time()。

  - PyTime_TimeRaw()。

  （由 Victor Stinner 和 Petr Viktorin 在 gh-110850 中貢獻。）

- Add the PyDict_ContainsString() function with the same behavior 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 the PyDict_GetItemRef() and PyDict_GetItemStringRef() functions, which behave similarly to PyDict_GetItemWithError(), but return a strong reference instead of a borrowed reference. Moreover, these functions return -1 on error, removing the need to check PyErr_Occurred(). (Contributed by Victor Stinner in gh-106004.)

- Add the PyDict_SetDefaultRef() function, which behaves similarly to PyDict_SetDefault(), but returns a strong reference instead of a borrowed reference. This function returns -1 on error, 0 on insertion, and 1 if the key was already present in the dictionary. (Contributed by Sam Gross in gh-112066.)

- Add the `PyDict_Pop()` and `PyDict_PopString()` functions to remove a key from a dictionary and optionally return the removed value. This is similar to `dict.pop()`, though there is no default value, and `KeyError` is not raised for missing keys. (Contributed by Stefan Behnel and Victor Stinner in gh-111262.)

- Add the `PyMapping_GetOptionalItem()` and `PyMapping_GetOptionalItemString()` functions as alternatives to `PyObject_GetItem()` and `PyMapping_GetItemString()` respectively. The new functions do not raise `KeyError` if the requested key is missing from the mapping. These variants are more convenient and faster if a missing key should not be treated as a failure. (Contributed by Serhiy Storchaka in gh-106307.)

- Add the `PyObject_GetOptionalAttr()` and `PyObject_GetOptionalAttrString()` functions as alternatives to `PyObject_GetAttr()` and `PyObject_GetAttrString()` respectively. The new functions do not raise `AttributeError` if the requested attribute is not found on the object. 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 the `PyErr_FormatUnraisable()` function as an extension to `PyErr_WriteUnraisable()` that allows customizing the warning message. (Contributed by Serhiy Storchaka in gh-108082.)

- Add new functions that return a strong reference instead of a borrowed reference for frame locals, globals, and builtins, as part of *PEP 667*:

  - `PyEval_GetFrameBuiltins()` 取代 `PyEval_GetBuiltins()`

  - `PyEval_GetFrameGlobals()` 取代 `PyEval_GetGlobals()`

  - `PyEval_GetFrameLocals()` 取代 `PyEval_GetLocals()`

  （由 Mark Shannon 和 Tian Gao 在 gh-74929 中貢獻。）

- Add the `Py_GetConstant()` and `Py_GetConstantBorrowed()` functions to get strong or borrowed references to constants. For example, `Py_GetConstant(Py_CONSTANT_ZERO)` returns a strong reference to the constant zero. (Contributed by Victor Stinner in gh-115754.)

- Add the `PyImport_AddModuleRef()` function as a replacement for `PyImport_AddModule()` that returns a strong reference instead of a borrowed reference. (Contributed by Victor Stinner in gh-105922.)

- Add the `Py_IsFinalizing()` function to check whether the main Python interpreter is shutting down. (Contributed by Victor Stinner in gh-108014.)

- Add the `PyList_GetItemRef()` function as a replacement for `PyList_GetItem()` that returns a strong reference instead of a borrowed reference. (Contributed by Sam Gross in gh-114329.)

- Add the `PyList_Extend()` and `PyList_Clear()` functions, mirroring the Python `list.extend()` and `list.clear()` methods. (Contributed by Victor Stinner in gh-111138.)

- Add the `PyLong_AsInt()` function. It behaves similarly to `PyLong_AsLong()`, but stores the result in a C `int` instead of a C `long`. (Contributed by Victor Stinner in gh-108014.)

- Add the `PyLong_AsNativeBytes()`, `PyLong_FromNativeBytes()`, and `PyLong_FromUnsignedNativeBytes()` functions to simplify converting between native integer types and Python `int` objects. (Contributed by Steve Dower in gh-111140.)

- Add `PyModule_Add()` function, which is similar to `PyModule_AddObjectRef()` and `PyModule_AddObject()`, but always steals a reference to the value. (Contributed by Serhiy Storchaka in gh-86493.)

- Add the `PyObject_GenericHash()` function that implements the default hashing function of a Python object. (Contributed by Serhiy Storchaka in gh-113024.)

- Add the `Py_HashPointer()` function to hash a raw pointer. (Contributed by Victor Stinner in gh-111545.)

- Add the `PyObject_VisitManagedDict()` and `PyObject_ClearManagedDict()` functions. which must be called by the traverse and clear functions of a type using the `Py_TPFLAGS_MANAGED_DICT` flag. The pythoncapi-compat project can be used to use these functions with Python 3.11 and 3.12. (Contributed by Victor Stinner in gh-107073.)

- Add the `PyRefTracer_SetTracer()` and `PyRefTracer_GetTracer()` functions, which enable tracking object creation and destruction in the same way that the `tracemalloc` module does. (Contributed by Pablo Galindo in gh-93502.)

- Add the `PySys_AuditTuple()` function as an alternative to `PySys_Audit()` that takes event arguments as a Python `tuple` object. (Contributed by Victor Stinner in gh-85283.)

- Add the `PyThreadState_GetUnchecked()` function as an alternative to `PyThreadState_Get()` that doesn't kill the process with a fatal error if it is `NULL`. The caller is responsible for checking if the result is `NULL`. (Contributed by Victor Stinner in gh-108867.)

- Add the `PyType_GetFullyQualifiedName()` function to get the type's fully qualified name. The module name is prepended if `type.__module__` is a string and is not equal to either `'builtins'` or `'__main__'`. (Contributed by Victor Stinner in gh-111696.)

- Add the `PyType_GetModuleName()` function to get the type's module name. This is equivalent to getting the `type.__module__` attribute. (Contributed by Eric Snow and Victor Stinner in gh-111696.)

- Add the `PyUnicode_EqualToUTF8AndSize()` and `PyUnicode_EqualToUTF8()` functions to compare a Unicode object with a `const char*` UTF-8 encoded string and `1` if they are equal or `0` otherwise. These functions do not raise exceptions. (Contributed by Serhiy Storchaka in gh-110289.)

- Add the `PyWeakref_GetRef()` function as an alternative to `PyWeakref_GetObject()` that returns a strong reference or `NULL` if the referent is no longer live. (Contributed by Victor Stinner in gh-105927.)

- Add fixed variants of functions which silently ignore errors:

    - `PyObject_HasAttrWithError()` 取代 `PyObject_HasAttr()`。

    - `PyObject_HasAttrStringWithError()` 取代 `PyObject_HasAttrString()`。

    - `PyMapping_HasKeyWithError()` 取代 `PyMapping_HasKey()`。

    - `PyMapping_HasKeyStringWithError()` 取代 `PyMapping_HasKeyString()`。

    The new functions return `-1` for errors and the standard `1` for true and `0` for false.

    （由 Serhiy Storchaka 在 gh-108511 中貢獻。）

10.2 Changed C APIs

- The *keywords* parameter of `PyArg_ParseTupleAndKeywords()` and `PyArg_VaParseTupleAndKeywords()` now has type `char *const*` in C and `const char *const*` in C++, instead of `char**`. In C++, this makes these functions compatible with arguments of type `const char *const*`, `const char**`, or `char *const*` without an explicit type cast. In C, the functions only support arguments of type `char *const*`. This can be overridden with the `PY_CXX_CONST` macro. (Contributed by Serhiy Storchaka in gh-65210.)

- `PyArg_ParseTupleAndKeywords()` now supports non-ASCII keyword parameter names. (Contributed by Serhiy Storchaka in gh-110815.)

- The `PyCode_GetFirstFree()` function is now unstable API and is now named `PyUnstable_Code_GetFirstFree()`. (Contributed by Bogdan Romanyuk in gh-115781.)

- The `PyDict_GetItem()`, `PyDict_GetItemString()`, `PyMapping_HasKey()`, `PyMapping_HasKeyString()`, `PyObject_HasAttr()`, `PyObject_HasAttrString()`, and `PySys_GetObject()` functions, each of which clears all errors which occurred when calling them now reports these errors using `sys.unraisablehook()`. You may replace them with other functions as recommended in the documentation. (Contributed by Serhiy Storchaka in gh-106672.)

- Add support for the `%T`, `%#T`, `%N` and `%#N` formats to `PyUnicode_FromFormat()`:

    - `%T`: Get the fully qualified name of an object type

    - `%#T`: As above, but use a colon as the separator

    - `%N`: Get the fully qualified name of a type

    - `%#N`: As above, but use a colon as the separator

See **PEP 737** for more information. (Contributed by Victor Stinner in gh-111696.)

- 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.)

- 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. (Contributed by Victor Stinner in gh-106168.)

10.3 Limited C API Changes

- The following functions are now included in the Limited C API:

  - `PyMem_RawMalloc()`

  - `PyMem_RawCalloc()`

  - `PyMem_RawRealloc()`

  - `PyMem_RawFree()`

  - `PySys_Audit()`

  - `PySys_AuditTuple()`

  - `PyType_GetModuleByDef()`

  （由 Victor Stinner 貢獻於 gh-85283、gh-85283 和 gh-116936。）

- Python built with `--with-trace-refs` (tracing references) now supports the Limited API. (Contributed by Victor Stinner in gh-108634.)

10.4 移除的 C API

- Remove several functions, macros, variables, etc with names prefixed by `_Py` or `_PY` (which are considered private). If your project is affected by one of these removals and you believe 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 old buffer protocols deprecated in Python 3.0. Use bufferobjects instead.

  - `PyObject_CheckReadBuffer()`: Use `PyObject_CheckBuffer()` to test whether 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()`: 請改用 `PyObject_GetBuffer()` 和 `PyBuffer_Release()`。

```
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()`: 請改用 `PyObject_GetBuffer()` 和 `PyBuffer_Release()`。

```
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);
```

  （由 Inada Naoki 貢獻於 gh-85275。）

- Remove various functions deprecated in Python 3.9:

  - `PyEval_CallObject()`、`PyEval_CallObjectWithKeywords()`: 請 改 用 `PyObject_CallNoArgs()` 或 `PyObject_Call()`。

    > ⚠️ **警告**
    >
    > In `PyObject_Call()`, positional arguments must be a `tuple` and must not be `NULL`, and keyword arguments must be a `dict` or `NULL`, whereas the removed functions checked argument types 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()`: 請改用 `PyObject_CallFunction()`。

  - `PyEval_CallMethod()`: 請改用 `PyObject_CallMethod()`。

  - `PyCFunction_Call()`: 請改用 `PyCFunction_Call()`。

  （由 Victor Stinner 貢獻於 gh-105107。）

- Remove the following old functions to configure the Python initialization, deprecated in Python 3.11:

  - `PySys_AddWarnOptionUnicode()`: 請改用 `PyConfig.warnoptions`。

  - `PySys_AddWarnOption()`: 請改用 `PyConfig.warnoptions`。

  - `PySys_AddXOption()`: 請改用 `PyConfig.xoptions`。

  - `PySys_HasWarnOptions()`: 請改用 `PyConfig.xoptions`。

  - `PySys_SetPath()`: 請改用 `PyConfig.module_search_paths`。

  - `Py_SetPath()`: 請改用 `PyConfig.module_search_paths`。

  - `Py_SetStandardStreamEncoding()`: 請改用 `PyConfig.stdio_encoding` Ｆ設定可能的 `PyConfig.legacy_windows_stdio`（在 Windows 上）。

  - `_Py_SetProgramFullPath()`: 請改用 `PyConfig.executable`。

  請改用 Python 初始化設定的新 `PyConfig` API (**PEP 587**)，這是在 Python 3.8 中新增的。（由 Victor Stinner 於 gh-105145 貢獻。）

- 移除 `PyEval_AcquireLock()` 和 `PyEval_ReleaseLock()` 函式，這些函式在 Python 3.2 中已被廢Ｆ。它們不會更新目前的執行緒狀態。可以用以下函式取代:

  - `PyEval_SaveThread()` 和 `PyEval_RestoreThread()`;

  - 低階的 `PyEval_AcquireThread()` 和 `PyEval_RestoreThread()`;

  - 或 `PyGILState_Ensure()` 與 `PyGILState_Release()`。

  （由 Victor Stinner 貢獻於 gh-105182。）

- Remove the `PyEval_ThreadsInitialized()` function, deprecated in Python 3.9. Since Python 3.7, `Py_Initialize()` always creates the GIL: calling `PyEval_InitThreads()` does nothing and `PyEval_ThreadsInitialized()` always returns non-zero. (Contributed by Victor Stinner in gh-105182.)

- Remove the `_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.)

- Remove the private `_PyObject_FastCall()` function: use `PyObject_Vectorcall()` which is available since Python 3.8 (**PEP 590**). (Contributed by Victor Stinner in gh-106023.)

- 移除只包含私有函式的 `cpython/pytime.h` 標頭檔。（由 Victor Stinner 於 gh-106316 貢獻。）

- Remove the undocumented `PY_TIMEOUT_MAX` constant from the limited C API. (Contributed by Victor Stinner in gh-110014.)

- Remove the old trashcan macros `Py_TRASHCAN_SAFE_BEGIN` and `Py_TRASHCAN_SAFE_END`. Replace both with the new macros `Py_TRASHCAN_BEGIN` and `Py_TRASHCAN_END`. (Contributed by Irit Katriel in gh-105111.)

10.5 器用的 C API

- Deprecate old Python initialization functions:

  - `PySys_ResetWarnOptions()`：請改Ｅ清除 `sys.warnoptions` 和 `warnings.filters`。

  - `Py_GetExecPrefix()`：請改用 `sys.exec_prefix`。

  - `Py_GetPath()`：請改用 `sys.path`。

  - `Py_GetPrefix()`：請改用 `sys.prefix`。

  - `Py_GetProgramFullPath()`：請改用 `sys.executable`。

  - `Py_GetProgramName()`：請改用 `sys.executable`。

  - `Py_GetPythonHome()`：請改用 `PyConfig.home` 或 `PYTHONHOME` 環境變數。

  （由 Victor Stinner 在 gh-105145 中貢獻。）

- Soft deprecate the `PyEval_GetBuiltins()`, `PyEval_GetGlobals()`, and `PyEval_GetLocals()` functions, which return a borrowed reference. (Soft deprecated as part of **PEP 667**.)

- Deprecate the `PyImport_ImportModuleNoBlock()` function, which is just an alias to `PyImport_ImportModule()` since Python 3.3. (Contributed by Victor Stinner in gh-105396.)

- Soft deprecate the `PyModule_AddObject()` function. It should be replaced with `PyModule_Add()` or `PyModule_AddObjectRef()`. (Contributed by Serhiy Storchaka in gh-86493.)

- Deprecate the old `Py_UNICODE` and `PY_UNICODE_TYPE` types and the `Py_UNICODE_WIDE` define. Use the `wchar_t` type directly 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 the `PyWeakref_GetObject()` and `PyWeakref_GET_OBJECT()` functions, which return a borrowed reference. Replace them with the new `PyWeakref_GetRef()` function, which 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.)

Python 3.14 中待移除的項目

- `PyDictObject` 中的 `ma_version_tag` 欄位，用於擴充模組 (**PEP 699**；gh-101193)。

- 使用可變基底建立不可變型Ｅ (gh-95388)。

- 設定 Python 初始化的函式，Python 3.11 中已被Ｅ用：

  - `PySys_SetArgvEx()`：請改以 `PyConfig.argv` 設定。

  - `PySys_SetArgv()`：請改以 `PyConfig.argv` 設定。

  - `Py_SetProgramName()`\`()：請改以 `PyConfig.program_name` 設定。

  - `Py_SetPythonHome()`：請改以 `PyConfig.home` 設定。

  `Py_InitializeFromConfig()` API 應該與 `PyConfig` 一起使用。

- 全域設定變數：

  - `Py_DebugFlag`：請改用 `PyConfig.parser_debug`。

  - `Py_VerboseFlag`：請改用 `PyConfig.verbose`。

  - `Py_QuietFlag`：請改用 `PyConfig.quiet`。

  - `Py_InteractiveFlag`：請改用 `PyConfig.interactive`。

  - `Py_InspectFlag`：請改用 `PyConfig.inspect`。

- `Py_OptimizeFlag`：請改用 `PyConfig.optimization_level`。

- `Py_NoSiteFlag`：請改用 `PyConfig.site_import`。

- `Py_BytesWarningFlag`：請改用 `PyConfig.bytes_warning`。

- `Py_FrozenFlag`：請改用 `PyConfig.pathconfig_warnings`。

- `Py_IgnoreEnvironmentFlag`：請改用 `PyConfig.use_environment`。

- `Py_DontWriteBytecodeFlag`：請改用 `PyConfig.write_bytecode`。

- `Py_NoUserSiteDirectory`：請改用 `PyConfig.user_site_directory`。

- `Py_UnbufferedStdioFlag`：請改用 `PyConfig.buffered_stdio`。

- `Py_HashRandomizationFlag`：請改用 `PyConfig.use_hash_seed` 和 `PyConfig.hash_seed`。

- `Py_IsolatedFlag`：請改用 `PyConfig.isolated`。

- `Py_LegacyWindowsFSEncodingFlag`：請改用 `PyPreConfig.legacy_windows_fs_encoding`。

- `Py_LegacyWindowsStdioFlag`：請改用 `PyConfig.legacy_windows_stdio`。

- `Py_FileSystemDefaultEncoding`：請改用 `PyConfig.filesystem_encoding`。

- `Py_HasFileSystemDefaultEncoding`：請改用 `PyConfig.filesystem_encoding`。

- `Py_FileSystemDefaultEncodeErrors`：請改用 `PyConfig.filesystem_errors`。

- `Py_UTF8Mode`：請改用 `PyPreConfig.utf8_mode`。（請見 `Py_PreInitialize()`）

`Py_InitializeFromConfig()` API 應該與 `PyConfig` 一起使用。

Python 3.15 中待移除的項目

- `libmpdecimal` 的打包副本 (bundled copy)。

- `PyImport_ImportModuleNoBlock()`：請改用 `PyImport_ImportModule()`。

- `PyWeakref_GetObject()` 和 `PyWeakref_GET_OBJECT()`：請改用 `PyWeakref_GetRef()`。

- `Py_UNICODE` 型別與 `Py_UNICODE_WIDE` 巨集：請改用 `wchar_t`。

- Python 初始化函式：

  - `PySys_ResetWarnOptions()`：請改為清除 `sys.warnoptions` 和 `warnings.filters`。

  - `Py_GetExecPrefix()`：請改用 `sys.base_exec_prefix` 與 `sys.exec_prefix`。

  - `Py_GetPath()`：請改用 `sys.path`。

  - `Py_GetPrefix()`：請改用 `sys.base_prefix` 與 `sys.prefix`。

  - `Py_GetProgramFullPath()`：請改用 `sys.executable`。

  - `Py_GetProgramName()`：請改用 `sys.executable`。

  - `Py_GetPythonHome()`：請改用 `PyConfig.home` 或 `PYTHONHOME` 環境變數。

未來版本中的待移除項目

下列 API 已被棄用且將會被移除，不過目前尚未訂定移除日期。

- `Py_TPFLAGS_HAVE_FINALIZE`：自 Python 3.8 起不再需要

- `PyErr_Fetch()`：請改用 `PyErr_GetRaisedException()`。

- `PyErr_NormalizeException()`：請改用 `PyErr_GetRaisedException()`。

- `PyErr_Restore()`：請改用 `PyErr_SetRaisedException()`。

- `PyModule_GetFilename()`：請改用 `PyModule_GetFilenameObject()`。

- `PyOS_AfterFork()`: 請改用 `PyOS_AfterFork_Child()`。

- `PySlice_GetIndicesEx()`: 請改用 `PySlice_Unpack()` 和 `PySlice_AdjustIndices()`。

- `PyUnicode_AsDecodedObject()`: 請改用 `PyCodec_Decode()`。

- `PyUnicode_AsDecodedUnicode()`: 請改用 `PyCodec_Decode()`。

- `PyUnicode_AsEncodedObject()`: 請改用 `PyCodec_Encode()`。

- `PyUnicode_AsEncodedUnicode()`: 請改用 `PyCodec_Encode()`。

- `PyUnicode_READY()`: 自 Python 3.12 起不再需要

- `PyErr_Display()`: 請改用 `PyErr_DisplayException()`。

- `_PyErr_ChainExceptions()`: 請改用 `_PyErr_ChainExceptions1`。

- `PyBytesObject.ob_shash` 成員: 請改Ⓕ呼叫 `PyObject_Hash()`。

- `PyDictObject.ma_version_tag` 成員。

- 執行緒局部儲存 (Thread Local Storage, TLS) API:

  - `PyThread_create_key()`: 請改用 `PyThread_tss_alloc()`。

  - `PyThread_delete_key()`: 請改用 `PyThread_tss_free()`。

  - `PyThread_set_key_value()`: 請改用 `PyThread_tss_set()`。

  - `PyThread_get_key_value()`: 請改用 `PyThread_tss_get()`。

  - `PyThread_delete_key_value()`: 請改用 `PyThread_tss_delete()`。

  - `PyThread_ReInitTLS()`: 自 Python 3.7 起不再需要。

11 建置變更

- `arm64-apple-ios` and `arm64-apple-ios-simulator` are both now **PEP 11** tier 3 platforms. (*PEP 730* written and implementation contributed by Russell Keith-Magee in gh-114099.)

- `aarch64-linux-android` and `x86_64-linux-android` are both now **PEP 11** tier 3 platforms. (*PEP 738* written and implementation contributed by Malcolm Smith in gh-116622.)

- `wasm32-wasi` is now a **PEP 11** tier 2 platform. (Contributed by Brett Cannon in gh-115192.)

- `wasm32-emscripten` is no longer a **PEP 11** supported platform. (Contributed by Brett Cannon in gh-115192.)

- Building CPython now requires a compiler with support for the C11 atomic library, GCC built-in atomic functions, or MSVC interlocked intrinsics.

- Autoconf 2.71 and aclocal 1.16.5 are now required to regenerate the `configure` script. (Contributed by Christian Heimes in gh-89886 and by Victor Stinner in gh-112090.)

- SQLite 3.15.2 or newer is required to build the `sqlite3` extension module. (Contributed by Erlend Aasland in gh-105875.)

- CPython now bundles the mimalloc library by default. It is licensed under the MIT license; see mimalloc license. The bundled mimalloc has custom changes, see gh-113141 for details. (Contributed by Dino Viehland in gh-109914.)

- The `configure` option `--with-system-libmpdec` now defaults to `yes`. The bundled copy of `libmpdecimal` will be removed in Python 3.15.

- Python built with `configure --with-trace-refs` (tracing references) is now ABI compatible with the Python release build and debug build. (Contributed by Victor Stinner in gh-108634.)

- On POSIX systems, the pkg-config (`.pc`) filenames now include the ABI flags. For example, the free-threaded build generates `python-3.13t.pc` and the debug build generates `python-3.13d.pc`.

- The `errno`, `fcntl`, `grp`, `md5`, `pwd`, `resource`, `termios`, `winsound`, `_ctypes_test`, `_multiprocessing.posixshmem`, `_scproxy`, `_stat`, `_statistics`, `_testconsole`, `_testimportmultiple` and `_uuid` C extensions are now built with the limited C API. (Contributed by Victor Stinner in gh-85283.)

12 移植至 Python 3.13

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

12.1 Python API 的變更

- *PEP 667* introduces several changes to the semantics of `locals()` and `f_locals`:

  - Calling `locals()` in an optimized scope now produces an independent snapshot on each call, and hence no longer implicitly updates previously returned references. Obtaining the legacy CPython behavior now requires explicit calls to update the initially returned dictionary with the results of subsequent calls to `locals()`. Code execution functions that implicitly target `locals()` (such as `exec` and `eval`) must be passed an explicit namespace to access their results in an optimized scope. (Changed as part of **PEP 667**.)

  - Calling `locals()` from a comprehension at module or class scope (including via `exec` or `eval`) once more behaves as if the comprehension were running as an independent nested function (i.e. the local variables from the containing scope are not included). In Python 3.12, this had changed to include the local variables from the containing scope when implementing **PEP 709**. (Changed as part of **PEP 667**.)

  - Accessing `FrameType.f_locals` in an optimized scope now returns a write-through proxy rather than a snapshot that gets updated at ill-specified times. If a snapshot is desired, it must be created explicitly with `dict` or the proxy's `.copy()` method. (Changed as part of **PEP 667**.)

- `functools.partial` now emits a `FutureWarning` when used as a method. The behavior will change in future Python versions. Wrap it in `staticmethod()` if you want to preserve the old behavior. (Contributed by Serhiy Storchaka in gh-121027.)

- An `OSError` is now raised by `getpass.getuser()` for any failure to retrieve a username, instead of `ImportError` on non-Unix platforms or `KeyError` on Unix platforms where the password database is empty.

- The value of the `mode` attribute of `gzip.GzipFile` is now a string (`'rb'` or `'wb'`) instead of an integer (`1` or `2`). The value of the `mode` attribute of the readable file-like object returned by `zipfile.ZipFile.open()` is now `'rb'` instead of `'r'`. (Contributed by Serhiy Storchaka in gh-115961.)

- `mailbox.Maildir` now ignores files with a leading dot (`.`). (Contributed by Zackery Spytz in gh-65559.)

- `pathlib.Path.glob()` and `rglob()` now return both files and directories if a pattern that ends with "`**`" is given, rather than directories only. Add a trailing slash to keep the previous behavior and only match directories.

- The `threading` module now expects the `_thread` module to have an `_is_main_interpreter()` function. This function takes no arguments and returns `True` if the current interpreter is the main interpreter.

  Any library or application that provides a custom `_thread` module must provide `_is_main_interpreter()`, just like the module's other "private" attributes. (gh-112826.)

12.2 C API 中的改動

- `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 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.)

- On Windows, `Python.h` no longer includes the `<stddef.h>` standard header file. If needed, it should now be included explicitly. For example, it provides `offsetof()` function, and `size_t` and `ptrdiff_t` types. Including `<stddef.h>` explicitly was already needed by all other platforms, the `HAVE_STDDEF_H` macro is only defined on Windows. (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.)

- 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`.

  有舊巨集的 `tp_dealloc` 函式，例如：

  ```
  static void
  mytype_dealloc(mytype *p)
  {
      PyObject_GC_UnTrack(p);
      Py_TRASHCAN_SAFE_BEGIN(p);
      ...
      Py_TRASHCAN_SAFE_END
  }
  ```

  應該改E使用新的巨集，如下所示：

  ```
  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. 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.)

- *PEP 667* introduces several changes to frame-related functions:

  - The effects of mutating the dictionary returned from `PyEval_GetLocals()` in an optimized scope have changed. New dict entries added this way will now *only* be visible to subsequent `PyEval_GetLocals()` calls in that frame, as `PyFrame_GetLocals()`, `locals()`, and `FrameType.f_locals` no longer access the same underlying cached dictionary. Changes made to entries for actual variable names and names added via the write-through proxy interfaces will be overwritten on subsequent calls to `PyEval_GetLocals()` in that frame. The recommended code update depends on how the function was being used, so refer to the deprecation notice on the function for details.

  - Calling `PyFrame_GetLocals()` in an optimized scope now returns a write-through proxy rather than a snapshot that gets updated at ill-specified times. If a snapshot is desired, it must be created explicitly (e.g. with `PyDict_Copy()`), or by calling the new `PyEval_GetFrameLocals()` API.

  - `PyFrame_FastToLocals()` and `PyFrame_FastToLocalsWithError()` no longer have any effect. Calling these functions has been redundant since Python 3.11, when `PyFrame_GetLocals()` was first introduced.

  - `PyFrame_LocalsToFast()` no longer has any effect. Calling this function is redundant now that `PyFrame_GetLocals()` returns a write-through proxy for optimized scopes.

- Python 3.13 removed many private functions. Some of them can be replaced using these alternatives:

  - `_PyDict_Pop()`: `PyDict_Pop()` 或 `PyDict_PopString()`;

  - `_PyDict_GetItemWithError()`: `PyDict_GetItemRef()`;

  - `_PyErr_WriteUnraisableMsg()`: `PyErr_FormatUnraisable()`;

- _PyEval_SetTrace(): PyEval_SetTrace() 或 PyEval_SetTraceAllThreads();

- _PyList_Extend(): PyList_Extend();

- _PyLong_AsInt(): PyLong_AsInt();

- _PyMem_RawStrdup(): strdup();

- _PyMem_Strdup(): strdup();

- _PyObject_ClearManagedDict(): PyObject_ClearManagedDict();

- _PyObject_VisitManagedDict(): PyObject_VisitManagedDict();

- _PyThreadState_UncheckedGet(): PyThreadState_GetUnchecked();

- _PyTime_AsSecondsDouble(): PyTime_AsSecondsDouble();

- _PyTime_GetMonotonicClock(): PyTime_Monotonic() 或 PyTime_MonotonicRaw();

- _PyTime_GetPerfCounter(): PyTime_PerfCounter() 或 PyTime_PerfCounterRaw();

- _PyTime_GetSystemClock(): PyTime_Time() 或 PyTime_TimeRaw();

- _PyTime_MAX: PyTime_MAX;

- _PyTime_MIN: PyTime_MIN;

- _PyTime_t: PyTime_t;

- _Py_HashPointer(): Py_HashPointer();

- _Py_IsFinalizing(): Py_IsFinalizing()。

The pythoncapi-compat project can be used to get most of these new functions on Python 3.12 and older.

13 Regression Test Changes

- Python built with configure --with-pydebug now supports a -X presite=package.module command-line option. If used, it specifies a module that should be imported early in the lifecycle of the interpreter, before site.py is executed. (Contributed by Łukasz Langa in gh-110769.)

14 Python 3.13.1 中顯著的變更

14.1 sys

- The previously undocumented special function sys.getobjects(), which only exists in specialized builds of Python, may now return objects from other interpreters than the one it's called in.

索引

非依字母順序

環境變數
    PYTHON_BASIC_REPL, 5
    PYTHON_COLORS, 4, 5, 12
    PYTHON_CPU_COUNT, 14
    PYTHON_FROZEN_MODULES, 9
    PYTHON_GIL, 6
    PYTHON_HISTORY, 9
    PYTHON_PERF_JIT_SUPPORT, 9
    PYTHONHOME, 35, 36
    PYTHONLEGACYWINDOWSFSENCODING, 24, 27

C

Common Vulnerabilities and Exposures
    CVE 2023-27043, 13
    CVE 2023-52425, 18
    CVE 2024-4030, 14, 17

P

Python Enhancement Proposals
    PEP 11, 8, 37
    PEP 11#tier-2, 4
    PEP 11#tier-3, 4
    PEP 587, 34
    PEP 590, 34
    PEP 594, 3, 19
    PEP 602, 5
    PEP 626, 26
    PEP 667, 3, 8, 35, 38
    PEP 669, 4, 29
    PEP 696, 4
    PEP 699, 35
    PEP 702, 4, 18
    PEP 703, 3, 7
    PEP 705, 4, 18
    PEP 709, 38
    PEP 719, 3
    PEP 730, 4, 8
    PEP 737, 33
    PEP 738, 4, 8
    PEP 742, 4, 18
    PEP 744, 3, 4, 7
PYTHON_BASIC_REPL, 5
PYTHON_COLORS, 4, 5, 12
PYTHON_CPU_COUNT, 14
PYTHON_FROZEN_MODULES, 9
PYTHON_GIL, 6
PYTHON_HISTORY, 9
PYTHON_PERF_JIT_SUPPORT, 9
PYTHONHOME, 35, 36
PYTHONLEGACYWINDOWSFSENCODING, 24, 27

R

RFC

RFC 5280, 16