What's New in Python

发布 *3.9.0a5*

A. M. Kuchling

四月 28, 2020

Python Software Foundation
Email: docs@python.org

Contents

Release 3.9.0a5

Date 四月 28, 2020

This article explains the new features in Python 3.9, compared to 3.8.

For full details, see the changelog.

注解: Prerelease users should be aware that this document is currently in draft form. It will be updated substantially as Python 3.9 moves towards release, so it's worth checking back even after reading earlier versions.

---

1 Summary -- Release highlights

2 You should check for DeprecationWarning in your code

When Python 2.7 was still supported, many functions were kept for backward compatibility with Python 2.7. With the end of Python 2.7 support, these backward compatibility layers have been removed, or will be removed soon. Most of them emitted a `DeprecationWarning` warning for several years. For example, using `collections.Mapping` instead of `collections.abc.Mapping` emits a `DeprecationWarning` since Python 3.3, released in 2012.

Test your application with the `-W default` command-line option to see `DeprecationWarning` and `PendingDeprecationWarning`, or even with `-W error` to treat them as errors. Warnings Filter can be used to ignore warnings from third-party code.

It has been decided to keep a few backward compatibility layers for one last release, to give more time to Python projects maintainers to organize the removal of the Python 2 support and add support for Python 3.9.

Aliases to Abstract Base Classes in the `collections` module, like `collections.Mapping` alias to `collections.abc.Mapping`, are kept for one last release for backward compatibility. They will be removed from Python 3.10.

More generally, try to run your tests in the Python Development Mode which helps to prepare your code to make it compatible with the next Python version.

3 New Features

3.1 Dictionary Merge & Update Operators

Merge (`|`) and update (`|=`) operators have been added to the built-in `dict` class. See **PEP 584** for a full description. (Contributed by Brandt Bucher in bpo-36144.)

3.2 PEP 616: New removeprefix() and removesuffix() string methods

`str.removeprefix(prefix)` and `str.removesuffix(suffix)` have been added to easily remove an unneeded prefix or a suffix from a string. Corresponding `bytes`, `bytearray`, and `collections.UserString` methods have also been added. See **PEP 616** for a full description. (Contributed by Dennis Sweeney in bpo-18939.)

3.3 PEP 585: Builtin Generic Types

In type annotations you can now use built-in collection types such as `list` and `dict` as generic types instead of importing the corresponding capitalized types (e.g. `List` or `Dict`) from `typing`. Some other types in the standard library are also now generic, for example `queue.Queue`.

Example:

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

See **PEP 585** for more details. (Contributed by Guido van Rossum, Ethan Smith, and Batuhan Taşkaya in bpo-39481.)

3.4 PEP 617: New Parser

Python 3.9 uses a new parser, based on PEG instead of LL(1). The new parser's performance is roughly comparable to that of the old parser, but the PEG formalism is more flexible than LL(1) when it comes to designing new language features. We'll start using this flexibility in Python 3.10 and later.

The `ast` module uses the new parser and produces the same AST as the old parser.

In Python 3.10, the old parser will be deleted and so will all functionality that depends on it (primarily the `parser` module, which has long been deprecated). In Python 3.9 *only*, you can switch back to the LL(1) parser using a command line switch (`-X oldparser`) or an environment variable (`PYTHONOLDPARSER=1`).

See **PEP 617** for more details. (Contributed by Guido van Rossum, Pablo Galindo and Lysandros Nikolau in bpo-40334.)

4 Other Language Changes

- `__import__()` now raises `ImportError` instead of `ValueError`, which used to occur when a relative import went past its top-level package. (Contributed by Ngalim Siregar in bpo-37444.)
- Python now gets the absolute path of the script filename specified on the command line (ex: `python3 script.py`): the `__file__` attribute of the `__main__` module became an absolute path, rather than a relative path. These paths now remain valid after the current directory is changed by `os.chdir()`. As a side effect, the traceback also displays the absolute path for `__main__` module frames in this case. (Contributed by Victor Stinner in bpo-20443.)
- In the Python Development Mode and in debug build, the *encoding* and *errors* arguments are now checked for string encoding and decoding operations. Examples: `open()`, `str.encode()` and `bytes.decode()`.

  By default, for best performance, the *errors* argument is only checked at the first encoding/decoding error and the *encoding* argument is sometimes ignored for empty strings. (Contributed by Victor Stinner in bpo-37388.)
- `"".replace("", s, n)` now returns `s` instead of an empty string for all non-zero `n`. It is now consistent with `"".replace("", s)`. There are similar changes for `bytes` and `bytearray` objects. (Contributed by Serhiy Storchaka in bpo-28029.)
- Any valid expression can now be used as a decorator. Previously, the grammar was much more restrictive. See **PEP 614** for details. (Contributed by Brandt Bucher in bpo-39702.)
- Improved help for the `typing` module. Docstrings are now shown for all special forms and special generic aliases (like `Union` and `List`). Using `help()` with generic alias like `List[int]` will show the help for the correspondent concrete type (`list` in this case). (Contributed by Serhiy Storchaka in bpo-40257.)

5 New Modules

- None yet.

6 Improved Modules

6.1 ast

Added the *indent* option to `dump()` which allows it to produce a multiline indented output. (Contributed by Serhiy Storchaka in bpo-37995.)

Added `ast.unparse()` as a function in the `ast` module that can be used to unparse an `ast.AST` object and produce a string with code that would produce an equivalent `ast.AST` object when parsed. (Contributed by Pablo Galindo and Batuhan Taskaya in bpo-38870.)

Added docstrings to AST nodes that contains the ASDL signature used to construct that node. (Contributed by Batuhan Taskaya in bpo-39638.)

6.2 asyncio

Due to significant security concerns, the *reuse_address* parameter of `asyncio.loop.create_datagram_endpoint()` is no longer supported. This is because of the behavior of the socket option `SO_REUSEADDR` in UDP. For more details, see the documentation for `loop.create_datagram_endpoint()`. (Contributed by Kyle Stanley, Antoine Pitrou, and Yury Selivanov in bpo-37228.)

Added a new coroutine `shutdown_default_executor()` that schedules a shutdown for the default executor that waits on the `ThreadPoolExecutor` to finish closing. Also, `asyncio.run()` has been updated to use the new coroutine. (Contributed by Kyle Stanley in bpo-34037.)

Added `asyncio.PidfdChildWatcher`, a Linux-specific child watcher implementation that polls process file descriptors. (bpo-38692)

6.3 concurrent.futures

Added a new *cancel_futures* parameter to `concurrent.futures.Executor.shutdown()` that cancels all pending futures which have not started running, instead of waiting for them to complete before shutting down the executor. (Contributed by Kyle Stanley in bpo-39349.)

Removed daemon threads from `ThreadPoolExecutor` and `ProcessPoolExecutor`. This improves compatibility with subinterpreters and predictability in their shutdown processes. (Contributed by Kyle Stanley in bpo-39812.)

Workers in `ProcessPoolExecutor` are now spawned on demand, only when there are no available idle workers to reuse. This optimizes startup overhead and reduces the amount of lost CPU time to idle workers. (Contributed by Kyle Stanley in bpo-39207.)

6.4 curses

Add `curses.get_escdelay()`, `curses.set_escdelay()`, `curses.get_tabsize()`, and `curses.set_tabsize()` functions. (Contributed by Anthony Sottile in bpo-38312.)

6.5 fcntl

Added constants `F_OFD_GETLK`, `F_OFD_SETLK` and `F_OFD_SETLKW`. (Contributed by Dong-hee Na in bpo-38602.)

6.6 ftplib

`FTP` and `FTP_TLS` now raise a `ValueError` if the given timeout for their constructor is zero to prevent the creation of a non-blocking socket. (Contributed by Dong-hee Na in bpo-39259.)

6.7 functools

Add the `functools.TopologicalSorter` class to offer functionality to perform topological sorting of graphs. (Contributed by Pablo Galindo, Tim Peters and Larry Hastings in bpo-17005.)

6.8 gc

When the garbage collector makes a collection in which some objects resurrect (they are reachable from outside the isolated cycles after the finalizers have been executed), do not block the collection of all objects that are still unreachable. (Contributed by Pablo Galindo and Tim Peters in bpo-38379.)

Added a new function `gc.is_finalized()` to check if an object has been finalized by the garbage collector. (Contributed by Pablo Galindo in bpo-39322.)

6.9 http

HTTP status codes `103 EARLY_HINTS`, `418 IM_A_TEAPOT` and `425 TOO_EARLY` are added to `http.HTTPStatus`. (Contributed by Dong-hee Na in bpo-39509 and Ross Rhodes in bpo-39507.)

6.10 imaplib

`IMAP4` and `IMAP4_SSL` now have an optional *timeout* parameter for their constructors. Also, the `open()` method now has an optional *timeout* parameter with this change. The overridden methods of `IMAP4_SSL` and `IMAP4_stream` were applied to this change. (Contributed by Dong-hee Na in bpo-38615.)

`imaplib.IMAP4.unselect()` is added. `imaplib.IMAP4.unselect()` frees server's resources associated with the selected mailbox and returns the server to the authenticated state. This command performs the same actions as `imaplib.IMAP4.close()`, except that no messages are permanently removed from the currently selected mailbox. (Contributed by Dong-hee Na in bpo-40375.)

6.11 importlib

To improve consistency with import statements, `importlib.util.resolve_name()` now raises `ImportError` instead of `ValueError` for invalid relative import attempts. (Contributed by Ngalim Siregar in bpo-37444.)

6.12 inspect

`inspect.BoundArguments.arguments` is changed from `OrderedDict` to regular dict. (Contributed by Inada Naoki in bpo-36350 and bpo-39775.)

6.13 ipaddress

`ipaddress` now supports IPv6 Scoped Addresses (IPv6 address with suffix `%<scope_id>`).

Scoped IPv6 addresses can be parsed using `ipaddress.IPv6Address`. If present, scope zone ID is available through the `scope_id` attribute. (Contributed by Oleksandr Pavliuk in bpo-34788.)

6.14 math

Expanded the `math.gcd()` function to handle multiple arguments. Formerly, it only supported two arguments. (Contributed by Serhiy Storchaka in bpo-39648.)

Add `math.lcm()`: return the least common multiple of specified arguments. (Contributed by Mark Dickinson, Ananthakrishnan and Serhiy Storchaka in bpo-39479 and bpo-39648.)

Add `math.nextafter()`: return the next floating-point value after *x* towards *y*. (Contributed by Victor Stinner in bpo-39288.)

Add `math.ulp()`: return the value of the least significant bit of a float. (Contributed by Victor Stinner in bpo-39310.)

6.15 nntplib

`NNTP` and `NNTP_SSL` now raise a `ValueError` if the given timeout for their constructor is zero to prevent the creation of a non-blocking socket. (Contributed by Dong-hee Na in bpo-39259.)

6.16 multiprocessing

The `multiprocessing.SimpleQueue` class has a new `close()` method to explicitly close the queue. (Contributed by Victor Stinner in bpo-30966.)

6.17 os

Added `CLD_KILLED` and `CLD_STOPPED` for `si_code`. (Contributed by Dong-hee Na in bpo-38493.)

Exposed the Linux-specific `os.pidfd_open()` (bpo-38692) and `os.P_PIDFD` (bpo-38713) for process management with file descriptors.

The `os.unsetenv()` function is now also available on Windows. (Contributed by Victor Stinner in bpo-39413.)

The `os.putenv()` and `os.unsetenv()` functions are now always available. (Contributed by Victor Stinner in bpo-39395.)

Add `os.waitstatus_to_exitcode()` function: convert a wait status to an exit code. (Contributed by Victor Stinner in bpo-40094.)

6.18 pathlib

Added `pathlib.Path.readlink()` which acts similarly to `os.readlink()`. (Contributed by Girts Folkmanis in bpo-30618)

6.19 poplib

`POP3` and `POP3_SSL` now raise a `ValueError` if the given timeout for their constructor is zero to prevent the creation of a non-blocking socket. (Contributed by Dong-hee Na in bpo-39259.)

6.20 pprint

`pprint` can now pretty-print `types.SimpleNamespace`. (Contributed by Carl Bordum Hansen in bpo-37376.)

6.21 pydoc

The documentation string is now shown not only for class, function, method etc, but for any object that has its own `__doc__` attribute. (Contributed by Serhiy Storchaka in bpo-40257.)

6.22 random

Add a new `random.Random.randbytes` method: generate random bytes. (Contributed by Victor Stinner in bpo-40286.)

6.23 signal

Exposed the Linux-specific `signal.pidfd_send_signal()` for sending to signals to a process using a file descriptor instead of a pid. (bpo-38712)

6.24 smtplib

`SMTP` and `SMTP_SSL` now raise a `ValueError` if the given timeout for their constructor is zero to prevent the creation of a non-blocking socket. (Contributed by Dong-hee Na in bpo-39259.)

`LMTP` constructor now has an optional *timeout* parameter. (Contributed by Dong-hee Na in bpo-39329.)

6.25 socket

The `socket` module now exports the `CAN_RAW_JOIN_FILTERS` constant on Linux 4.1 and greater. (Contributed by Stefan Tatschner and Zackery Spytz in bpo-25780.)

6.26 sys

Add a new `sys.platlibdir` attribute: name of the platform-specific library directory. It is used to build the path of platform-specific dynamic libraries and the path of the standard library. It is equal to `"lib"` on most platforms. On Fedora and SuSE, it is equal to `"lib64"` on 64-bit platforms. (Contributed by Jan Matějek, Matěj Cepl, Charalampos Stratakis and Victor Stinner in bpo-1294959.)

6.27 typing

PEP 593 introduced an `typing.Annotated` type to decorate existing types with context-specific metadata and new `include_extras` parameter to `typing.get_type_hints()` to access the metadata at runtime. (Contributed by Till Varoquaux and Konstantin Kashin.)

6.28 unicodedata

The Unicode database has been updated to version 13.0.0. (bpo-39926).

6.29 venv

The activation scripts provided by `venv` now all specify their prompt customization consistently by always using the value specified by `__VENV_PROMPT__`. Previously some scripts unconditionally used `__VENV_PROMPT__`, others only if it happened to be set (which was the default case), and one used `__VENV_NAME__` instead. (Contributed by Brett Cannon in bpo-37663.)

6.30 xml

White space characters within attributes are now preserved when serializing `xml.etree.ElementTree` to XML file. EOLNs are no longer normalized to "n". This is the result of discussion about how to interpret section 2.11 of XML spec. (Contributed by Mefistotelis in bpo-39011.)

7 Optimizations

- Optimized the idiom for assignment a temporary variable in comprehensions. Now `for y in [expr]` in comprehensions is as fast as a simple assignment `y = expr`. For example:

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

  Unlike to the `:=` operator this idiom does not leak a variable to the outer scope.

  (Contributed by Serhiy Storchaka in bpo-32856.)

- Optimize signal handling in multithreaded applications. If a thread different than the main thread gets a signal, the bytecode evaluation loop is no longer interrupted at each bytecode instruction to check for pending signals which cannot be handled. Only the main thread of the main interpreter can handle signals.

  Previously, the bytecode evaluation loop was interrupted at each instruction until the main thread handles signals. (Contributed by Victor Stinner in bpo-40010.)

- Optimize the `subprocess` module on FreeBSD using `closefrom()`. (Contributed by Ed Maste, Conrad Meyer, Kyle Evans, Kubilay Kocak and Victor Stinner in bpo-38061.)

8 Build and C API Changes

- New `PyThreadState_GetInterpreter()` and `PyInterpreterState_Get()` functions to get the interpreter. New `PyThreadState_GetFrame()` function to get the current frame of a Python thread state. New `PyThreadState_GetID()` function: get the unique identifier of a Python thread state. (Contributed by Victor Stinner in bpo-39947.)
- Add `--with-platlibdir` option to the `configure` script: name of the platform-specific library directory, stored in the new `sys.platlibdir` attribute. See `sys.platlibdir` attribute for more information. (Contributed by Jan Matějek, Matěj Cepl, Charalampos Stratakis and Victor Stinner in bpo-1294959.)
- Add a new public `PyObject_CallNoArgs()` function to the C API, which calls a callable Python object without any arguments. It is the most efficient way to call a callable Python object without any argument. (Contributed by Victor Stinner in bpo-37194.)
- The global variable `PyStructSequence_UnnamedField` is now a constant and refers to a constant string. (Contributed by Serhiy Storchaka in bpo-38650.)
- Exclude `PyFPE_START_PROTECT()` and `PyFPE_END_PROTECT()` macros of `pyfpe.h` from `Py_LIMITED_API` (stable API). (Contributed by Victor Stinner in bpo-38835.)
- Remove `PyMethod_ClearFreeList()` and `PyCFunction_ClearFreeList()` functions: the free lists of bound method objects have been removed. (Contributed by Inada Naoki and Victor Stinner in bpo-37340.)
- Remove `PyUnicode_ClearFreeList()` function: the Unicode free list has been removed in Python 3.3. (Contributed by Victor Stinner in bpo-38896.)
- The `tp_print` slot of PyTypeObject has been removed. It was used for printing objects to files in Python 2.7 and before. Since Python 3.0, it has been ignored and unused. (Contributed by Jeroen Demeyer in bpo-36974.)
- On non-Windows platforms, the `setenv()` and `unsetenv()` functions are now required to build Python. (Contributed by Victor Stinner in bpo-39395.)
- The `COUNT_ALLOCS` special build macro has been removed. (Contributed by Victor Stinner in bpo-39489.)
- Changes in the limited C API (if `Py_LIMITED_API` macro is defined):
  - Provide `Py_EnterRecursiveCall()` and `Py_LeaveRecursiveCall()` as regular functions for the limited API. Previously, there were defined as macros, but these macros didn't compile with the limited C API which cannot access `PyThreadState.recursion_depth` field (the structure is opaque in the limited C API).
  - Exclude the following functions from the limited C API:
    * `_Py_CheckRecursionLimit`
    * `_Py_NewReference()`
    * `_Py_ForgetReference()`
    * `_PyTraceMalloc_NewReference()`
    * `_Py_GetRefTotal()`
    * The trashcan mechanism which never worked in the limited C API.
    * `PyTrash_UNWIND_LEVEL`
    * `Py_TRASHCAN_BEGIN_CONDITION`
    * `Py_TRASHCAN_BEGIN`
    * `Py_TRASHCAN_END`
    * `Py_TRASHCAN_SAFE_BEGIN`

    * `Py_TRASHCAN_SAFE_END`

  – The following static inline functions or macros become regular "opaque" function to hide implementation details:

    * `_Py_NewReference()`

    * `PyObject_INIT()` and `PyObject_INIT_VAR()` become aliases to `PyObject_Init()` and `PyObject_InitVar()` in the limited C API, but are overriden with static inline function otherwise. Thanks to that, it was possible to exclude `_Py_NewReference()` from the limited C API.

  – Move following functions and definitions to the internal C API:

    * `_PyDebug_PrintTotalRefs()`

    * `_Py_PrintReferences()`

    * `_Py_PrintReferenceAddresses()`

    * `_Py_tracemalloc_config`

    * `_Py_AddToAllObjects()` (specific to `Py_TRACE_REFS` build)

  (Contributed by Victor Stinner in bpo-38644 and bpo-39542.)

- `PyInterpreterState.eval_frame` (**PEP 523**) now requires a new mandatory *tstate* parameter (`PyThreadState*`). (Contributed by Victor Stinner in bpo-38500.)

- Extension modules: `m_traverse`, `m_clear` and `m_free` functions of `PyModuleDef` are no longer called if the module state was requested but is not allocated yet. This is the case immediately after the module is created and before the module is executed (`Py_mod_exec` function). More precisely, these functions are not called if `m_size` is greater than 0 and the module state (as returned by `PyModule_GetState()`) is `NULL`.

  Extension modules without module state (`m_size <= 0`) are not affected.

- If `Py_AddPendingCall()` is called in a subinterpreter, the function is now scheduled to be called from the subinterpreter, rather than being called from the main interpreter. Each subinterpreter now has its own list of scheduled calls. (Contributed by Victor Stinner in bpo-39984.)

- Remove `_PyRuntime.getframe` hook and remove `_PyThreadState_GetFrame` macro which was an alias to `_PyRuntime.getframe`. They were only exposed by the internal C API. Remove also `PyThreadFrameGetter` type. (Contributed by Victor Stinner in bpo-39946.)

- The `PyModule_AddType()` function is added to help adding a type to a module. (Contributed by Dong-hee Na in bpo-40024.)

- The Windows registry is no longer used to initialize `sys.path` when the `-E` option is used. This is significant when embedding Python on Windows. (Contributed by Zackery Spytz in bpo-8901.)

- Add the functions `PyObject_GC_IsTracked()` and `PyObject_GC_IsFinalized()` to the public API to allow to query if Python objects are being currently tracked or have been already finalized by the garbage collector respectively. (Contributed by Pablo Galindo in bpo-40241.)

9 Deprecated

- The distutils `bdist_msi` command is now deprecated, use `bdist_wheel` (wheel packages) instead. (Contributed by Hugo van Kemenade in bpo-39586.)
- Currently `math.factorial()` accepts `float` instances with non-negative integer values (like `5.0`). It raises a `ValueError` for non-integral and negative floats. It is now deprecated. In future Python versions it will raise a `TypeError` for all floats. (Contributed by Serhiy Storchaka in bpo-37315.)
- The `parser` module is deprecated and will be removed in future versions of Python. For the majority of use cases, users can leverage the Abstract Syntax Tree (AST) generation and compilation stage, using the `ast` module.
- Using `NotImplemented` in a boolean context has been deprecated, as it is almost exclusively the result of incorrect rich comparator implementations. It will be made a `TypeError` in a future version of Python. (Contributed by Josh Rosenberg in bpo-35712.)
- The `random` module currently accepts any hashable type as a possible seed value. Unfortunately, some of those types are not guaranteed to have a deterministic hash value. After Python 3.9, the module will restrict its seeds to `None`, `int`, `float`, `str`, `bytes`, and `bytearray`.
- Opening the `GzipFile` file for writing without specifying the *mode* argument is deprecated. In future Python versions it will always be opened for reading by default. Specify the *mode* argument for opening it for writing and silencing a warning. (Contributed by Serhiy Storchaka in bpo-28286.)
- Deprecated the `split()` method of `_tkinter.TkappType` in favour of the `splitlist()` method which has more consistent and predicable behavior. (Contributed by Serhiy Storchaka in bpo-38371.)
- The explicit passing of coroutine objects to `asyncio.wait()` has been deprecated and will be removed in version 3.11. (Contributed by Yury Selivanov and Kyle Stanley in bpo-34790.)
- binhex4 and hexbin4 standards are now deprecated. The :*binhex* module and the following `binascii` functions are now deprecated:
  - `b2a_hqx()`, `a2b_hqx()`
  - `rlecode_hqx()`, `rledecode_hqx()`

  (Contributed by Victor Stinner in bpo-39353.)
- `ast` classes `slice`, `Index` and `ExtSlice` are considered deprecated and will be removed in future Python versions. `value` itself should be used instead of `Index(value)`. `Tuple(slices, Load())` should be used instead of `ExtSlice(slices)`. (Contributed by Serhiy Storchaka in bpo-32892.)
- `ast` classes `Suite`, `Param`, `AugLoad` and `AugStore` are considered deprecated and will be removed in future Python versions. They were not generated by the parser and not accepted by the code generator in Python 3. (Contributed by Batuhan Taskaya in bpo-39639 and bpo-39969 and Serhiy Storchaka in bpo-39988.)
- The `PyEval_InitThreads()` and `PyEval_ThreadsInitialized()` functions are now deprecated and will be removed in Python 3.11. Calling `PyEval_InitThreads()` now does nothing. The GIL is initialized by `Py_Initialize()` since Python 3.7. (Contributed by Victor Stinner in bpo-39877.)
- Passing `None` as the first argument to the `shlex.split()` function has been deprecated. (Contributed by Zackery Spytz in bpo-33262.)

10 Removed

- The erroneous version at `unittest.mock.__version__` has been removed.
- `nntplib.NNTP`: `xpath()` and `xgtitle()` methods have been removed. These methods are deprecated since Python 3.3. Generally, these extensions are not supported or not enabled by NNTP server administrators. For `xgtitle()`, please use `nntplib.NNTP.descriptions()` or `nntplib.NNTP.description()` instead. (Contributed by Dong-hee Na in bpo-39366.)
- `array.array`: `tostring()` and `fromstring()` methods have been removed. They were aliases to `tobytes()` and `frombytes()`, deprecated since Python 3.2. (Contributed by Victor Stinner in bpo-38916.)
- The undocumented `sys.callstats()` function has been removed. Since Python 3.7, it was deprecated and always returned `None`. It required a special build option `CALL_PROFILE` which was already removed in Python 3.7. (Contributed by Victor Stinner in bpo-37414.)
- The `sys.getcheckinterval()` and `sys.setcheckinterval()` functions have been removed. They were deprecated since Python 3.2. Use `sys.getswitchinterval()` and `sys.setswitchinterval()` instead. (Contributed by Victor Stinner in bpo-37392.)
- The C function `PyImport_Cleanup()` has been removed. It was documented as: "Empty the module table. For internal use only." (Contributed by Victor Stinner in bpo-36710.)
- `_dummy_thread` and `dummy_threading` modules have been removed. These modules were deprecated since Python 3.7 which requires threading support. (Contributed by Victor Stinner in bpo-37312.)
- `aifc.openfp()` alias to `aifc.open()`, `sunau.openfp()` alias to `sunau.open()`, and `wave.openfp()` alias to `wave.open()` have been removed. They were deprecated since Python 3.7. (Contributed by Victor Stinner in bpo-37320.)
- The `isAlive()` method of `threading.Thread` has been removed. It was deprecated since Python 3.8. Use `is_alive()` instead. (Contributed by Dong-hee Na in bpo-37804.)
- Methods `getchildren()` and `getiterator()` of classes `ElementTree` and `Element` in the `ElementTree` module have been removed. They were deprecated in Python 3.2. Use `iter(x)` or `list(x)` instead of `x.getchildren()` and `x.iter()` or `list(x.iter())` instead of `x.getiterator()`. The `xml.etree.cElementTree` module has been removed, use the `xml.etree.ElementTree` module instead. Since Python 3.3 the `xml.etree.cElementTree` module has been deprecated, the `xml.etree.ElementTree` module uses a fast implementation whenever available. (Contributed by Serhiy Storchaka in bpo-36543.)
- The old `plistlib` API has been removed, it was deprecated since Python 3.4. Use the `load()`, `loads()`, `dump()`, and `dumps()` functions. Additionally, the *use_builtin_types* parameter was removed, standard `bytes` objects are always used instead. (Contributed by Jon Janzen in bpo-36409.)
- The C function `PyThreadState_DeleteCurrent()` has been removed. It was not documented. (Contributed by Joannah Nanjekye in bpo-37878.)
- The C function `PyGen_NeedsFinalizing` has been removed. It was not documented, tested, or used anywhere within CPython after the implementation of **PEP 442**. Patch by Joannah Nanjekye. (Contributed by Joannah Nanjekye in bpo-15088)
- `base64.encodestring()` and `base64.decodestring()`, aliases deprecated since Python 3.1, have been removed: use `base64.encodebytes()` and `base64.decodebytes()` instead. (Contributed by Victor Stinner in bpo-39351.)
- `fractions.gcd()` function has been removed, it was deprecated since Python 3.5 (bpo-22486): use `math.gcd()` instead. (Contributed by Victor Stinner in bpo-39350.)

- The *buffering* parameter of `bz2.BZ2File` has been removed. Since Python 3.0, it was ignored and using it emitted a `DeprecationWarning`. Pass an open file object to control how the file is opened. (Contributed by Victor Stinner in bpo-39357.)
- The *encoding* parameter of `json.loads()` has been removed. As of Python 3.1, it was deprecated and ignored; using it has emitted a `DeprecationWarning` since Python 3.8. (Contributed by Inada Naoki in bpo-39377)
- `with (await asyncio.lock):` and `with (yield from asyncio.lock):` statements are not longer supported, use `async with lock` instead. The same is correct for `asyncio.Condition` and `asyncio.Semaphore`. (Contributed by Andrew Svetlov in bpo-34793.)
- The `sys.getcounts()` function, the `-X showalloccount` command line option and the `show_alloc_count` field of the C structure `PyConfig` have been removed. They required a special Python build by defining `COUNT_ALLOCS` macro. (Contributed by Victor Stinner in bpo-39489.)
- The `_field_types` attribute of the `typing.NamedTuple` class has been removed. It was deprecated deprecated since Python 3.8. Use the `__annotations__` attribute instead. (Contributed by Serhiy Storchaka in bpo-40182.)
- The `symtable.SymbolTable.has_exec()` method has been removed. It was deprecated since 2006, and only returning `False` when it's called. (Contributed by Batuhan Taskaya in bpo-40208)

11 Porting to Python 3.9

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

11.1 Changes in the Python API

- `__import__()` and `importlib.util.resolve_name()` now raise `ImportError` where it previously raised `ValueError`. Callers catching the specific exception type and supporting both Python 3.9 and earlier versions will need to catch both using `except (ImportError, ValueError):`.
- The `venv` activation scripts no longer special-case when `__VENV_PROMPT__` is set to `""`.
- The `select.epoll.unregister()` method no longer ignores the `EBADF` error. (Contributed by Victor Stinner in bpo-39239.)
- The *compresslevel* parameter of `bz2.BZ2File` became keyword-only, since the *buffering* parameter has been removed. (Contributed by Victor Stinner in bpo-39357.)
- Simplified AST for subscription. Simple indices will be represented by their value, extended slices will be represented as tuples. `Index(value)` will return a `value` itself, `ExtSlice(slices)` will return `Tuple(slices, Load())`. (Contributed by Serhiy Storchaka in bpo-34822.)
- The `importlib` module now ignores the `PYTHONCASEOK` environment variable when the `-E` or `-I` command line options are being used.
- The *encoding* parameter has been added to the classes `ftplib.FTP` and `ftplib.FTP_TLS` as a keyword-only parameter, and the default encoding is changed from Latin-1 to UTF-8 to follow **RFC 2640**.
- `inspect.getdoc()` no longer returns docstring inherited from the type of the object or from parent class if it is a class if it is not defined in the object itself. (Contributed by Serhiy Storchaka in bpo-40257.)
- `asyncio.loop.shutdown_default_executor()` has been added to `AbstractEventLoop`, meaning alternative event loops that inherit from it should have this method defined. (Contributed by Kyle Stanley in bpo-34037.)

- The constant values of future flags in the `__future__` module is updated in order to prevent collision with compiler flags. Previously `PyCF_ALLOW_TOP_LEVEL_AWAIT` was clashing with `CO_FUTURE_DIVISION`. (Contributed by Batuhan Taskaya in bpo-39562)

11.2 CPython bytecode changes

- The `LOAD_ASSERTION_ERROR` opcode was added for handling the `assert` statement. Previously, the assert statement would not work correctly if the `AssertionError` exception was being shadowed. (Contributed by Zackery Spytz in bpo-34880.)

索引

非字母

P

R