What's New in Python

Release 3.9.0a3

A. M. Kuchling

February 25, 2020

Python Software Foundation
Email: docs@python.org

Contents

Release 3.9.0a3

Date February 25, 2020

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

For full details, see the changelog.

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

---

1 Summary – Release highlights

2 New Features

3 Other Language Changes

- `__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 and `sys.path[0]` become 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, a 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.)

4 New Modules

- None yet.

5 Improved Modules

5.1 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.)

5.2 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)

5.3 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.)

5.4 curses

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

5.5 fcntl

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

5.6 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.)

5.7 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.)

5.8 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.)

5.9 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.)

5.10 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.)

5.11 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.)

5.12 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.)

5.13 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.)

5.14 pathlib

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

5.15 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.)

5.16 pprint

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

5.17 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)

5.18 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.)

5.19 threading

In a subinterpreter, spawning a daemon thread now raises a `RuntimeError`. Daemon threads were never supported in subinterpreters. Previously, the subinterpreter finalization crashed with a Python fatal error if a daemon thread was still running. (Contributed by Victor Stinner in bpo-37266.)

5.20 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.)

5.21 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 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.)

7 Build and C API Changes

- 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.)

8 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.
- 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.)

9 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()` in the `ElementTree` module have been removed. They were deprecated in Python 3.2. Use functions `list()` and `iter()` instead. The `xml.etree.cElementTree` module has been removed. (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.)

10 Porting to Python 3.9

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

10.1 Changes in the Python API

- `open()`, `io.open()`, `codecs.open()` and `fileinput.FileInput` no longer accept `'U'` ("universal newline") in the file mode. This flag was deprecated since Python 3.3. In Python 3, the "universal newline" is used by default when a file is open in text mode. The newline parameter of `open()` controls how universal newlines works. (Contributed by Victor Stinner in bpo-37330.)
- `__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.)

10.2 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.)

Index

P