What's New in Python

Release 3.9.0a0

A. M. Kuchling

November 19, 2019

Python Software Foundation
Email: docs@python.org

Contents

Release 3.9.0a0

Date November 19, 2019

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, `sys.argv[0]` 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 development mode and in debug build, *encoding* and *errors* arguments are now checked on 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.)

5.2 asyncio

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 curses

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

5.4 fcntl

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

5.5 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.

5.6 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.)

5.7 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.)

5.8 pathlib

Added `readlink()` which acts similar to `readlink()`. (Contributed by Girts Folkmanis in bpo-30618)

5.9 pprint

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

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.)

6 Optimizations

7 Build and C API Changes

- Provide `Py_EnterRecursiveCall()` and `Py_LeaveRecursiveCall()` as regular functions for the limited API. Previously, there were defined as macros, but these macros didn't work with the limited API which cannot access `PyThreadState.recursion_depth` field. Remove `_Py_CheckRecursionLimit` from the stable ABI. (Contributed by Victor Stinner in bpo-38644.)
- 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.)

8 Deprecated

- 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.)

9 Removed

- The abstract base classes in `collections.abc` no longer are exposed in the regular `collections` module. This will help create a clearer distinction between the concrete classes and the abstract base classes.
- 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)

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 `""`.

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