
What's New in Python

Release 3.12.0a0

A. M. Kuchling

October 24, 2022

Python Software Foundation
Email: docs@python.org

Contents

1	Summary – Release highlights	2
2	New Features	2
3	Other Language Changes	2
4	New Modules	3
5	Improved Modules	3
5.1	asyncio	3
5.2	pathlib	3
5.3	dis	3
5.4	os	3
5.5	shutil	3
5.6	sqlite3	4
5.7	threading	4
5.8	unicodedata	4
5.9	tempfile	4
6	Optimizations	4
7	CPython bytecode changes	4
8	Demos and Tools	4
9	Deprecated	5
9.1	Pending Removal in Python 3.13	5
10	Pending Removal in Python 3.14	6
10.1	Pending Removal in Future Versions	6
11	Removed	6
12	Porting to Python 3.12	8
12.1	Changes in the Python API	8

13 Build Changes	9
14 C API Changes	9
14.1 New Features	9
14.2 Porting to Python 3.12	10
14.3 Deprecated	11
14.4 Removed	11
Index	13

Release 3.12.0a0

Date October 24, 2022

This article explains the new features in Python 3.12, compared to 3.11.

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 3.12 moves towards release, so it's worth checking back even after reading earlier versions.

1 Summary – Release highlights

Important deprecations, removals or restrictions:

- [PEP 623](#), Remove wstr from Unicode

2 New Features

3 Other Language Changes

- `types.MappingProxyType` instances are now hashable if the underlying mapping is hashable. (Contributed by Serhiy Storchaka in [gh-87995](#).)
- `memoryview` now supports the half-float type (the “e” format code). (Contributed by Dong-hee Na and Antoine Pitrou in [gh-90751](#).)
- The parser now raises `SyntaxError` when parsing source code containing null bytes. (Contributed by Pablo Galindo in [gh-96670](#).)
- `ast.parse()` now raises `SyntaxError` instead of `ValueError` when parsing source code containing null bytes. (Contributed by Pablo Galindo in [gh-96670](#).)
- The Garbage Collector now runs only on the eval breaker mechanism of the Python bytecode evaluation loop instead on object allocations. The GC can also run when `PyErr_CheckSignals()` is called so C extensions that need to run for a long time without executing any Python code also have a chance to execute the GC periodically. (Contributed by Pablo Galindo in [gh-97922](#).)

4 New Modules

- None yet.

5 Improved Modules

5.1 asyncio

- On Linux, `asyncio` uses `PidfdChildWatcher` by default if `os.pidfd_open()` is available and functional instead of `ThreadedChildWatcher`. (Contributed by Kumar Aditya in [gh-98024](#).)
- The child watcher classes `MultiLoopChildWatcher`, `FastChildWatcher` and `SafeChildWatcher` are deprecated and will be removed in Python 3.14. It is recommended to not manually configure a child watcher as the event loop now uses the best available child watcher for each platform (`PidfdChildWatcher` if supported and `ThreadedChildWatcher` otherwise). (Contributed by Kumar Aditya in [gh-94597](#).)
- `asyncio.set_child_watcher()`, `asyncio.get_child_watcher()`, `asyncio.AbstractEventLoopPolicy.set_child_watcher()` and `asyncio.AbstractEventLoopPolicy.get_child_watcher()` are deprecated and will be removed in Python 3.14. (Contributed by Kumar Aditya in [gh-94597](#).)

5.2 pathlib

- Add `walk()` for walking the directory trees and generating all file or directory names within them, similar to `os.walk()`. (Contributed by Stanislav Zmiev in [gh-90385](#).)

5.3 dis

- Pseudo instruction opcodes (which are used by the compiler but do not appear in executable bytecode) are now exposed in the `dis` module. `HAVE_ARGUMENT` is still relevant to real opcodes, but it is not useful for pseudo instructions. Use the new `hasarg` collection instead. (Contributed by Irit Katriel in [gh-94216](#).)

5.4 os

- Add `os.PIDFD_NONBLOCK` to open a file descriptor for a process with `os.pidfd_open()` in non-blocking mode. (Contributed by Kumar Aditya in [gh-93312](#).)

5.5 shutil

- `shutil.make_archive()` now passes the `root_dir` argument to custom archivers which support it. In this case it no longer temporarily changes the current working directory of the process to `root_dir` to perform archiving. (Contributed by Serhiy Storchaka in [gh-74696](#).)

5.6 sqlite3

- Add a command-line interface. (Contributed by Erlend E. Aasland in [gh-77617](#).)

5.7 threading

- Add `threading.settrace_all_threads()` and `threading.setprofile_all_threads()` that allow to set tracing and profiling functions in all running threads in addition to the calling one. (Contributed by Pablo Galindo in [gh-93503](#).)

5.8 unicodedata

- The Unicode database has been updated to version 15.0.0. (Contributed by Benjamin Peterson in [gh-96734](#).)

5.9 tempfile

The `tempfile.NamedTemporaryFile` function has a new optional parameter *delete_on_close* (Contributed by Evgeny Zorin in [gh-58451](#).)

6 Optimizations

- Removed `wstr` and `wstr_length` members from Unicode objects. It reduces object size by 8 or 16 bytes on 64bit platform. ([PEP 623](#)) (Contributed by Inada Naoki in [gh-92536](#).)
- Added experimental support for using the BOLT binary optimizer in the build process, which improves performance by 1-5%. (Contributed by Kevin Modzelewski in [gh-90536](#).)
- Speed up the regular expression substitution (functions `re.sub()` and `re.subn()` and corresponding `re.Pattern` methods) for replacement strings containing group references by 2–3 times. (Contributed by Serhiy Storchaka in [gh-91524](#).)

7 CPython bytecode changes

- Removed the `LOAD_METHOD` instruction. It has been merged into `LOAD_ATTR`. `LOAD_ATTR` will now behave like the old `LOAD_METHOD` instruction if the low bit of its `oparg` is set. (Contributed by Ken Jin in [gh-93429](#).)

8 Demos and Tools

- Remove the `Tools/demo/` directory which contained old demo scripts. A copy can be found in the [old-demos project](#). (Contributed by Victor Stinner in [gh-97681](#).)
- Remove outdated example scripts of the `Tools/scripts/` directory. A copy can be found in the [old-demos project](#). (Contributed by Victor Stinner in [gh-97669](#).)

9 Deprecated

- `typing.Hashable` and `typing.Sized` aliases for `collections.abc.Hashable` and `collections.abc.Sized`. ([gh-94309](#).)
- The `sqlite3` default adapters and converters are now deprecated. Instead, use the `sqlite3-adapter-converter-recipes` and tailor them to your needs. (Contributed by Erlend E. Aasland in [gh-90016](#).)
- The 3-arg signatures (type, value, traceback) of `throw()`, `throw()` and `athrow()` are deprecated and may be removed in a future version of Python. Use the single-arg versions of these functions instead. (Contributed by Ofey Chan in [gh-89874](#).)
- `DeprecationWarning` is now raised when `__package__` on a module differs from `__spec__.parent` (previously it was `ImportWarning`). (Contributed by Brett Cannon in [gh-65961](#).)

9.1 Pending Removal in Python 3.13

The following modules and APIs have been deprecated in earlier Python releases, and will be removed in Python 3.13.

Modules (see [PEP 594](#)):

- `aifc`
- `audioop`
- `cgi`
- `cgitb`
- `chunk`
- `crypt`
- `imghdr`
- `mailcap`
- `msilib`
- `nis`
- `nntplib`
- `ossaudiodev`
- `pipes`
- `sndhdr`
- `spwd`
- `sunau`
- `telnetlib`
- `uu`
- `xdrlib`

APIs:

- `configparser.LegacyInterpolation` ([gh-90765](#))
- `locale.getdefaultlocale()` ([gh-90817](#))
- `turtle.RawTurtle.settiltangle()` ([gh-50096](#))

- `unittest.findTestCases()` ([gh-50096](#))
- `unittest.makeSuite()` ([gh-50096](#))
- `unittest.getTestCaseNames()` ([gh-50096](#))
- `webbrowser.MacOSX` ([gh-86421](#))

10 Pending Removal in Python 3.14

- Deprecated the following `importlib.abc` classes, scheduled for removal in Python 3.14:

- `importlib.abc.ResourceReader`
- `importlib.abc.Traversable`
- `importlib.abc.TraversableResources`

Use `importlib.resources.abc` classes instead:

- `importlib.resources.abc.TraversableResources`
- `importlib.resources.abc.Traversable`
- `importlib.resources.abc.TraversableResources`

(Contributed by Jason R. Coombs and Hugo van Kemenade in [gh-93963](#).)

- Creating immutable types with mutable bases using the C API.
- `__package__` and `__cached__` will cease to be set or taken into consideration by the import system ([gh-97879](#)).

10.1 Pending Removal in Future Versions

The following APIs were deprecated in earlier Python versions and will be removed, although there is currently no date scheduled for their removal.

- `typing.Text` ([gh-92332](#))
- Currently Python accepts numeric literals immediately followed by keywords, for example `0 in x`, `1 or x`, `0 if 1 else 2`. It allows confusing and ambiguous expressions like `[0x1 for x in y]` (which can be interpreted as `[0x1 for x in y]` or `[0x1f or x in y]`). A syntax warning is raised if the numeric literal is immediately followed by one of keywords and, else, for, if, in, is and or. In a future release it will be changed to a syntax error. ([gh-87999](#))

11 Removed

- Removed many old deprecated `unittest` features:
 - A number of `TestCase` method aliases:

Deprecated alias	Method Name	Deprecated in
<code>failUnless</code>	<code>assertTrue()</code>	3.1
<code>failIf</code>	<code>assertFalse()</code>	3.1
<code>failUnlessEqual</code>	<code>assertEqual()</code>	3.1
<code>failIfEqual</code>	<code>assertNotEqual()</code>	3.1
<code>failUnlessAlmostEqual</code>	<code>assertAlmostEqual()</code>	3.1
<code>failIfAlmostEqual</code>	<code>assertNotAlmostEqual()</code>	3.1
<code>failUnlessRaises</code>	<code>assertRaises()</code>	3.1
<code>assert_</code>	<code>assertTrue()</code>	3.2
<code>assertEquals</code>	<code>assertEqual()</code>	3.2
<code>assertNotEquals</code>	<code>assertNotEqual()</code>	3.2
<code>assertAlmostEquals</code>	<code>assertAlmostEqual()</code>	3.2
<code>assertNotAlmostEquals</code>	<code>assertNotAlmostEqual()</code>	3.2
<code>assertRegexpMatches</code>	<code>assertRegex()</code>	3.2
<code>assertRaisesRegexp</code>	<code>assertRaisesRegex()</code>	3.2
<code>assertNotRegexpMatches</code>	<code>assertNotRegex()</code>	3.5

You can use <https://github.com/isidentical/teyit> to automatically modernise your unit tests.

- Undocumented and broken `TestCase` method `assertDictContainsSubset` (deprecated in Python 3.2).
- Undocumented `TestLoader.loadTestsFromModule` parameter `use_load_tests` (deprecated and ignored since Python 3.2).
- An alias of the `TextTestResult` class: `_TextTestResult` (deprecated in Python 3.2).

(Contributed by Serhiy Storchaka in [bpo-45162](#).)

- Several names deprecated in the `configparser` way back in 3.2 have been removed per [gh-89336](#):
 - `configparser.ParsingError` no longer has a `filename` attribute or argument. Use the `source` attribute and argument instead.
 - `configparser` no longer has a `SafeConfigParser` class. Use the shorter `ConfigParser` name instead.
 - `configparser.ConfigParser` no longer has a `readfp` method. Use `read_file()` instead.
- The following undocumented `sqlite3` features, deprecated in Python 3.10, are now removed:
 - `sqlite3.enable_shared_cache()`
 - `sqlite3.OptimizedUnicode`

If a shared cache must be used, open the database in URI mode using the `cache=shared` query parameter.

The `sqlite3.OptimizedUnicode` text factory has been an alias for `str` since Python 3.3. Code that previously set the text factory to `OptimizedUnicode` can either use `str` explicitly, or rely on the default value which is also `str`.

(Contributed by Erlend E. Aasland in [gh-92548](#).)

- `smtpd` has been removed according to the schedule in [PEP 594](#), having been deprecated in Python 3.4.7 and 3.5.4. Use [aiosmtpd](#) PyPI module or any other `asyncio`-based server instead. (Contributed by Oleg Iarygin in [gh-93243](#).)
- Remove `io.OpenWrapper` and `_pyio.OpenWrapper`, deprecated in Python 3.10: just use `open()` instead. The `open()` (`io.open()`) function is a built-in function. Since Python 3.10, `_pyio.open()` is also a static method. (Contributed by Victor Stinner in [gh-94169](#).)

- Remove the `ssl.RAND_pseudo_bytes()` function, deprecated in Python 3.6: use `os.urandom()` or `ssl.RAND_bytes()` instead. (Contributed by Victor Stinner in [gh-94199](#).)
- `gzip`: Remove the `filename` attribute of `gzip.GzipFile`, deprecated since Python 2.6, use the `name` attribute instead. In write mode, the `filename` attribute added `'.gz'` file extension if it was not present. (Contributed by Victor Stinner in [gh-94196](#).)
- Remove the `ssl.match_hostname()` function. The `ssl.match_hostname()` was deprecated in Python 3.7. OpenSSL performs hostname matching since Python 3.7, Python no longer uses the `ssl.match_hostname()` function. (Contributed by Victor Stinner in [gh-94199](#).)
- Remove the `locale.format()` function, deprecated in Python 3.7: use `locale.format_string()` instead. (Contributed by Victor Stinner in [gh-94226](#).)
- `hashlib`: Remove the pure Python implementation of `hashlib.pbkdf2_hmac()`, deprecated in Python 3.10. Python 3.10 and newer requires OpenSSL 1.1.1 ([PEP 644](#)): this OpenSSL version provides a C implementation of `pbkdf2_hmac()` which is faster. (Contributed by Victor Stinner in [gh-94199](#).)
- `xml.etree`: Remove the `ElementTree.Element.copy()` method of the pure Python implementation, deprecated in Python 3.10, use the `copy.copy()` function instead. The C implementation of `xml.etree` has no `copy()` method, only a `__copy__()` method. (Contributed by Victor Stinner in [gh-94383](#).)
- `zipimport`: Remove `find_loader()` and `find_module()` methods, deprecated in Python 3.10: use the `find_spec()` method instead. See [PEP 451](#) for the rationale. (Contributed by Victor Stinner in [gh-94379](#).)
- Remove the `ssl.wrap_socket()` function, deprecated in Python 3.7: instead, create a `ssl.SSLContext` object and call its `ssl.SSLContext.wrap_socket` method. Any package that still uses `ssl.wrap_socket()` is broken and insecure. The function neither sends a SNI TLS extension nor validates server hostname. Code is subject to [CWE-295: Improper Certificate Validation](#). (Contributed by Victor Stinner in [gh-94199](#).)
- Many previously deprecated cleanups in `importlib` have now been completed:
 - References to, and support for `module_repr()` has been eradicated.
- `importlib.util.set_package` has been removed. (Contributed by Brett Cannon in [gh-65961](#).)
- Removed the `suspicious` rule from the documentation Makefile, and removed `Doc/tools/rstlint.py`, both in favor of [sphinx-lint](#). (Contributed by Julien Palard in [gh-98179](#).)

12 Porting to Python 3.12

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

12.1 Changes in the Python API

- More strict rules are now applied for numerical group references and group names in regular expressions. Only sequence of ASCII digits is now accepted as a numerical reference. The group name in bytes patterns and replacement strings can now only contain ASCII letters and digits and underscore. (Contributed by Serhiy Storchaka in [gh-91760](#).)
- Removed `randrange()` functionality deprecated since Python 3.10. Formerly, `randrange(10.0)` losslessly converted to `randrange(10)`. Now, it raises a `TypeError`. Also, the exception raised for non-integral values such as `randrange(10.5)` or `randrange('10')` has been changed from `ValueError` to `TypeError`. This also prevents bugs where `randrange(1e25)` would silently select from a larger range than `randrange(10**25)`. (Originally suggested by Serhiy Storchaka [gh-86388](#).)

- `argparse.ArgumentParser` changed encoding and error handler for reading arguments from file (e.g. `fromfile_prefix_chars` option) from default text encoding (e.g. `locale.getpreferredencoding(False)`) to filesystem encoding and error handler. Argument files should be encoded in UTF-8 instead of ANSI Codepage on Windows.
- Removed the `asyncore`-based `smtpd` module deprecated in Python 3.4.7 and 3.5.4. A recommended replacement is the `asyncio`-based [aiosmtpd](#) PyPI module.
- `shlex.split()`: Passing `None` for `s` argument now raises an exception, rather than reading `sys.stdin`. The feature was deprecated in Python 3.9. (Contributed by Victor Stinner in [gh-94352](#).)
- The `os` module no longer accepts bytes-like paths, like `bytearray` and `memoryview` types: only the exact `bytes` type is accepted for bytes strings. (Contributed by Victor Stinner in [gh-98393](#).)

13 Build Changes

- Python no longer uses `setup.py` to build shared C extension modules. Build parameters like headers and libraries are detected in `configure` script. Extensions are built by `Makefile`. Most extensions use `pkg-config` and fall back to manual detection. (Contributed by Christian Heimes in [gh-93939](#).)
- `va_start()` with two parameters, like `va_start(args, format)`, is now required to build Python. `va_start()` is no longer called with a single parameter. (Contributed by Kumar Aditya in [gh-93207](#).)
- CPython now uses the `ThinLTO` option as the default link time optimization policy if the Clang compiler accepts the flag. (Contributed by Dong-hee Na in [gh-89536](#).)

14 C API Changes

14.1 New Features

- Added the new limited C API function `PyType_FromMetaclass()`, which generalizes the existing `PyType_FromModuleAndSpec()` using an additional metaclass argument. (Contributed by Wenzel Jakob in [gh-93012](#).)
- API for creating objects that can be called using the vectorcall protocol was added to the Limited API:
 - `Py_TPFLAGS_HAVE_VECTORCALL`
 - `PyVectorcall_NARGS()`
 - `PyVectorcall_Call()`
 - `vectorcallfunc`

The `Py_TPFLAGS_HAVE_VECTORCALL` flag is now removed from a class when the class's `__call__()` method is reassigned. This makes vectorcall safe to use with mutable types (i.e. heap types without the `immutable` flag). Mutable types that do not override `tp_call` now inherit the `Py_TPFLAGS_HAVE_VECTORCALL` flag. (Contributed by Petr Viktorin in [gh-93274](#).)

The `Py_TPFLAGS_MANAGED_DICT` and `Py_TPFLAGS_MANAGED_WEAKREF` flags have been added. This allows extensions classes to support object `__dict__` and weakrefs with less bookkeeping, using less memory and with faster access.

- Added two new public functions, `PyEval_SetProfileAllThreads()` and `PyEval_SetTraceAllThreads()`, that allow to set tracing and profiling functions in all running threads in addition to the calling one. (Contributed by Pablo Galindo in [gh-93503](#).)

- Added new function `PyFunction_SetVectorcall()` to the C API which sets the `vectorcall` field of a given `PyFunctionObject`. (Contributed by Andrew Frost in [gh-92257](#).)
- The C API now permits registering callbacks via `PyDict_AddWatcher()`, `PyDict_AddWatch()` and related APIs to be called whenever a dictionary is modified. This is intended for use by optimizing interpreters, JIT compilers, or debuggers. (Contributed by Carl Meyer in [gh-91052](#).)
- Added `PyType_AddWatcher()` and `PyType_Watch()` API to register callbacks to receive notification on changes to a type. (Contributed by Carl Meyer in [gh-91051](#).)

14.2 Porting to Python 3.12

- Legacy Unicode APIs based on `Py_UNICODE*` representation has been removed. Please migrate to APIs based on UTF-8 or `wchar_t*`.
- Argument parsing functions like `PyArg_ParseTuple()` doesn't support `Py_UNICODE*` based format (e.g. `u`, `Z`) anymore. Please migrate to other formats for Unicode like `s`, `z`, `es`, and `U`.
- `tp_weaklist` for all static builtin types is always `NULL`. This is an internal-only field on `PyTypeObject` but we're pointing out the change in case someone happens to be accessing the field directly anyway. To avoid breakage, consider using the existing public C-API instead, or, if necessary, the (internal-only) `_PyObject_GET_WEAKREFS_LISTPTR()` macro.
- This internal-only `PyTypeObject.tp_subclasses` may now not be a valid object pointer. Its type was changed to `void*` to reflect this. We mention this in case someone happens to be accessing the internal-only field directly.

To get a list of subclasses, call the Python method `__subclasses__()` (using `PyObject_CallMethod()`, for example).
- An unrecognized format character in `PyUnicode_FromFormat()` and `PyUnicode_FromFormatV()` now sets a `SystemError`. In previous versions it caused all the rest of the format string to be copied as-is to the result string, and any extra arguments discarded. (Contributed by Serhiy Storchaka in [gh-95781](#).)
- Fixed wrong sign placement in `PyUnicode_FromFormat()` and `PyUnicode_FromFormatV()`. (Contributed by Philip Georgi in [gh-95504](#).)
- Extension classes wanting to add a `__dict__` or weak reference slot should use `Py_TPFLAGS_MANAGED_DICT` and `Py_TPFLAGS_MANAGED_WEAKREF` instead of `tp_dictoffset` and `tp_weaklistoffset`, respectively. The use of `tp_dictoffset` and `tp_weaklistoffset` is still supported, but does not fully support multiple inheritance ([gh-95589](#)), and performance may be worse. Classes declaring `Py_TPFLAGS_MANAGED_DICT` should call `_PyObject_VisitManagedDict()` and `_PyObject_ClearManagedDict()` to traverse and clear their instance's dictionaries. To clear weakrefs, call `PyObject_ClearWeakRefs()`, as before.
- The `PyUnicode_FSDecoder()` function no longer accepts bytes-like paths, like `bytearray` and `memoryview` types: only the exact `bytes` type is accepted for bytes strings. (Contributed by Victor Stinner in [gh-98393](#).)

14.3 Deprecated

- Deprecate global configuration variable:
 - `Py_DebugFlag`; use `PyConfig.parser_debug`
 - `Py_VerboseFlag`; use `PyConfig.verbose`
 - `Py_QuietFlag`; use `PyConfig.quiet`
 - `Py_InteractiveFlag`; use `PyConfig.interactive`
 - `Py_InspectFlag`; use `PyConfig.inspect`
 - `Py_OptimizeFlag`; use `PyConfig.optimization_level`
 - `Py_NoSiteFlag`; use `PyConfig.site_import`
 - `Py_BytesWarningFlag`; use `PyConfig.bytes_warning`
 - `Py_FrozenFlag`; use `PyConfig.pathconfig_warnings`
 - `Py_IgnoreEnvironmentFlag`; use `PyConfig.use_environment`
 - `Py_DontWriteBytecodeFlag`; use `PyConfig.write_bytecode`
 - `Py_NoUserSiteDirectory`; use `PyConfig.user_site_directory`
 - `Py_UnbufferedStdioFlag`; use `PyConfig.buffered_stdio`
 - `Py_HashRandomizationFlag`; use `PyConfig.use_hash_seed` and `PyConfig.hash_seed`
 - `Py_IsolatedFlag`; use `PyConfig.isolated`
 - `Py_LegacyWindowsFSEncodingFlag`; use `PyConfig.legacy_windows_fs_encoding`
 - `Py_LegacyWindowsStdioFlag`; use `PyConfig.legacy_windows_stdio`
 - `Py_FileSystemDefaultEncoding`; use `PyConfig.filesystem_encoding`
 - `Py_FileSystemDefaultEncodeErrors`; use `PyConfig.filesystem_errors`
 - `Py_UTF8Mode`; use `PyPreConfig.utf8_mode` (see `Py_PreInitialize()`)

The `Py_InitializeFromConfig()` API should be used with `PyConfig` instead. (Contributed by Victor Stinner in [gh-77782](#).)

- Creating `immutable` types with mutable bases is deprecated and will be disabled in Python 3.14.

14.4 Removed

- Remove the `token.h` header file. There was never any public tokenizer C API. The `token.h` header file was only designed to be used by Python internals. (Contributed by Victor Stinner in [gh-92651](#).)
- Legacy Unicode APIs has been removed. See [PEP 623](#) for detail.
 - `PyUnicode_WCHAR_KIND`
 - `PyUnicode_AS_UNICODE()`
 - `PyUnicode_AsUnicode()`
 - `PyUnicode_AsUnicodeAndSize()`
 - `PyUnicode_AS_DATA()`
 - `PyUnicode_FromUnicode()`

- `PyUnicode_GET_SIZE()`
- `PyUnicode_GetSize()`
- `PyUnicode_GET_DATA_SIZE()`
- Remove the `PyUnicode_InternImmortal()` function and the `SSTATE_INTERNED_IMMORTAL` macro.
(Contributed by Victor Stinner in [gh-85858](#).)

Index

P

Python Enhancement Proposals

PEP 451, 8

PEP 594, 5, 7

PEP 623, 2, 4, 11

PEP 644, 8