What's new in Python 3.16

- Summary -- Release highlights
- New features
- Other language changes
- Default interactive shell
- New modules
- Improved modules
  - argparse
  - asyncio
  - bdb
  - bz2
  - codecs
  - compression.zstd
  - concurrent.futures
  - concurrent.interpreters
  - csv
  - ctypes
  - curses
  - datetime
  - dbm.gnu
  - dbm.ndbm
  - decimal
  - difflib
  - encodings
  - gzip
  - imaplib
  - io
  - ipaddress
  - logging

- lzma
- math
- multiprocessing
- os
- pydoc
- pickle
- quopri
- re
- readline
- shlex
- shutil
- socket
- sqlite3
- ssl
- symtable
- tkinter
- turtle
- unicodedata
- uuid
- xml
- zipfile
- zlib

- Optimizations
  - datetime and time
  - re
  - uuid
- Removed
  - annotationlib
  - array
  - asyncio
  - functools
  - logging
  - mimetypes
  - shutil

- symtable
- sys
- sysconfig
- tarfile
- xml.etree.ElementTree

- Deprecated
  - New deprecations
  - Pending removal in Python 3.17
  - Pending removal in Python 3.18
  - Pending removal in Python 3.19
  - Pending removal in Python 3.20
  - Pending removal in Python 3.21
  - Pending removal in future versions
- Porting to Python 3.16
- Build changes
- C API changes
  - New features
  - Porting to Python 3.16
  - Deprecated C APIs
    - Pending removal in Python 3.18
    - Pending removal in Python 3.19
    - Pending removal in Python 3.20
    - Pending removal in future versions
  - Removed C APIs

Editor
Savannah Ostrowski

> * Anyone can add text to this document. Do not spend very much time on the wording of your changes, because your text will probably get rewritten to some degree.
>
> * The maintainer will go through Misc/NEWS periodically and add changes; it's therefore more important to add your changes to Misc/NEWS than to this file.

> * This is not a complete list of every single change; completeness is the purpose of Misc/NEWS. Some changes I consider too small or esoteric to include. If such a change is added to the text, I'll just remove it. (This is another reason you shouldn't spend too much time on writing your addition.)
>
> * If you want to draw your new text to the attention of the maintainer, add 'XXX' to the beginning of the paragraph or section.
>
> * It's OK to just add a fragmentary note about a change. For example: "XXX Describe the transmogrify() function added to the socket module." The maintainer will research the change and write the necessary text.
>
> * You can comment out your additions if you like, but it's not necessary (especially when a final release is some months away).
>
> * Credit the author of a patch or bugfix. Just the name is sufficient; the e-mail address isn't necessary.
>
> - It's helpful to add the issue number as a comment:
>
> XXX Describe the transmogrify() function added to the socket module. (Contributed by P.Y. Developer in `12345`.)
>
> This saves the maintainer the effort of going through the VCS log when researching a change.

This article explains the new features in Python 3.16, compared to 3.15.

For full details, see the `changelog <changelog>`.

> [!NOTE] Prerelease users should be aware that this document is currently in draft form. It will be updated substantially as Python 3.16 moves towards release, so it's worth checking back even after reading earlier versions.

Summary -- Release highlights

New features

Other language changes

- The `iter` function now accepts the *stop_exception* parameter. The created iterator stops when the callable raises the specified exception. The second parameter is now named *stop_value* and can be passed by keyword. `aiter` now accepts the same *stop_value* and *stop_exception* parameters, calling an asynchronous callable and awaiting the result. (Contributed by Serhiy Storchaka in `64862`.)
- `memoryview.cast` now allows casting a multidimensional F-contiguous view to a one-dimensional view. (Contributed by Jaemin Park in `91484`.)
- `memoryview.cast` now accepts an *order* parameter. With `order='F'` it returns a zero-copy Fortran-contiguous (column-major) view of a flat buffer, which is useful for interfacing with column-major libraries. (Contributed by Serhiy Storchaka in `78959`.)
- `Frame objects <frame-objects>` now support `weak references <weakref>`. This allows associating extra data with active frames, for example in debuggers, without keeping the frames (and everything they reference) alive indefinitely. (Contributed by Łukasz Langa in `102960`.)
- `TypeError` messages for some instance methods called with too many positional arguments now suggest checking whether the method definition is missing the `self` parameter. (Contributed by Suren Nihalani in `152315`.)
- `anext` and `aiter` are now implemented in Python instead of C. The awaitable returned by `anext(it, default)` and the `__anext__()` of the iterator returned by `aiter(callable, ...)` are now plain coroutines, so introspection tools such as `asyncio.print_call_graph` can see through them into the awaited code. (Contributed by Kumar Aditya in `157361`.)

Default interactive shell

- The `REPL` now displays error messages when editing commands cannot be performed, such as when tab completion finds no matches or a history search fails. (Contributed by Bartosz Sławecki in `148228`.)

New modules

- None yet.

Improved modules

argparse

- `~argparse.ArgumentGroup` and `~argparse.MutuallyExclusiveGroup`, the types returned by `~argparse.ArgumentParser.add_argument_group` and `~argparse.ArgumentParser.add_mutually_exclusive_group`, are now public. The `_`-prefixed names remain as aliases. (Contributed by Bernát Gábor in `144812`.)

asyncio

- Add the *mode* parameter to `asyncio.loop.create_unix_server` and `asyncio.start_unix_server` to set the permissions of the Unix socket file created for *path*. (Contributed by Sam Bull in `94984`.)
- Add *eager_tasks* parameter to `asyncio.run` function and `asyncio.Runner` class constructor. (Contributed by Andrew Svetlov in `128289`.)

bdb

- `bdb.BdbQuit` is now a subclass of `BaseException`, instead of `Exception`, to allow it to be used for cleanly exiting the debugger without being accidentally caught by user code. (Contributed by Tian Gao in `149337`.)

bz2

- Added constants `~bz2.bzlib_version` and `~bz2.bzlib_version_info` which provide information about the version of the bzip2 compression library in use. (Contributed by Serhiy Storchaka in `117404`.)

codecs

- On platforms that provide the C library's `iconv(3)` function, every encoding known to `iconv` for which Python has no built-in codec is now available (for example `cp1133`). Prefixing an encoding name with `iconv:` forces the `iconv`-based codec even when a built-in codec of the same name exists. (Contributed by Serhiy Storchaka in `152997`.)

compression.zstd

- Added constants `~compression.zstd.ZSTD_VERSION` and `~compression.zstd.ZSTD_VERSION_INFO` which provide information about the version of the zstd library that was used for building

the module. `~compression.zstd.zstd_version_info` is now a named tuple. (Contributed by Serhiy Storchaka in `117404`.)

concurrent.futures

- The iterator returned by `concurrent.futures.Executor.map` is no longer automatically closed if a function call raises an exception. Use the `!close` method to explicitly close the iterator. (Contributed by xzmeng and Serhiy Storchaka in `108518`.)

concurrent.interpreters

- `~concurrent.interpreters.Interpreter` objects now support the `context manager` protocol. Exiting the `with` block calls `~concurrent.interpreters.Interpreter.close`. (Contributed by Peter Bierma in `158370`.)

csv

- `csv.Sniffer.sniff` now deduces the dialect by trial parsing instead of heuristics based on matching isolated fragments and on character frequencies, so the result is consistent with how `csv.reader` will parse the sample. It can now detect the *escapechar* parameter, accepts non-ASCII delimiters in the *delimiters* argument, no longer misdetects the delimiter when the sample contains delimiter characters inside quoted fields, and no longer takes quadratic time on quoted samples. The results may differ from those of earlier Python versions. (Contributed by Serhiy Storchaka in `83273`.)
- `csv.Sniffer.sniff` now detects the *lineterminator* parameter by a majority vote among the line endings of the sample, instead of always returning `'\r\n'`. (Contributed by Serhiy Storchaka in `75008`.)

ctypes

- Add `ctypes.util.struct` for generating `~ctypes.Structure` types from an annotation-based syntax, similar to how the `dataclasses` module is used. (Contributed by Peter Bierma in `104533`.)
- Add `ctypes.util.wrap_dll_function` for generating function pointers through a function signature. (Contributed by Peter Bierma in `153903`.)
- Added constants `~ctypes.LIBFFI_VERSION`, `~ctypes.libffi_version`, `~ctypes.LIBFFI_VERSION_INFO` and `~ctypes.libffi_version_info`, which provide information about the version of the libffi library in use. They are only available with libffi 3.5 or later. (Contributed by Serhiy Storchaka in `157487`.)
- Objects with an `!_as_parameter_` attribute are now implicitly converted when passed for a `~ctypes.POINTER` argument. (Contributed by Peter Bierma in `71990`.)

- `Callback functions <ctypes-callback-functions>` can now return `~ctypes.Structure` and `~ctypes.Union` instances by value. (Contributed by Russell Keith-Magee in `49960`.)

curses

- The `curses` character-cell window methods now accept a full character cell --- a spacing character optionally followed by combining characters --- in addition to a single integer or byte character. This affects `~curses.window.addch`, `~curses.window.bkgd`, `~curses.window.bkgdset`, `~curses.window.border`, `~curses.window.box`, `~curses.window.echochar`, `~curses.window.hline`, `~curses.window.insch` and `~curses.window.vline`. Also add the wide-character read methods `~curses.window.get_wstr` and `~curses.window.in_wstr`, the counterparts of `~curses.window.getstr` and `~curses.window.instr` that return a `str` rather than `bytes`, and the module functions `curses.erasewchar`, `curses.killwchar` and `curses.wunctrl`, the wide-character counterparts of `curses.erasechar`, `curses.killchar` and `curses.unctrl`. On a narrow (non-ncursesw) build the character cell holds a single character without combining marks, representable as one byte in the window's encoding, and `~curses.window.in_wstr` returns its decoded text. (Contributed by Serhiy Storchaka in `151757`.)
- Add the `curses.complexchar` type, representing a styled character cell (its text, attributes and color pair), and the window methods `~curses.window.in_wch` and `~curses.window.getbkgrnd` that return one --- the counterparts of `~curses.window.inch` and `~curses.window.getbkgd`. The character-cell methods, such as `~curses.window.addch` and `~curses.window.border`, now also accept a `~curses.complexchar`. These work whether or not Python was built against a wide-character-aware curses library; on a narrow build a cell holds a single character representable as one byte in the window's encoding (so only 8-bit locales are supported). (Contributed by Serhiy Storchaka in `152233`.)
- Add the `curses.complexstr` type, an immutable run of styled cells (the string counterpart of `~curses.complexchar`), and the window method `~curses.window.in_wchstr` that returns one. The string-cell methods `~curses.window.addstr`, `~curses.window.addnstr`, `~curses.window.insstr` and `~curses.window.insnstr` now also accept a `~curses.complexstr`. Like `~curses.complexchar`, it works whether or not Python was built against a wide-character-aware curses library. (Contributed by Serhiy Storchaka in `152233`.)
- The wide-character `curses` functions and methods `~curses.window.get_wch`, `~curses.window.get_wstr`, `curses.unget_wch`, `curses.erasewchar`, `curses.killwchar` and `curses.wunctrl` now also work when Python is not built against a wide-character-aware curses library, on an 8-bit locale, where each character is a single byte in the relevant encoding. `curses.ungetch` now also accepts a one-character string, like `curses.unget_wch`; on a wide-character build it can be any character (previously a multibyte character raised `OverflowError`). (Contributed by Serhiy Storchaka in `152470`.)
- Add support for multiple terminals to the `curses` module: the new functions `curses.newterm`, `curses.set_term` and `curses.new_prescr`, the corresponding `screen <curses-screen-objects>` object, and the `window.use() <curses.window.use>` method. (Contributed by Serhiy Storchaka in `90092`.)

- Add the `curses` window methods `~curses.window.attr_get`, `~curses.window.attr_set`, `~curses.window.attr_on`, `~curses.window.attr_off` and `~curses.window.color_set`, which pass the color pair as a separate argument instead of packing it into the attribute value, and the corresponding `WA_*` attribute constants. (Contributed by Serhiy Storchaka in `152219`.)
- Add the `curses.term_attrs` function, which returns the supported video attributes as `WA_* <curses-wa-constants>` values, the counterpart of `curses.termattrs`. (Contributed by Serhiy Storchaka in `152332`.)
- Add the `WACS_*` constants to the `curses` module, the counterparts of the `ACS_* <curses-acs-codes>` line-drawing codes as `curses.complexchar` cells. (Contributed by Serhiy Storchaka in `155863`.)
- Add the `curses` functions `curses.alloc_pair`, `curses.find_pair`, `curses.free_pair` and `curses.reset_color_pairs` for dynamic color-pair management, available when built against a wide-character ncurses with extended-color support. (Contributed by Serhiy Storchaka in `151774`.)
- Add the `curses` functions `~curses.scr_dump`, `~curses.scr_restore`, `~curses.scr_init` and `~curses.scr_set`, which dump the whole screen to a file and restore it. (Contributed by Serhiy Storchaka in `152260`.)
- Add the `curses` window method `~curses.window.dupwin`, which returns a new window that is an independent duplicate of an existing one. (Contributed by Serhiy Storchaka in `152258`.)
- Add the soft-label-key functions to the `curses` module, which manage a row of labels along the bottom line of the screen: `~curses.slk_init`, `~curses.slk_set`, `~curses.slk_label`, `~curses.slk_refresh`, `~curses.slk_noutrefresh`, `~curses.slk_clear`, `~curses.slk_restore`, `~curses.slk_touch`, the attribute functions `~curses.slk_attron`, `~curses.slk_attroff`, `~curses.slk_attrset`, `~curses.slk_attr`, `~curses.slk_attr_on`, `~curses.slk_attr_off`, `~curses.slk_attr_set`, and `~curses.slk_color`. (Contributed by Serhiy Storchaka in `152263`.)
- Add the `curses` key-management functions `~curses.define_key`, `~curses.key_defined` and `~curses.keyok`, available when built against an ncurses with `NCURSES_EXT_FUNCS`. (Contributed by Serhiy Storchaka in `152334`.)
- Add the `curses.has_mouse` function and the `curses.window.mouse_trafo` method, completing the `curses` mouse interface. (Contributed by Serhiy Storchaka in `152325`.)
- Add `curses` functions and window methods that report state which could previously only be set, such as `curses.window.is_keypad`, `curses.window.getparent` and `curses.is_cbreak`, available when built against an ncurses with `NCURSES_EXT_FUNCS`. (Contributed by Serhiy Storchaka in `151776`.)
- Add `curses.nofilter`, which undoes the effect of `curses.filter`. (Contributed by Serhiy Storchaka in `151744`.)
- `curses.textpad.Textbox` now supports entering and reading back the full Unicode range, including combining characters, when curses is built with wide-character support. (Contributed by Serhiy Storchaka in `133031`.)

datetime

- Add the `%Nf` format code (where *N* is from 1 to 6) to `datetime.datetime.strftime`, `datetime.time.strftime`, `datetime.date.strftime`, `datetime.datetime.strptime` and `datetime.time.strptime`. (Contributed by Stan Ulbrych and Karoline Pauls in `157290`.)
- Add `datetime.timezone.name`, the *name* passed to the constructor, or `None` if no name was passed. (Contributed by Stan Ulbrych in `131210`.)

dbm.gnu

- Added constants `~dbm.gnu.GDBM_VERSION_INFO`, `~dbm.gnu.gdbm_version_info` and `~dbm.gnu.gdbm_version`, which provide information about the version of the GDBM library in use. (Contributed by Serhiy Storchaka in `157478`.)

dbm.ndbm

- Added constants which provide information about the version of the underlying library in use: `~dbm.ndbm.GDBM_VERSION_INFO`, `~dbm.ndbm.gdbm_version_info` and `~dbm.ndbm.gdbm_version` if it is GDBM, or `~dbm.ndbm.BDB_VERSION`, `~dbm.ndbm.bdb_version`, `~dbm.ndbm.BDB_VERSION_INFO` and `~dbm.ndbm.bdb_version_info` if it is Berkeley DB. `dbm.ndbm.library` is now `'ndbm'` instead of `'GNU gdbm'` for a classic NDBM library. (Contributed by Serhiy Storchaka in `157480`.)

decimal

- Added constants `~decimal.LIBMPDEC_VERSION`, `~decimal.libmpdec_version`, `~decimal.LIBMPDEC_VERSION_INFO`, and `~decimal.libmpdec_version_info` in the C implementation, which provide information about the version of the libmpdec library in use. (Contributed by Serhiy Storchaka in `157475`.)

difflib

- Expose the optional *autojunk* parameter of `difflib.SequenceMatcher` in public functions and class methods in `difflib`, allowing callers to control the automatic junk heuristic. (Contributed by Tomasz Kazimierczak in `118150`.)

encodings

- Add the `utf-7-imap` codec, implementing the modified UTF-7 encoding used for international IMAP4 mailbox names (`3501`). (Contributed by Serhiy Storchaka in `66788`.)

- `encodings.idna.nameprep` now takes a *limit* argument that allows rejecting extremely large input early. (Contributed by Petr Viktorin in `157675`.)

gzip

- `gzip.open` now accepts an optional argument `mtime` which is passed on to the constructor of the `~gzip.GzipFile` class. (Contributed by Marin Misur in `91372`.)

imaplib

- Add the `~imaplib.IMAP4.id` method, a wrapper for the `ID` command (`2971`). (Contributed by Serhiy Storchaka in `98092`.)
- Add the `~imaplib.IMAP4.move` method, a wrapper for the `MOVE` command (`6851`). (Contributed by Serhiy Storchaka in `77508`.)
- Add the `~imaplib.IMAP4.login_plain` method, which authenticates using the `PLAIN` SASL mechanism (`4616`) and supports non-ASCII user names and passwords. (Contributed by Przemysław Buczkowski and Serhiy Storchaka in `89869`.)
- Non-ASCII mailbox names are now automatically encoded as modified UTF-7 (`3501`, section 5.1.3) in the default mode, so international mailbox names can be passed as ordinary `str` without enabling `UTF8=ACCEPT` (under which they are sent as UTF-8). (Contributed by Serhiy Storchaka in `49555`.)
- The `~imaplib.IMAP4.copy`, `~imaplib.IMAP4.move`, `~imaplib.IMAP4.fetch`, `~imaplib.IMAP4.store`, `~imaplib.IMAP4.search`, `~imaplib.IMAP4.sort`, `~imaplib.IMAP4.thread` and `~imaplib.IMAP4.expunge` methods now accept a keyword-only *uid* argument that selects the corresponding `UID` command, as a more convenient alternative to `~imaplib.IMAP4.uid`. (Contributed by Serhiy Storchaka in `153502`.)
- `~imaplib.IMAP4.search`, `~imaplib.IMAP4.sort` and `~imaplib.IMAP4.thread` (and the corresponding `uid` commands) now encode `str` search criteria to the declared *charset*, so international search text can be passed as an ordinary `str`. When *charset* is `None` (as it must be under `UTF8=ACCEPT`), the criteria are sent using the connection encoding instead. (Contributed by Serhiy Storchaka in `153494`.)
- Command methods now accept structured arguments, so the module takes care of quoting instead of the caller. A *message_set* and lists of flags or other atoms can be passed as sequences instead of preformatted strings, and the `~imaplib.IMAP4.search`, `~imaplib.IMAP4.fetch`, `~imaplib.IMAP4.sort`, `~imaplib.IMAP4.thread` and `~imaplib.IMAP4.uid` methods accept a *params* keyword argument that substitutes and quotes `?` placeholders, in the manner of `sqlite3` parameter substitution. (Contributed by Serhiy Storchaka in `153521`.)

io

- Add `io.BytesIO.peek` method to read without advancing position. (Contributed by Marcel Martin in `90533`.)

ipaddress

- Add `IPv4Network.next_network() <ipaddress.IPv4Network.next_network>` and `IPv6Network.next_network() <ipaddress.IPv6Network.next_network>` methods to find the next nearest network with a specific prefix size. (Contributed by Faisal Mahmood in `87027`.)

logging

- `~logging.handlers.TimedRotatingFileHandler` now uses the creation time instead of the last modification time of an existing log file as the basis for the first rotation after handler creation, if supported by the OS and file system. This allows it to be used in short-running programs that start and end before the rotation interval expires. (Contributed by Iván Márton and Serhiy Storchaka in `84649`.)

lzma

- Add support of new BCJ filters ARM64 and RISC-V via `!lzma.FILTER_ARM64` and `!lzma.FILTER_RISCV`. Note that the new filters will work only if the runtime library supports them. ARM64 filter requires `lzma` 5.4.0 or newer while RISC-V requires 5.6.0 or newer. (Contributed by Chien Wong in `115988`.)
- Added constants `~lzma.LZMA_VERSION`, `~lzma.lzma_version`, `~lzma.LZMA_VERSION_INFO`, and `~lzma.lzma_version_info`, which provide information about the version of the lzma library in use. (Contributed by Serhiy Storchaka in `117404`.)

math

- Added trigonometric functions that work in units of half turns, rather than radians. The new functions `math.acospi`, `math.asinpi`, `math.atanpi`, and `math.atan2pi` return half-turn angles. The new functions `math.cospi`, `math.sinpi`, and `math.tanpi` take half-turn angle arguments. These functions are recommended by IEEE 754-2019 and standardized in C23. (Contributed by Jeff Epler in `150534`.)

multiprocessing

- Add the optional *buffersize* parameter to `multiprocessing.pool.Pool.imap` and `multiprocessing.pool.Pool.imap_unordered` to limit the number of submitted tasks whose

results have not yet been yielded. If the buffer is full, iteration over the *iterable* pauses until a result is yielded from the buffer. To fully utilize pool's capacity when using this feature, set *buffersize* at least to the number of processes in pool (to consume *iterable* as you go), or even higher (to prefetch the next `N=buffersize-processes` arguments). (Contributed by Oleksandr Baltian in `136871`.)

os

- Add `os.pidfd_getfd` for duplicating a file descriptor from another process via a pidfd. Available on Linux 5.6+. (Contributed by Maurycy Pawłowski-Wieroński in `149464`.)
- The following functions are now available on Android: `os.chroot`, `os.initgroups`, `os.setegid`, `os.seteuid`, `os.setgid`, `os.setregid`, `os.setresgid`, `os.setresuid`, `os.setreuid`, and `os.setuid`. Calling them without sufficient privileges now raises `PermissionError` instead of the functions being unavailable. (Contributed by Md Arif in `152936`.)
- `os.readlink` now accepts a file descriptor referring to a symlink on Linux, Android, macOS and iOS. (Contributed by OOTS in `157899`.)

pydoc

- Modernize the HTML output of `pydoc`. The pages generated by the `pydoc` HTTP server now use semantic HTML5 markup and a new style sheet based on the python-docs-theme used by docs.python.org, with dark mode support. Class members inherited from other classes are collapsed by default. (Contributed by Serhiy Storchaka in `153906`.)

pickle

- Weak references (`weakref.ref`), `weakref.WeakMethod`, `weakref.WeakSet`, `weakref.WeakValueDictionary` and `weakref.WeakKeyDictionary` can now be pickled if the weakly referenced objects are strongly referenced elsewhere in the same pickle; otherwise pickling raises `pickle.PicklingError`. `!WeakMethod` and `!WeakSet` can now also be copied. (Contributed by Serhiy Storchaka in `74876`.)

quopri

- `quopri.decode`, `quopri.decodestring` and `binascii.a2b_qp` gained a *strip_ws* parameter. When true, trailing whitespace is stripped from each line while decoding, as required by `2045` for a quoted-printable body. (Contributed by Serhiy Storchaka in `62222`.)

re

- `re` now supports set operations and nested sets in character classes, as described in Unicode Technical Standard #18: set difference (`[A--B]`), intersection (`[A&&B]`) and union (`[A||B]`), where an operand may be a nested set written in square brackets. For example, `[a-z--[aeiou]]` matches an ASCII lowercase consonant. (Contributed by Serhiy Storchaka in `152100`.)
- Regular expressions now support Unicode property escapes `\p{...}` and `\P{...}`, which match a character by a Unicode property -- for example `\p{Lu}` (an uppercase letter), `\p{Cased}` or `\p{ASCII}`. See `the regular expression syntax <re-syntax>` for the supported properties. (Contributed by Serhiy Storchaka in `95555`.)

readline

- Added constants `~readline.READLINE_VERSION_INFO`, `~readline.readline_version_info` and `~readline.readline_version`, which provide information about the version of the Readline library in use. (Contributed by Serhiy Storchaka in `157485`.)

shlex

- Add keyword-only parameter *force* to `shlex.quote` to force quoting a string, even if it is already safe for a shell without being quoted. (Contributed by Jay Berry in `148846`.)

shutil

- Add keyword-only *copy_dir_stat* parameter to `shutil.copytree`: set it to false to skip copying directory metadata with `shutil.copystat`, e.g. when copying a read-only tree. (Contributed by Łukasz Langa in `89721`.)
- `shutil.unpack_archive` can now unpack wheel files using the `whl` format. (Contributed by Emma Harper Smith and Stan Ulbrych in `84027`.)

socket

- The `socket.sethostname` function is now available on Android. Calling it without sufficient privileges now raises `PermissionError` instead of the function being unavailable. (Contributed by Md Arif in `152936`.)

sqlite3

- `sqlite3.Blob` now supports negative-step slices for reading and writing (for example, `blob[9:0:-2]`). Previously, such slices would raise `SystemError` or `ValueError`. (Contributed by Jiseok CHOI in `150449`.)
- Added constants `~sqlite3.SQLITE_VERSION` and `~sqlite3.SQLITE_VERSION_INFO` which provide information about the version of the SQLite library that was used for building the module. `~sqlite3.sqlite_version_info` is now a named tuple. (Contributed by Serhiy Storchaka in `157470`.)

ssl

- Added support for `Encrypted Client Hello <ssl-encrypted-client-hello>` (ECH) with OpenSSL 4.0 or later. Clients can supply an ECHConfigList discovered through DNS, for example using a third-party library such as `dnspython` to query HTTPS or SVCB records, as the *ech_config* argument of `~ssl.SSLContext.wrap_socket` and `~ssl.SSLContext.wrap_bio` or with `~ssl.SSLContext.set_ech_config`. Servers can use `~ssl.SSLContext.load_ech_keys` to load ECH private keys and their configurations from PEM files. (Contributed by Peter Bierma and Iain R. Learmonth in `89730`.)
- `ssl.OPENSSL_VERSION_INFO` is now a named tuple. Added `~ssl.OPENSSL_API_VERSION_INFO` which provides information about the version of the OpenSSL library that was used for building the module. (Contributed by Serhiy Storchaka in `157513`.)
- The `ssl` module now rejects TLS certificate chains that contain one of a set of compromised eMudhra intermediate CA certificates. The affected certificates are identified by their public key, so cross-signed copies that reuse the same key are blocked as well. It is a defense-in-depth backstop that applies even when an application supplies its own root CA bundle, and regardless of whether the platform trust store has removed trust. (Contributed by Christian Heimes in `160040` for `2026-115557`.)

symtable

- `symtable.symtable` now accepts an AST object, like the built-in `compile`. (Contributed by Serhiy Storchaka in `153844`.)
- Inlined list, set and dict comprehensions (`709`) are now represented as their own symbol table entries, of type `~symtable.SymbolTableType.INLINED_COMPREHENSION`. Each such entry is a lexical child of the enclosing scope and records the comprehension's own locals, cells, and free names. It does not correspond to a separate compilation unit. For names loaded only inside an inlined list, set, or dict comprehension, `symtable.Symbol.is_referenced` can now return `True` on the enclosing function. (Contributed by Irit Katriel in `124697`.)

tkinter

- Added many `tkinter.ttk.Treeview` methods wrapping the enhanced `ttk::treeview` widget commands from Tk 9.1, such as `~tkinter.ttk.Treeview.sort`, `~tkinter.ttk.Treeview.search`

, `~tkinter.ttk.Treeview.expand`, `~tkinter.ttk.Treeview.collapse`, `~tkinter.ttk.Treeview.hide`, `~tkinter.ttk.Treeview.unhide`, and methods for cell focus, selection and tagging. The `~tkinter.ttk.Treeview.expand` and `~tkinter.ttk.Treeview.collapse` methods (without recursion) also work on Tk older than 9.1. (Contributed by Serhiy Storchaka in `151910`.)

- Added constants `~tkinter.TCL_VERSION_INFO` and `~tkinter.TK_VERSION_INFO`, named tuples with the versions of the Tcl and Tk libraries that were used for building the module. (Contributed by Serhiy Storchaka in `157514`.)
- Added new `!tkinter.Text` methods `~tkinter.Text.edit_canundo` and `~tkinter.Text.edit_canredo` which return whether an undo or redo is possible. (Contributed by Serhiy Storchaka in `151674`.)
- Added new `!tkinter.Text` methods `~tkinter.Text.sync` and `~tkinter.Text.pendingsync` which control and report the synchronization of the displayed view with the underlying text. (Contributed by Serhiy Storchaka in `151675`.)
- Added the `ttk.Style.theme_styles <tkinter.ttk.Style.theme_styles>` method which returns the list of styles defined in a theme. (Contributed by Serhiy Storchaka in `151920`.)
- Added new `!tkinter.Canvas` methods `~tkinter.Canvas.rchars` which replaces the text or coordinates of canvas items, and `~tkinter.Canvas.rotate` which rotates the coordinates of canvas items. (Contributed by Serhiy Storchaka in `151876`.)
- Added a `!validate` method to the `!tkinter.Entry` and `!tkinter.Spinbox` widgets, which forces an evaluation of the validation command. (Contributed by Serhiy Storchaka in `151878`.)
- Added the `tkinter.Menu.postcascade` method, and the `~tkinter.Misc.tk_scaling` and `~tkinter.Misc.tk_inactive` methods which respectively query or set the display scaling factor and report the user idle time. (Contributed by Serhiy Storchaka in `151881`.)
- Added the `!tk_print` method to `tkinter.Canvas` and `tkinter.Text` which prints the contents of the widget using the native print dialog. It requires Tk 8.7/9.0 or newer. (Contributed by Serhiy Storchaka in `153256`.)
- Added new window-management methods `~tkinter.Misc.winfo_isdark` (dark mode detection), `~tkinter.Wm.wm_iconbadge` (application icon badge) and `~tkinter.Wm.wm_stackorder` (toplevel stacking order). (Contributed by Serhiy Storchaka in `151874`.)
- Added the `~tkinter.Misc.tk_appname`, `~tkinter.Misc.tk_useinputmethods` and `~tkinter.Misc.tk_caret` methods, exposing the application send name, the X Input Methods state and the input method caret location. (Contributed by Serhiy Storchaka in `151886`.)
- Added support for more options in `!tkinter.PhotoImage` methods: the *format* parameter of `~tkinter.PhotoImage.put`, the *metadata* parameter of `~tkinter.PhotoImage.put`, `~tkinter.PhotoImage.read`, `~tkinter.PhotoImage.write` and `~tkinter.PhotoImage.data`, and the *withalpha* parameter of `~tkinter.PhotoImage.get`. (Contributed by Serhiy Storchaka in `151890`.)
- Added the `~tkinter.PhotoImage.redither` method which recalculates the dithered image when its data was supplied in pieces. (Contributed by Serhiy Storchaka in `151888`.)
- `tkinter.OptionMenu` now accepts arbitrary `!tkinter.Menubutton` options as keyword arguments, which can also override its default appearance. (Contributed by Serhiy

Storchaka in `101284`.)

- The `tkinter.simpledialog` dialogs were modernized to match the look and feel of the native Tk dialogs. `!tkinter.simpledialog.SimpleDialog` and the `~tkinter.simpledialog.askinteger`, `~tkinter.simpledialog.askfloat` and `~tkinter.simpledialog.askstring` dialogs are now built from the themed `tkinter.ttk` widgets instead of the classic `tkinter` widgets; the `!tkinter.simpledialog.Dialog` base class still defaults to the classic widgets for compatibility. Both `!Dialog` and `!SimpleDialog` gained a *use_ttk* parameter that selects between the classic Tk widgets and the themed ttk widgets. `!SimpleDialog` also gained *bitmap* and *detail* parameters, draws the standard icons with themed images in the ttk version, and accepts mappings of button options as *buttons* entries. (Contributed by Serhiy Storchaka in `59396`.)
- The `!tkinter.filedialog.FileDialog` dialog and its `!tkinter.filedialog.LoadFileDialog` and `!tkinter.filedialog.SaveFileDialog` subclasses are now built from the themed `tkinter.ttk` widgets by default instead of the classic `tkinter` widgets, and gained a *use_ttk* parameter to select between them. (Contributed by Serhiy Storchaka in `59396`.)
- Added the `tkinter.systray` module which provides the `~tkinter.systray.SysTrayIcon` class as an interface to the system tray icon and the `~tkinter.systray.notify` function which sends a desktop notification. They require Tk 8.7/9.0 or newer. (Contributed by Serhiy Storchaka in `153259`.)
- `tkinter.scrolledtext.ScrolledText` gained a *use_ttk* parameter to use the themed `tkinter.ttk` frame and scroll bar instead of the classic `tkinter` widgets. (Contributed by Serhiy Storchaka in `59396`.)
- `tkinter.font.Font` can now wrap a font description without creating a new named font, by passing it as *font* with `exists=True` and no *name*. This avoids a loss of precision in `~tkinter.font.Font.actual`, `~tkinter.font.Font.measure` and `~tkinter.font.Font.metrics`. (Contributed by Serhiy Storchaka in `143990`.)
- Added the `tkinter.fontchooser` module which provides the `~tkinter.fontchooser.FontChooser` class as an interface to the native font selection dialog. (Contributed by Serhiy Storchaka in `72880`.)
- Values of several Tcl object types returned by `tkinter` are now converted to the corresponding Python type instead of being wrapped in a `!_tkinter.Tcl_Obj`: `index`, `window`, `nsName` and `parsedVarName` objects to `str`, and `pixel` screen distances with no unit suffix to `int` or `float`. (Contributed by Serhiy Storchaka in `153513`.)

turtle

- Translations of the `turtle` docstrings are now distributed on PyPI in the `turtle-translations` package. Additionally, added the `PYTHON_TURTLE_LANG` environment variable to select the language. (Contributed by Stan Ulbrych in `156360`.)

unicodedata

- The Unicode database has been updated to Unicode 18.0.0.

uuid

- `uuid.UUID`, `uuid.uuid4`, and `uuid.uuid7` now have C implementations, with a pure Python fallback. In microbenchmarks, UUIDv4 and UUIDv7 generation are approximately 15 and 30 times faster, respectively, than the pure Python implementations. UUID construction and string conversion are also faster; see `whatsnew316-uuid-optimizations` for the benchmark results. (Contributed by Yury Selivanov in `139123`.)

xml

- Add support for multiple multi-byte encodings in the `XML parser <xml.parsers.expat>`: "cp932", "cp949", "cp950", "Big5", "EUC-JP", "GB2312", "GBK", "johab", and "Shift_JIS". Add partial support (only BMP characters) for multi-byte encodings "Big5-HKSCS", "EUC_JIS-2004", "EUC_JISX0213", "Shift_JIS-2004", "Shift_JISX0213", "utf-8-sig" and non-standard aliases like "UTF8" (without hyphen). The parser now raises `ValueError` for known unsupported multi-byte encodings such as "ISO-2022-JP" or "raw-unicode-escape" instead of failing later, when encountering non-ASCII data. (Contributed by Serhiy Storchaka in `62259`.)
- `xml.dom.minidom` now conforms closer to the DOM Level 1 specification. It checks names passed to the factory methods, rejects inserting or setting a node created by another document or making a node a descendant of itself, rejects a second element or document type child of a document and a notation child of a document fragment, reports attributes defaulted in the DTD and whether an attribute was given in the start tag, and implements `!EntityReference` nodes and `!Document.createEntityReference`. (Contributed by Jason Orendorff and Serhiy Storchaka in `44871`.)
- `xml.dom.minidom` now validates namespaces in the factory methods `~xml.dom.Document.createElementNS`, `~xml.dom.Document.createAttributeNS` and `~xml.dom.Element.setAttributeNS`. `~xml.dom.NamespaceErr` is now raised for a malformed qualified name, for a prefix with an empty namespace, and for illegal use of the `xml` and `xmlns` prefixes. (Contributed by Serhiy Storchaka in `156665`.)
- `~xml.dom.minidom.Node.toprettyxml` in `xml.dom.minidom` and `~xml.etree.ElementTree.indent` in `xml.etree.ElementTree` no longer add whitespace inside an element which is marked with `xml:space="preserve"` or which contains text. `!toprettyxml` also takes into account the content model declared in the DTD. (Contributed by Serhiy Storchaka in `81623`.)
- Add `!GetSpecifiedAttributeCount` method to the `XML parser <xml.parsers.expat>` objects. It tells how many of the reported attributes were given in the start tag rather than defaulted from the DTD. (Contributed by Jason Orendorff and Serhiy Storchaka in `44871`.)
- The `~xml.dom.minidom.Node.writexml`, `~xml.dom.minidom.Node.toxml` and `~xml.dom.minidom.Node.toprettyxml` methods now work for `!DocumentFragment` nodes in `xml.dom.minidom`. (Contributed by Serhiy Storchaka in `54092`.)
- `~xml.etree.ElementTree.XMLPullParser` and `~xml.etree.ElementTree.iterparse` now support the *target* parameter. The reported object is the value returned by the corresponding method of the target, so a large document can be parsed incrementally without building a

tree for it. (Contributed by Serhiy Storchaka in `63102`.)

- Added constant `~xml.parsers.expat.VERSION_INFO` in the `XML parser <xml.parsers.expat>` module, which provides information about the version of the Expat library that was used for building the module. `~xml.parsers.expat.version_info` is now a named tuple. (Contributed by Serhiy Storchaka in `157473`.)

zipfile

- Add `ZipFile.remove() <zipfile.ZipFile.remove>` to remove a member from an archive's central directory, and `ZipFile.repack() <zipfile.ZipFile.repack>` to reclaim the space used by the local file entries of removed members. (Contributed by Danny Lin in `51067`.)

zlib

- Added constants `~zlib.ZLIB_VERSION_INFO`, `~zlib.zlib_version_info`, and `~zlib.ZLIBNG_VERSION_INFO`, which provide information about the version of the zlib and the zlib-ng libraries in use. Added `~zlib.zlib_version` as an alias of `~zlib.ZLIB_RUNTIME_VERSION`. (Contributed by Serhiy Storchaka in `117404`.)

Optimizations

datetime and time

- `datetime.datetime.strptime`, `datetime.date.strptime`, `datetime.time.strptime`, and `time.strptime` are faster for common numeric formats such as `%Y-%m-%d` and `%Y-%m-%d %H:%M:%S`. (Contributed by Paul Ganssle in `158252`, based on work by Stan Ulbrych.)

re

- Character class escapes (`\d`, `\D`, `\s`, `\S`, `\w` and `\W`) outside a character set, and character sets containing a single such escape (such as `[\d]` or `[^\s]`), are now compiled to a single `CATEGORY` opcode instead of being wrapped in an `IN` block. This speeds up matching of patterns such as `\d+` and reduces the size of the compiled byte code. (Contributed by Serhiy Storchaka in `152033` and Pieter Eendebak in `152056`.)

uuid

- C implementations accelerate UUID generation, construction, and string conversion. Microbenchmarks comparing the C and pure Python implementations measured the following speedups:

Operation	Speedup
`uuid.uuid4` generation	15.01x
`uuid.uuid7` generation	29.64x
`uuid.UUID` from a string	6.76x
`uuid.UUID` from bytes	5.16x
`str` conversion of a UUID	6.66x

  (Contributed by Yury Selivanov in `139123`.)

Removed

annotationlib

- The `!annotationlib.ForwardRef._evaluate` method which has been deprecated since Python 3.14. Use `annotationlib.ForwardRef.evaluate` or `typing.evaluate_forward_ref` instead.

array

- The `'u'` format code (`wchar_t`) which has been deprecated in documentation since Python 3.3 and at runtime since Python 3.13. Use `'w'` format code instead (`Py_UCS4`, always 4 bytes).

asyncio

- The `!asyncio.iscoroutinefunction` which has been deprecated since Python 3.14. Use `inspect.iscoroutinefunction` instead.
- The event loop policy system, including the `!asyncio.AbstractEventLoopPolicy`, `!asyncio.DefaultEventLoopPolicy`, `!asyncio.WindowsSelectorEventLoopPolicy` and `!asyncio.WindowsProactorEventLoopPolicy` classes and the `!asyncio.get_event_loop_policy` and `!asyncio.set_event_loop_policy` functions, which have been deprecated since Python 3.14. Use `asyncio.run` or `asyncio.Runner` with a custom *loop_factory* instead.

functools

- Remove support for calling the Python implementation of `functools.reduce` with *function* or *sequence* as keyword arguments, deprecated since Python 3.14. These parameters are now positional-only. (Contributed by Nikita Sobolev in `149537`.)

logging

- Remove support for custom logging handlers with the *strm* argument, deprecated since Python 3.14. Define handlers with the *stream* argument instead. (Contributed by Nikita Sobolev in `149598`.)

mimetypes

- Remove support for undotted extensions in `mimetypes.MimeTypes.add_type`, deprecated since Python 3.14. Valid extensions must now start with a `'.'` or be empty; undotted extensions raise a `ValueError`. (Contributed by Nikita Sobolev in `149720`.)

shutil

- The `!ExecError` exception which has been deprecated since Python 3.14. It has not been used by any function in `!shutil` since Python 3.4. (Contributed by Stan Ulbrych in `149567`.)

symtable

- The `!symtable.Class.get_methods` method which has been deprecated since Python 3.14.

sys

- The `!_enablelegacywindowsfsencoding` function which has been deprecated since Python 3.13. Use the `PYTHONLEGACYWINDOWSFSENCODING` environment variable instead. (Contributed by Stan Ulbrych in `149595`.)

sysconfig

- The `!sysconfig.expand_makefile_vars` function which has been deprecated since Python 3.14. Use the `vars` argument of `sysconfig.get_paths` instead. (Contributed by Stan Ulbrych in `149499`.)

tarfile

- The undocumented and unused `!tarfile.TarFile.tarfile` attribute has been deprecated since Python 3.13.

xml.etree.ElementTree

- The undocumented `!xml.etree.ElementTree.ElementTree.write_c14n` method and the support of `method="c14n"` in `~xml.etree.ElementTree.ElementTree.write` and `~xml.etree.ElementTree.tostring`. They never worked and always raised `ValueError`. Use `xml.etree.ElementTree.canonicalize` instead. (Contributed by Daniel Hillier in `118318`.)

Deprecated

New deprecations

- `abc`:
  - `abc.abstractclassmethod`, `abc.abstractstaticmethod`, and `abc.abstractproperty`, soft-deprecated since Python 3.3, now raise a `DeprecationWarning`. These classes will be removed in Python 3.21; use `abc.abstractmethod` with `classmethod`, `staticmethod`, and `property` respectively instead.
- `ast`:
  - Classes `slice`, `Index`, `ExtSlice`, `Suite`, `Param`, `AugLoad` and `AugStore`, deprecated since Python 3.9, are no longer imported by `from ast import *` and issue a deprecation warning on use. The classes are slated for removal in Python 3.21. These types are not generated by the parser or accepted by the code generator.
  - The `dims` property of `ast.Tuple` objects, deprecated since Python 3.9, now issues a deprecation warning on use. This property is slated for removal in 3.21. Use `ast.Tuple.elts` instead.
- `__author__`, `__credits__`, `__copyright__` and `__date__`:
  - The `__author__`, `__credits__`, `__copyright__` and `__date__` attributes have been deprecated in these standard library modules and will be removed in Python 3.20. Attribution is preserved in the source code and the version control history.
    - `bz2`
    - `concurrent.futures`
    - `inspect`
    - `io`
    - `json`
    - `logging`
    - `optparse`

    - `platform`
    - `pydoc`
    - `tarfile`
    - `tkinter.ttk`
    - `tokenize`
    - `uuid`

    (Contributed by Stan Ulbrych in `159240`.)
- `functools`:
    - Using a return annotation to infer the dispatch type for `functools.singledispatch` or `functools.singledispatchmethod` now emits a `DeprecationWarning` and will raise `TypeError` in Python 3.21. Annotate the dispatch parameter or pass the dispatch type explicitly instead. (Contributed by Bartosz Sławecki in `143465`.)
- `struct`:
    - The `'F'` and `'D'` type codes, soft-deprecated since Python 3.15, are now deprecated. These codes will be removed in Python 3.21. Use the two-letter forms `'Zf'` and `'Zd'` instead. (Contributed by Sergey B Kirpichev in `121249`.)
- `tempfile`:
    - The private `tempfile._TemporaryFileWrapper` name is deprecated and is slated for removal in Python 3.21. Use the new public `tempfile.TemporaryFileWrapper` instead, which is the return type of `tempfile.NamedTemporaryFile`.
- `tkinter`:
    - `tkinter.filedialog.askopenfiles` is deprecated and slated for removal in Python 3.19. Opening several files at once is error-prone, and the returned list cannot be used in a `with` statement. Iterate over the names returned by `~tkinter.filedialog.askopenfilenames` and open them one by one instead. (Contributed by Serhiy Storchaka in `152638`.)

Pending removal in Python 3.17

- `datetime`:
    - `~datetime.datetime.strptime` calls using a format string containing `%e` (day of month) without a year. This has been deprecated since Python 3.15. (Contributed by Stan Ulbrych in `70647`.)
- `collections.abc`:
    - `collections.abc.ByteString` is scheduled for removal in Python 3.17.
      Use `isinstance(obj, collections.abc.Buffer)` to test if `obj` implements the `buffer protocol <bufferobjects>` at runtime. For use in type annotations, either use `~collections.abc.Buffer` or a union that explicitly specifies the types your code supports (e.g., `bytes | bytearray | memoryview`).
      `!ByteString` was originally intended to be an abstract class that would serve as a supertype of both `bytes` and `bytearray`. However, since the ABC never had any methods, knowing that an object was an instance of `!ByteString` never actually told you anything useful about the object. Other common buffer types such as `memoryview` were also never understood as subtypes of `!ByteString` (either at

runtime or by static type checkers).
  See `PEP 688 <688#current-options>` for more details. (Contributed by Shantanu Jain in `91896`.)
- `encodings`:
  - Passing non-ascii *encoding* names to `encodings.normalize_encoding` is deprecated and scheduled for removal in Python 3.17. (Contributed by Stan Ulbrych in `136702`.)
- `profile`:
  - The `!profile` module is deprecated and will be removed in Python 3.17. Use `profiling.tracing` instead, which provides a compatible API. See `799` for details. (Contributed by Pablo Galindo and László Kiss Kollár in `138122`.)
- `webbrowser`:
  - `!webbrowser.MacOSXOSAScript` is deprecated in favour of `!webbrowser.MacOS`. (`137586`)
- `typing`:
  - Before Python 3.14, old-style unions were implemented using the private class `typing._UnionGenericAlias`. This class is no longer needed for the implementation, but it has been retained for backward compatibility, with removal scheduled for Python 3.17. Users should use documented introspection helpers like `typing.get_origin` and `typing.get_args` instead of relying on private implementation details.
  - `typing.ByteString`, deprecated since Python 3.9, is scheduled for removal in Python 3.17.
    Use `isinstance(obj, collections.abc.Buffer)` to test if `obj` implements the `buffer protocol <bufferobjects>` at runtime. For use in type annotations, either use `~collections.abc.Buffer` or a union that explicitly specifies the types your code supports (e.g., `bytes | bytearray | memoryview`).
    `!ByteString` was originally intended to be an abstract class that would serve as a supertype of both `bytes` and `bytearray`. However, since the ABC never had any methods, knowing that an object was an instance of `!ByteString` never actually told you anything useful about the object. Other common buffer types such as `memoryview` were also never understood as subtypes of `!ByteString` (either at runtime or by static type checkers).
    See `PEP 688 <688#current-options>` for more details. (Contributed by Shantanu Jain in `91896`.)
- `tkinter`:
  - The `!tkinter.Variable` methods `!trace_variable`, `!trace` (an alias of `!trace_variable`), `!trace_vdelete` and `!trace_vinfo`, deprecated since Python 3.14, are scheduled for removal in Python 3.17. Use `!trace_add`, `!trace_remove` and `!trace_info` instead. (Contributed by Serhiy Storchaka in `120220`.)

Pending removal in Python 3.18

- No longer accept a boolean value when a file descriptor is expected. (Contributed by Serhiy Storchaka in `82626`.)
- `decimal`:

- The non-standard and undocumented `~decimal.Decimal` format specifier `'N'`, which is only supported in the `!decimal` module's C implementation, has been deprecated since Python 3.13. (Contributed by Serhiy Storchaka in `89902`.)

- Deprecations defined by `829`:
  - `import` lines in `{name}.pth` files are silently ignored.

  (Contributed by Barry Warsaw in `148641`.)

Pending removal in Python 3.19

- `builtins`:
  - Passing a complex number as the *real* or *imag* argument in the `complex` constructor is now deprecated; it should only be passed as a single positional argument. (Contributed by Serhiy Storchaka in `109218`.)
- `ctypes`:
  - Implicitly switching to the MSVC-compatible struct layout by setting `~ctypes.Structure._pack_` but not `~ctypes.Structure._layout_` on non-Windows platforms.
- `hashlib`:
  - In hash function constructors such as `~hashlib.new` or the direct hash-named constructors such as `~hashlib.md5` and `~hashlib.sha256`, their optional initial data parameter could also be passed a keyword argument named `data=` or `string=` in various `!hashlib` implementations.
    Support for the `string` keyword argument name is now deprecated and slated for removal in Python 3.19.
    Before Python 3.13, the `string` keyword parameter was not correctly supported depending on the backend implementation of hash functions. Prefer passing the initial data as a positional argument for maximum backwards compatibility.
- `http.cookies`:
  - `http.cookies.Morsel.js_output` is deprecated and will be removed in Python 3.19.
  - `http.cookies.BaseCookie.js_output` is deprecated and will be removed in Python 3.19.
- `imaplib`:
  - Altering `IMAP4.file <imaplib.IMAP4.file>` is now deprecated and slated for removal in Python 3.19. This property is now unused and changing its value does not automatically close the current file.
    Before Python 3.14, this property was used to implement the corresponding `read()` and `readline()` methods for `~imaplib.IMAP4` but this is no longer the case since then.
- `tkinter`:
  - `tkinter.filedialog.askopenfiles` has been deprecated since Python 3.16. Iterate over the names returned by `~tkinter.filedialog.askopenfilenames` and open them one by one instead. (Contributed by Serhiy Storchaka in `152638`.)

Pending removal in Python 3.20

- Calling the `__new__()` method of `struct.Struct` without the *format* argument is deprecated and will be removed in Python 3.20. Calling `~object.__init__` method on initialized `~struct.Struct` objects is deprecated and will be removed in Python 3.20. (Contributed by Sergey B Kirpichev and Serhiy Storchaka in `143715`.)
- The `__version__`, `version` and `VERSION` attributes have been deprecated in these standard library modules and will be removed in Python 3.20. Use `sys.version_info` instead.
  - `argparse`
  - `csv`
  - `ctypes`
  - `!ctypes.macholib`
  - `decimal` (use `decimal.SPEC_VERSION` instead)
  - `http.server`
  - `imaplib`
  - `ipaddress`
  - `json`
  - `logging` (`__date__` also deprecated)
  - `optparse`
  - `pickle`
  - `platform`
  - `re`
  - `socketserver`
  - `tabnanny`
  - `tarfile`
  - `tkinter.font`
  - `tkinter.ttk`
  - `wsgiref.simple_server`
  - `xml.etree.ElementTree`
  - `!xml.sax.expatreader`
  - `xml.sax.handler`
  - `zlib`

  (Contributed by Hugo van Kemenade and Stan Ulbrych in `76007`.)
- The `__author__`, `__credits__`, `__copyright__` and `__date__` attributes have been deprecated in these standard library modules and will be removed in Python 3.20. Attribution is preserved in the source code and the version control history.
  - `bz2`
  - `concurrent.futures`
  - `inspect`
  - `io`
  - `json`
  - `logging`
  - `optparse`
  - `platform`
  - `pydoc`
  - `tarfile`
  - `tkinter.ttk`

    - `tokenize`
    - `uuid`

  (Contributed by Stan Ulbrych in `159240`.)
- Deprecations defined by `829`:
    - Warnings are produced for `import` lines found in `{name}.pth` files.
    - `{name}.pth` files are no longer decoded in the locale encoding by default. They **MUST** be encoded in `utf-8-sig`.

  (Contributed by Barry Warsaw in `148641`.)
- `ast`:
    - Creating instances of abstract AST nodes (such as `ast.AST` or `!ast.expr`) is deprecated and will raise an error in Python 3.20.
- `typing`:
    - It is deprecated to call `isinstance` and `issubclass` checks on protocol classes that were not explicitly decorated with `!runtime_checkable` but that inherit from a runtime-checkable protocol class. This will raise a `TypeError` in Python 3.20. (Contributed by Bartosz Sławecki in `132604`.)

Pending removal in Python 3.21

- `abc`

  > - Soft-deprecated since Python 3.3 `abc.abstractclassmethod`, `abc.abstractstaticmethod`, and `abc.abstractproperty` now raise a `DeprecationWarning`. These classes will be removed in Python 3.21, instead use `abc.abstractmethod` with `classmethod`, `staticmethod`, and `property` respectively.

- `ast`:
    - Classes `slice`, `Index`, `ExtSlice`, `Suite`, `Param`, `AugLoad` and `AugStore`, will be removed in Python 3.21. These types are not generated by the parser or accepted by the code generator.
    - The `dims` property of `ast.Tuple` will be removed in Python 3.21. Use the `ast.Tuple.elts` property instead.
- `functools`:
    - Using a return annotation to infer the dispatch type for `functools.singledispatch` or `functools.singledispatchmethod` is deprecated since Python 3.16 and will raise `TypeError` in Python 3.21. Annotate the dispatch parameter or pass the dispatch type explicitly instead. (Contributed by Bartosz Sławecki in `143465`.)
- `struct`:
    - Soft-deprecated since Python 3.15, using `'F'` and `'D'` type codes are now deprecated. These codes will be removed in Python 3.21. Use instead two-letter forms `'Zf'` and `'Zd'`.
- `tempfile`:

- `tempfile._TemporaryFileWrapper` will be removed in Python 3.21. Use the public `tempfile.TemporaryFileWrapper` instead.

Pending removal in future versions

The following APIs will be removed in the future, although there is currently no date scheduled for their removal.

- `argparse`:
  - Nesting argument groups and nesting mutually exclusive groups are deprecated.
  - Passing the undocumented keyword argument *prefix_chars* to `~argparse.ArgumentParser.add_argument_group` is now deprecated.
  - The `argparse.FileType` type converter is deprecated.
- `builtins`:
  - Generators: `throw(type, exc, tb)` and `athrow(type, exc, tb)` signature is deprecated: use `throw(exc)` and `athrow(exc)` instead, the single argument signature.
  - Currently Python accepts numeric literals immediately followed by keywords, for example `0in x`, `1or x`, `0if 1else 2`. It allows confusing and ambiguous expressions like `[0x1for 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. (`87999`)
  - Support for `__index__()` and `__int__()` method returning non-int type: these methods will be required to return an instance of a strict subclass of `int`.
  - Support for `__float__()` method returning a strict subclass of `float`: these methods will be required to return an instance of `float`.
  - Support for `__complex__()` method returning a strict subclass of `complex`: these methods will be required to return an instance of `complex`.
- `calendar`: `calendar.January` and `calendar.February` constants are deprecated and replaced by `calendar.JANUARY` and `calendar.FEBRUARY`. (Contributed by Prince Roshan in `103636`.)
- `codecs`: use `open` instead of `codecs.open`. (`133038`)
- `!codeobject.co_lnotab`: use the `codeobject.co_lines` method instead.
- `datetime`:
  - `~datetime.datetime.utcnow`: use `datetime.datetime.now(tz=datetime.UTC)`.
  - `~datetime.datetime.utcfromtimestamp`: use `datetime.datetime.fromtimestamp(timestamp, tz=datetime.UTC)`.
- `gettext`: Plural value must be an integer.
- `importlib`:
  - `~importlib.util.cache_from_source` *debug_override* parameter is deprecated: use the *optimization* parameter instead.
- `importlib.metadata`:
  - `EntryPoints` tuple interface.
  - Implicit `None` on return values.

- `logging`: the `warn()` method has been deprecated since Python 3.3, use `~logging.warning` instead.
- `mailbox`: Use of StringIO input and text mode is deprecated, use BytesIO and binary mode instead.
- `os`: Calling `os.register_at_fork` in a multi-threaded process.
- `os.path`: `os.path.commonprefix` is deprecated, use `os.path.commonpath` for path prefixes. The `os.path.commonprefix` function is being deprecated due to having a misleading name and module. The function is not safe to use for path prefixes despite being included in a module about path manipulation, meaning it is easy to accidentally introduce path traversal vulnerabilities into Python programs by using this function.
- `!pydoc.ErrorDuringImport`: A tuple value for *exc_info* parameter is deprecated, use an exception instance.
- `re`: 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 `91760`.)
- `shutil`: `~shutil.rmtree`'s *onerror* parameter is deprecated in Python 3.12; use the *onexc* parameter instead.
- `ssl` options and protocols:
  - `ssl.SSLContext` without protocol argument is deprecated.
  - `ssl.SSLContext`: `~ssl.SSLContext.set_npn_protocols` and `!selected_npn_protocol` are deprecated: use ALPN instead.
  - `ssl.OP_NO_SSL*` options
  - `ssl.OP_NO_TLS*` options
  - `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` methods:
  - `!threading.Condition.notifyAll`: use `~threading.Condition.notify_all`.
  - `!threading.Event.isSet`: use `~threading.Event.is_set`.
  - `!threading.Thread.isDaemon`, `threading.Thread.setDaemon`: use `threading.Thread.daemon` attribute.
  - `!threading.Thread.getName`, `threading.Thread.setName`: use `threading.Thread.name` attribute.
  - `!threading.currentThread`: use `threading.current_thread`.
  - `!threading.activeCount`: use `threading.active_count`.
- `typing.Text` (`92332`).
- The internal class `typing._UnionGenericAlias` is no longer used to implement `typing.Union`. To preserve compatibility with users using this private class, a compatibility shim will be

provided until at least Python 3.17. (Contributed by Jelle Zijlstra in `105499`.)

- `unittest.IsolatedAsyncioTestCase`: it is deprecated to return a value that is not `None` from a test case.
- `urllib.parse` deprecated functions: `~urllib.parse.urlparse` instead
  - `splitattr()`
  - `splithost()`
  - `splitnport()`
  - `splitpasswd()`
  - `splitport()`
  - `splitquery()`
  - `splittag()`
  - `splittype()`
  - `splituser()`
  - `splitvalue()`
  - `to_bytes()`
- `wsgiref`: `SimpleHandler.stdout.write()` should not do partial writes.
- `xml.etree.ElementTree`: Testing the truth value of an `~xml.etree.ElementTree.Element` is deprecated. In a future release it will always return `True`. Prefer explicit `len(elem)` or `elem is not None` tests instead.
- `sys._clear_type_cache` is deprecated: use `sys._clear_internal_caches` instead.

Porting to Python 3.16

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

- In `tkinter`, the *name* parameter of the `~tkinter.Misc.wait_variable`, `~tkinter.Misc.setvar` and `~tkinter.Misc.getvar` methods and the *value* parameter of `!setvar` are now required. Calling these methods without them, which formerly defaulted to `'PY_VAR'` and `'1'`, now raises `TypeError`. (Contributed by Serhiy Storchaka in `152587`.)
- `uuid.UUID` now consistently raises `TypeError` when the *bytes* or *bytes_le* argument is not a `bytes` object. Previously, some invalid inputs raised `AssertionError` or `ValueError`. (Contributed by Yury Selivanov in `139123`.)
- `xml.dom.minidom` now raises `~xml.dom.InvalidCharacterErr` for a name which is not a valid XML name, `~xml.dom.WrongDocumentErr` for inserting or setting a node created by another document, `~xml.dom.InuseAttributeErr` for setting an attribute node which belongs to another element, and `~xml.dom.HierarchyRequestErr` for inserting a node into itself or its descendant. It also rejects a second element or document type child of a document and a notation child of a document fragment. Such operations formerly succeeded and produced an invalid document or an endless loop. On the other hand, `!Element.removeAttribute` and `!Element.removeAttributeNS` no longer raise `~xml.dom.NotFoundErr` if there is no matching attribute. Attributes defaulted in the DTD are no longer omitted when parsing. (Contributed by Jason Orendorff and Serhiy Storchaka in `44871`.)

- `xml.dom.minidom` now raises `~xml.dom.NamespaceErr` for a malformed qualified name, for a prefix with an empty namespace, and for illegal use of the `xml` and `xmlns` prefixes. Such operations formerly succeeded and produced an invalid document. (Contributed by Serhiy Storchaka in `156665`.)
- `~xml.dom.minidom.Node.toprettyxml` in `xml.dom.minidom` and `~xml.etree.ElementTree.indent` in `xml.etree.ElementTree` no longer add whitespace inside an element which is marked with `xml:space="preserve"` or which contains text, because this changed the content of the element. `!toprettyxml` also takes into account the content model declared in the DTD. (Contributed by Serhiy Storchaka in `81623`.)
- On Windows, seeking a pipe now fails instead of silently appearing to succeed: `os.lseek` and `~io.IOBase.seek` raise `OSError`, and `~io.IOBase.seekable` returns `False`. As a consequence, opening a pipe in a read-write binary mode (`'r+b'` or `'w+b'`) now raises `io.UnsupportedOperation` unless buffering is disabled. (Contributed by An Long in `86768`.)

Build changes

- Remove the bundled copy of the libmpdec decimal library from the CPython source tree to simplify maintenance and updates. The `decimal` module will now unconditionally use the system's libmpdec decimal library. Also remove the now unused `!--with-system-libmpdec` `configure` flag. This change has no impact on binary releases of Python, which have been built against a separate copy of libmpdec for the past several releases.
  (Contributed by Sergey B Kirpichev in `115119`.)
- Add a `--with-build-details-suffix` configure flag to allow Linux distributions that co-install multiple versions of Python in the same tree to avoid `build-details.json` clashes.
  (Contributed by Stefano Rivera in `131372`.)
- Add the `--with-curses` `configure` option to select the curses backend for the `curses` and `curses.panel` modules. In addition to `ncursesw` and `ncurses`, it can now build against the system's native `curses` library (for example on NetBSD or Solaris), with wide-character support when the library provides it. `--without-curses` excludes the modules from the build.
  (Contributed by Serhiy Storchaka in `136687`.)
- Add the `--with-zlib` and `--with-bzip2` `configure` options to select the zlib and bzip2 implementations used to build the `zlib` and `bz2` modules. In addition to the default system libraries, they can build against zlib-rs and libbzip2-rs, and `--with-zlib=zlib-ng` verifies that the detected zlib is zlib-ng. `--without-zlib` and `--without-bzip2` exclude the modules from the build.
  (Contributed by Stan Ulbrych in `139314`.)
- Add the `--with-hwaddress-sanitizer` `configure` option to build with HWAddressSanitizer.
  (Contributed by Stan Ulbrych, Florian Mayer and AnnaAr321 in `156049`.)

C API changes

New features

- Add `Py_HashBuffer` to the limited C API. (Contributed by Victor Stinner in `148233`.)
- Add `Py_SETREF` and `Py_XSETREF` macros to the limited C API. (Contributed by Nathan Goldbaum and Victor Stinner in `158842`.)

Porting to Python 3.16

- `PyType_ClearCache` is now a no-op as the type cache is now implemented per-type. It still returns the current version tag.
- Functions reading marshalled data from a `FILE*`, such as `PyMarshal_ReadObjectFromFile`, now raise `OSError` for I/O errors and `KeyboardInterrupt` for interrupted reading, instead of `EOFError`. (Contributed by Serhiy Storchaka in `155907`.)
- `PyMarshal_WriteLongToFile` and `PyMarshal_WriteObjectToFile` now set the error indicator for I/O errors and interrupted writing, instead of ignoring them. `PyMarshal_WriteObjectToFile` now also sets the error indicator if the value cannot be marshalled. (Contributed by Serhiy Storchaka in `155907`.)
- `_Py_c_abs` no longer sets `errno` to zero on success, but rather leaves it unchanged. (Contributed by Sergey B Kirpichev in `155526`.)

Deprecated C APIs

- `PyGen_New`, `PyGen_NewWithQualName`, `PyCoro_New`, and `PyAsyncGen_New` are deprecated. They are scheduled for removal in 3.18.
- Deprecate `PyInitConfig_GetExitCode`, `PyStatus_Exit`, and `PyStatus_IsExit` functions, and `PyStatus.exitcode` member. `Py_InitializeFromInitConfig` and `Py_InitializeFromConfig` can no longer return an exit code. Instead, if a command line option wants to exit Python, the option is now processed in `Py_RunMain`. (Contributed by Victor Stinner in `158080`.)
- `PyModule_GetFilename` is no longer deprecated, but using `PyModule_GetFilenameObject` instead is still recommended. (Contributed by Victor Stinner in `154757`.)
- Soft deprecate functions modifying Unicode strings: `PyUnicode_New`, `PyUnicode_CopyCharacters`, `PyUnicode_Fill`, `PyUnicode_Resize`, `PyUnicode_WRITE` and `PyUnicode_WriteChar`. Use the safer `PyUnicodeWriter` API instead. (Contributed by Victor Stinner in `157710`.)

Pending removal in Python 3.18

- The following private functions are deprecated and planned for removal in Python 3.18:

- `!_PyBytes_Join`: use `PyBytes_Join`.
- `!_PyDict_GetItemStringWithError`: use `PyDict_GetItemStringRef`.
- `!_PyDict_Pop()`: use `PyDict_Pop`.
- `!_PyLong_Sign()`: use `PyLong_GetSign`.
- `!_PyLong_FromDigits` and `!_PyLong_New`: use `PyLongWriter_Create`.
- `!_PyThreadState_UncheckedGet`: use `PyThreadState_GetUnchecked`.
- `!_PyUnicode_AsString`: use `PyUnicode_AsUTF8`.
- `!_PyUnicodeWriter_Init`: replace `_PyUnicodeWriter_Init(&writer)` with `writer = PyUnicodeWriter_Create(0) <PyUnicodeWriter_Create>`.
- `!_PyUnicodeWriter_Finish`: replace `_PyUnicodeWriter_Finish(&writer)` with `PyUnicodeWriter_Finish(writer) <PyUnicodeWriter_Finish>`.
- `!_PyUnicodeWriter_Dealloc`: replace `_PyUnicodeWriter_Dealloc(&writer)` with `PyUnicodeWriter_Discard(writer) <PyUnicodeWriter_Discard>`.
- `!_PyUnicodeWriter_WriteChar`: replace `_PyUnicodeWriter_WriteChar(&writer, ch)` with `PyUnicodeWriter_WriteChar(writer, ch) <PyUnicodeWriter_WriteChar>`.
- `!_PyUnicodeWriter_WriteStr`: replace `_PyUnicodeWriter_WriteStr(&writer, str)` with `PyUnicodeWriter_WriteStr(writer, str) <PyUnicodeWriter_WriteStr>`.
- `!_PyUnicodeWriter_WriteSubstring`: replace `_PyUnicodeWriter_WriteSubstring(&writer, str, start, end)` with `PyUnicodeWriter_WriteSubstring(writer, str, start, end) <PyUnicodeWriter_WriteSubstring>`.
- `!_PyUnicodeWriter_WriteASCIIString`: replace `_PyUnicodeWriter_WriteASCIIString(&writer, str, size)` with `PyUnicodeWriter_WriteASCII(writer, str, size) <PyUnicodeWriter_WriteASCII>`.
- `!_PyUnicodeWriter_WriteLatin1String`: replace `_PyUnicodeWriter_WriteLatin1String(&writer, str, size)` by decoding the string with `PyUnicode_DecodeLatin1`, writing the result with `PyUnicodeWriter_WriteStr`, and releasing the result with `Py_DECREF`.
- `!_PyUnicodeWriter_Prepare`: (no replacement).
- `!_PyUnicodeWriter_PrepareKind`: (no replacement).
- `!_Py_HashPointer`: use `Py_HashPointer`.
- `!_Py_fopen_obj`: use `Py_fopen`.
- `PyGen_New`: (no replacement).
- `PyGen_NewWithQualName`: (no replacement).
- `PyCoro_New`: (no replacement).
- `PyAsyncGen_New`: (no replacement).

The pythoncapi-compat project can be used to get these new public functions on Python 3.13 and older. (Contributed by Victor Stinner in `128863`.)

Pending removal in Python 3.19

- `456` embedders support for the string hashing scheme definition.

Pending removal in Python 3.20

- `!_PyObject_CallMethodId`, `!_PyObject_GetAttrId` and `!_PyUnicode_FromId` are deprecated since 3.15 and will be removed in 3.20. Instead, use `PyUnicode_InternFromString()` and cache the result in the module state, then call `PyObject_CallMethod` or `PyObject_GetAttr`. (Contributed by Victor Stinner in `141049`.)
- The `cval` field in `PyComplexObject` (`128813`). Use `PyComplex_AsCComplex` and `PyComplex_FromCComplex` to convert a Python complex number to/from the C `Py_complex` representation.
- Macros `!Py_MATH_PIl` and `!Py_MATH_El`.

Pending removal in future versions

The following APIs are deprecated and will be removed, although there is currently no date scheduled for their removal.

- `Py_TPFLAGS_HAVE_FINALIZE`: Unneeded since Python 3.8.
- `PyErr_Fetch`: Use `PyErr_GetRaisedException` instead.
- `PyErr_NormalizeException`: Use `PyErr_GetRaisedException` instead.
- `PyErr_Restore`: Use `PyErr_SetRaisedException` instead.
- `PyOS_AfterFork`: Use `PyOS_AfterFork_Child` instead.
- `PySlice_GetIndicesEx`: Use `PySlice_Unpack` and `PySlice_AdjustIndices` instead.
- `PyUnicode_READY`: Unneeded since Python 3.12
- `!PyErr_Display`: Use `PyErr_DisplayException` instead.
- `!_PyErr_ChainExceptions`: Use `!_PyErr_ChainExceptions1` instead.
- `!PyBytesObject.ob_shash` member: call `PyObject_Hash` instead.
- Thread Local Storage (TLS) API:
  - `PyThread_create_key`: Use `PyThread_tss_alloc` instead.
  - `PyThread_delete_key`: Use `PyThread_tss_free` instead.
  - `PyThread_set_key_value`: Use `PyThread_tss_set` instead.
  - `PyThread_get_key_value`: Use `PyThread_tss_get` instead.
  - `PyThread_delete_key_value`: Use `PyThread_tss_delete` instead.
  - `PyThread_ReInitTLS`: Unneeded since Python 3.7.

Removed C APIs

- Remove 21 global configuration variables, deprecated in Python 3.12. To initialize Python, the `PyInitConfig C API <pyinitconfig_api>` can be used instead. To read these variables at runtime, `PyConfig_Get` can be used instead. Removed variables with their `PyInitConfig C API <pyinitconfig_api>` and `PyConfig_Get` replacement:
  - `!Py_DebugFlag`: Use `"parser_debug"` instead.
  - `!Py_VerboseFlag`: Use `"verbose"` instead.
  - `!Py_QuietFlag`: Use `"quiet"` instead.
  - `!Py_InteractiveFlag`: Use `"interactive"` instead.
  - `!Py_InspectFlag`: Use `"inspect"` instead.
  - `!Py_OptimizeFlag`: Use `"optimization_level"` instead.
  - `!Py_NoSiteFlag`: Use `"site_import"` instead (opposite value).

- `!Py_BytesWarningFlag`: Use `"bytes_warning"` instead.
- `!Py_FrozenFlag`: Use `"pathconfig_warnings"` instead (opposite value).
- `!Py_IgnoreEnvironmentFlag`: Use `"use_environment"` instead (opposite value).
- `!Py_DontWriteBytecodeFlag`: Use `"write_bytecode"` instead (opposite value).
- `!Py_NoUserSiteDirectory`: Use `"user_site_directory"` instead (opposite value).
- `!Py_UnbufferedStdioFlag`: Use `"buffered_stdio"` instead (opposite value).
- `!Py_HashRandomizationFlag`: Use `"hash_seed"` instead.
- `!Py_IsolatedFlag`: Use `"isolated"` instead.
- `!Py_LegacyWindowsFSEncodingFlag`: Use `"legacy_windows_fs_encoding"` instead.
- `!Py_LegacyWindowsStdioFlag`: Use `"legacy_windows_stdio"` instead.
- `!Py_FileSystemDefaultEncoding`, `!Py_HasFileSystemDefaultEncoding`: Use `"filesystem_encoding"` instead.
- `!Py_FileSystemDefaultEncodeErrors`: Use `"filesystem_errors"` instead.
- `!Py_UTF8Mode`: Use `"utf8_mode"` instead.

(Contributed by Victor Stinner in `158001`.)

- Remove `!PyEval_InitThreads` function which did nothing since Python 3.7 and was deprecated since Python 3.9. (Contributed by Victor Stinner in `154757`.)
- Remove `!Py_SetProgramName` and `!Py_SetPythonHome` functions, deprecated since Python 3.11. Instead, use `PyInitConfig C API <pyinitconfig_api>` to set respectively `"program_name"` and `"home"`. (Contributed by Victor Stinner in `158001`.)
- Remove `!PySys_SetArgv` and `!PySys_SetArgvEx` functions, deprecated since Python 3.11. Instead, set directly `sys.argv`, and maybe also `sys.path`. (Contributed by Victor Stinner in `158001`.)
- The `!PyUnstable_ExecutableKinds` array, as well as the macros `!PyUnstable_EXECUTABLE_KIND_SKIP`, `!PyUnstable_EXECUTABLE_KIND_PY_FUNCTION`, `!PyUnstable_EXECUTABLE_KIND_BUILTIN_FUNCTION`, `!PyUnstable_EXECUTABLE_KIND_METHOD_DESCRIPTOR` and `!PyUnstable_EXECUTABLE_KINDS`, were removed.
  (Contributed by Peters Bierma and Viktorin in `152105`.)

□□□ #2
□ □□□ □□□ 2026-10-31 17:29:40 UTC
□ □□□ □□□ 2026-10-31 17:29:44 UTC