Deprecations

- Pending removal in Python 3.14
- Pending removal in Python 3.15
- Pending removal in Python 3.16
- Pending removal in Python 3.18
- Pending removal in Python 3.19
- Pending removal in Python 3.20
- Pending removal in future versions
- Deprecations
- Pending removal in Python 3.13
- Pending removal in Python 3.14
- Pending removal in Python 3.15
- Pending removal in Python 3.16
- 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
- Soft deprecations

Pending removal in Python 3.14

- The `ma_version_tag` field in `PyDictObject` for extension modules (`699`; `101193`).
- Creating `immutable types <Py_TPFLAGS_IMMUTABLETYPE>` with mutable bases (`95388`).

Pending removal in Python 3.15

- The `!PyImport_ImportModuleNoBlock`: Use `PyImport_ImportModule` instead.
- `!PyWeakref_GetObject` and `!PyWeakref_GET_OBJECT`: Use `PyWeakref_GetRef` instead. The pythoncapi-compat project can be used to get `PyWeakref_GetRef` on Python 3.12 and older.
- `!PyUnicode_AsDecodedObject`: Use `PyCodec_Decode` instead.
- `!PyUnicode_AsDecodedUnicode`: Use `PyCodec_Decode` instead; Note that some codecs (for example, "base64") may return a type other than `str`, such as `bytes`.
- `!PyUnicode_AsEncodedObject`: Use `PyCodec_Encode` instead.
- `!PyUnicode_AsEncodedUnicode`: Use `PyCodec_Encode` instead; Note that some codecs (for example, "base64") may return a type other than `bytes`, such as `str`.
- Python initialization functions, deprecated in Python 3.13:
  - `!Py_GetPath`: Use `PyConfig_Get("module_search_paths") <PyConfig_Get>` (`sys.path`) instead.
  - `!Py_GetPrefix`: Use `PyConfig_Get("base_prefix") <PyConfig_Get>` (`sys.base_prefix`) instead. Use `PyConfig_Get("prefix") <PyConfig_Get>` (`sys.prefix`) if `virtual environments <venv-def>` need to be handled.
  - `!Py_GetExecPrefix`: Use `PyConfig_Get("base_exec_prefix") <PyConfig_Get>` (`sys.base_exec_prefix`) instead. Use `PyConfig_Get("exec_prefix") <PyConfig_Get>` (`sys.exec_prefix`) if `virtual environments <venv-def>` need to be handled.
  - `!Py_GetProgramFullPath`: Use `PyConfig_Get("executable") <PyConfig_Get>` (`sys.executable`) instead.
  - `!Py_GetProgramName`: Use `PyConfig_Get("executable") <PyConfig_Get>` (`sys.executable`) instead.
  - `!Py_GetPythonHome`: Use `PyConfig_Get("home") <PyConfig_Get>` or the `PYTHONHOME` environment variable instead.

  The pythoncapi-compat project can be used to get `PyConfig_Get` on Python 3.13 and older.

Pending removal in Python 3.16

- Functions to configure Python's initialization, deprecated in Python 3.11:
  - `!PySys_SetArgvEx()`: Set `PyConfig.argv` instead.
  - `!PySys_SetArgv()`: Set `PyConfig.argv` instead.
  - `!Py_SetProgramName()`: Set `PyConfig.program_name` instead.
  - `!Py_SetPythonHome()`: Set `PyConfig.home` instead.

  The `Py_InitializeFromConfig` API should be used with `PyConfig` instead.
- Global configuration variables:
  - `!Py_DebugFlag`: Use `PyConfig.parser_debug` or `PyConfig_Get("parser_debug") <PyConfig_Get>` instead.
  - `!Py_VerboseFlag`: Use `PyConfig.verbose` or `PyConfig_Get("verbose") <PyConfig_Get>` instead.
  - `!Py_QuietFlag`: Use `PyConfig.quiet` or `PyConfig_Get("quiet") <PyConfig_Get>` instead.
  - `!Py_InteractiveFlag`: Use `PyConfig.interactive` or `PyConfig_Get("interactive") <PyConfig_Get>` instead.
  - `!Py_InspectFlag`: Use `PyConfig.inspect` or `PyConfig_Get("inspect") <PyConfig_Get>` instead.
  - `!Py_OptimizeFlag`: Use `PyConfig.optimization_level` or `PyConfig_Get("optimization_level") <PyConfig_Get>` instead.
  - `!Py_NoSiteFlag`: Use `PyConfig.site_import` or `PyConfig_Get("site_import") <PyConfig_Get>` instead.
  - `!Py_BytesWarningFlag`: Use `PyConfig.bytes_warning` or `PyConfig_Get("bytes_warning") <PyConfig_Get>` instead.
  - `!Py_FrozenFlag`: Use `PyConfig.pathconfig_warnings` or `PyConfig_Get("pathconfig_warnings") <PyConfig_Get>` instead.
  - `!Py_IgnoreEnvironmentFlag`: Use `PyConfig.use_environment` or `PyConfig_Get("use_environment") <PyConfig_Get>` instead.
  - `!Py_DontWriteBytecodeFlag`: Use `PyConfig.write_bytecode` or `PyConfig_Get("write_bytecode") <PyConfig_Get>` instead.
  - `!Py_NoUserSiteDirectory`: Use `PyConfig.user_site_directory` or `PyConfig_Get("user_site_directory") <PyConfig_Get>` instead.
  - `!Py_UnbufferedStdioFlag`: Use `PyConfig.buffered_stdio` or `PyConfig_Get("buffered_stdio") <PyConfig_Get>` instead.
  - `!Py_HashRandomizationFlag`: Use `PyConfig.use_hash_seed` and `PyConfig.hash_seed` or `PyConfig_Get("hash_seed") <PyConfig_Get>` instead.
  - `!Py_IsolatedFlag`: Use `PyConfig.isolated` or `PyConfig_Get("isolated") <PyConfig_Get>` instead.

- `!Py_LegacyWindowsFSEncodingFlag`: Use `PyPreConfig.legacy_windows_fs_encoding` or `PyConfig_Get("legacy_windows_fs_encoding") <PyConfig_Get>` instead.
- `!Py_LegacyWindowsStdioFlag`: Use `PyConfig.legacy_windows_stdio` or `PyConfig_Get("legacy_windows_stdio") <PyConfig_Get>` instead.
- `!Py_FileSystemDefaultEncoding`, `!Py_HasFileSystemDefaultEncoding`: Use `PyConfig.filesystem_encoding` or `PyConfig_Get("filesystem_encoding") <PyConfig_Get>` instead.
- `!Py_FileSystemDefaultEncodeErrors`: Use `PyConfig.filesystem_errors` or `PyConfig_Get("filesystem_errors") <PyConfig_Get>` instead.
- `!Py_UTF8Mode`: Use `PyPreConfig.utf8_mode` or `PyConfig_Get("utf8_mode") <PyConfig_Get>` instead. (see `Py_PreInitialize`)

  The `Py_InitializeFromConfig` API should be used with `PyConfig` to set these options. Or `PyConfig_Get` can be used to get these options at runtime.
- `Py_UNICODE` which was deprecated by `393`.

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.

Deprecations

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.

Soft deprecations

There are no plans to remove `soft deprecated` APIs.

- `re.match` and `re.Pattern.match` are now `soft deprecated` in favor of the new `re.prefixmatch` and `re.Pattern.prefixmatch` APIs, which have been added as alternate, more explicit names. These are intended to be used to alleviate confusion around what *match* means by following the Zen of Python's "*Explicit is better than implicit*" mantra. Most other language regular expression libraries use an API named *match* to mean what Python has always called *search*.
  We **do not** plan to remove the older `!match` name, as it has been used in code for over 30 years. Code supporting older versions of Python should continue to use `!match`, while new code should prefer `!prefixmatch`. See `prefixmatch-vs-match`.
  (Contributed by Gregory P. Smith in `86519` and Hugo van Kemenade in `148100`.)

C API deprecations

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.

Pending removal in Python 3.13

Modules (see `594`):

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

Other modules:

- `!lib2to3`, and the `2to3` program (`84540`)

APIs:

- `!configparser.LegacyInterpolation` (`90765`)
- `locale.resetlocale()` (`90817`)
- `!turtle.RawTurtle.settiltangle` (`50096`)
- `!unittest.findTestCases` (`50096`)
- `!unittest.getTestCaseNames` (`50096`)
- `!unittest.makeSuite` (`50096`)
- `!unittest.TestProgram.usageExit` (`67048`)
- `!webbrowser.MacOSX` (`86421`)
- `classmethod` descriptor chaining (`89519`)

Pending removal in Python 3.14

- `argparse`: The *type*, *choices*, and *metavar* parameters of `!argparse.BooleanOptionalAction` are deprecated and will be removed in 3.14. (Contributed by Nikita Sobolev in `92248`.)
- `ast`: The following features have been deprecated in documentation since Python 3.8, now cause a `DeprecationWarning` to be emitted at runtime when they are accessed or used, and will be removed in Python 3.14:
  - `!ast.Num`
  - `!ast.Str`
  - `!ast.Bytes`
  - `!ast.NameConstant`
  - `!ast.Ellipsis`

  Use `ast.Constant` instead. (Contributed by Serhiy Storchaka in `90953`.)
- `asyncio`:
  - The child watcher classes `!asyncio.MultiLoopChildWatcher`, `!asyncio.FastChildWatcher`, `!asyncio.AbstractChildWatcher` and `!asyncio.SafeChildWatcher` are deprecated and will be removed in Python 3.14. (Contributed by Kumar Aditya in `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 `94597`.)
  - The `~asyncio.get_event_loop` method of the default event loop policy now emits a `DeprecationWarning` if there is no current event loop set and it decides to create one. (Contributed by Serhiy Storchaka and Guido van Rossum in `100160`.)
- `email`: Deprecated the *isdst* parameter in `email.utils.localtime`. (Contributed by Alan Williams in `72346`.)
- `importlib.abc` deprecated classes:
  - `!importlib.abc.ResourceReader`
  - `!importlib.abc.Traversable`
  - `!importlib.abc.TraversableResources`

  Use `importlib.resources.abc` classes instead:
  - `importlib.resources.abc.Traversable`
  - `importlib.resources.abc.TraversableResources`

  (Contributed by Jason R. Coombs and Hugo van Kemenade in `93963`.)
- `itertools` had undocumented, inefficient, historically buggy, and inconsistent support for copy, deepcopy, and pickle operations. This will be removed in 3.14 for a significant reduction in code volume and maintenance burden. (Contributed by Raymond Hettinger in `101588`.)

- `multiprocessing`: The default start method will change to a safer one on Linux, BSDs, and other non-macOS POSIX platforms where `'fork'` is currently the default (`84559`). Adding a runtime warning about this was deemed too disruptive as the majority of code is not expected to care. Use the `~multiprocessing.get_context` or `~multiprocessing.set_start_method` APIs to explicitly specify when your code *requires* `'fork'`. See `multiprocessing-start-methods`.
- `pathlib`: `~pathlib.PurePath.is_relative_to` and `~pathlib.PurePath.relative_to`: passing additional arguments is deprecated.
- `pkgutil`: `!pkgutil.find_loader` and `!pkgutil.get_loader` now raise `DeprecationWarning`; use `importlib.util.find_spec` instead. (Contributed by Nikita Sobolev in `97850`.)
- `pty`:
  - `master_open()`: use `pty.openpty`.
  - `slave_open()`: use `pty.openpty`.
- `sqlite3`:
  - `!version` and `!version_info`.
  - `~sqlite3.Cursor.execute` and `~sqlite3.Cursor.executemany` if `named placeholders <sqlite3-placeholders>` are used and *parameters* is a sequence instead of a `dict`.
- `urllib`: `!urllib.parse.Quoter` is deprecated: it was not intended to be a public API. (Contributed by Gregory P. Smith in `88168`.)

Pending removal in Python 3.15

- The import system:
  - Setting `__cached__` on a module while failing to set `__spec__.cached` `<importlib.machinery.ModuleSpec.cached>` is deprecated. In Python 3.15, `__cached__` will cease to be set or take into consideration by the import system or standard library. (`97879`)
- `ctypes`:
  - The undocumented `!ctypes.SetPointerType` function has been deprecated since Python 3.13.
- `http.server`:
  - The obsolete and rarely used `!CGIHTTPRequestHandler` has been deprecated since Python 3.13. No direct replacement exists. *Anything* is better than CGI to interface a web server with a request handler.
  - The `!--cgi` flag to the `python -m http.server` command-line interface has been deprecated since Python 3.13.
- `importlib`:
  - `load_module()` method: use `exec_module()` instead.
- `pathlib`:
  - `!.PurePath.is_reserved` has been deprecated since Python 3.13. Use `os.path.isreserved` to detect reserved paths on Windows.
- `platform`:
  - `!platform.java_ver` has been deprecated since Python 3.13. This function is only useful for Jython support, has a confusing API, and is largely untested.
- `sysconfig`:
  - The *check_home* argument of `sysconfig.is_python_build` has been deprecated since Python 3.12.
- `threading`:
  - `~threading.RLock` will take no arguments in Python 3.15. Passing any arguments has been deprecated since Python 3.14, as the Python version does not permit any arguments, but the C version allows any number of positional or keyword arguments, ignoring every argument.
- `types`:
  - `types.CodeType`: Accessing `!codeobject.co_lnotab` was deprecated in `626` since 3.10 and was planned to be removed in 3.12, but it only got a proper `DeprecationWarning` in 3.12. May be removed in 3.15. (Contributed by Nikita Sobolev in `101866`.)
- `typing`:
  - The undocumented keyword argument syntax for creating `~typing.NamedTuple` classes (for example, `Point = NamedTuple("Point", x=int, y=int)`) has been

deprecated since Python 3.13. Use the class-based syntax or the functional syntax instead.
    - When using the functional syntax of `~typing.TypedDict`s, failing to pass a value to the *fields* parameter (`TD = TypedDict("TD")`) or passing `None` (`TD = TypedDict("TD", None)`) has been deprecated since Python 3.13. Use `class TD(TypedDict): pass` or `TD = TypedDict("TD", {})` to create a TypedDict with zero field.
    - The `!typing.no_type_check_decorator` decorator function has been deprecated since Python 3.13. After eight years in the `typing` module, it has yet to be supported by any major type checker.
- `!sre_compile`, `!sre_constants` and `!sre_parse` modules.
- `wave`:
    - The `getmark()`, `setmark()` and `getmarkers()` methods of the `~wave.Wave_read` and `~wave.Wave_write` classes have been deprecated since Python 3.13.
- `zipimport`:
    - `!zipimport.zipimporter.load_module` has been deprecated since Python 3.10. Use `~zipimport.zipimporter.exec_module` instead. (`125746`.)

Pending removal in Python 3.16

- The import system:
  - Setting `~module.__loader__` on a module while failing to set `__spec__.loader <importlib.machinery.ModuleSpec.loader>` is deprecated. In Python 3.16, `!__loader__` will cease to be set or taken into consideration by the import system or the standard library.
  - Setting `~module.__package__` on a module while failing to set `__spec__.parent <importlib.machinery.ModuleSpec.parent>` is deprecated. In Python 3.16, `!__package__` will cease to be taken into consideration by the import system or standard library. (`97879`)
- The bundled copy of `libmpdec`.
- `array`:
  - The `'u'` format code (`wchar_t`) has been deprecated in documentation since Python 3.3 and at runtime since Python 3.13. Use the `'w'` format code (`Py_UCS4`) for Unicode characters instead.
- `asyncio`:
  - `!asyncio.iscoroutinefunction` is deprecated and will be removed in Python 3.16; use `inspect.iscoroutinefunction` instead. (Contributed by Jiahao Li and Kumar Aditya in `122875` .)
  - `asyncio` policy system is deprecated and will be removed in Python 3.16. In particular, the following classes and functions are deprecated:
    - `!asyncio.AbstractEventLoopPolicy`
    - `!asyncio.DefaultEventLoopPolicy`
    - `!asyncio.WindowsSelectorEventLoopPolicy`
    - `!asyncio.WindowsProactorEventLoopPolicy`
    - `!asyncio.get_event_loop_policy`
    - `!asyncio.set_event_loop_policy`

    Users should use `asyncio.run` or `asyncio.Runner` with *loop_factory* to use the desired event loop implementation.
    For example, to use `asyncio.SelectorEventLoop` on Windows:

    ```
    import asyncio


    async def main():
        ...
    ```

```
asyncio.run(main(), loop_factory=asyncio.SelectorEventLoop)
```

(Contributed by Kumar Aditya in `127949`.)

- `builtins`:
  - Bitwise inversion on boolean types, `~True` or `~False` has been deprecated since Python 3.12, as it produces surprising and unintuitive results (`-2` and `-1`). Use `not x` instead for the logical negation of a Boolean. In the rare case that you need the bitwise inversion of the underlying integer, convert to `int` explicitly (`~int(x)`).
- `functools`:
  - Calling the Python implementation of `functools.reduce` with *function* or *sequence* as keyword arguments has been deprecated since Python 3.14.
- `logging`:
  - Support for custom logging handlers with the *strm* argument is deprecated and scheduled for removal in Python 3.16. Define handlers with the *stream* argument instead. (Contributed by Mariusz Felisiak in `115032`.)
- `mimetypes`:
  - Valid extensions start with a '.' or are empty for `mimetypes.MimeTypes.add_type`. Undotted extensions are deprecated and will raise a `ValueError` in Python 3.16. (Contributed by Hugo van Kemenade in `75223`.)
- `shutil`:
  - The `!ExecError` exception has been deprecated since Python 3.14. It has not been used by any function in `!shutil` since Python 3.4, and is now an alias of `RuntimeError`.
- `symtable`:
  - The `!symtable.Class.get_methods` method has been deprecated since Python 3.14.
- `sys`:
  - The `!_enablelegacywindowsfsencoding` function has been deprecated since Python 3.13. Use the `PYTHONLEGACYWINDOWSFSENCODING` environment variable instead.
- `sysconfig`:
  - The `!sysconfig.expand_makefile_vars` function has been deprecated since Python 3.14. Use the `vars` argument of `sysconfig.get_paths` instead.
- `tarfile`:
  - The undocumented and unused `!TarInfo.tarfile` attribute has been deprecated since Python 3.13.

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.

Soft deprecations

There are no plans to remove `soft deprecated` APIs.

- `re.match` and `re.Pattern.match` are now `soft deprecated` in favor of the new `re.prefixmatch` and `re.Pattern.prefixmatch` APIs, which have been added as alternate, more explicit names. These are intended to be used to alleviate confusion around what *match* means by following the Zen of Python's "*Explicit is better than implicit*" mantra. Most other language regular expression libraries use an API named *match* to mean what Python has always called *search*.
  We **do not** plan to remove the older `!match` name, as it has been used in code for over 30 years. Code supporting older versions of Python should continue to use `!match`, while new code should prefer `!prefixmatch`. See `prefixmatch-vs-match`.
  (Contributed by Gregory P. Smith in `86519` and Hugo van Kemenade in `148100`.)