What's New in Python

Release 3.9.1

A. M. Kuchling

January 27, 2021

Python Software Foundation
Email: docs@python.org

Contents

Release 3.9.1

Date January 27, 2021

Editor Łukasz Langa

This article explains the new features in Python 3.9, compared to 3.8. Python 3.9 was released on October 5th, 2020.

For full details, see the changelog.

See also:

PEP 596 - Python 3.9 Release Schedule

1 Summary – Release highlights

New syntax features:

- **PEP 584**, union operators added to `dict`;
- **PEP 585**, type hinting generics in standard collections;
- **PEP 614**, relaxed grammar restrictions on decorators.

New built-in features:

- **PEP 616**, string methods to remove prefixes and suffixes.

New features in the standard library:

- **PEP 593**, flexible function and variable annotations;
- `os.pidfd_open()` added that allows process management without races and signals.

Interpreter improvements:

- **PEP 573**, fast access to module state from methods of C extension types;
- **PEP 617**, CPython now uses a new parser based on PEG;
- a number of Python builtins (range, tuple, set, frozenset, list, dict) are now sped up using **PEP 590** vectorcall;
- garbage collection does not block on resurrected objects;
- a number of Python modules (`_abc`, `audioop`, `_bz2`, `_codecs`, `_contextvars`, `_crypt`, `_functools`, `_json`, `_locale`, `math`, `operator`, `resource`, `time`, `_weakref`) now use multiphase initialization as defined by PEP 489;
- a number of standard library modules (`audioop`, `ast`, `grp`, `_hashlib`, `pwd`, `_posixsubprocess`, `random`, `select`, `struct`, `termios`, `zlib`) are now using the stable ABI defined by PEP 384.

New library modules:

- **PEP 615**, the IANA Time Zone Database is now present in the standard library in the `zoneinfo` module;
- an implementation of a topological sort of a graph is now provided in the new `graphlib` module.

Release process changes:

- **PEP 602**, CPython adopts an annual release cycle.

2 You should check for DeprecationWarning in your code

When Python 2.7 was still supported, a lot of functionality in Python 3 was kept for backward compatibility with Python 2.7. With the end of Python 2 support, these backward compatibility layers have been removed, or will be removed soon. Most of them emitted a `DeprecationWarning` warning for several years. For example, using `collections.Mapping` instead of `collections.abc.Mapping` emits a `DeprecationWarning` since Python 3.3, released in 2012.

Test your application with the `-W default` command-line option to see `DeprecationWarning` and `PendingDeprecationWarning`, or even with `-W error` to treat them as errors. Warnings Filter can be used to ignore warnings from third-party code.

Python 3.9 is the last version providing those Python 2 backward compatibility layers, to give more time to Python projects maintainers to organize the removal of the Python 2 support and add support for Python 3.9.

Aliases to Abstract Base Classes in the `collections` module, like `collections.Mapping` alias to `collections.abc.Mapping`, are kept for one last release for backward compatibility. They will be removed from Python 3.10.

More generally, try to run your tests in the Python Development Mode which helps to prepare your code to make it compatible with the next Python version.

Note: a number of pre-existing deprecations were removed in this version of Python as well. Consult the *Removed* section.

3 New Features

3.1 Dictionary Merge & Update Operators

Merge (`|`) and update (`|=`) operators have been added to the built-in `dict` class. Those complement the existing `dict.update` and `{**d1, **d2}` methods of merging dictionaries.

Example:

```
>>> x = {"key1": "value1 from x", "key2": "value2 from x"}
>>> y = {"key2": "value2 from y", "key3": "value3 from y"}
>>> x | y
{'key1': 'value1 from x', 'key2': 'value2 from y', 'key3': 'value3 from y'}
>>> y | x
{'key2': 'value2 from x', 'key3': 'value3 from y', 'key1': 'value1 from x'}
```

See **PEP 584** for a full description. (Contributed by Brandt Bucher in bpo-36144.)

3.2 New String Methods to Remove Prefixes and Suffixes

`str.removeprefix(prefix)` and `str.removesuffix(suffix)` have been added to easily remove an unneeded prefix or a suffix from a string. Corresponding `bytes`, `bytearray`, and `collections.UserString` methods have also been added. See **PEP 616** for a full description. (Contributed by Dennis Sweeney in bpo-39939.)

3.3 Type Hinting Generics in Standard Collections

In type annotations you can now use built-in collection types such as `list` and `dict` as generic types instead of importing the corresponding capitalized types (e.g. `List` or `Dict`) from `typing`. Some other types in the standard library are also now generic, for example `queue.Queue`.

Example:

```
def greet_all(names: list[str]) -> None:
    for name in names:
        print("Hello", name)
```

See **PEP 585** for more details. (Contributed by Guido van Rossum, Ethan Smith, and Batuhan Taşkaya in bpo-39481.)

3.4 New Parser

Python 3.9 uses a new parser, based on PEG instead of LL(1). The new parser's performance is roughly comparable to that of the old parser, but the PEG formalism is more flexible than LL(1) when it comes to designing new language features. We'll start using this flexibility in Python 3.10 and later.

The `ast` module uses the new parser and produces the same AST as the old parser.

In Python 3.10, the old parser will be deleted and so will all functionality that depends on it (primarily the `parser` module, which has long been deprecated). In Python 3.9 *only*, you can switch back to the LL(1) parser using a command line switch (`-X oldparser`) or an environment variable (`PYTHONOLDPARSER=1`).

See **PEP 617** for more details. (Contributed by Guido van Rossum, Pablo Galindo and Lysandros Nikolaou in bpo-40334.)

4 Other Language Changes

- `__import__()` now raises `ImportError` instead of `ValueError`, which used to occur when a relative import went past its top-level package. (Contributed by Ngalim Siregar in bpo-37444.)
- Python now gets the absolute path of the script filename specified on the command line (ex: `python3 script.py`): the `__file__` attribute of the `__main__` module became an absolute path, rather than a relative path. These paths now remain valid after the current directory is changed by `os.chdir()`. As a side effect, the traceback also displays the absolute path for `__main__` module frames in this case. (Contributed by Victor Stinner in bpo-20443.)
- In the Python Development Mode and in debug build, the *encoding* and *errors* arguments are now checked for string encoding and decoding operations. Examples: `open()`, `str.encode()` and `bytes.decode()`.

  By default, for best performance, the *errors* argument is only checked at the first encoding/decoding error and the *encoding* argument is sometimes ignored for empty strings. (Contributed by Victor Stinner in bpo-37388.)
- `"".replace("", s, n)` now returns `s` instead of an empty string for all non-zero `n`. It is now consistent with `"".replace("", s)`. There are similar changes for `bytes` and `bytearray` objects. (Contributed by Serhiy Storchaka in bpo-28029.)
- Any valid expression can now be used as a decorator. Previously, the grammar was much more restrictive. See **PEP 614** for details. (Contributed by Brandt Bucher in bpo-39702.)
- Improved help for the `typing` module. Docstrings are now shown for all special forms and special generic aliases (like `Union` and `List`). Using `help()` with generic alias like `List[int]` will show the help for the correspondent concrete type (`list` in this case). (Contributed by Serhiy Storchaka in bpo-40257.)
- Parallel running of `aclose()` / `asend()` / `athrow()` is now prohibited, and `ag_running` now reflects the actual running status of the async generator. (Contributed by Yury Selivanov in bpo-30773.)
- Unexpected errors in calling the `__iter__` method are no longer masked by `TypeError` in the `in` operator and functions `contains()`, `indexOf()` and `countOf()` of the `operator` module. (Contributed by Serhiy Storchaka in bpo-40824.)

5 New Modules

5.1 zoneinfo

The `zoneinfo` module brings support for the IANA time zone database to the standard library. It adds `zoneinfo.ZoneInfo`, a concrete `datetime.tzinfo` implementation backed by the system's time zone data.

Example:

```
>>> from zoneinfo import ZoneInfo
>>> from datetime import datetime, timedelta

>>> # Daylight saving time
>>> dt = datetime(2020, 10, 31, 12, tzinfo=ZoneInfo("America/Los_Angeles"))
>>> print(dt)
2020-10-31 12:00:00-07:00
>>> dt.tzname()
'PDT'

>>> # Standard time
>>> dt += timedelta(days=7)
>>> print(dt)
2020-11-07 12:00:00-08:00
>>> print(dt.tzname())
PST
```

As a fall-back source of data for platforms that don't ship the IANA database, the `tzdata` module was released as a first-party package – distributed via PyPI and maintained by the CPython core team.

See also:

PEP 615 – Support for the IANA Time Zone Database in the Standard Library PEP written and implemented by Paul Ganssle

5.2 graphlib

A new module, `graphlib`, was added that contains the `graphlib.TopologicalSorter` class to offer functionality to perform topological sorting of graphs. (Contributed by Pablo Galindo, Tim Peters and Larry Hastings in bpo-17005.)

6 Improved Modules

6.1 ast

Added the *indent* option to `dump()` which allows it to produce a multiline indented output. (Contributed by Serhiy Storchaka in bpo-37995.)

Added `ast.unparse()` as a function in the `ast` module that can be used to unparse an `ast.AST` object and produce a string with code that would produce an equivalent `ast.AST` object when parsed. (Contributed by Pablo Galindo and Batuhan Taskaya in bpo-38870.)

Added docstrings to AST nodes that contains the ASDL signature used to construct that node. (Contributed by Batuhan Taskaya in bpo-39638.)

6.2 asyncio

Due to significant security concerns, the *reuse_address* parameter of `asyncio.loop.create_datagram_endpoint()` is no longer supported. This is because of the behavior of the socket option `SO_REUSEADDR` in UDP. For more details, see the documentation for `loop.create_datagram_endpoint()`. (Contributed by Kyle Stanley, Antoine Pitrou, and Yury Selivanov in bpo-37228.)

Added a new coroutine `shutdown_default_executor()` that schedules a shutdown for the default executor that waits on the `ThreadPoolExecutor` to finish closing. Also, `asyncio.run()` has been updated to use the new coroutine. (Contributed by Kyle Stanley in bpo-34037.)

Added `asyncio.PidfdChildWatcher`, a Linux-specific child watcher implementation that polls process file descriptors. (bpo-38692)

Added a new coroutine `asyncio.to_thread()`. It is mainly used for running IO-bound functions in a separate thread to avoid blocking the event loop, and essentially works as a high-level version of `run_in_executor()` that can directly take keyword arguments. (Contributed by Kyle Stanley and Yury Selivanov in bpo-32309.)

When cancelling the task due to a timeout, `asyncio.wait_for()` will now wait until the cancellation is complete also in the case when *timeout* is <= 0, like it does with positive timeouts. (Contributed by Elvis Pranskevichus in bpo-32751.)

`asyncio` now raises `TyperError` when calling incompatible methods with an `ssl.SSLSocket` socket. (Contributed by Ido Michael in bpo-37404.)

6.3 compileall

Added new possibility to use hardlinks for duplicated `.pyc` files: *hardlink_dupes* parameter and –hardlink-dupes command line option. (Contributed by Lumír 'Frenzy' Balhar in bpo-40495.)

Added new options for path manipulation in resulting `.pyc` files: *stripdir*, *prependdir*, *limit_sl_dest* parameters and -s, -p, -e command line options. Added the possibility to specify the option for an optimization level multiple times. (Contributed by Lumír 'Frenzy' Balhar in bpo-38112.)

6.4 concurrent.futures

Added a new *cancel_futures* parameter to `concurrent.futures.Executor.shutdown()` that cancels all pending futures which have not started running, instead of waiting for them to complete before shutting down the executor. (Contributed by Kyle Stanley in bpo-39349.)

Removed daemon threads from `ThreadPoolExecutor` and `ProcessPoolExecutor`. This improves compatibility with subinterpreters and predictability in their shutdown processes. (Contributed by Kyle Stanley in bpo-39812.)

Workers in `ProcessPoolExecutor` are now spawned on demand, only when there are no available idle workers to reuse. This optimizes startup overhead and reduces the amount of lost CPU time to idle workers. (Contributed by Kyle Stanley in bpo-39207.)

6.5 curses

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

6.6 datetime

The `isocalendar()` of `datetime.date` and `isocalendar()` of `datetime.datetime` methods now returns a `namedtuple()` instead of a `tuple`. (Contributed by Dong-hee Na in bpo-24416.)

6.7 distutils

The **`upload`** command now creates SHA2-256 and Blake2b-256 hash digests. It skips MD5 on platforms that block MD5 digest. (Contributed by Christian Heimes in bpo-40698.)

6.8 fcntl

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

6.9 ftplib

`FTP` and `FTP_TLS` now raise a `ValueError` if the given timeout for their constructor is zero to prevent the creation of a non-blocking socket. (Contributed by Dong-hee Na in bpo-39259.)

6.10 gc

When the garbage collector makes a collection in which some objects resurrect (they are reachable from outside the isolated cycles after the finalizers have been executed), do not block the collection of all objects that are still unreachable. (Contributed by Pablo Galindo and Tim Peters in bpo-38379.)

Added a new function `gc.is_finalized()` to check if an object has been finalized by the garbage collector. (Contributed by Pablo Galindo in bpo-39322.)

6.11 hashlib

The `hashlib` module can now use SHA3 hashes and SHAKE XOF from OpenSSL when available. (Contributed by Christian Heimes in bpo-37630.)

Builtin hash modules can now be disabled with `./configure --without-builtin-hashlib-hashes` or selectively enabled with e.g. `./configure --with-builtin-hashlib-hashes=sha3,blake2` to force use of OpenSSL based implementation. (Contributed by Christian Heimes in bpo-40479)

6.12 http

HTTP status codes `103 EARLY_HINTS`, `418 IM_A_TEAPOT` and `425 TOO_EARLY` are added to `http.HTTPStatus`. (Contributed by Dong-hee Na in bpo-39509 and Ross Rhodes in bpo-39507.)

6.13 IDLE and idlelib

Added option to toggle cursor blink off. (Contributed by Zackery Spytz in bpo-4603.)

Escape key now closes IDLE completion windows. (Contributed by Johnny Najera in bpo-38944.)

Added keywords to module name completion list. (Contributed by Terry J. Reedy in bpo-37765.)

The changes above have been backported to 3.8 maintenance releases.

6.14 imaplib

`IMAP4` and `IMAP4_SSL` now have an optional *timeout* parameter for their constructors. Also, the `open()` method now has an optional *timeout* parameter with this change. The overridden methods of `IMAP4_SSL` and `IMAP4_stream` were applied to this change. (Contributed by Dong-hee Na in bpo-38615.)

`imaplib.IMAP4.unselect()` is added. `imaplib.IMAP4.unselect()` frees server's resources associated with the selected mailbox and returns the server to the authenticated state. This command performs the same actions as `imaplib.IMAP4.close()`, except that no messages are permanently removed from the currently selected mailbox. (Contributed by Dong-hee Na in bpo-40375.)

6.15 importlib

To improve consistency with import statements, `importlib.util.resolve_name()` now raises `ImportError` instead of `ValueError` for invalid relative import attempts. (Contributed by Ngalim Siregar in bpo-37444.)

Import loaders which publish immutable module objects can now publish immutable packages in addition to individual modules. (Contributed by Dino Viehland in bpo-39336.)

Added `importlib.resources.files()` function with support for subdirectories in package data, matching backport in `importlib_resources` version 1.5. (Contributed by Jason R. Coombs in bpo-39791.)

Refreshed `importlib.metadata` from `importlib_metadata` version 1.6.1.

6.16 inspect

`inspect.BoundArguments.arguments` is changed from `OrderedDict` to regular dict. (Contributed by Inada Naoki in bpo-36350 and bpo-39775.)

6.17 ipaddress

`ipaddress` now supports IPv6 Scoped Addresses (IPv6 address with suffix `%<scope_id>`).

Scoped IPv6 addresses can be parsed using `ipaddress.IPv6Address`. If present, scope zone ID is available through the `scope_id` attribute. (Contributed by Oleksandr Pavliuk in bpo-34788.)

6.18 math

Expanded the `math.gcd()` function to handle multiple arguments. Formerly, it only supported two arguments. (Contributed by Serhiy Storchaka in bpo-39648.)

Added `math.lcm()`: return the least common multiple of specified arguments. (Contributed by Mark Dickinson, Ananthakrishnan and Serhiy Storchaka in bpo-39479 and bpo-39648.)

Added `math.nextafter()`: return the next floating-point value after *x* towards *y*. (Contributed by Victor Stinner in bpo-39288.)

Added `math.ulp()`: return the value of the least significant bit of a float. (Contributed by Victor Stinner in bpo-39310.)

6.19 multiprocessing

The `multiprocessing.SimpleQueue` class has a new `close()` method to explicitly close the queue. (Contributed by Victor Stinner in bpo-30966.)

6.20 nntplib

`NNTP` and `NNTP_SSL` now raise a `ValueError` if the given timeout for their constructor is zero to prevent the creation of a non-blocking socket. (Contributed by Dong-hee Na in bpo-39259.)

6.21 os

Added `CLD_KILLED` and `CLD_STOPPED` for `si_code`. (Contributed by Dong-hee Na in bpo-38493.)

Exposed the Linux-specific `os.pidfd_open()` (bpo-38692) and `os.P_PIDFD` (bpo-38713) for process management with file descriptors.

The `os.unsetenv()` function is now also available on Windows. (Contributed by Victor Stinner in bpo-39413.)

The `os.putenv()` and `os.unsetenv()` functions are now always available. (Contributed by Victor Stinner in bpo-39395.)

Added `os.waitstatus_to_exitcode()` function: convert a wait status to an exit code. (Contributed by Victor Stinner in bpo-40094.)

6.22 pathlib

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

6.23 pdb

On Windows now `Pdb` supports `~/.pdbrc`. (Contributed by Tim Hopper and Dan Lidral-Porter in bpo-20523.)

6.24 poplib

`POP3` and `POP3_SSL` now raise a `ValueError` if the given timeout for their constructor is zero to prevent the creation of a non-blocking socket. (Contributed by Dong-hee Na in bpo-39259.)

6.25 pprint

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

6.26 pydoc

The documentation string is now shown not only for class, function, method etc, but for any object that has its own `__doc__` attribute. (Contributed by Serhiy Storchaka in bpo-40257.)

6.27 random

Added a new `random.Random.randbytes` method: generate random bytes. (Contributed by Victor Stinner in bpo-40286.)

6.28 signal

Exposed the Linux-specific `signal.pidfd_send_signal()` for sending to signals to a process using a file descriptor instead of a pid. (bpo-38712)

6.29 smtplib

`SMTP` and `SMTP_SSL` now raise a `ValueError` if the given timeout for their constructor is zero to prevent the creation of a non-blocking socket. (Contributed by Dong-hee Na in bpo-39259.)

`LMTP` constructor now has an optional *timeout* parameter. (Contributed by Dong-hee Na in bpo-39329.)

6.30 socket

The `socket` module now exports the `CAN_RAW_JOIN_FILTERS` constant on Linux 4.1 and greater. (Contributed by Stefan Tatschner and Zackery Spytz in bpo-25780.)

The socket module now supports the `CAN_J1939` protocol on platforms that support it. (Contributed by Karl Ding in bpo-40291.)

The socket module now has the `socket.send_fds()` and `socket.recv.fds()` methods. (Contributed by Joannah Nanjekye, Shinya Okano and Victor Stinner in bpo-28724.)

6.31 time

On AIX, `thread_time()` is now implemented with `thread_cputime()` which has nanosecond resolution, rather than `clock_gettime(CLOCK_THREAD_CPUTIME_ID)` which has a resolution of 10 ms. (Contributed by Batuhan Taskaya in bpo-40192)

6.32 sys

Added a new `sys.platlibdir` attribute: name of the platform-specific library directory. It is used to build the path of standard library and the paths of installed extension modules. It is equal to `"lib"` on most platforms. On Fedora and SuSE, it is equal to `"lib64"` on 64-bit platforms. (Contributed by Jan Matějek, Matěj Cepl, Charalampos Stratakis and Victor Stinner in bpo-1294959.)

Previously, `sys.stderr` was block-buffered when non-interactive. Now `stderr` defaults to always being line-buffered. (Contributed by Jendrik Seipp in bpo-13601.)

6.33 tracemalloc

Added `tracemalloc.reset_peak()` to set the peak size of traced memory blocks to the current size, to measure the peak of specific pieces of code. (Contributed by Huon Wilson in bpo-40630.)

6.34 typing

PEP 593 introduced an `typing.Annotated` type to decorate existing types with context-specific metadata and new `include_extras` parameter to `typing.get_type_hints()` to access the metadata at runtime. (Contributed by Till Varoquaux and Konstantin Kashin.)

6.35 unicodedata

The Unicode database has been updated to version 13.0.0. (bpo-39926).

6.36 venv

The activation scripts provided by `venv` now all specify their prompt customization consistently by always using the value specified by `__VENV_PROMPT__`. Previously some scripts unconditionally used `__VENV_PROMPT__`, others only if it happened to be set (which was the default case), and one used `__VENV_NAME__` instead. (Contributed by Brett Cannon in bpo-37663.)

6.37 xml

White space characters within attributes are now preserved when serializing `xml.etree.ElementTree` to XML file. EOLNs are no longer normalized to "n". This is the result of discussion about how to interpret section 2.11 of XML spec. (Contributed by Mefistotelis in bpo-39011.)

7 Optimizations

- Optimized the idiom for assignment a temporary variable in comprehensions. Now `for y in [expr]` in comprehensions is as fast as a simple assignment `y = expr`. For example:

  sums = [s for s in [0] for x in data for s in [s + x]]

  Unlike the `:=` operator this idiom does not leak a variable to the outer scope.

  (Contributed by Serhiy Storchaka in bpo-32856.)

- Optimized signal handling in multithreaded applications. If a thread different than the main thread gets a signal, the bytecode evaluation loop is no longer interrupted at each bytecode instruction to check for pending signals which cannot be handled. Only the main thread of the main interpreter can handle signals.

  Previously, the bytecode evaluation loop was interrupted at each instruction until the main thread handles signals. (Contributed by Victor Stinner in bpo-40010.)

- Optimized the `subprocess` module on FreeBSD using `closefrom()`. (Contributed by Ed Maste, Conrad Meyer, Kyle Evans, Kubilay Kocak and Victor Stinner in bpo-38061.)

- `PyLong_FromDouble()` is now up to 1.87x faster for values that fit into `long`. (Contributed by Sergey Fedoseev in bpo-37986.)

- A number of Python builtins (`range`, `tuple`, `set`, `frozenset`, `list`, `dict`) are now sped up by using **PEP 590** vectorcall protocol. (Contributed by Dong-hee Na, Mark Shannon, Jeroen Demeyer and Petr Viktorin in bpo-37207.)

- Optimized `difference_update()` for the case when the other set is much larger than the base set. (Suggested by Evgeny Kapun with code contributed by Michele Orrù in bpo-8425.)

- Python's small object allocator (`obmalloc.c`) now allows (no more than) one empty arena to remain available for immediate reuse, without returning it to the OS. This prevents thrashing in simple loops where an arena could be created and destroyed anew on each iteration. (Contributed by Tim Peters in bpo-37257.)

- floor division of float operation now has a better performance. Also the message of `ZeroDivisionError` for this operation is updated. (Contributed by Dong-hee Na in bpo-39434.)

- Decoding short ASCII strings with UTF-8 and ascii codecs is now about 15% faster. (Contributed by Inada Naoki in bpo-37348.)

Here's a summary of performance improvements from Python 3.4 through Python 3.9:

Python version	3.4	3.5	3.6	3.7	3.8	3.9
Variable and attribute read access:						
read_local	7.1	7.1	5.4	5.1	3.9	3.9
read_nonlocal	7.1	8.1	5.8	5.4	4.4	4.5
read_global	15.5	19.0	14.3	13.6	7.6	7.8
read_builtin	21.1	21.6	18.5	19.0	7.5	7.8
read_classvar_from_class	25.6	26.5	20.7	19.5	18.4	17.9
read_classvar_from_instance	22.8	23.5	18.8	17.1	16.4	16.9
read_instancevar	32.4	33.1	28.0	26.3	25.4	25.3
read_instancevar_slots	27.8	31.3	20.8	20.8	20.2	20.5
read_namedtuple	73.8	57.5	45.0	46.8	18.4	18.7
read_boundmethod	37.6	37.9	29.6	26.9	27.7	41.1

(continues on next page)

(continued from previous page)

```
Variable and attribute write access:
    write_local                      8.7     9.3     5.5     5.3     4.3     4.3
    write_nonlocal                  10.5    11.1     5.6     5.5     4.7     4.8
    write_global                    19.7    21.2    18.0    18.0    15.8    16.7
    write_classvar                  92.9    96.0   104.6   102.1    39.2    39.8
    write_instancevar               44.6    45.8    40.0    38.9    35.5    37.4
    write_instancevar_slots         35.6    36.1    27.3    26.6    25.7    25.8

Data structure read access:
    read_list                       24.2    24.5    20.8    20.8    19.0    19.5
    read_deque                      24.7    25.5    20.2    20.6    19.8    20.2
    read_dict                       24.3    25.7    22.3    23.0    21.0    22.4
    read_strdict                    22.6    24.3    19.5    21.2    18.9    21.5

Data structure write access:
    write_list                      27.1    28.5    22.5    21.6    20.0    20.0
    write_deque                     28.7    30.1    22.7    21.8    23.5    21.7
    write_dict                      31.4    33.3    29.3    29.2    24.7    25.4
    write_strdict                   28.4    29.9    27.5    25.2    23.1    24.5

Stack (or queue) operations:
    list_append_pop                 93.4   112.7    75.4    74.2    50.8    50.6
    deque_append_pop                43.5    57.0    49.4    49.2    42.5    44.2
    deque_append_popleft            43.7    57.3    49.7    49.7    42.8    46.4

Timing loop:
    loop_overhead                    0.5     0.6     0.4     0.3     0.3     0.3
```

These results were generated from the variable access benchmark script at: `Tools/scripts/var_access_benchmark.py`. The benchmark script displays timings in nanoseconds. The benchmarks were measured on an Intel® Core™ i7-4960HQ processor running the macOS 64-bit builds found at python.org.

8 Deprecated

- The distutils `bdist_msi` command is now deprecated, use `bdist_wheel` (wheel packages) instead. (Contributed by Hugo van Kemenade in bpo-39586.)
- Currently `math.factorial()` accepts `float` instances with non-negative integer values (like `5.0`). It raises a `ValueError` for non-integral and negative floats. It is now deprecated. In future Python versions it will raise a `TypeError` for all floats. (Contributed by Serhiy Storchaka in bpo-37315.)
- The `parser` and `symbol` modules are deprecated and will be removed in future versions of Python. For the majority of use cases, users can leverage the Abstract Syntax Tree (AST) generation and compilation stage, using the `ast` module.
- The Public C API functions `PyParser_SimpleParseStringFlags()`, `PyParser_SimpleParseStringFlagsFilename()`, `PyParser_SimpleParseFileFlags()` and `PyNode_Compile()` are deprecated and will be removed in Python 3.10 together with the old parser.
- Using `NotImplemented` in a boolean context has been deprecated, as it is almost exclusively the result of incorrect rich comparator implementations. It will be made a `TypeError` in a future version of Python. (Contributed by Josh Rosenberg in bpo-35712.)
- The `random` module currently accepts any hashable type as a possible seed value. Unfortunately, some of those types are not guaranteed to have a deterministic hash value. After Python 3.9, the module will restrict its seeds to `None`, `int`, `float`, `str`, `bytes`, and `bytearray`.
- Opening the `GzipFile` file for writing without specifying the *mode* argument is deprecated. In future Python versions it will always be opened for reading by default. Specify the *mode* argument for opening it for writing

and silencing a warning. (Contributed by Serhiy Storchaka in bpo-28286.)

- Deprecated the `split()` method of `_tkinter.TkappType` in favour of the `splitlist()` method which has more consistent and predicable behavior. (Contributed by Serhiy Storchaka in bpo-38371.)
- The explicit passing of coroutine objects to `asyncio.wait()` has been deprecated and will be removed in version 3.11. (Contributed by Yury Selivanov and Kyle Stanley in bpo-34790.)
- binhex4 and hexbin4 standards are now deprecated. The `binhex` module and the following `binascii` functions are now deprecated:
  - `b2a_hqx()`, `a2b_hqx()`
  - `rlecode_hqx()`, `rledecode_hqx()`

  (Contributed by Victor Stinner in bpo-39353.)
- `ast` classes `slice`, `Index` and `ExtSlice` are considered deprecated and will be removed in future Python versions. `value` itself should be used instead of `Index(value)`. `Tuple(slices, Load())` should be used instead of `ExtSlice(slices)`. (Contributed by Serhiy Storchaka in bpo-34822.)
- `ast` classes `Suite`, `Param`, `AugLoad` and `AugStore` are considered deprecated and will be removed in future Python versions. They were not generated by the parser and not accepted by the code generator in Python 3. (Contributed by Batuhan Taskaya in bpo-39639 and bpo-39969 and Serhiy Storchaka in bpo-39988.)
- The `PyEval_InitThreads()` and `PyEval_ThreadsInitialized()` functions are now deprecated and will be removed in Python 3.11. Calling `PyEval_InitThreads()` now does nothing. The GIL is initialized by `Py_Initialize()` since Python 3.7. (Contributed by Victor Stinner in bpo-39877.)
- Passing `None` as the first argument to the `shlex.split()` function has been deprecated. (Contributed by Zackery Spytz in bpo-33262.)
- `smtpd.MailmanProxy()` is now deprecated as it is unusable without an external module, `mailman`. (Contributed by Samuel Colvin in bpo-35800.)
- The `lib2to3` module now emits a `PendingDeprecationWarning`. Python 3.9 switched to a PEG parser (see **PEP 617**), and Python 3.10 may include new language syntax that is not parsable by lib2to3's LL(1) parser. The `lib2to3` module may be removed from the standard library in a future Python version. Consider third-party alternatives such as LibCST or parso. (Contributed by Carl Meyer in bpo-40360.)
- The *random* parameter of `random.shuffle()` has been deprecated. (Contributed by Raymond Hettinger in bpo-40465)

9 Removed

- The erroneous version at `unittest.mock.__version__` has been removed.
- `nntplib.NNTP`: `xpath()` and `xgtitle()` methods have been removed. These methods are deprecated since Python 3.3. Generally, these extensions are not supported or not enabled by NNTP server administrators. For `xgtitle()`, please use `nntplib.NNTP.descriptions()` or `nntplib.NNTP.description()` instead. (Contributed by Dong-hee Na in bpo-39366.)
- `array.array`: `tostring()` and `fromstring()` methods have been removed. They were aliases to `tobytes()` and `frombytes()`, deprecated since Python 3.2. (Contributed by Victor Stinner in bpo-38916.)
- The undocumented `sys.callstats()` function has been removed. Since Python 3.7, it was deprecated and always returned `None`. It required a special build option `CALL_PROFILE` which was already removed in Python 3.7. (Contributed by Victor Stinner in bpo-37414.)
- The `sys.getcheckinterval()` and `sys.setcheckinterval()` functions have been removed. They were deprecated since Python 3.2. Use `sys.getswitchinterval()` and `sys.setswitchinterval()` instead. (Contributed by Victor Stinner in bpo-37392.)

- The C function `PyImport_Cleanup()` has been removed. It was documented as: "Empty the module table. For internal use only." (Contributed by Victor Stinner in bpo-36710.)
- `_dummy_thread` and `dummy_threading` modules have been removed. These modules were deprecated since Python 3.7 which requires threading support. (Contributed by Victor Stinner in bpo-37312.)
- `aifc.openfp()` alias to `aifc.open()`, `sunau.openfp()` alias to `sunau.open()`, and `wave.openfp()` alias to `wave.open()` have been removed. They were deprecated since Python 3.7. (Contributed by Victor Stinner in bpo-37320.)
- The `isAlive()` method of `threading.Thread` has been removed. It was deprecated since Python 3.8. Use `is_alive()` instead. (Contributed by Dong-hee Na in bpo-37804.)
- Methods `getchildren()` and `getiterator()` of classes `ElementTree` and `Element` in the `ElementTree` module have been removed. They were deprecated in Python 3.2. Use `iter(x)` or `list(x)` instead of `x.getchildren()` and `x.iter()` or `list(x.iter())` instead of `x.getiterator()`. (Contributed by Serhiy Storchaka in bpo-36543.)
- The old `plistlib` API has been removed, it was deprecated since Python 3.4. Use the `load()`, `loads()`, `dump()`, and `dumps()` functions. Additionally, the *use_builtin_types* parameter was removed, standard `bytes` objects are always used instead. (Contributed by Jon Janzen in bpo-36409.)
- The C function `PyGen_NeedsFinalizing` has been removed. It was not documented, tested, or used anywhere within CPython after the implementation of **PEP 442**. Patch by Joannah Nanjekye. (Contributed by Joannah Nanjekye in bpo-15088)
- `base64.encodestring()` and `base64.decodestring()`, aliases deprecated since Python 3.1, have been removed: use `base64.encodebytes()` and `base64.decodebytes()` instead. (Contributed by Victor Stinner in bpo-39351.)
- `fractions.gcd()` function has been removed, it was deprecated since Python 3.5 (bpo-22486): use `math.gcd()` instead. (Contributed by Victor Stinner in bpo-39350.)
- The *buffering* parameter of `bz2.BZ2File` has been removed. Since Python 3.0, it was ignored and using it emitted a `DeprecationWarning`. Pass an open file object to control how the file is opened. (Contributed by Victor Stinner in bpo-39357.)
- The *encoding* parameter of `json.loads()` has been removed. As of Python 3.1, it was deprecated and ignored; using it has emitted a `DeprecationWarning` since Python 3.8. (Contributed by Inada Naoki in bpo-39377)
- `with (await asyncio.lock):` and `with (yield from asyncio.lock):` statements are not longer supported, use `async with lock` instead. The same is correct for `asyncio.Condition` and `asyncio.Semaphore`. (Contributed by Andrew Svetlov in bpo-34793.)
- The `sys.getcounts()` function, the `-X showalloccount` command line option and the `show_alloc_count` field of the C structure `PyConfig` have been removed. They required a special Python build by defining `COUNT_ALLOCS` macro. (Contributed by Victor Stinner in bpo-39489.)
- The `_field_types` attribute of the `typing.NamedTuple` class has been removed. It was deprecated since Python 3.8. Use the `__annotations__` attribute instead. (Contributed by Serhiy Storchaka in bpo-40182.)
- The `symtable.SymbolTable.has_exec()` method has been removed. It was deprecated since 2006, and only returning `False` when it's called. (Contributed by Batuhan Taskaya in bpo-40208)
- The `asyncio.Task.current_task()` and `asyncio.Task.all_tasks()` have been removed. They were deprecated since Python 3.7 and you can use `asyncio.current_task()` and `asyncio.all_tasks()` instead. (Contributed by Rémi Lapeyre in bpo-40967)
- The `unescape()` method in the `html.parser.HTMLParser` class has been removed (it was deprecated since Python 3.4). `html.unescape()` should be used for converting character references to the corresponding unicode characters.

10 Porting to Python 3.9

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

10.1 Changes in the Python API

- `__import__()` and `importlib.util.resolve_name()` now raise `ImportError` where it previously raised `ValueError`. Callers catching the specific exception type and supporting both Python 3.9 and earlier versions will need to catch both using `except (ImportError, ValueError):`.
- The `venv` activation scripts no longer special-case when `__VENV_PROMPT__` is set to `""`.
- The `select.epoll.unregister()` method no longer ignores the `EBADF` error. (Contributed by Victor Stinner in bpo-39239.)
- The *compresslevel* parameter of `bz2.BZ2File` became keyword-only, since the *buffering* parameter has been removed. (Contributed by Victor Stinner in bpo-39357.)
- Simplified AST for subscription. Simple indices will be represented by their value, extended slices will be represented as tuples. `Index(value)` will return a `value` itself, `ExtSlice(slices)` will return `Tuple(slices, Load())`. (Contributed by Serhiy Storchaka in bpo-34822.)
- The `importlib` module now ignores the `PYTHONCASEOK` environment variable when the `-E` or `-I` command line options are being used.
- The *encoding* parameter has been added to the classes `ftplib.FTP` and `ftplib.FTP_TLS` as a keyword-only parameter, and the default encoding is changed from Latin-1 to UTF-8 to follow **RFC 2640**.
- `asyncio.loop.shutdown_default_executor()` has been added to `AbstractEventLoop`, meaning alternative event loops that inherit from it should have this method defined. (Contributed by Kyle Stanley in bpo-34037.)
- The constant values of future flags in the `__future__` module is updated in order to prevent collision with compiler flags. Previously `PyCF_ALLOW_TOP_LEVEL_AWAIT` was clashing with `CO_FUTURE_DIVISION`. (Contributed by Batuhan Taskaya in bpo-39562)
- `array('u')` now uses `wchar_t` as C type instead of `Py_UNICODE`. This change doesn't affect to its behavior because `Py_UNICODE` is alias of `wchar_t` since Python 3.3. (Contributed by Inada Naoki in bpo-34538.)
- The `logging.getLogger()` API now returns the root logger when passed the name `'root'`, whereas previously it returned a non-root logger named `'root'`. This could affect cases where user code explicitly wants a non-root logger named `'root'`, or instantiates a logger using `logging.getLogger(__name__)` in some top-level module called `'root.py'`. (Contributed by Vinay Sajip in bpo-37742.)
- Division handling of `PurePath` now returns `NotImplemented` instead of raising a `TypeError` when passed something other than an instance of `str` or `PurePath`. This allows creating compatible classes that don't inherit from those mentioned types. (Contributed by Roger Aiudi in bpo-34775).

10.2 Changes in the C API

- Instances of heap-allocated types (such as those created with `PyType_FromSpec()` and similar APIs) hold a reference to their type object since Python 3.8. As indicated in the "Changes in the C API" of Python 3.8, for the vast majority of cases, there should be no side effect but for types that have a custom `tp_traverse` function, ensure that all custom `tp_traverse` functions of heap-allocated types visit the object's type.

  Example:

```
int
foo_traverse(foo_struct *self, visitproc visit, void *arg) {
// Rest of the traverse function
#if PY_VERSION_HEX >= 0x03090000
    // This was not needed before Python 3.9 (Python issue 35810 and␣
↪40217)
    Py_VISIT(Py_TYPE(self));
#endif
}
```

If your traverse function delegates to `tp_traverse` of its base class (or another type), ensure that `Py_TYPE(self)` is visited only once. Note that only heap types are expected to visit the type in `tp_traverse`.

For example, if your `tp_traverse` function includes:

```
base->tp_traverse(self, visit, arg)
```

then add:

```
#if PY_VERSION_HEX >= 0x03090000
    // This was not needed before Python 3.9 (Python issue 35810 and␣
↪40217)
    if (base->tp_flags & Py_TPFLAGS_HEAPTYPE) {
        // a heap type's tp_traverse already visited Py_TYPE(self)
    } else {
        Py_VISIT(Py_TYPE(self));
    }
#else
```

(See bpo-35810 and bpo-40217 for more information.)

- The functions `PyEval_CallObject`, `PyEval_CallFunction`, `PyEval_CallMethod` and `PyEval_CallObjectWithKeywords` are deprecated. Use `PyObject_Call()` and its variants instead. (See more details in bpo-29548.)

10.3 CPython bytecode changes

- The `LOAD_ASSERTION_ERROR` opcode was added for handling the `assert` statement. Previously, the assert statement would not work correctly if the `AssertionError` exception was being shadowed. (Contributed by Zackery Spytz in bpo-34880.)
- The `COMPARE_OP` opcode was split into four distinct instructions:
  - `COMPARE_OP` for rich comparisons
  - `IS_OP` for 'is' and 'is not' tests
  - `CONTAINS_OP` for 'in' and 'not in' tests
  - `JUMP_IF_NOT_EXC_MATCH` for checking exceptions in 'try-except' statements.

  (Contributed by Mark Shannon in bpo-39156.)

11 Build Changes

- Added `--with-platlibdir` option to the `configure` script: name of the platform-specific library directory, stored in the new `sys.platlibdir` attribute. See `sys.platlibdir` attribute for more information. (Contributed by Jan Matějek, Matěj Cepl, Charalampos Stratakis and Victor Stinner in bpo-1294959.)
- The `COUNT_ALLOCS` special build macro has been removed. (Contributed by Victor Stinner in bpo-39489.)
- On non-Windows platforms, the `setenv()` and `unsetenv()` functions are now required to build Python. (Contributed by Victor Stinner in bpo-39395.)
- On non-Windows platforms, creating `bdist_wininst` installers is now officially unsupported. (See bpo-10945 for more details.)
- When building Python on macOS from source, `_tkinter` now links with non-system Tcl and Tk frameworks if they are installed in `/Library/Frameworks`, as had been the case on older releases of macOS. If a macOS SDK is explicitly configured, by using `--enable-universalsdk=` or `-isysroot`, only the SDK itself is searched. The default behavior can still be overridden with `--with-tcltk-includes` and `--with-tcltk-libs`. (Contributed by Ned Deily in bpo-34956.)
- Python can now be built for Windows 10 ARM64. (Contributed by Steve Dower in bpo-33125.)
- Some individual tests are now skipped when `--pgo` is used. The tests in question increased the PGO task time significantly and likely didn't help improve optimization of the final executable. This speeds up the task by a factor of about 15x. Running the full unit test suite is slow. This change may result in a slightly less optimized build since not as many code branches will be executed. If you are willing to wait for the much slower build, the old behavior can be restored using `./configure [..] PROFILE_TASK="-m test --pgo-extended"`. We make no guarantees as to which PGO task set produces a faster build. Users who care should run their own relevant benchmarks as results can depend on the environment, workload, and compiler tool chain. (See bpo-36044 and bpo-37707 for more details.)

12 C API Changes

12.1 New Features

- **PEP 573**: Added `PyType_FromModuleAndSpec()` to associate a module with a class; `PyType_GetModule()` and `PyType_GetModuleState()` to retrieve the module and its state; and `PyCMethod` and `METH_METHOD` to allow a method to access the class it was defined in. (Contributed by Marcel Plch and Petr Viktorin in bpo-38787.)
- Added `PyFrame_GetCode()` function: get a frame code. Added `PyFrame_GetBack()` function: get the frame next outer frame. (Contributed by Victor Stinner in bpo-40421.)
- Added `PyFrame_GetLineNumber()` to the limited C API. (Contributed by Victor Stinner in bpo-40421.)
- Added `PyThreadState_GetInterpreter()` and `PyInterpreterState_Get()` functions to get the interpreter. Added `PyThreadState_GetFrame()` function to get the current frame of a Python thread state. Added `PyThreadState_GetID()` function: get the unique identifier of a Python thread state. (Contributed by Victor Stinner in bpo-39947.)
- Added a new public `PyObject_CallNoArgs()` function to the C API, which calls a callable Python object without any arguments. It is the most efficient way to call a callable Python object without any argument. (Contributed by Victor Stinner in bpo-37194.)
- Changes in the limited C API (if `Py_LIMITED_API` macro is defined):
  - Provide `Py_EnterRecursiveCall()` and `Py_LeaveRecursiveCall()` as regular functions for the limited API. Previously, there were defined as macros, but these macros didn't compile with the limited C API which cannot access `PyThreadState.recursion_depth` field (the structure is opaque in the limited C API).

  - `PyObject_INIT()` and `PyObject_INIT_VAR()` become regular "opaque" function to hide implementation details.

  (Contributed by Victor Stinner in bpo-38644 and bpo-39542.)

- The `PyModule_AddType()` function is added to help adding a type to a module. (Contributed by Dong-hee Na in bpo-40024.)
- Added the functions `PyObject_GC_IsTracked()` and `PyObject_GC_IsFinalized()` to the public API to allow to query if Python objects are being currently tracked or have been already finalized by the garbage collector respectively. (Contributed by Pablo Galindo Salgado in bpo-40241.)
- Added `_PyObject_FunctionStr()` to get a user-friendly string representation of a function-like object. (Patch by Jeroen Demeyer in bpo-37645.)
- Added `PyObject_CallOneArg()` for calling an object with one positional argument (Patch by Jeroen Demeyer in bpo-37483.)

12.2 Porting to Python 3.9

- `PyInterpreterState.eval_frame` (**PEP 523**) now requires a new mandatory *tstate* parameter (`PyThreadState*`). (Contributed by Victor Stinner in bpo-38500.)
- Extension modules: `m_traverse`, `m_clear` and `m_free` functions of `PyModuleDef` are no longer called if the module state was requested but is not allocated yet. This is the case immediately after the module is created and before the module is executed (`Py_mod_exec` function). More precisely, these functions are not called if `m_size` is greater than 0 and the module state (as returned by `PyModule_GetState()`) is `NULL`.

  Extension modules without module state (`m_size <= 0`) are not affected.
- If `Py_AddPendingCall()` is called in a subinterpreter, the function is now scheduled to be called from the subinterpreter, rather than being called from the main interpreter. Each subinterpreter now has its own list of scheduled calls. (Contributed by Victor Stinner in bpo-39984.)
- The Windows registry is no longer used to initialize `sys.path` when the `-E` option is used (if `PyConfig.use_environment` is set to `0`). This is significant when embedding Python on Windows. (Contributed by Zackery Spytz in bpo-8901.)
- The global variable `PyStructSequence_UnnamedField` is now a constant and refers to a constant string. (Contributed by Serhiy Storchaka in bpo-38650.)
- The `PyGC_Head` structure is now opaque. It is only defined in the internal C API (`pycore_gc.h`). (Contributed by Victor Stinner in bpo-40241.)
- The `Py_UNICODE_COPY`, `Py_UNICODE_FILL`, `PyUnicode_WSTR_LENGTH`, `PyUnicode_FromUnicode()`, `PyUnicode_AsUnicode()`, `_PyUnicode_AsUnicode`, and `PyUnicode_AsUnicodeAndSize()` are marked as deprecated in C. They have been deprecated by **PEP 393** since Python 3.3. (Contributed by Inada Naoki in bpo-36346.)
- The `Py_FatalError()` function is replaced with a macro which logs automatically the name of the current function, unless the `Py_LIMITED_API` macro is defined. (Contributed by Victor Stinner in bpo-39882.)
- The vectorcall protocol now requires that the caller passes only strings as keyword names. (See bpo-37540 for more information.)
- Implementation details of a number of macros and functions are now hidden:
  - `PyObject_IS_GC()` macro was converted to a function.
  - The `PyObject_NEW()` macro becomes an alias to the `PyObject_New()` macro, and the `PyObject_NEW_VAR()` macro becomes an alias to the `PyObject_NewVar()` macro. They no longer access directly the `PyTypeObject.tp_basicsize` member.
  - `PyType_HasFeature()` now always calls `PyType_GetFlags()`. Previously, it accessed directly the `PyTypeObject.tp_flags` member when the limited C API was not used.

- `PyObject_GET_WEAKREFS_LISTPTR()` macro was converted to a function: the macro accessed directly the `PyTypeObject.tp_weaklistoffset` member.
- `PyObject_CheckBuffer()` macro was converted to a function: the macro accessed directly the `PyTypeObject.tp_as_buffer` member.
- `PyIndex_Check()` is now always declared as an opaque function to hide implementation details: removed the `PyIndex_Check()` macro. The macro accessed directly the `PyTypeObject.tp_as_number` member.

(See bpo-40170 for more details.)

12.3 Removed

- Excluded `PyFPE_START_PROTECT()` and `PyFPE_END_PROTECT()` macros of `pyfpe.h` from the limited C API. (Contributed by Victor Stinner in bpo-38835.)
- The `tp_print` slot of PyTypeObject has been removed. It was used for printing objects to files in Python 2.7 and before. Since Python 3.0, it has been ignored and unused. (Contributed by Jeroen Demeyer in bpo-36974.)
- Changes in the limited C API (if `Py_LIMITED_API` macro is defined):
  - Excluded the following functions from the limited C API:
    * `PyThreadState_DeleteCurrent()` (Contributed by Joannah Nanjekye in bpo-37878.)
    * `_Py_CheckRecursionLimit`
    * `_Py_NewReference()`
    * `_Py_ForgetReference()`
    * `_PyTraceMalloc_NewReference()`
    * `_Py_GetRefTotal()`
    * The trashcan mechanism which never worked in the limited C API.
    * `PyTrash_UNWIND_LEVEL`
    * `Py_TRASHCAN_BEGIN_CONDITION`
    * `Py_TRASHCAN_BEGIN`
    * `Py_TRASHCAN_END`
    * `Py_TRASHCAN_SAFE_BEGIN`
    * `Py_TRASHCAN_SAFE_END`
  - Moved following functions and definitions to the internal C API:
    * `_PyDebug_PrintTotalRefs()`
    * `_Py_PrintReferences()`
    * `_Py_PrintReferenceAddresses()`
    * `_Py_tracemalloc_config`
    * `_Py_AddToAllObjects()` (specific to `Py_TRACE_REFS` build)

  (Contributed by Victor Stinner in bpo-38644 and bpo-39542.)
- Removed `_PyRuntime.getframe` hook and removed `_PyThreadState_GetFrame` macro which was an alias to `_PyRuntime.getframe`. They were only exposed by the internal C API. Removed also `PyThreadFrameGetter` type. (Contributed by Victor Stinner in bpo-39946.)
- Removed the following functions from the C API. Call `PyGC_Collect()` explicitly to clear all free lists. (Contributed by Inada Naoki and Victor Stinner in bpo-37340, bpo-38896 and bpo-40428.)
  - `PyAsyncGen_ClearFreeLists()`

  - `PyContext_ClearFreeList()`
  - `PyDict_ClearFreeList()`
  - `PyFloat_ClearFreeList()`
  - `PyFrame_ClearFreeList()`
  - `PyList_ClearFreeList()`
  - `PyMethod_ClearFreeList()` and `PyCFunction_ClearFreeList()`: the free lists of bound method objects have been removed.
  - `PySet_ClearFreeList()`: the set free list has been removed in Python 3.4.
  - `PyTuple_ClearFreeList()`
  - `PyUnicode_ClearFreeList()`: the Unicode free list has been removed in Python 3.3.

- Removed `_PyUnicode_ClearStaticStrings()` function. (Contributed by Victor Stinner in bpo-39465.)
- Removed `Py_UNICODE_MATCH`. It has been deprecated by **PEP 393**, and broken since Python 3.3. The `PyUnicode_Tailmatch()` function can be used instead. (Contributed by Inada Naoki in bpo-36346.)
- Cleaned header files of interfaces defined but with no implementation. The public API symbols being removed are: `_PyBytes_InsertThousandsGroupingLocale`, `_PyBytes_InsertThousandsGrouping`, `_Py_InitializeFromArgs`, `_Py_InitializeFromWideArgs`, `_PyFloat_Repr`, `_PyFloat_Digits`, `_PyFloat_DigitsInit`, `PyFrame_ExtendStack`, `_PyAIterWrapper_Type`, `PyNullImporter_Type`, `PyCmpWrapper_Type`, `PySortWrapper_Type`, `PyNoArgsFunction`. (Contributed by Pablo Galindo Salgado in bpo-39372.)

13 Notable changes in Python 3.9.1

13.1 typing

The behavior of `typing.Literal` was changed to conform with **PEP 586** and to match the behavior of static type checkers specified in the PEP.

1. `Literal` now de-duplicates parameters.
2. Equality comparisons between `Literal` objects are now order independent.
3. `Literal` comparisons now respect types. For example, `Literal[0] == Literal[False]` previously evaluated to `True`. It is now `False`. To support this change, the internally used type cache now supports differentiating types.
4. `Literal` objects will now raise a `TypeError` exception during equality comparisons if one of their parameters are not immutable. Note that declaring `Literal` with mutable parameters will not throw an error:

```
>>> from typing import Literal
>>> Literal[{0}]
>>> Literal[{0}] == Literal[{False}]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unhashable type: 'set'
```

(Contributed by Yurii Karabas in bpo-42345.)

13.2 macOS 11.0 (Big Sur) and Apple Silicon Mac support

As of 3.9.1, Python now fully supports building and running on macOS 11.0 (Big Sur) and on Apple Silicon Macs (based on the `ARM64` architecture). A new universal build variant, `universal2`, is now available to natively support both `ARM64` and `Intel 64` in one set of executables. Binaries can also now be built on current versions of macOS to be deployed on a range of older macOS versions (tested to 10.9) while making some newer OS functions and options conditionally available based on the operating system version in use at runtime ("weaklinking").

(Contributed by Ronald Oussoren and Lawrence D'Anna in bpo-41100.)

14 Notable changes in Python 3.9.2

14.1 collections.abc

`collections.abc.Callable` generic now flattens type parameters, similar to what `typing.Callable` currently does. This means that `collections.abc.Callable[[int, str], str]` will have `__args__` of `(int, str, str)`; previously this was `([int, str], str)`. To allow this change, `types.GenericAlias` can now be subclassed, and a subclass will be returned when subscripting the `collections.abc.Callable` type. Code which accesses the arguments via `typing.get_args()` or `__args__` need to account for this change. A `DeprecationWarning` may be emitted for invalid forms of parameterizing `collections.abc.Callable` which may have passed silently in Python 3.9.1. This `DeprecationWarning` will become a `TypeError` in Python 3.10. (Contributed by Ken Jin in bpo-42195.)

Index

E

P

R