What's New in Python

Release 3.8.0b1

A. M. Kuchling

July 04, 2019

Python Software Foundation
Email: docs@python.org

Contents

This article explains the new features in Python 3.8, compared to 3.7.

For full details, see the changelog.

Note: Prerelease users should be aware that this document is currently in draft form. It will be updated substantially as Python 3.8 moves towards release, so it's worth checking back even after reading earlier versions.

Some notable items not yet covered here:

- **PEP 578** - Runtime audit hooks for potentially sensitive operations
- `python -m asyncio` runs a natively async REPL

- …

---

1 Summary – Release highlights

2 New Features

2.1 Assignment expressions

There is new syntax (the “walrus operator”, `:=`) to assign values to variables as part of an expression. Example:

```
if (n := len(a)) > 10:
    print(f"List is too long ({n} elements, expected <= 10)")
```

See **PEP 572** for a full description.

(Contributed by Emily Morehouse in bpo-35224.)

2.2 Positional-only parameters

There is new syntax (`/`) to indicate that some function parameters must be specified positionally (i.e., cannot be used as keyword arguments). This is the same notation as shown by `help()` for functions implemented in C (produced by Larry Hastings’ “Argument Clinic” tool). Example:

```
def pow(x, y, z=None, /):
    r = x**y
    if z is not None:
        r %= z
    return r
```

Now `pow(2, 10)` and `pow(2, 10, 17)` are valid calls, but `pow(x=2, y=10)` and `pow(2, 10, z=17)` are invalid.

See **PEP 570** for a full description.

(Contributed by Pablo Galindo in bpo-36540.)

2.3 Parallel filesystem cache for compiled bytecode files

The new `PYTHONPYCACHEPREFIX` setting (also available as `-X pycache_prefix`) configures the implicit bytecode cache to use a separate parallel filesystem tree, rather than the default `__pycache__` subdirectories within each source directory.

The location of the cache is reported in `sys.pycache_prefix` (`None` indicates the default location in `__pycache__` subdirectories).

(Contributed by Carl Meyer in bpo-33499.)

2.4 Debug build uses the same ABI as release build

Python now uses the same ABI whether it built in release or debug mode. On Unix, when Python is built in debug mode, it is now possible to load C extensions built in release mode and C extensions built using the stable ABI.

Release builds and debug builds are now ABI compatible: defining the `Py_DEBUG` macro no longer implies the `Py_TRACE_REFS` macro, which introduces the only ABI incompatibility. The `Py_TRACE_REFS` macro, which adds the `sys.getobjects()` function and the `PYTHONDUMPREFS` environment variable, can be set using the new `./configure --with-trace-refs` build option. (Contributed by Victor Stinner in bpo-36465.)

On Unix, C extensions are no longer linked to libpython except on Android and Cygwin. It is now possible for a statically linked Python to load a C extension built using a shared library Python. (Contributed by Victor Stinner in bpo-21536.)

On Unix, when Python is built in debug mode, import now also looks for C extensions compiled in release mode and for C extensions compiled with the stable ABI. (Contributed by Victor Stinner in bpo-36722.)

To embed Python into an application, a new `--embed` option must be passed to `python3-config --libs --embed` to get `-lpython3.8` (link the application to libpython). To support both 3.8 and older, try `python3-config --libs --embed` first and fallback to `python3-config --libs` (without `--embed`) if the previous command fails.

Add a pkg-config `python-3.8-embed` module to embed Python into an application: `pkg-config python-3.8-embed --libs` includes `-lpython3.8`. To support both 3.8 and older, try `pkg-config python-X.Y-embed --libs` first and fallback to `pkg-config python-X.Y --libs` (without `--embed`) if the previous command fails (replace `X.Y` with the Python version).

On the other hand, `pkg-config python3.8 --libs` no longer contains `-lpython3.8`. C extensions must not be linked to libpython (except on Android and Cygwin, whose cases are handled by the script); this change is backward incompatible on purpose. (Contributed by Victor Stinner in bpo-36721.)

2.5 f-strings now support = for quick and easy debugging

Add = specifier to f-strings. `f'{expr=}'` expands to the text of the expression, an equal sign, then the repr of the evaluated expression. So:

```
x = 3
print(f'{x*9 + 15=}')
```

Would print `x*9 + 15=42`.

(Contributed by Eric V. Smith and Larry Hastings in bpo-36817.)

2.6 PEP 587: Python Initialization Configuration

The **PEP 587** adds a new C API to configure the Python Initialization providing finer control on the whole configuration and better error reporting.

New structures:

- `PyConfig`
- `PyPreConfig`
- `PyStatus`
- `PyWideStringList`

New functions:

- `PyConfig_Clear()`
- `PyConfig_InitIsolatedConfig()`
- `PyConfig_InitPythonConfig()`
- `PyConfig_Read()`
- `PyConfig_SetArgv()`
- `PyConfig_SetBytesArgv()`
- `PyConfig_SetBytesString()`
- `PyConfig_SetString()`
- `PyPreConfig_InitIsolatedConfig()`
- `PyPreConfig_InitPythonConfig()`
- `PyStatus_Error()`
- `PyStatus_Exception()`
- `PyStatus_Exit()`
- `PyStatus_IsError()`
- `PyStatus_IsExit()`
- `PyStatus_NoMemory()`
- `PyStatus_Ok()`
- `PyWideStringList_Append()`
- `PyWideStringList_Insert()`
- `Py_BytesMain()`
- `Py_ExitStatusException()`
- `Py_InitializeFromConfig()`
- `Py_PreInitialize()`
- `Py_PreInitializeFromArgs()`
- `Py_PreInitializeFromBytesArgs()`
- `Py_RunMain()`

This PEP also adds `_PyRuntimeState.preconfig` (`PyPreConfig` type) and `PyInterpreterState.config` (`PyConfig` type) fields to these internal structures. `PyInterpreterState.config` becomes the new reference configuration, replacing global configuration variables and other private variables.

See Python Initialization Configuration for the documentation.

See **PEP 587** for a full description.

(Contributed by Victor Stinner in bpo-36763.)

2.7 Vectorcall: a fast calling protocol for CPython

The "vectorcall" protocol is added to the Python/C API. It is meant to formalize existing optimizations which were already done for various classes. Any extension type implementing a callable can use this protocol.

This is currently provisional, the aim is to make it fully public in Python 3.9.

See **PEP 590** for a full description.

(Contributed by Jeroen Demeyer and Mark Shannon in bpo-36974.)

2.8 Pickle protocol 5 with out-of-band data buffers

When `pickle` is used to transfer large data between Python processes in order to take advantage of multi-core or multi-machine processing, it is important to optimize the transfer by reducing memory copies, and possibly by applying custom techniques such as data-dependent compression.

The `pickle` protocol 5 introduces support for out-of-band buffers where **PEP 3118**-compatible data can be transmitted separately from the main pickle stream, at the discretion of the communication layer.

See **PEP 574** for a full description.

(Contributed by Antoine Pitrou in bpo-36785.)

3 Other Language Changes

- A `continue` statement was illegal in the `finally` clause due to a problem with the implementation. In Python 3.8 this restriction was lifted. (Contributed by Serhiy Storchaka in bpo-32489.)
- The `int` type now has a new `as_integer_ratio()` method compatible with the existing `float.as_integer_ratio()` method. (Contributed by Lisa Roach in bpo-33073.)
- Constructors of `int`, `float` and `complex` will now use the `__index__()` special method, if available and the corresponding method `__int__()`, `__float__()` or `__complex__()` is not available. (Contributed by Serhiy Storchaka in bpo-20092.)
- Added support of `\N{name}` escapes in `regular expressions`. (Contributed by Jonathan Eunice and Serhiy Storchaka in bpo-30688.)
- Dict and dictviews are now iterable in reversed insertion order using `reversed()`. (Contributed by Rémi Lapeyre in bpo-33462.)
- The syntax allowed for keyword names in function calls was further restricted. In particular, `f((keyword)=arg)` is no longer allowed. It was never intended to permit more than a bare name on the left-hand side of a keyword argument assignment term. See bpo-34641.
- Iterable unpacking is now allowed without parentheses in `yield` and `return` statements. (Contributed by David Cuthbert and Jordan Chapman in bpo-32117.)
- A backslash-character pair that is not a valid escape sequence generates a `DeprecationWarning` since Python 3.6. In Python 3.8 it generates a `SyntaxWarning` instead. (Contributed by Serhiy Storchaka in bpo-32912.)
- The compiler now produces a `SyntaxWarning` in some cases when a comma is missed before tuple or list. For example:

```
data = [
    (1, 2, 3) # oops, missing comma!
    (4, 5, 6)
]
```

  (Contributed by Serhiy Storchaka in bpo-15248.)
- Arithmetic operations between subclasses of `datetime.date` or `datetime.datetime` and `datetime.timedelta` objects now return an instance of the subclass, rather than the base class. This also affects the return

type of operations whose implementation (directly or indirectly) uses `datetime.timedelta` arithmetic, such as `datetime.datetime.astimezone()`. (Contributed by Paul Ganssle in bpo-32417.)

- When the Python interpreter is interrupted by Ctrl-C (SIGINT) and the resulting `KeyboardInterrupt` exception is not caught, the Python process now exits via a SIGINT signal or with the correct exit code such that the calling process can detect that it died due to a Ctrl-C. Shells on POSIX and Windows use this to properly terminate scripts in interactive sessions. (Contributed by Google via Gregory P. Smith in bpo-1054041.)
- Added new `replace()` method to the code type (`types.CodeType`). (Contributed by Victor Stinner in bpo-37032.)
- For integers, the three-argument form of the `pow()` function now permits the exponent to be negative in the case where the base is relatively prime to the modulus. It then computes a modular inverse to the base when the exponent is `-1`, and a suitable power of that inverse for other negative exponents. (Contributed by Mark Dickinson in bpo-36027.)
- When dictionary comprehensions are evaluated, the key is now evaluated before the value, as proposed by **PEP 572**.

4 New Modules

- None yet.

5 Improved Modules

5.1 ast

AST nodes now have `end_lineno` and `end_col_offset` attributes, which give the precise location of the end of the node. (This only applies to nodes that have `lineno` and `col_offset` attributes.)

The `ast.parse()` function has some new flags:

- `type_comments=True` causes it to return the text of **PEP 484** and **PEP 526** type comments associated with certain AST nodes;
- `mode='func_type'` can be used to parse **PEP 484** "signature type comments" (returned for function definition AST nodes);
- `feature_version=(3, N)` allows specifying an earlier Python 3 version. (For example, `feature_version=(3, 4)` will treat `async` and `await` as non-reserved words.)

New function `ast.get_source_segment()` returns the source code for a specific AST node.

5.2 asyncio

On Windows, the default event loop is now `ProactorEventLoop`. (Contributed by Victor Stinner in bpo-34687.)

`ProactorEventLoop` now also supports UDP. (Contributed by Adam Meily and Andrew Svetlov in bpo-29883.)

`ProactorEventLoop` can now be interrupted by `KeyboardInterrupt` ("CTRL+C"). (Contributed by Vladimir Matveev in bpo-23057.)

5.3 builtins

The `compile()` built-in has been improved to accept the `ast.PyCF_ALLOW_TOP_LEVEL_AWAIT` flag. With this new flag passed, `compile()` will allow top-level `await`, `async for` and `async with` constructs that are usually considered invalid syntax. Asynchronous code object marked with the `CO_COROUTINE` flag may then be returned.

(Contributed by Matthias Bussonnier in bpo-34616)

5.4 collections

The `_asdict()` method for `collections.namedtuple()` now returns a `dict` instead of a `collections.OrderedDict`. This works because regular dicts have guaranteed ordering since Python 3.7. If the extra features of `OrderedDict` are required, the suggested remediation is to cast the result to the desired type: `OrderedDict(nt._asdict())`. (Contributed by Raymond Hettinger in bpo-35864.)

5.5 ctypes

On Windows, `CDLL` and subclasses now accept a *winmode* parameter to specify flags for the underlying `LoadLibraryEx` call. The default flags are set to only load DLL dependencies from trusted locations, including the path where the DLL is stored (if a full or partial path is used to load the initial DLL) and paths added by `add_dll_directory()`.

5.6 functools

`functools.lru_cache()` can now be used as a straight decorator rather than as a function returning a decorator. So both of these are now supported:

```
@lru_cache
def f(x):
    ...

@lru_cache(maxsize=256)
def f(x):
    ...
```

(Contributed by Raymond Hettinger in bpo-36772.)

5.7 datetime

Added new alternate constructors `datetime.date.fromisocalendar()` and `datetime.datetime.fromisocalendar()`, which construct `date` and `datetime` objects respectively from ISO year, week number and weekday; these are the inverse of each class's `isocalendar` method. (Contributed by Paul Ganssle in bpo-36004.)

5.8 gettext

Added `pgettext()` and its variants. (Contributed by Franz Glasner, Éric Araujo, and Cheryl Sabella in bpo-2504.)

5.9 inspect

The `inspect.getdoc()` function can now find docstrings for `__slots__` if that attribute is a `dict` where the values are docstrings. This provides documentation options similar to what we already have for `property()`, `classmethod()`, and `staticmethod()`:

```
class AudioClip:
    __slots__ = {'bit_rate': 'expressed in kilohertz to one decimal place',
                 'duration': 'in seconds, rounded up to an integer'}
    def __init__(self, bit_rate, duration):
        self.bit_rate = round(bit_rate / 1000.0, 1)
        self.duration = ceil(duration)
```

5.10 io

In development mode (`-X env`) and in debug build, the `io.IOBase` finalizer now logs the exception if the `close()` method fails. The exception is ignored silently by default in release build. (Contributed by Victor Stinner in bpo-18748.)

5.11 gc

`get_objects()` can now receive an optional *generation* parameter indicating a generation to get objects from. Contributed in bpo-36016 by Pablo Galindo.

5.12 gzip

Added the *mtime* parameter to `gzip.compress()` for reproducible output. (Contributed by Guo Ci Teo in bpo-34898.)

A `BadGzipFile` exception is now raised instead of `OSError` for certain types of invalid or corrupt gzip files. (Contributed by Filip Gruszczyński, Michele Orrù, and Zackery Spytz in bpo-6584.)

5.13 idlelib and IDLE

Output over N lines (50 by default) is squeezed down to a button. N can be changed in the PyShell section of the General page of the Settings dialog. Fewer, but possibly extra long, lines can be squeezed by right clicking on the output. Squeezed output can be expanded in place by double-clicking the button or into the clipboard or a separate window by right-clicking the button. (Contributed by Tal Einat in bpo-1529353.)

The changes above have been backported to 3.7 maintenance releases.

5.14 json.tool

Add option `--json-lines` to parse every input line as separate JSON object. (Contributed by Weipeng Hong in bpo-31553.)

5.15 math

Added new function `math.dist()` for computing Euclidean distance between two points. (Contributed by Raymond Hettinger in bpo-33089.)

Expanded the `math.hypot()` function to handle multiple dimensions. Formerly, it only supported the 2-D case. (Contributed by Raymond Hettinger in bpo-33089.)

Added new function, `math.prod()`, as analogous function to `sum()` that returns the product of a 'start' value (default: 1) times an iterable of numbers. (Contributed by Pablo Galindo in bpo-35606)

Added new function `math.isqrt()` for computing integer square roots. (Contributed by Mark Dickinson in bpo-36887.)

The function `math.factorial()` no longer accepts arguments that are not int-like. (Contributed by Pablo Galindo in bpo-33083.)

5.16 mmap

The `mmap.mmap` class now has an `madvise()` method to access the `madvise()` system call. (Contributed by Zackery Spytz in bpo-32941.)

5.17 multiprocessing

Added new `multiprocessing.shared_memory` module. (Contributed Davin Potts in bpo-35813.)

On macOS, the *spawn* start method is now used by default. (Contributed by Victor Stinner in bpo-33725.)

5.18 os

Added new function `add_dll_directory()` on Windows for providing additional search paths for native dependencies when importing extension modules or loading DLLs using `ctypes`.

A new `os.memfd_create()` function was added to wrap the `memfd_create()` syscall. (Contributed by Zackery Spytz and Christian Heimes in bpo-26836.)

5.19 os.path

`os.path` functions that return a boolean result like `exists()`, `lexists()`, `isdir()`, `isfile()`, `islink()`, and `ismount()` now return `False` instead of raising `ValueError` or its subclasses `UnicodeEncodeError` and `UnicodeDecodeError` for paths that contain characters or bytes unrepresentable at the OS level. (Contributed by Serhiy Storchaka in bpo-33721.)

`expanduser()` on Windows now prefers the `USERPROFILE` environment variable and does not use `HOME`, which is not normally set for regular user accounts.

5.20 ncurses

Added a new variable holding structured version information for the underlying ncurses library: `ncurses_version`. (Contributed by Serhiy Storchaka in bpo-31680.)

5.21 pathlib

`pathlib.Path` methods that return a boolean result like `exists()`, `is_dir()`, `is_file()`, `is_mount()`, `is_symlink()`, `is_block_device()`, `is_char_device()`, `is_fifo()`, `is_socket()` now return

`False` instead of raising `ValueError` or its subclass `UnicodeEncodeError` for paths that contain characters unrepresentable at the OS level. (Contributed by Serhiy Storchaka in bpo-33721.)

Added `pathlib.Path.link_to()` which creates a hard link pointing to a path. (Contributed by Joannah Nanjekye in bpo-26978)

5.22 pickle

Reduction methods can now include a 6th item in the tuple they return. This item should specify a custom state-setting method that's called instead of the regular `__setstate__` method. (Contributed by Pierre Glaser and Olivier Grisel in bpo-35900)

`pickle` extensions subclassing the C-optimized `Pickler` can now override the pickling logic of functions and classes by defining the special `reducer_override()` method. (Contributed by Pierre Glaser and Olivier Grisel in bpo-35900)

5.23 plistlib

Added new `plistlib.UID` and enabled support for reading and writing NSKeyedArchiver-encoded binary plists. (Contributed by Jon Janzen in bpo-26707.)

5.24 py_compile

`py_compile.compile()` now supports silent mode. (Contributed by Joannah Nanjekye in bpo-22640.)

5.25 socket

Added `create_server()` and `has_dualstack_ipv6()` convenience functions to automate the necessary tasks usually involved when creating a server socket, including accepting both IPv4 and IPv6 connections on the same socket. (Contributed by Giampaolo Rodola in bpo-17561.)

The `socket.if_nameindex()`, `socket.if_nametoindex()`, and `socket.if_indextoname()` functions have been implemented on Windows. (Contributed by Zackery Spytz in bpo-37007.)

5.26 shlex

The new `shlex.join()` function acts as the inverse of `shlex.split()`. (Contributed by Bo Bayles in bpo-32102.)

5.27 shutil

`shutil.copytree()` now accepts a new `dirs_exist_ok` keyword argument. (Contributed by Josh Bronson in bpo-20849.)

`shutil.make_archive()` now defaults to the modern pax (POSIX.1-2001) format for new archives to improve portability and standards conformance, inherited from the corresponding change to the `tarfile` module. (Contributed by C.A.M. Gerlach in bpo-30661.)

5.28 ssl

Added `SSLContext.post_handshake_auth` to enable and `ssl.SSLSocket.verify_client_post_handshake()` to initiate TLS 1.3 post-handshake authentication. (Contributed by Christian Heimes in bpo-34670.)

5.29 statistics

Added `statistics.fmean()` as a faster, floating point variant of `statistics.mean()`. (Contributed by Raymond Hettinger and Steven D'Aprano in bpo-35904.)

Added `statistics.geometric_mean()` (Contributed by Raymond Hettinger in bpo-27181.)

Added `statistics.multimode()` that returns a list of the most common values. (Contributed by Raymond Hettinger in bpo-35892.)

Added `statistics.quantiles()` that divides data or a distribution in to equiprobable intervals (e.g. quartiles, deciles, or percentiles). (Contributed by Raymond Hettinger in bpo-36546.)

Added `statistics.NormalDist`, a tool for creating and manipulating normal distributions of a random variable. (Contributed by Raymond Hettinger in bpo-36018.)

```
>>> temperature_feb = NormalDist.from_samples([4, 12, -3, 2, 7, 14])
>>> temperature_feb.mean
6.0
>>> temperature_feb.stdev
6.356099432828281

>>> temperature_feb.cdf(3)           # Chance of being under 3 degrees
0.3184678262814532
>>> # Relative chance of being 7 degrees versus 10 degrees
>>> temperature_feb.pdf(7) / temperature_feb.pdf(10)
1.2039930378537762

>>> el_niño = NormalDist(4, 2.5)
>>> temperature_feb += el_niño       # Add in a climate effect
>>> temperature_feb
NormalDist(mu=10.0, sigma=6.830080526611674)

>>> temperature_feb * (9/5) + 32     # Convert to Fahrenheit
NormalDist(mu=50.0, sigma=12.294144947901014)
>>> temperature_feb.samples(3)       # Generate random samples
[7.672102882379219, 12.000027119750287, 4.647488369766392]
```

5.30 sys

Add new `sys.unraisablehook()` function which can be overridden to control how "unraisable exceptions" are handled. It is called when an exception has occurred but there is no way for Python to handle it. For example, when a destructor raises an exception or during garbage collection (`gc.collect()`). (Contributed by Victor Stinner in bpo-36829.)

5.31 tarfile

The `tarfile` module now defaults to the modern pax (POSIX.1-2001) format for new archives, instead of the previous GNU-specific one. This improves cross-platform portability with a consistent encoding (UTF-8) in a standardized and

extensible format, and offers several other benefits. (Contributed by C.A.M. Gerlach in bpo-36268.)

5.32 threading

Add a new `threading.excepthook()` function which handles uncaught `threading.Thread.run()` exception. It can be overridden to control how uncaught `threading.Thread.run()` exceptions are handled. (Contributed by Victor Stinner in bpo-1230540.)

5.33 tokenize

The `tokenize` module now implicitly emits a `NEWLINE` token when provided with input that does not have a trailing new line. This behavior now matches what the C tokenizer does internally. (Contributed by Ammar Askar in bpo-33899.)

5.34 tkinter

Added methods `selection_from()`, `selection_present()`, `selection_range()` and `selection_to()` in the `tkinter.Spinbox` class. (Contributed by Juliette Monsel in bpo-34829.)

Added method `moveto()` in the `tkinter.Canvas` class. (Contributed by Juliette Monsel in bpo-23831.)

The `tkinter.PhotoImage` class now has `transparency_get()` and `transparency_set()` methods. (Contributed by Zackery Spytz in bpo-25451.)

5.35 time

Added new clock `CLOCK_UPTIME_RAW` for macOS 10.12. (Contributed by Joannah Nanjekye in bpo-35702.)

5.36 typing

The `typing` module incorporates several new features:

- Protocol definitions. See **PEP 544**, `typing.Protocol` and `typing.runtime_checkable()`. Simple ABCs like `typing.SupportsInt` are now `Protocol` subclasses.
- A dictionary type with per-key types. See **PEP 589** and `typing.TypedDict`.
- Literal types. See **PEP 586** and `typing.Literal`.
- "Final" variables, functions, methods and classes. See **PEP 591**, `typing.Final` and `typing.final()`.
- New protocol class `typing.SupportsIndex`.
- New functions `typing.get_origin()` and `typing.get_args()`.

5.37 unicodedata

- The `unicodedata` module has been upgraded to use the Unicode 12.1.0 release.
- New function `is_normalized()` can be used to verify a string is in a specific normal form. (Contributed by Max Belanger and David Euresti in bpo-32285).

5.38 unittest

- Added `AsyncMock` to support an asynchronous version of `Mock`. Appropriate new assert functions for testing have been added as well. (Contributed by Lisa Roach in bpo-26467).
- Added `addModuleCleanup()` and `addClassCleanup()` to unittest to support cleanups for `setUpModule()` and `setUpClass()`. (Contributed by Lisa Roach in bpo-24412.)
- Several mock assert functions now also print a list of actual calls upon failure. (Contributed by Petter Strandmark in bpo-35047.)

5.39 venv

- `venv` now includes an `Activate.ps1` script on all platforms for activating virtual environments under PowerShell Core 6.1. (Contributed by Brett Cannon in bpo-32718.)

5.40 weakref

- The proxy objects returned by `weakref.proxy()` now support the matrix multiplication operators `@` and `@=` in addition to the other numeric operators. (Contributed by Mark Dickinson in bpo-36669.)

5.41 xml

- As mitigation against DTD and external entity retrieval, the `xml.dom.minidom` and `xml.sax` modules no longer process external entities by default. (Contributed by Christian Heimes in bpo-17239.)
- The `.find*()` methods in the `xml.etree.ElementTree` module support wildcard searches like `{*}tag` which ignores the namespace and `{namespace}*` which returns all tags in the given namespace. (Contributed by Stefan Behnel in bpo-28238.)
- The `xml.etree.ElementTree` module provides a new function `–xml.etree.ElementTree.canonicalize()` that implements C14N 2.0. (Contributed by Stefan Behnel in bpo-13611.)
- The target object of `xml.etree.ElementTree.XMLParser` can receive namespace declaration events through the new callback methods `start_ns()` and `end_ns()`. Additionally, the `xml.etree.ElementTree.TreeBuilder` target can be configured to process events about comments and processing instructions to include them in the generated tree. (Contributed by Stefan Behnel in bpo-36676 and bpo-36673.)

6 Optimizations

- The `subprocess` module can now use the `os.posix_spawn()` function in some cases for better performance. Currently, it is only used on macOS and Linux (using glibc 2.24 or newer) if all these conditions are met:
  - *close_fds* is false;
  - *preexec_fn*, *pass_fds*, *cwd* and *start_new_session* parameters are not set;
  - the *executable* path contains a directory.

  (Contributed by Joannah Nanjekye and Victor Stinner in bpo-35537.)

- `shutil.copyfile()`, `shutil.copy()`, `shutil.copy2()`, `shutil.copytree()` and `shutil.move()` use platform-specific "fast-copy" syscalls on Linux and macOS in order to copy the file more efficiently. "fast-copy" means that the copying operation occurs within the kernel, avoiding the use of userspace buffers in Python as in "`outfd.write(infd.read())`". On Windows `shutil.copyfile()` uses a bigger default buffer size (1 MiB instead of 16 KiB) and a `memoryview()`-based variant of `shutil.copyfileobj()` is used. The speedup for copying a 512 MiB file within the same partition is about +26% on Linux, +50% on macOS and +40% on Windows. Also, much less CPU cycles are consumed. See shutil-platform-dependent-efficient-copy-operations section. (Contributed by Giampaolo Rodola' in bpo-33671.)
- `shutil.copytree()` uses `os.scandir()` function and all copy functions depending from it use cached `os.stat()` values. The speedup for copying a directory with 8000 files is around +9% on Linux, +20% on Windows and +30% on a Windows SMB share. Also the number of `os.stat()` syscalls is reduced by 38% making `shutil.copytree()` especially faster on network filesystems. (Contributed by Giampaolo Rodola' in bpo-33695.)
- The default protocol in the `pickle` module is now Protocol 4, first introduced in Python 3.4. It offers better performance and smaller size compared to Protocol 3 available since Python 3.0.
- Removed one `Py_ssize_t` member from `PyGC_Head`. All GC tracked objects (e.g. tuple, list, dict) size is reduced 4 or 8 bytes. (Contributed by Inada Naoki in bpo-33597)
- `uuid.UUID` now uses `__slots__` to reduce its memory footprint.
- Improved performance of `operator.itemgetter()` by 33%. Optimized argument handling and added a fast path for the common case of a single non-negative integer index into a tuple (which is the typical use case in the standard library). (Contributed by Raymond Hettinger in bpo-35664.)
- Sped-up field lookups in `collections.namedtuple()`. They are now more than two times faster, making them the fastest form of instance variable lookup in Python. (Contributed by Raymond Hettinger, Pablo Galindo, and Joe Jevnik, Serhiy Storchaka in bpo-32492.)
- The `list` constructor does not overallocate the internal item buffer if the input iterable has a known length (the input implements `__len__`). This makes the created list 12% smaller on average. (Contributed by Raymond Hettinger and Pablo Galindo in bpo-33234.)
- Doubled the speed of class variable writes. When a non-dunder attribute was updated, there was an unnecessary call to update slots. (Contributed by Stefan Behnel, Pablo Galindo Salgado, Raymond Hettinger, Neil Schemenauer, and Serhiy Storchaka in bpo-36012.)
- Reduced an overhead of converting arguments passed to many builtin functions and methods. This sped up calling some simple builtin functions and methods up to 20–50%. (Contributed by Serhiy Storchaka in bpo-23867, bpo-35582 and bpo-36127.)
- `LOAD_GLOBAL` instruction now uses new "per opcode cache" mechanism. It is about 40% faster now. (Contributed by Yury Selivanov and Inada Naoki in bpo-26219.)

7 Build and C API Changes

- Default `sys.abiflags` became an empty string: the `m` flag for pymalloc became useless (builds with and without pymalloc are ABI compatible) and so has been removed. (Contributed by Victor Stinner in bpo-36707.)

  Example of changes:

  - Only `python3.8` program is installed, `python3.8m` program is gone.
  - Only `python3.8-config` script is installed, `python3.8m-config` script is gone.
  - The `m` flag has been removed from the suffix of dynamic library filenames: extension modules in the standard library as well as those produced and installed by third-party packages, like those downloaded from PyPI.

On Linux, for example, the Python 3.7 suffix `.cpython-37m-x86_64-linux-gnu.so` became `.cpython-38-x86_64-linux-gnu.so` in Python 3.8.

- The header files have been reorganized to better separate the different kinds of APIs:
  - `Include/*.h` should be the portable public stable C API.
  - `Include/cpython/*.h` should be the unstable C API specific to CPython; public API, with some private API prefixed by `_Py` or `_PY`.
  - `Include/internal/*.h` is the private internal C API very specific to CPython. This API comes with no backward compatibility warranty and should not be used outside CPython. It is only exposed for very specific needs like debuggers and profiles which has to access to CPython internals without calling functions. This API is now installed by `make install`.

  (Contributed by Victor Stinner in bpo-35134 and bpo-35081, work initiated by Eric Snow in Python 3.7)

- Some macros have been converted to static inline functions: parameter types and return type are well defined, they don't have issues specific to macros, variables have a local scopes. Examples:
  - `Py_INCREF()`, `Py_DECREF()`
  - `Py_XINCREF()`, `Py_XDECREF()`
  - `PyObject_INIT()`, `PyObject_INIT_VAR()`
  - Private functions: `_PyObject_GC_TRACK()`, `_PyObject_GC_UNTRACK()`, `_Py_Dealloc()`

  (Contributed by Victor Stinner in bpo-35059.)

- The `PyByteArray_Init()` and `PyByteArray_Fini()` functions have been removed. They did nothing since Python 2.7.4 and Python 3.2.0, were excluded from the limited API (stable ABI), and were not documented. (Contributed by Victor Stinner in bpo-35713.)

- The result of `PyExceptionClass_Name()` is now of type `const char *` rather of `char *`. (Contributed by Serhiy Storchaka in bpo-33818.)

- The duality of `Modules/Setup.dist` and `Modules/Setup` has been removed. Previously, when updating the CPython source tree, one had to manually copy `Modules/Setup.dist` (inside the source tree) to `Modules/Setup` (inside the build tree) in order to reflect any changes upstream. This was of a small benefit to packagers at the expense of a frequent annoyance to developers following CPython development, as forgetting to copy the file could produce build failures.

  Now the build system always reads from `Modules/Setup` inside the source tree. People who want to customize that file are encouraged to maintain their changes in a git fork of CPython or as patch files, as they would do for any other change to the source tree.

  (Contributed by Antoine Pitrou in bpo-32430.)

- Functions that convert Python number to C integer like `PyLong_AsLong()` and argument parsing functions like `PyArg_ParseTuple()` with integer converting format units like `'i'` will now use the `__index__()` special method instead of `__int__()`, if available. The deprecation warning will be emitted for objects with the `__int__()` method but without the `__index__()` method (like `Decimal` and `Fraction`). `PyNumber_Check()` will now return `1` for objects implementing `__index__()`. `PyNumber_Long()`, `PyNumber_Float()` and `PyFloat_AsDouble()` also now use the `__index__()` method if available. (Contributed by Serhiy Storchaka in bpo-36048 and bpo-20092.)

- Heap-allocated type objects will now increase their reference count in `PyObject_Init()` (and its parallel macro `PyObject_INIT`) instead of in `PyType_GenericAlloc()`. Types that modify instance allocation or deallocation may need to be adjusted. (Contributed by Eddie Elizondo in bpo-35810.)

- The new function `PyCode_NewWithPosOnlyArgs()` allows to create code objects like `PyCode_New()`, but with an extra *posonlyargcount* parameter for indicating the number of positional-only arguments. (Contributed by Pablo Galindo in bpo-37221.)

8 Deprecated

- Deprecated methods `getchildren()` and `getiterator()` in the `ElementTree` module emit now a `DeprecationWarning` instead of `PendingDeprecationWarning`. They will be removed in Python 3.9. (Contributed by Serhiy Storchaka in bpo-29209.)
- Passing an object that is not an instance of `concurrent.futures.ThreadPoolExecutor` to `asyncio.loop.set_default_executor()` is deprecated and will be prohibited in Python 3.9. (Contributed by Elvis Pranskevichus in bpo-34075.)
- The `__getitem__()` methods of `xml.dom.pulldom.DOMEventStream`, `wsgiref.util.FileWrapper` and `fileinput.FileInput` have been deprecated.

  Implementations of these methods have been ignoring their *index* parameter, and returning the next item instead.

  (Contributed by Berker Peksag in bpo-9372.)
- The `typing.NamedTuple` class has deprecated the `_field_types` attribute in favor of the `__annotations__` attribute which has the same information. (Contributed by Raymond Hettinger in bpo-36320.)
- `ast` classes `Num`, `Str`, `Bytes`, `NameConstant` and `Ellipsis` are considered deprecated and will be removed in future Python versions. `Constant` should be used instead. (Contributed by Serhiy Storchaka in bpo-32892.)
- The following functions and methods are deprecated in the `gettext` module: `lgettext()`, `ldgettext()`, `lngettext()` and `ldngettext()`. They return encoded bytes, and it's possible that you will get unexpected Unicode-related exceptions if there are encoding problems with the translated strings. It's much better to use alternatives which return Unicode strings in Python 3. These functions have been broken for a long time.

  Function `bind_textdomain_codeset()`, methods `output_charset()` and `set_output_charset()`, and the *codeset* parameter of functions `translation()` and `install()` are also deprecated, since they are only used for for the `l*gettext()` functions.

  (Contributed by Serhiy Storchaka in bpo-33710.)
- The `isAlive()` method of `threading.Thread` has been deprecated. (Contributed by Dong-hee Na in bpo-35283.)
- Many builtin and extension functions that take integer arguments will now emit a deprecation warning for `Decimal`s, `Fraction`s and any other objects that can be converted to integers only with a loss (e.g. that have the `__int__()` method but do not have the `__index__()` method). In future version they will be errors. (Contributed by Serhiy Storchaka in bpo-36048.)
- Deprecated passing the following arguments as keyword arguments:
  - *func* in `functools.partialmethod()`, `weakref.finalize()`, `profile.Profile.runcall()`, `cProfile.Profile.runcall()`, `bdb.Bdb.runcall()`, `trace.Trace.runfunc()` and `curses.wrapper()`.
  - *function* in `unittest.TestCase.addCleanup()`.
  - *fn* in the `submit()` method of `concurrent.futures.ThreadPoolExecutor` and `concurrent.futures.ProcessPoolExecutor`.

- *callback* in `contextlib.ExitStack.callback()`, `contextlib.AsyncExitStack.callback()` and `contextlib.AsyncExitStack.push_async_callback()`.
- *c* and *typeid* in the `create()` method of `multiprocessing.managers.Server` and `multiprocessing.managers.SharedMemoryServer`.
- *obj* in `weakref.finalize()`.

In future releases of Python they will be positional-only. (Contributed by Serhiy Storchaka in bpo-36492.)

9 API and Feature Removals

The following features and APIs have been removed from Python 3.8:

- The `macpath` module, deprecated in Python 3.7, has been removed. (Contributed by Victor Stinner in bpo-35471.)
- The function `platform.popen()` has been removed, it was deprecated since Python 3.3: use `os.popen()` instead. (Contributed by Victor Stinner in bpo-35345.)
- The function `time.clock()` has been removed, it was deprecated since Python 3.3: use `time.perf_counter()` or `time.process_time()` instead, depending on your requirements, to have a well defined behavior. (Contributed by Matthias Bussonnier in bpo-36895.)
- The `pyvenv` script has been removed in favor of `python3.8 -m venv` to help eliminate confusion as to what Python interpreter the `pyvenv` script is tied to. (Contributed by Brett Cannon in bpo-25427.)
- `parse_qs`, `parse_qsl`, and `escape` are removed from `cgi` module. They are deprecated from Python 3.2 or older.
- `filemode` function is removed from `tarfile` module. It is not documented and deprecated since Python 3.3.
- The `XMLParser` constructor no longer accepts the *html* argument. It never had effect and was deprecated in Python 3.4. All other parameters are now keyword-only. (Contributed by Serhiy Storchaka in bpo-29209.)
- Removed the `doctype()` method of `XMLParser`. (Contributed by Serhiy Storchaka in bpo-29209.)
- "unicode_internal" codec is removed. (Contributed by Inada Naoki in bpo-36297.)
- The `Cache` and `Statement` objects of the `sqlite3` module are not exposed to the user. (Contributed by Aviv Palivoda in bpo-30262.)
- The `bufsize` keyword argument of `fileinput.input()` and `fileinput.FileInput()` which was ignored and deprecated since Python 3.6 has been removed. bpo-36952 (Contributed by Matthias Bussonnier)
- The functions `sys.set_coroutine_wrapper()` and `sys.get_coroutine_wrapper()` deprecated in Python 3.7 have been removed; bpo-36933 (Contributed by Matthias Bussonnier)

10 Porting to Python 3.8

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

10.1 Changes in Python behavior

- Yield expressions (both `yield` and `yield from` clauses) are now disallowed in comprehensions and generator expressions (aside from the iterable expression in the leftmost `for` clause). (Contributed by Serhiy Storchaka in bpo-10544.)

- The compiler now produces a `SyntaxWarning` when identity checks (`is` and `is not`) are used with certain types of literals (e.g. strings, ints). These can often work by accident in CPython, but are not guaranteed by the language spec. The warning advises users to use equality tests (== and !=) instead. (Contributed by Serhiy Storchaka in bpo-34850.)
- The CPython interpreter can swallow exceptions in some circumstances. In Python 3.8 this happens in less cases. In particular, exceptions raised when getting the attribute from the type dictionary are no longer ignored. (Contributed by Serhiy Storchaka in bpo-35459.)
- Removed `__str__` implementations from builtin types `bool`, `int`, `float`, `complex` and few classes from the standard library. They now inherit `__str__()` from `object`. As result, defining the `__repr__()` method in the subclass of these classes will affect they string representation. (Contributed by Serhiy Storchaka in bpo-36793.)
- On AIX, `sys.platform` doesn't contain the major version anymore. It is always `'aix'`, instead of `'aix3'` .. `'aix7'`. Since older Python versions include the version number, it is recommended to always use the `sys.platform.startswith('aix')`. (Contributed by M. Felt in bpo-36588.)
- `PyEval_AcquireLock()` and `PyEval_AcquireThread()` now terminate the current thread if called while the interpreter is finalizing, making them consistent with `PyEval_RestoreThread()`, `Py_END_ALLOW_THREADS()`, and `PyGILState_Ensure()`. If this behaviour is not desired, guard the call by checking `_Py_IsFinalizing()` or `sys.is_finalizing()`.

10.2 Changes in the Python API

- The `os.getcwdb()` function now uses the UTF-8 encoding on Windows, rather than the ANSI code page: see **PEP 529** for the rationale. The function is no longer deprecated on Windows. (Contributed by Victor Stinner in bpo-37412.)
- `subprocess.Popen` can now use `os.posix_spawn()` in some cases for better performance. On Windows Subsystem for Linux and QEMU User Emulation, Popen constructor using `os.posix_spawn()` no longer raise an exception on errors like missing program, but the child process fails with a non-zero `returncode`. (Contributed by Joannah Nanjekye and Victor Stinner in bpo-35537.)
- The `imap.IMAP4.logout()` method no longer ignores silently arbitrary exceptions.
- The function `platform.popen()` has been removed, it was deprecated since Python 3.3: use `os.popen()` instead. (Contributed by Victor Stinner in bpo-35345.)
- The `statistics.mode()` function no longer raises an exception when given multimodal data. Instead, it returns the first mode encountered in the input data. (Contributed by Raymond Hettinger in bpo-35892.)
- The `selection()` method of the `tkinter.ttk.Treeview` class no longer takes arguments. Using it with arguments for changing the selection was deprecated in Python 3.6. Use specialized methods like `selection_set()` for changing the selection. (Contributed by Serhiy Storchaka in bpo-31508.)
- The `writexml()`, `toxml()` and `toprettyxml()` methods of the `xml.dom.minidom` module, and `xml.etree` now preserve the attribute order specified by the user. (Contributed by Diego Rojas and Raymond Hettinger in bpo-34160.)
- A `dbm.dumb` database opened with flags `'r'` is now read-only. `dbm.dumb.open()` with flags `'r'` and `'w'` no longer creates a database if it does not exist. (Contributed by Serhiy Storchaka in bpo-32749.)
- The `doctype()` method defined in a subclass of `XMLParser` will no longer be called and will cause emitting a `RuntimeWarning` instead of a `DeprecationWarning`. Define the `doctype()` method on a target for handling an XML doctype declaration. (Contributed by Serhiy Storchaka in bpo-29209.)
- A `RuntimeError` is now raised when the custom metaclass doesn't provide the `__classcell__` entry in the namespace passed to `type.__new__`. A `DeprecationWarning` was emitted in Python 3.6–3.7. (Contributed by Serhiy Storchaka in bpo-23722.)

- The `cProfile.Profile` class can now be used as a context manager. (Contributed by Scott Sanderson in bpo-29235.)
- `shutil.copyfile()`, `shutil.copy()`, `shutil.copy2()`, `shutil.copytree()` and `shutil.move()` use platform-specific "fast-copy" syscalls (see shutil-platform-dependent-efficient-copy-operations section).
- `shutil.copyfile()` default buffer size on Windows was changed from 16 KiB to 1 MiB.
- `PyGC_Head` struct is changed completely. All code touched the struct member should be rewritten. (See bpo-33597)
- The `PyInterpreterState` struct has been moved into the "internal" header files (specifically Include/internal/pycore_pystate.h). An opaque `PyInterpreterState` is still available as part of the public API (and stable ABI). The docs indicate that none of the struct's fields are public, so we hope no one has been using them. However, if you do rely on one or more of those private fields and have no alternative then please open a BPO issue. We'll work on helping you adjust (possibly including adding accessor functions to the public API). (See bpo-35886.)
- Asyncio tasks can now be named, either by passing the `name` keyword argument to `asyncio.create_task()` or the `create_task()` event loop method, or by calling the `set_name()` method on the task object. The task name is visible in the `repr()` output of `asyncio.Task` and can also be retrieved using the `get_name()` method.
- The `mmap.flush()` method now returns `None` on success and raises an exception on error under all platforms. Previously, its behavior was platform-depended: a nonzero value was returned on success; zero was returned on error under Windows. A zero value was returned on success; an exception was raised on error under Unix. (Contributed by Berker Peksag in bpo-2122.)
- `xml.dom.minidom` and `xml.sax` modules no longer process external entities by default. (Contributed by Christian Heimes in bpo-17239.)
- Deleting a key from a read-only `dbm` database (`dbm.dumb`, `dbm.gnu` or `dbm.ndbm`) raises `error` (`dbm.dumb.error`, `dbm.gnu.error` or `dbm.ndbm.error`) instead of `KeyError`. (Contributed by Xiang Zhang in bpo-33106.)
- `expanduser()` on Windows now prefers the `USERPROFILE` environment variable and does not use `HOME`, which is not normally set for regular user accounts.
- DLL dependencies for extension modules and DLLs loaded with `ctypes` on Windows are now resolved more securely. Only the system paths, the directory containing the DLL or PYD file, and directories added with `add_dll_directory()` are searched for load-time dependencies. Specifically, `PATH` and the current working directory are no longer used, and modifications to these will no longer have any effect on normal DLL resolution. If your application relies on these mechanisms, you should check for `add_dll_directory()` and if it exists, use it to add your DLLs directory while loading your library. Note that Windows 7 users will need to ensure that Windows Update KB2533625 has been installed (this is also verified by the installer). (See bpo-36085.)
- The header files and functions related to pgen have been removed after its replacement by a pure Python implementation. (Contributed by Pablo Galindo in bpo-36623.)
- `types.CodeType` has a new parameter in the second position of the constructor (*posonlyargcount*) to support positional-only arguments defined in **PEP 570**. The first argument (*argcount*) now represents the total number of positional arguments (including positional-only arguments). A new `replace()` method of `types.CodeType` can be used to make the code future-proof.

10.3 Changes in the C API

- The `PyCompilerFlags` structure gets a new *cf_feature_version* field. It should be initialized to `PY_MINOR_VERSION`. The field is ignored by default, it is used if and only if `PyCF_ONLY_AST` flag is set

in *cf_flags*.

- The `PyEval_ReInitThreads()` function has been removed from the C API. It should not be called explicitly: use `PyOS_AfterFork_Child()` instead. (Contributed by Victor Stinner in bpo-36728.)

- On Unix, C extensions are no longer linked to libpython except on Android and Cygwin. When Python is embedded, `libpython` must not be loaded with `RTLD_LOCAL`, but `RTLD_GLOBAL` instead. Previously, using `RTLD_LOCAL`, it was already not possible to load C extensions which were not linked to `libpython`, like C extensions of the standard library built by the `*shared*` section of `Modules/Setup`. (Contributed by Victor Stinner in bpo-21536.)

- Use of `#` variants of formats in parsing or building value (e.g. `PyArg_ParseTuple()`, `Py_BuildValue()`, `PyObject_CallFunction()`, etc.) without `PY_SSIZE_T_CLEAN` defined raises `DeprecationWarning` now. It will be removed in 3.10 or 4.0. Read arg-parsing for detail. (Contributed by Inada Naoki in bpo-36381.)

- Instances of heap-allocated types (such as those created with `PyType_FromSpec()`) hold a reference to their type object. Increasing the reference count of these type objects has been moved from `PyType_GenericAlloc()` to the more low-level functions, `PyObject_Init()` and `PyObject_INIT()`. This makes types created through `PyType_FromSpec()` behave like other classes in managed code.

  Statically allocated types are not affected.

  For the vast majority of cases, there should be no side effect. However, types that manually increase the reference count after allocating an instance (perhaps to work around the bug) may now become immortal. To avoid this, these classes need to call Py_DECREF on the type object during instance deallocation.

  To correctly port these types into 3.8, please apply the following changes:

  – Remove `Py_INCREF` on the type object after allocating an instance - if any. This may happen after calling `PyObject_New()`, `PyObject_NewVar()`, `PyObject_GC_New()`, `PyObject_GC_NewVar()`, or any other custom allocator that uses `PyObject_Init()` or `PyObject_INIT()`.

    Example:

```
static foo_struct *
foo_new(PyObject *type) {
    foo_struct *foo = PyObject_GC_New(foo_struct, (PyTypeObject *) type);
    if (foo == NULL)
        return NULL;
#if PY_VERSION_HEX < 0x03080000
    // Workaround for Python issue 35810; no longer necessary in Python 3.8
    PY_INCREF(type)
#endif
    return foo;
}
```

  – Ensure that all custom `tp_dealloc` functions of heap-allocated types decrease the type's reference count.

    Example:

```
static void
foo_dealloc(foo_struct *instance) {
    PyObject *type = Py_TYPE(instance);
    PyObject_GC_Del(instance);
#if PY_VERSION_HEX >= 0x03080000
    // This was not needed before Python 3.8 (Python issue 35810)
    Py_DECREF(type);
```

(continues on next page)

(continued from previous page)

```
#endif
}
```

(Contributed by Eddie Elizondo in bpo-35810.)

- The `Py_DEPRECATED()` macro has been implemented for MSVC. The macro now must be placed before the symbol name.

  Example:

```
Py_DEPRECATED(3.8) PyAPI_FUNC(int) Py_OldFunction(void);
```

  (Contributed by Zackery Spytz in bpo-33407.)

- The interpreter does not pretend to support binary compatibility of extension types across feature releases, anymore. A `PyTypeObject` exported by a third-party extension module is supposed to have all the slots expected in the current Python version, including `tp_finalize` (`Py_TPFLAGS_HAVE_FINALIZE` is not checked anymore before reading `tp_finalize`).

  (Contributed by Antoine Pitrou in bpo-32388.)

- The `PyCode_New()` has a new parameter in the second position (*posonlyargcount*) to support **PEP 570**, indicating the number of positional-only arguments.

- The functions `PyNode_AddChild()` and `PyParser_AddToken()` now accept two additional `int` arguments *end_lineno* and *end_col_offset*.

- The `libpython38.a` file to allow MinGW tools to link directly against `python38.dll` is no longer included in the regular Windows distribution. If you require this file, it may be generated with the `gendef` and `dlltool` tools, which are part of the MinGW binutils package:

```
gendef python38.dll > tmp.def
dlltool --dllname python38.dll --def tmp.def --output-lib libpython38.a
```

  The location of an installed `pythonXY.dll` will depend on the installation options and the version and language of Windows. See using-on-windows for more information. The resulting library should be placed in the same directory as `pythonXY.lib`, which is generally the `libs` directory under your Python installation.

10.4 CPython bytecode changes

- The interpreter loop has been simplified by moving the logic of unrolling the stack of blocks into the compiler. The compiler emits now explicit instructions for adjusting the stack of values and calling the cleaning-up code for `break`, `continue` and `return`.

  Removed opcodes `BREAK_LOOP`, `CONTINUE_LOOP`, `SETUP_LOOP` and `SETUP_EXCEPT`. Added new opcodes `ROT_FOUR`, `BEGIN_FINALLY`, `CALL_FINALLY` and `POP_FINALLY`. Changed the behavior of `END_FINALLY` and `WITH_CLEANUP_START`.

  (Contributed by Mark Shannon, Antoine Pitrou and Serhiy Storchaka in bpo-17611.)

- Added new opcode `END_ASYNC_FOR` for handling exceptions raised when awaiting a next item in an `async for` loop. (Contributed by Serhiy Storchaka in bpo-33041.)

- The `MAP_ADD` now expects the value as the first element in the stack and the key as the second element. This change was made so the key is always evaluated before the value in dictionary comprehensions, as proposed by **PEP 572**. (Contributed by Jörn Heissler in bpo-35224.)

10.5 Demos and Tools

- Added a benchmark script for timing various ways to access variables: `Tools/scripts/var_access_benchmark.py`. (Contributed by Raymond Hettinger in bpo-35884.)

Index