What's New in Python

3.7.2

A. M. Kuchling

12, 2019

Python Software Foundation
Email: docs@python.org

Contents

1 - 4

2 5
2.1 PEP 563 . . . 5
2.2 PEP 538: C . . . 5
2.3 PEP 540: UTF-8 . . . 6
2.4 PEP 553: `breakpoint()` . . . 6
2.5 PEP 539: C API . . . 6
2.6 PEP 562: . . . 6
2.7 PEP 564: . . . 6
2.8 PEP 565: `__main__` DeprecationWarning . . . 7
2.9 PEP 560: `typing` . . . 7
2.10 PEP 552: .pyc . . . 7
2.11 PEP 545: Python . . . 8
2.12 : -X dev . . . 8

3 8

4 8
4.1 contextvars . . . 8
4.2 dataclasses . . . 9
4.3 importlib.resources . . . 9

5 9
5.1 argparse . . . 9
5.2 asyncio . . . 9
5.3 binascii . . . 10
5.4 calendar . . . 10
5.5 collections . . . 11
5.6 compileall . . . 11
5.7 concurrent.futures . . . 11
5.8 contextlib . . . 11
5.9 cProfile . . . 11
5.10 crypt . . . 11

5.11 datetime . . . 11
5.12 dbm . . . 11
5.13 decimal . . . 11
5.14 dis . . . 12
5.15 distutils . . . 12
5.16 enum . . . 12
5.17 functools . . . 12
5.18 gc . . . 12
5.19 hmac . . . 12
5.20 http.client . . . 12
5.21 http.server . . . 12
5.22 idlelib IDLE . . . 13
5.23 importlib . . . 13
5.24 io . . . 13
5.25 ipaddress . . . 13
5.26 itertools . . . 13
5.27 locale . . . 14
5.28 logging . . . 14
5.29 math . . . 14
5.30 mimetypes . . . 14
5.31 msilib . . . 14
5.32 multiprocessing . . . 14
5.33 os . . . 14
5.34 pathlib . . . 14
5.35 pdb . . . 15
5.36 py_compile . . . 15
5.37 pydoc . . . 15
5.38 queue . . . 15
5.39 re . . . 15
5.40 signal . . . 15
5.41 socket . . . 15
5.42 socketserver . . . 16
5.43 sqlite3 . . . 16
5.44 ssl . . . 16
5.45 string . . . 16
5.46 subprocess . . . 17
5.47 sys . . . 17
5.48 time . . . 17
5.49 tkinter . . . 18
5.50 tracemalloc . . . 18
5.51 types . . . 18
5.52 unicodedata . . . 18
5.53 unittest . . . 18
5.54 unittest.mock . . . 18
5.55 urllib.parse . . . 18
5.56 uu . . . 19
5.57 uuid . . . 19
5.58 warnings . . . 19
5.59 xml . . . 19
5.60 xml.etree . . . 19
5.61 xmlrpc.server . . . 20
5.62 zipapp . . . 20
5.63 zipfile . . . 20

6 C API **20**

7 **21**

8 **22**

9 CPython **23**

10 Python **23**

11 Python **24**
11.1 aifc . . . 24
11.2 asyncio . . . 24
11.3 collections . . . 24
11.4 dbm . . . 24
11.5 enum . . . 24
11.6 gettext . . . 24
11.7 importlib . . . 24
11.8 locale . . . 25
11.9 macpath . . . 25
11.10threading . . . 25
11.11socket . . . 25
11.12ssl . . . 25
11.13sunau . . . 25
11.14sys . . . 25
11.15wave . . . 25

12 Deprecated functions and types of the C API **26**

13 Platform Support Removals **26**

14 API and Feature Removals **26**

15 Module Removals **27**

16 Windows-only Changes **27**

17 Porting to Python 3.7 **27**
17.1 Changes in Python Behavior . . . 27
17.2 Changes in the Python API . . . 28
17.3 C API . . . 30
17.4 CPython . . . 30
17.5 Windows-only Changes . . . 30
17.6 CPython . . . 30

18 Python 3.7.1 **31**

19 Python 3.7.2 **31**

32

Elvis Pranskevichus <elvis@magic.io>

Python 3.7 3.6 Python 3.7 2018 6 27

1 -

- *PEP 563*

- `async await`

- `contextvars`: *PEP 567* –
- `dataclasses`: *PEP 557* –
- *importlib.resources*

- *PEP 553*, `breakpoint()`

Python

- *PEP 562*,
- *PEP 560*, typing
- dict Python

- `asyncio`
- `time`

CPython

- ASCII
  - *PEP 538* C
  - *PEP 540* UTF-8
- *PEP 552* .pycs
-
- *PEP 565* `DeprecationWarning`

C API

- *PEP 539* C API

- *PEP 545*, Python
- Japanese French Korean

Python *Porting to Python 3.7*

2

2.1 PEP 563

PEP 3107 **PEP 526** Python

- ——
- Python

AST `typing.get_type_hints()`

:

```
class C:
    @classmethod
    def from_string(cls, source: str) -> C:
        ...

    def validate_b(self, obj: B) -> bool:
        ...

class B:
    ...
```

Python 3.7 `__future__` :

```
from __future__ import annotations
```

Python 4.0

:

PEP 563 – PEP Łukasz Langa

2.2 PEP 538: C

Python 3 7 ASCII Windows C POSIX

PEP 538 UTF-8 `PYTHONCOERCECLOCALE` `LC_CTYPE`
C (`readline`) UTF-8 ASCII

PEP 11 ASCII

(`C.UTF-8`, `C.utf8` `UTF-8`) `stdin` `stdout` `surrogateescape` (`strict`)
`stderr` `backslashreplace`

`PYTHONCOERCECLOCALE=warn` `stderr` Python C

PEP 538 (GNU `readline`) (Python Python)
(RHEL/CentOS 7) Python 3.7 *PEP 540: UTF-8*

:

PEP 538 – **C** **UTF-8** PEP Nick Coghlan

2.3 PEP 540: UTF-8

`-X utf8` `PYTHONUTF8` CPython *UTF-8*

UTF-8 CPython UTF-8 `sys.stdin` `sys.stdout` `surrogateescape`

UTF-8 Python

PEP 540 UTF-8 (GNU `readline`) Python Python Python 3.7 *PEP 540: UTF-8*)

UTF-8 `C` `POSIX` **PEP 538** UTF-8 `PYTHONCOERCECLOCALE=0`, `LC_ALL`

:

PEP 540 – **UTF-8** PEP Victor Stinner

2.4 PEP 553: `breakpoint()`

Python 3.7 `breakpoint()` Python

`breakpoint()` `sys.breakpointhook()` `pdb` `pdb.set_trace()` `sys.breakpointhook()` `breakpoint()` `PYTHONBREAKPOINT` `PYTHONBREAKPOINT=0` `breakpoint()`

:

PEP 553 – **breakpoint()** PEP Barry Warsaw

2.5 PEP 539: C API

Python C API (TLS) API `int` TLS POSIX

PEP 539 CPython (TSS) API CPython TLS API API TSS API `Py_tss_t` `int` TSS – TLS `int` TLS CPython

TLS `int` TLS API API

:

PEP 539 – **CPython** **C-API** PEP Erik M. Bray Masayuki Yamamoto

2.6 PEP 562:

Python 3.7 `__getattr__()` `__dir__()`

:

PEP 562 – `__getattr__` `__dir__` PEP Ivan Levkivskyi

2.7 PEP 564:

`time.time()` **PEP 564** `time` “ ” :

- `time.clock_gettime_ns()`

- `time.clock_settime_ns()`
- `time.monotonic_ns()`
- `time.perf_counter_ns()`
- `time.process_time_ns()`
- `time.time_ns()`

Linux Windows `time.time_ns()` `time.time()` 3

:

PEP 564 – PEP Victor Stinner

2.8 PEP 565: `__main__` DeprecationWarning

`DeprecationWarning` `__main__` Python API

:

- `FutureWarning`:
- `DeprecationWarning`: `__main__` Python
- `PendingDeprecationWarning`: `DeprecationWarning` `FutureWarning`

`DeprecationWarning` `PendingDeprecationWarning` Python API

:

PEP 565 – `__main__` **DeprecationWarning** PEP Nick Coghlan

2.9 PEP 560: `typing`

PEP 484 CPython `typing` PEP `__class_getitem__()` `__mro_entries__` `typing` 7 `typing`

:

PEP 560 – **typing** PEP Ivan Levkivskyi

2.10 PEP 552: .pyc

Python (`.pyc`) (Python

PEP 552 pyc `.pyc` “ ” Python `.pyc` `.pyc` `py_compile` `compileall`

`.pyc` Python `.pyc` pyc `.pyc` Python `.pyc`

pyc-invalidation

:

PEP 552 – **pyc** PEP Benjamin Peterson

2.11 PEP 545: Python

PEP 545 Python

:

- : https://docs.python.org/ja/
- : https://docs.python.org/fr/
- : https://docs.python.org/ko/

:

PEP 545 – Python PEP Julien Palard, Inada Naoki Victor Stinner

2.12 : -X dev

`-X dev` `PYTHONDEVMODE` CPython CPython `-X dev`

3

- 255 255 Serhiy Storchaka bpo-12844 bpo-18896
- `bytes.fromhex()` `bytearray.fromhex()` ASCII . Robert Xiao bpo-28927
- `str`, `bytes` `bytearray` `isascii()` ASCII INADA Naoki bpo-32677
- `from ... import ...` `ImportError` `__file__` Matthias Bussonnier bpo-29546
- Serhiy Storchaka bpo-30024
- `object.__format__(x, '')` `str(x)` `format(str(self), '')` Serhiy Storchaka d bpo-28974
- `types.TracebackType` Python `tb_next` Nathaniel J. Smith bpo-30579
- `-m` `sys.path[0]` Nick Coghlan bpo-33053
- `-X importtime` `PYTHONPROFILEIMPORTTIME` Victor Stinner bpo-31415

4

4.1 contextvars

`contextvars` C API TLS

`asyncio` `decimal` decimal

:

PEP 567 – PEP Yury Selivanov

4.2 dataclasses

`dataclass()` `__repr__()`, `__eq__()` `__hash__()`

:

```
@dataclass
class Point:
    x: float
    y: float
    z: float = 0.0

p = Point(1.5, 2.5)
print(p)   # produces "Point(x=1.5, y=2.5, z=0.0)"
```

:

PEP 557 – PEP Eric V. Smith

4.3 importlib.resources

`importlib.resources` API ABC `get_resource_reader()` `importlib.abc.ResourceReader` API zip

Barry Warsaw Brett Cannon bpo-32248

:

importlib_resources – Python PyPI

5

5.1 argparse

`ArgumentParser.parse_intermixed_args()` paul.j3 bpo-14191

5.2 asyncio

`asyncio` :

- `asyncio.run()` Yury Selivanov bpo-32314
- asyncio `contextvars`. `loop.call_soon()`, `loop.call_soon_threadsafe()`, `loop.call_later()`, `loop.call_at()` `Future.add_done_callback()` *context* `Tasks` **PEP 567** Yury Selivanov bpo-32436
- `asyncio.create_task()` `asyncio.get_event_loop().create_task()` Andrew Svetlov bpo-32311
- `loop.start_tls()` TLS Yury Selivanov bpo-23749
- `loop.sock_recv_into()` Antoine Pitrou bpo-31819
- `asyncio.current_task()` `Task` `asyncio.all_tasks()` `Task` `Task.current_task()` `Task.all_tasks()` Andrew Svetlov bpo-32250

- `BufferedProtocol` Yury Selivanov bpo-32251
- `asyncio.get_running_loop()` `RuntimeError` `asyncio.get_event_loop()` Yury Selivanov bpo-32269
- `StreamWriter.wait_closed()` `StreamWriter.is_closing()` Andrew Svetlov bpo-32391
- `loop.sock_sendfile()` `os.sendfile` Andrew Svetlov bpo-32410
- `Future.get_loop()` `Task.get_loop()` task future `Server.get_loop()` `asyncio.Server` Yury Selivanov bpo-32415 Srinivas Reddy Thatiparthy bpo-32418
- `asyncio.Server` *start_serving* `loop.create_server()` `loop.create_unix_server()` `Server.start_serving()`, `Server.serve_forever()` `Server.is_serving()` `True` `Server` :

```
srv = await loop.create_server(...)

async with srv:
    # some code

# At this point, srv is closed and no longer accepts new connections.
```

  Yury Selivanov bpo-32662
- `loop.call_later()` `when()` Andrew Svetlov bpo-32741
- `loop.create_datagram_endpoint()` Unix Quentin Dawans bpo-31245
- `asyncio.open_connection()`, `asyncio.start_server()` functions, `loop.create_connection()`, `loop.create_server()`, `loop.create_accepted_socket()` UNIX *ssl_handshake_timeout* Neil Aspinall bpo-29970
- `Handle.cancelled()` `True` Marat Sharafutdinov bpo-31943
- asyncio `async/await` Andrew Svetlov bpo-32193
- `ReadTransport.is_reading()` `ReadTransport.resume_reading()` `ReadTransport.pause_reading()` Yury Selivanov bpo-32356
- Yury Selivanov bpo-32066
- `asyncio` Linux TCP `TCP_NODELAY` Yury Selivanov Victor Stinner bpo-27456
- Yury Selivanov bpo-30508
- `WindowsSelectorEventLoopPolicy` `WindowsProactorEventLoopPolicy` Yury Selivanov bpo-33792

`asyncio` API

5.3 binascii

`b2a_uu()` *backtick* `` `` `` Xiang Zhang bpo-30103

5.4 calendar

`HTMLCalendar` HTML CSS Oz Tiram bpo-30095

5.5 collections

`collections.namedtuple()` Raymond Hettinger bpo-32320

5.6 compileall

`compileall.compile_dir()` *invalidation_mode* *.pyc* `--invalidation-mode` Benjamin Peterson bpo-31650

5.7 concurrent.futures

`ProcessPoolExecutor` `ThreadPoolExecutor` *initargs* Antoine Pitrou bpo-21423

`ProcessPoolExecutor` *mp_context* Thomas Moreau bpo-31540

5.8 contextlib

`nullcontext()` `ExitStack` Jesse-Bakker bpo-10049

`asynccontextmanager()`, `AbstractAsyncContextManager` `AsyncExitStack` Jelle Zijlstra bpo-29679 bpo-30241 Alexander Mohr Ilya Kulakov bpo-29302

5.9 cProfile

`cProfile` `-m module_name` Sanyam Khurana bpo-21862

5.10 crypt

`crypt` Blowfish Serhiy Storchaka bpo-31664

`mksalt()` Serhiy Storchaka bpo-31702

5.11 datetime

`datetime.fromisoformat()` `datetime.isoformat()` `datetime` Paul Ganssle bpo-15873

`tzinfo` Alexander Belopolsky bpo-5288

5.12 dbm

`dbm.dumb`

5.13 decimal

`decimal` Yury Selivanov bpo-32630

5.14 dis

`dis()` *depth* Serhiy Storchaka bpo-11822

5.15 distutils

`README.rst` distutils README Ryan Gonzalez bpo-11913

5.16 enum

`Enum` `_ignore_` Ethan Furman bpo-31801

Python 3.8 `Enum` Enum `TypeError` (`1 in Color`) `Flag` Flag `TypeError` (`1 in Perm.RW`) `False` Ethan Furman bpo-33217

5.17 functools

`functools.singledispatch()` Łukasz Langa bpo-32227

5.18 gc

`gc.freeze()` POSIX `fork()` GC `gc.unfreeze()` `gc.get_freeze_count()` Li Zekun bpo-31558

5.19 hmac

`hmac` `digest()` `HMAC()` Christian Heimes bpo-32433

5.20 http.client

`HTTPConnection` `HTTPSConnection` *blocksize* Nir Soffer bpo-31945

5.21 http.server

`SimpleHTTPRequestHandler` HTTP `If-Modified-Since` 304 Pierre Quentel bpo-29654

`SimpleHTTPRequestHandler` *directory* `--directory` Stéphane Wirtel Julien Palard bpo-28707

`ThreadingHTTPServer` `ThreadingMixin` `http.server` `-m` Julien Palard bpo-31639

5.22 idlelib IDLE

Louie Lu bpo-15786

Module Browser (File Class Browser) Guilherme Polo, Cheryl Sabella Terry Jan Reedy bpo-1612262

Settings (Options Configure IDLE) Cheryl Sabella Terry Jan Reedy

Terry Jan Reedy bpo-13802 Serhiy Storchaka bpo-31860

IDLE Extensions Charles Wohlganger Terry Jan Reedy bpo-27099

Box Settings Highlights Cheryl Sabella Terry Jan Reedy bpo-33642, bpo-33768 bpo-33679

Windows API tk DPI Windows Windows 8.1+ 10 Python DPI 96 DPI Terry Jan Reedy bpo-33656

3.7.1 :

N 50 N Settings General PyShell Tal Einat bpo-1529353

3.6

5.23 importlib

`importlib.abc.ResourceReader` ABC *importlib.resources* Barry Warsaw, Brett Cannon bpo-32248

`importlib.reload()` `ModuleNotFoundError` Garvit Khatri bpo-29851

(`__path__`) `importlib.find_spec()` `ModuleNotFoundError` `AttributeError` Milan Oberkirch bpo-30436

`importlib.source_hash()` *.pyc*

5.24 io

`TextIOWrapper.reconfigure()` Antoine Pitrou bpo-30526 INADA Naoki bpo-15216

5.25 ipaddress

methods of `ipaddress.IPv6Network` `ipaddress.IPv4Network` `subnet_of()` `supernet_of()` Michel Albert Cheryl Sabella bpo-20825

5.26 itertools

`itertools.islice()` start, stop slice Will Roberts bpo-30537

5.27 locale

`locale.format_string()` *monetary* Garvit bpo-10379

`locale.getpreferredencoding()` Android *UTF-8* `'UTF-8'`

5.28 logging

`Logger` pickle Vinay Sajip bpo-30520

`StreamHandler.setStream()` Vinay Sajip bpo-30522

`logging.config.fileConfig()` Preston Landers bpo-31080)

5.29 math

`math.remainder()` IEEE 754 Mark Dickinson bpo-29962

5.30 mimetypes

.bmp MIME type `'image/x-ms-bmp'` `'image/bmp'` Nitish Chandra bpo-22589

5.31 msilib

`Database.Close()` MSI Berker Peksag bpo-20486

5.32 multiprocessing

`Process.close()` `ValueError` Antoine Pitrou bpo-30596

`Process.kill()` Unix `SIGKILL` Vitor Pereira bpo-30794

`Process` Antoine Pitrou bpo-18966

5.33 os

`os.fwalk()` `bytes` *path* Serhiy Storchaka bpo-28682

`os.scandir()` Serhiy Storchaka bpo-25996

`register_at_fork()` Python Antoine Pitrou bpo-16500

`os.preadv()` (`os.readv()` `os.pread()`) `os.pwritev()` (`os.writev()` `os.pwrite()`) Pablo Galindo bpo-31368

`os.makedirs()` mode Serhiy Storchaka bpo-19930

`os.dup2()` `None` Benjamin Peterson bpo-32441

Solaris `os.stat()` `st_fstype` Jesús Cea Avión bpo-32659

5.34 pathlib

POSIX `Path.is_mount()` Cooper Ry Lees bpo-30897

5.35 pdb

`pdb.set_trace()` *header* Barry Warsaw bpo-31389

`pdb` `-m module_name` Mario Corchero bpo-32206

5.36 py_compile

`py_compile.compile()` – `compileall` – `.pyc` `SOURCE_DATE_EPOCH` `.pyc` Bernhard M. Wiedemann bpo-29708

5.37 pydoc

pydoc `-n` Feanil Patel bpo-31128

5.38 queue

`SimpleQueue` FIFO Antoine Pitrou bpo-14976

5.39 re

`re.ASCII`, `re.LOCALE` `re.UNICODE` Serhiy Storchaka bpo-31690

`re.split()` `r'\b'`, `'^$'` `(?=-)` Serhiy Storchaka bpo-25054

`re.LOCALE` Serhiy Storchaka bpo-30215

`FutureWarning` Serhiy Storchaka bpo-30349

`copy.copy()` `copy.deepcopy()` Serhiy Storchaka bpo-10076

5.40 signal

`signal.set_wakeup_fd()` *warn_on_full_buffer* Python stderr Nathaniel J. Smith bpo-30050

5.41 socket

`socket.getblocking()` `True` `False` Yury Selivanov bpo-32373

`socket.close()` `os.close()` Christian Heimes bpo-32454

`socket` `socket.TCP_CONGESTION` (Linux 2.6.13), `socket.TCP_USER_TIMEOUT` (Linux 2.6.37) `socket.TCP_NOTSENT_LOWAT` (Linux 3.12) Omar Sandoval bpo-26273 Nathaniel J. Smith bpo-29728

`socket.AF_VSOCK` Cathy Avery bpo-27584

Christian Heimes bpo-28134

5.42 socketserver

`socketserver.ThreadingMixIn.server_close()` `socketserver.ForkingMixIn.server_close()`

`socketserver.ForkingMixIn` `socketserver.ThreadingMixIn` `socketserver.ForkingMixIn.block_on_close` `False` 3.7

5.43 sqlite3

SQLite 3.6.11 `sqlite3.Connection` `backup()` Lele Gaifax bpo-27645

`sqlite3.connect()` *database* path-like object Anders Lorentsen bpo-31843

5.44 ssl

`ssl` OpenSSL API `match_hostname()` IP TLS `SSLCertVerificationError` TLS Alert `SSLContext.host_flags` Christian Heimes bpo-31399

`:` OpenSSL 1.0.2 1.1 *libssl* OpenSSL 0.9.8 1.0.1 *Platform Support Removals* ssl LibreSSL 2.7.2

`ssl` SNI TLS IP Christian Heimes bpo-32185

`match_hostname()` `www*.example.org` `SSLContext.host_flags` Mandeep Singh bpo-23033 Christian Heimes bpo-31399

`ssl` Python OpenSSL Christian Heimes bpo-31429

(IDN) `SSLSocket.server_hostname` A (`"xn--pythn-mua.org"`) U (`"pythön.org"`) Nathaniel J. Smith Christian Heimes bpo-28414

`ssl` TLS 1.3 OpenSSL 1.1.1 Python 3.7.0 OpenSSL 1.1.1 TLS 1.3 TLS 1.3 TLS 1.2 ssl-tlsv1_3 Christian Heimes bpo-32947, bpo-20995, bpo-29136, bpo-30622 bpo-33618

`SSLSocket` `SSLObject` `SSLContext` `wrap_socket()` `wrap_bio()` Christian Heimes bpo-32951

TLS OpenSSL 1.1 API `SSLContext.minimum_version` `SSLContext.maximum_version` `HAS_TLSv1_1` Christian Heimes bpo-32609

`SSLContext.post_handshake_auth` `ssl.SSLSocket.verify_client_post_handshake()` TLS 1.3 Christian Heimes bpo-34670

5.45 string

`string.Template` Barry Warsaw bpo-1198569

5.46 subprocess

`subprocess.run()` *capture_output* stdout stderr `subprocess.PIPE` *stdout* *stderr* Bo Bayles bpo-32102

`subprocess.run` `subprocess.Popen` *text* *universal_newlines* Andrew Clegg bpo-31756

Windows *close_fds* `False` `True` *close_fds* `subprocess.Popen` *close_fds* `True` Segev Finer bpo-19764

`subprocess.call()`, `subprocess.run()` `Popen` subprocess `KeyboardInterrupt` `KeyboardInterrupt` Gregory P. Smith bpo-25942

5.47 sys

The new `sys.breakpointhook()` hook function is called by the built-in `breakpoint()`. (Contributed by Barry Warsaw in bpo-31353.)

On Android, the new `sys.getandroidapilevel()` returns the build-time Android API version. (Contributed by Victor Stinner in bpo-28740.)

The new `sys.get_coroutine_origin_tracking_depth()` function returns the current coroutine origin tracking depth, as set by the new `sys.set_coroutine_origin_tracking_depth()`. `asyncio` has been converted to use this new API instead of the deprecated `sys.set_coroutine_wrapper()`. (Contributed by Nathaniel J. Smith in bpo-32591.)

5.48 time

PEP 564 adds six new functions with nanosecond resolution to the `time` module:

- `time.clock_gettime_ns()`
- `time.clock_settime_ns()`
- `time.monotonic_ns()`
- `time.perf_counter_ns()`
- `time.process_time_ns()`
- `time.time_ns()`

New clock identifiers have been added:

- `time.CLOCK_BOOTTIME` (Linux): Identical to `time.CLOCK_MONOTONIC`, except it also includes any time that the system is suspended.
- `time.CLOCK_PROF` (FreeBSD, NetBSD and OpenBSD): High-resolution per-process CPU timer.
- `time.CLOCK_UPTIME` (FreeBSD, OpenBSD): Time whose absolute value is the time the system has been running and not suspended, providing accurate uptime measurement.

The new `time.thread_time()` and `time.thread_time_ns()` functions can be used to get per-thread CPU time measurements. (Contributed by Antoine Pitrou in bpo-32025.)

The new `time.pthread_getcpuclockid()` function returns the clock ID of the thread-specific CPU-time clock.

5.49 tkinter

The new `tkinter.ttk.Spinbox` class is now available. (Contributed by Alan Moore in bpo-32585.)

5.50 tracemalloc

`tracemalloc.Traceback` behaves more like regular tracebacks, sorting the frames from oldest to most recent. `Traceback.format()` now accepts negative *limit*, truncating the result to the `abs(limit)` oldest frames. To get the old behaviour, use the new *most_recent_first* argument to `Traceback.format()`. (Contributed by Jesse Bakker in bpo-32121.)

5.51 types

The new `WrapperDescriptorType`, `MethodWrapperType`, `MethodDescriptorType`, and `ClassMethodDescriptorType` classes are now available. (Contributed by Manuel Krebber and Guido van Rossum in bpo-29377, and Serhiy Storchaka in bpo-32265.)

The new `types.resolve_bases()` function resolves MRO entries dynamically as specified by **PEP 560**. (Contributed by Ivan Levkivskyi in bpo-32717.)

5.52 unicodedata

The internal `unicodedata` database has been upgraded to use Unicode 11. (Contributed by Benjamin Peterson.)

5.53 unittest

The new `-k` command-line option allows filtering tests by a name substring or a Unix shell-like pattern. For example, `python -m unittest -k foo` runs `foo_tests.SomeTest.test_something`, `bar_tests.SomeTest.test_foo`, but not `bar_tests.FooTest.test_something`. (Contributed by Jonas Haag in bpo-32071.)

5.54 unittest.mock

The `sentinel` attributes now preserve their identity when they are `copied` or `pickled`. (Contributed by Serhiy Storchaka in bpo-20804.)

The new `seal()` function allows sealing `Mock` instances, which will disallow further creation of attribute mocks. The seal is applied recursively to all attributes that are themselves mocks. (Contributed by Mario Corchero in bpo-30541.)

5.55 urllib.parse

`urllib.parse.quote()` has been updated from **RFC 2396** to **RFC 3986**, adding `~` to the set of characters that are never quoted by default. (Contributed by Christian Theune and Ratnadeep Debnath in bpo-16285.)

5.56 uu

The `uu.encode()` function now accepts an optional *backtick* keyword argument. When it's true, zeros are represented by `'`'` instead of spaces. (Contributed by Xiang Zhang in bpo-30103.)

5.57 uuid

The new `UUID.is_safe` attribute relays information from the platform about whether generated UUIDs are generated with a multiprocessing-safe method. (Contributed by Barry Warsaw in bpo-22807.)

`uuid.getnode()` now prefers universally administered MAC addresses over locally administered MAC addresses. This makes a better guarantee for global uniqueness of UUIDs returned from `uuid.uuid1()`. If only locally administered MAC addresses are available, the first such one found is returned. (Contributed by Barry Warsaw in bpo-32107.)

5.58 warnings

The initialization of the default warnings filters has changed as follows:

- warnings enabled via command line options (including those for `-b` and the new CPython-specific `-X dev` option) are always passed to the warnings machinery via the `sys.warnoptions` attribute.
- warnings filters enabled via the command line or the environment now have the following order of precedence:
  - the `BytesWarning` filter for `-b` (or `-bb`)
  - any filters specified with the `-W` option
  - any filters specified with the `PYTHONWARNINGS` environment variable
  - any other CPython specific filters (e.g. the `default` filter added for the new `-X dev` mode)
  - any implicit filters defined directly by the warnings machinery
- in CPython debug builds, all warnings are now displayed by default (the implicit filter list is empty)

(Contributed by Nick Coghlan and Victor Stinner in bpo-20361, bpo-32043, and bpo-32230.)

Deprecation warnings are once again shown by default in single-file scripts and at the interactive prompt. See *PEP 565: __main__ DeprecationWarning* for details. (Contributed by Nick Coghlan in bpo-31975.)

5.59 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.)

5.60 xml.etree

ElementPath predicates in the `find()` methods can now compare text of the current node with `[. = "text"]`, not only text in children. Predicates also allow adding spaces for better readability. (Contributed by Stefan Behnel in bpo-31648.)

5.61 xmlrpc.server

`SimpleXMLRPCDispatcher.register_function` can now be used as a decorator. (Contributed by Xiang Zhang in bpo-7769.)

5.62 zipapp

Function `create_archive()` now accepts an optional *filter* argument to allow the user to select which files should be included in the archive. (Contributed by Irmen de Jong in bpo-31072.)

Function `create_archive()` now accepts an optional *compressed* argument to generate a compressed archive. A command line option `--compress` has also been added to support compression. (Contributed by Zhiming Wang in bpo-31638.)

5.63 zipfile

`ZipFile` now accepts the new *compresslevel* parameter to control the compression level. (Contributed by Bo Bayles in bpo-21417.)

Subdirectories in archives created by `ZipFile` are now stored in alphabetical order. (Contributed by Bernhard M. Wiedemann in bpo-30693.)

6 C API

A new API for thread-local storage has been implemented. See *PEP 539: C API* for an overview and thread-specific-storage-api for a complete reference. (Contributed by Masayuki Yamamoto in bpo-25658.)

The new *context variables* functionality exposes a number of new C APIs.

The new `PyImport_GetModule()` function returns the previously imported module with the given name. (Contributed by Eric Snow in bpo-28411.)

The new `Py_RETURN_RICHCOMPARE` macro eases writing rich comparison functions. (Contributed by Petr Victorin in bpo-23699.)

The new `Py_UNREACHABLE` macro can be used to mark unreachable code paths. (Contributed by Barry Warsaw in bpo-31338.)

The `tracemalloc` now exposes a C API through the new `PyTraceMalloc_Track()` and `PyTraceMalloc_Untrack()` functions. (Contributed by Victor Stinner in bpo-30054.)

The new `import__find__load__start()` and `import__find__load__done()` static markers can be used to trace module imports. (Contributed by Christian Heimes in bpo-31574.)

The fields `name` and `doc` of structures `PyMemberDef`, `PyGetSetDef`, `PyStructSequence_Field`, `PyStructSequence_Desc`, and `wrapperbase` are now of type `const char *` rather of `char *`. (Contributed by Serhiy Storchaka in bpo-28761.)

The result of `PyUnicode_AsUTF8AndSize()` and `PyUnicode_AsUTF8()` is now of type `const char *` rather of `char *`. (Contributed by Serhiy Storchaka in bpo-28769.)

The result of `PyMapping_Keys()`, `PyMapping_Values()` and `PyMapping_Items()` is now always a list, rather than a list or a tuple. (Contributed by Oren Milman in bpo-28280.)

Added functions `PySlice_Unpack()` and `PySlice_AdjustIndices()`. (Contributed by Serhiy Storchaka in bpo-27867.)

`PyOS_AfterFork()` is deprecated in favour of the new functions `PyOS_BeforeFork()`, `PyOS_AfterFork_Parent()` and `PyOS_AfterFork_Child()`. (Contributed by Antoine Pitrou in bpo-16500.)

The `PyExc_RecursionErrorInst` singleton that was part of the public API has been removed as its members being never cleared may cause a segfault during finalization of the interpreter. Contributed by Xavier de Gaye in bpo-22898 and bpo-30697.

Added C API support for timezones with timezone constructors `PyTimeZone_FromOffset()` and `PyTimeZone_FromOffsetAndName()`, and access to the UTC singleton with `PyDateTime_TimeZone_UTC`. Contributed by Paul Ganssle in bpo-10381.

The type of results of `PyThread_start_new_thread()` and `PyThread_get_thread_ident()`, and the *id* parameter of `PyThreadState_SetAsyncExc()` changed from `long` to `unsigned long`. (Contributed by Serhiy Storchaka in bpo-6532.)

`PyUnicode_AsWideCharString()` now raises a `ValueError` if the second argument is *NULL* and the `wchar_t*` string contains null characters. (Contributed by Serhiy Storchaka in bpo-30708.)

Changes to the startup sequence and the management of dynamic memory allocators mean that the long documented requirement to call `Py_Initialize()` before calling most C API functions is now relied on more heavily, and failing to abide by it may lead to segfaults in embedding applications. See the *Porting to Python 3.7* section in this document and the pre-init-safe section in the C API documentation for more details.

The new `PyInterpreterState_GetID()` returns the unique ID for a given interpreter. (Contributed by Eric Snow in bpo-29102.)

`Py_DecodeLocale()`, `Py_EncodeLocale()` now use the UTF-8 encoding when the *UTF-8 mode* is enabled. (Contributed by Victor Stinner in bpo-29240.)

`PyUnicode_DecodeLocaleAndSize()` and `PyUnicode_EncodeLocale()` now use the current locale encoding for `surrogateescape` error handler. (Contributed by Victor Stinner in bpo-29240.)

The *start* and *end* parameters of `PyUnicode_FindChar()` are now adjusted to behave like string slices. (Contributed by Xiang Zhang in bpo-28822.)

7

Support for building `--without-threads` has been removed. The `threading` module is now always available. (Contributed by Antoine Pitrou in bpo-31370.).

A full copy of libffi is no longer bundled for use when building the `_ctypes` module on non-OSX UNIX platforms. An installed copy of libffi is now required when building `_ctypes` on such platforms. (Contributed by Zachary Ware in bpo-27979.)

The Windows build process no longer depends on Subversion to pull in external sources, a Python script is used to download zipfiles from GitHub instead. If Python 3.6 is not found on the system (via `py -3.6`), NuGet is used to download a copy of 32-bit Python for this purpose. (Contributed by Zachary Ware in bpo-30450.)

The `ssl` module requires OpenSSL 1.0.2 or 1.1 compatible libssl. OpenSSL 1.0.1 has reached end of lifetime on 2016-12-31 and is no longer supported. LibreSSL is temporarily not supported as well. LibreSSL releases up to version 2.6.4 are missing required OpenSSL 1.0.2 APIs.

8

The overhead of calling many methods of various standard library classes implemented in C has been significantly reduced by porting more code to use the `METH_FASTCALL` convention. (Contributed by Victor Stinner in bpo-29300, bpo-29507, bpo-29452, and bpo-29286.)

Various optimizations have reduced Python startup time by 10% on Linux and up to 30% on macOS. (Contributed by Victor Stinner, INADA Naoki in bpo-29585, and Ivan Levkivskyi in bpo-31333.)

Method calls are now up to 20% faster due to the bytecode changes which avoid creating bound method instances. (Contributed by Yury Selivanov and INADA Naoki in bpo-26110.)

The `asyncio` module received a number of notable optimizations for commonly used functions:

- The `asyncio.get_event_loop()` function has been reimplemented in C to make it up to 15 times faster. (Contributed by Yury Selivanov in bpo-32296.)
- `asyncio.Future` callback management has been optimized. (Contributed by Yury Selivanov in bpo-32348.)
- `asyncio.gather()` is now up to 15% faster. (Contributed by Yury Selivanov in bpo-32355.)
- `asyncio.sleep()` is now up to 2 times faster when the *delay* argument is zero or negative. (Contributed by Andrew Svetlov in bpo-32351.)
- The performance overhead of asyncio debug mode has been reduced. (Contributed by Antoine Pitrou in bpo-31970.)

As a result of *PEP 560 work*, the import time of `typing` has been reduced by a factor of 7, and many typing operations are now faster. (Contributed by Ivan Levkivskyi in bpo-32226.)

`sorted()` and `list.sort()` have been optimized for common cases to be up to 40-75% faster. (Contributed by Elliot Gorokhovsky in bpo-28685.)

`dict.copy()` is now up to 5.5 times faster. (Contributed by Yury Selivanov in bpo-31179.)

`hasattr()` and `getattr()` are now about 4 times faster when *name* is not found and *obj* does not override `object.__getattr__()` or `object.__getattribute__()`. (Contributed by INADA Naoki in bpo-32544.)

Searching for certain Unicode characters (like Ukrainian capital ” ”) in a string was up to 25 times slower than searching for other characters. It is now only 3 times slower in the worst case. (Contributed by Serhiy Storchaka in bpo-24821.)

The `collections.namedtuple()` factory has been reimplemented to make the creation of named tuples 4 to 6 times faster. (Contributed by Jelle Zijlstra with further improvements by INADA Naoki, Serhiy Storchaka, and Raymond Hettinger in bpo-28638.)

`date.fromordinal()` and `date.fromtimestamp()` are now up to 30% faster in the common case. (Contributed by Paul Ganssle in bpo-32403.)

The `os.fwalk()` function is now up to 2 times faster thanks to the use of `os.scandir()`. (Contributed by Serhiy Storchaka in bpo-25996.)

The speed of the `shutil.rmtree()` function has been improved by 20–40% thanks to the use of the `os.scandir()` function. (Contributed by Serhiy Storchaka in bpo-28564.)

Optimized case-insensitive matching and searching of `regular expressions`. Searching some patterns can now be up to 20 times faster. (Contributed by Serhiy Storchaka in bpo-30285.)

`re.compile()` now converts `flags` parameter to int object if it is `RegexFlag`. It is now as fast as Python 3.5, and faster than Python 3.6 by about 10% depending on the pattern. (Contributed by INADA Naoki in bpo-31671.)

The `modify()` methods of classes `selectors.EpollSelector`, `selectors.PollSelector` and `selectors.DevpollSelector` may be around 10% faster under heavy loads. (Contributed by Giampaolo Rodola' in bpo-30014)

Constant folding has been moved from the peephole optimizer to the new AST optimizer, which is able perform optimizations more consistently. (Contributed by Eugene Toder and INADA Naoki in bpo-29469 and bpo-11549.)

Most functions and methods in `abc` have been rewritten in C. This makes creation of abstract base classes, and calling `isinstance()` and `issubclass()` on them 1.5x faster. This also reduces Python start-up time by up to 10%. (Contributed by Ivan Levkivskyi and INADA Naoki in bpo-31333)

Significant speed improvements to alternate constructors for `datetime.date` and `datetime.datetime` by using fast-path constructors when not constructing subclasses. (Contributed by Paul Ganssle in bpo-32403)

The speed of comparison of `array.array` instances has been improved considerably in certain cases. It is now from 10x to 70x faster when comparing arrays holding values of the same integer type. (Contributed by Adrian Wielgosik in bpo-24700.)

The `math.erf()` and `math.erfc()` functions now use the (faster) C library implementation on most platforms. (Contributed by Serhiy Storchaka in bpo-26121.)

9 CPython

- Trace hooks may now opt out of receiving the `line` and opt into receiving the `opcode` events from the interpreter by setting the corresponding new `f_trace_lines` and `f_trace_opcodes` attributes on the frame being traced. (Contributed by Nick Coghlan in bpo-31344.)
- Fixed some consistency problems with namespace package module attributes. Namespace module objects now have an `__file__` that is set to `None` (previously unset), and their `__spec__.origin` is also set to `None` (previously the string `"namespace"`). See bpo-32305. Also, the namespace module object's `__spec__.loader` is set to the same value as `__loader__` (previously, the former was set to `None`). See bpo-32303.
- The `locals()` dictionary now displays in the lexical order that variables were defined. Previously, the order was undefined. (Contributed by Raymond Hettinger in bpo-32690.)
- The `distutils upload` command no longer tries to change CR end-of-line characters to CRLF. This fixes a corruption issue with sdists that ended with a byte equivalent to CR. (Contributed by Bo Bayles in bpo-32304.)

10 Python

Yield expressions (both `yield` and `yield from` clauses) are now deprecated in comprehensions and generator expressions (aside from the iterable expression in the leftmost `for` clause). This ensures that comprehensions always immediately return a container of the appropriate type (rather than potentially returning a generator iterator object), while generator expressions won't attempt to interleave their implicit output with the output from any explicit yield expressions. In Python 3.7, such expressions emit `DeprecationWarning` when compiled, in Python 3.8 this will be a `SyntaxError`. (Contributed by Serhiy Storchaka in bpo-10544.)

Returning a subclass of `complex` from `object.__complex__()` is deprecated and will be an error in future Python versions. This makes `__complex__()` consistent with `object.__int__()` and `object.__float__()`. (Contributed by Serhiy Storchaka in bpo-28894.)

11 Python

11.1 aifc

`aifc.openfp()` has been deprecated and will be removed in Python 3.9. Use `aifc.open()` instead. (Contributed by Brian Curtin in bpo-31985.)

11.2 asyncio

Support for directly `await`-ing instances of `asyncio.Lock` and other asyncio synchronization primitives has been deprecated. An asynchronous context manager must be used in order to acquire and release the synchronization resource. (Contributed by Andrew Svetlov in bpo-32253.)

The `asyncio.Task.current_task()` and `asyncio.Task.all_tasks()` methods have been deprecated. (Contributed by Andrew Svetlov in bpo-32250.)

11.3 collections

In Python 3.8, the abstract base classes in `collections.abc` will no longer be exposed in the regular `collections` module. This will help create a clearer distinction between the concrete classes and the abstract base classes. (Contributed by Serhiy Storchaka in bpo-25988.)

11.4 dbm

`dbm.dumb` now supports reading read-only files and no longer writes the index file when it is not changed. A deprecation warning is now emitted if the index file is missing and recreated in the `'r'` and `'w'` modes (this will be an error in future Python releases). (Contributed by Serhiy Storchaka in bpo-28847.)

11.5 enum

In Python 3.8, attempting to check for non-Enum objects in `Enum` classes will raise a `TypeError` (e.g. `1 in Color`); similarly, attempting to check for non-Flag objects in a `Flag` member will raise `TypeError` (e.g. `1 in Perm.RW`); currently, both operations return `False` instead. (Contributed by Ethan Furman in bpo-33217.)

11.6 gettext

Using non-integer value for selecting a plural form in `gettext` is now deprecated. It never correctly worked. (Contributed by Serhiy Storchaka in bpo-28692.)

11.7 importlib

Methods `MetaPathFinder.find_module()` (replaced by `MetaPathFinder.find_spec()`) and `PathEntryFinder.find_loader()` (replaced by `PathEntryFinder.find_spec()`) both deprecated in Python 3.4 now emit `DeprecationWarning`. (Contributed by Matthias Bussonnier in bpo-29576)

The `importlib.abc.ResourceLoader` ABC has been deprecated in favour of `importlib.abc.ResourceReader`.

11.8 locale

`locale.format()` has been deprecated, use `locale.format_string()` instead. (Contributed by Garvit in bpo-10379.)

11.9 macpath

The `macpath` is now deprecated and will be removed in Python 3.8. (Contributed by Chi Hsuan Yen in bpo-9850.)

11.10 threading

`dummy_threading` and `_dummy_thread` have been deprecated. It is no longer possible to build Python with threading disabled. Use `threading` instead. (Contributed by Antoine Pitrou in bpo-31370.)

11.11 socket

The silent argument value truncation in `socket.htons()` and `socket.ntohs()` has been deprecated. In future versions of Python, if the passed argument is larger than 16 bits, an exception will be raised. (Contributed by Oren Milman in bpo-28332.)

11.12 ssl

`ssl.wrap_socket()` is deprecated. Use `ssl.SSLContext.wrap_socket()` instead. (Contributed by Christian Heimes in bpo-28124.)

11.13 sunau

`sunau.openfp()` has been deprecated and will be removed in Python 3.9. Use `sunau.open()` instead. (Contributed by Brian Curtin in bpo-31985.)

11.14 sys

Deprecated `sys.set_coroutine_wrapper()` and `sys.get_coroutine_wrapper()`.

The undocumented `sys.callstats()` function has been deprecated and will be removed in a future Python version. (Contributed by Victor Stinner in bpo-28799.)

11.15 wave

`wave.openfp()` has been deprecated and will be removed in Python 3.9. Use `wave.open()` instead. (Contributed by Brian Curtin in bpo-31985.)

12 Deprecated functions and types of the C API

Function `PySlice_GetIndicesEx()` is deprecated and replaced with a macro if `Py_LIMITED_API` is not set or set to a value in the range between `0x03050400` and `0x03060000` (not inclusive), or is `0x03060100` or higher. (Contributed by Serhiy Storchaka in bpo-27867.)

`PyOS_AfterFork()` has been deprecated. Use `PyOS_BeforeFork()`, `PyOS_AfterFork_Parent()` or `PyOS_AfterFork_Child()` instead. (Contributed by Antoine Pitrou in bpo-16500.)

13 Platform Support Removals

- FreeBSD 9 and older are no longer officially supported.
- For full Unicode support, including within extension modules, *nix platforms are now expected to provide at least one of `C.UTF-8` (full locale), `C.utf8` (full locale) or `UTF-8` (`LC_CTYPE`-only locale) as an alternative to the legacy `ASCII`-based `C` locale.
- OpenSSL 0.9.8 and 1.0.1 are no longer supported, which means building CPython 3.7 with SSL/TLS support on older platforms still using these versions requires custom build options that link to a more recent version of OpenSSL.

  Notably, this issue affects the Debian 8 (aka "jessie") and Ubuntu 14.04 (aka "Trusty") LTS Linux distributions, as they still use OpenSSL 1.0.1 by default.

  Debian 9 ("stretch") and Ubuntu 16.04 ("xenial"), as well as recent releases of other LTS Linux releases (e.g. RHEL/CentOS 7.5, SLES 12-SP3), use OpenSSL 1.0.2 or later, and remain supported in the default build configuration.

  CPython's own CI configuration file provides an example of using the SSL compatibility testing infrastructure in CPython's test suite to build and link against OpenSSL 1.1.0 rather than an outdated system provided OpenSSL.

14 API and Feature Removals

The following features and APIs have been removed from Python 3.7:

- The `os.stat_float_times()` function has been removed. It was introduced in Python 2.3 for backward compatibility with Python 2.2, and was deprecated since Python 3.1.
- Unknown escapes consisting of `'\'` and an ASCII letter in replacement templates for `re.sub()` were deprecated in Python 3.5, and will now cause an error.
- Removed support of the *exclude* argument in `tarfile.TarFile.add()`. It was deprecated in Python 2.7 and 3.2. Use the *filter* argument instead.
- The `splitunc()` function in the `ntpath` module was deprecated in Python 3.1, and has now been removed. Use the `splitdrive()` function instead.
- `collections.namedtuple()` no longer supports the *verbose* parameter or `_source` attribute which showed the generated source code for the named tuple class. This was part of an optimization designed to speed-up class creation. (Contributed by Jelle Zijlstra with further improvements by INADA Naoki, Serhiy Storchaka, and Raymond Hettinger in bpo-28638.)
- Functions `bool()`, `float()`, `list()` and `tuple()` no longer take keyword arguments. The first argument of `int()` can now be passed only as positional argument.

- Removed previously deprecated in Python 2.4 classes `Plist`, `Dict` and `_InternalDict` in the `plistlib` module. Dict values in the result of functions `readPlist()` and `readPlistFromBytes()` are now normal dicts. You no longer can use attribute access to access items of these dictionaries.
- The `asyncio.windows_utils.socketpair()` function has been removed. Use the `socket.socketpair()` function instead, it is available on all platforms since Python 3.5. `asyncio.windows_utils.socketpair` was just an alias to `socket.socketpair` on Python 3.5 and newer.
- `asyncio` no longer exports the `selectors` and `_overlapped` modules as `asyncio.selectors` and `asyncio._overlapped`. Replace `from asyncio import selectors` with `import selectors`.
- Direct instantiation of `ssl.SSLSocket` and `ssl.SSLObject` objects is now prohibited. The constructors were never documented, tested, or designed as public constructors. Users were supposed to use `ssl.wrap_socket()` or `ssl.SSLContext`. (Contributed by Christian Heimes in bpo-32951.)
- The unused `distutils install_misc` command has been removed. (Contributed by Eric N. Vander Weele in bpo-29218.)

15 Module Removals

The `fpectl` module has been removed. It was never enabled by default, never worked correctly on x86-64, and it changed the Python ABI in ways that caused unexpected breakage of C extensions. (Contributed by Nathaniel J. Smith in bpo-29137.)

16 Windows-only Changes

The python launcher, (py.exe), can accept 32 & 64 bit specifiers **without** having to specify a minor version as well. So `py -3-32` and `py -3-64` become valid as well as `py -3.7-32`, also the *-m*-64 and *-m.n*-64 forms are now accepted to force 64 bit python even if 32 bit would have otherwise been used. If the specified version is not available py.exe will error exit. (Contributed by Steve Barnes in bpo-30291.)

The launcher can be run as `py -0` to produce a list of the installed pythons, *with default marked with an asterisk*. Running `py -0p` will include the paths. If py is run with a version specifier that cannot be matched it will also print the *short form* list of available specifiers. (Contributed by Steve Barnes in bpo-30362.)

17 Porting to Python 3.7

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

17.1 Changes in Python Behavior

- `async` and `await` names are now reserved keywords. Code using these names as identifiers will now raise a `SyntaxError`. (Contributed by Jelle Zijlstra in bpo-30406.)
- **PEP 479** is enabled for all code in Python 3.7, meaning that `StopIteration` exceptions raised directly or indirectly in coroutines and generators are transformed into `RuntimeError` exceptions. (Contributed by Yury Selivanov in bpo-32670.)
- `object.__aiter__()` methods can no longer be declared as asynchronous. (Contributed by Yury Selivanov in bpo-31709.)
- Due to an oversight, earlier Python versions erroneously accepted the following syntax:

```
f(1 for x in [1],)

class C(1 for x in [1]):
    pass
```

Python 3.7 now correctly raises a `SyntaxError`, as a generator expression always needs to be directly inside a set of parentheses and cannot have a comma on either side, and the duplication of the parentheses can be omitted only on calls. (Contributed by Serhiy Storchaka in bpo-32012 and bpo-32023.)

- When using the `-m` switch, the initial working directory is now added to `sys.path`, rather than an empty string (which dynamically denoted the current working directory at the time of each import). Any programs that are checking for the empty string, or otherwise relying on the previous behaviour, will need to be updated accordingly (e.g. by also checking for `os.getcwd()` or `os.path.dirname(__main__.__file__)`, depending on why the code was checking for the empty string in the first place).

17.2 Changes in the Python API

- `socketserver.ThreadingMixIn.server_close()` now waits until all non-daemon threads complete. Set the new `socketserver.ThreadingMixIn.block_on_close` class attribute to `False` to get the pre-3.7 behaviour. (Contributed by Victor Stinner in bpo-31233 and bpo-33540.)
- `socketserver.ForkingMixIn.server_close()` now waits until all child processes complete. Set the new `socketserver.ForkingMixIn.block_on_close` class attribute to `False` to get the pre-3.7 behaviour. (Contributed by Victor Stinner in bpo-31151 and bpo-33540.)
- The `locale.localeconv()` function now temporarily sets the `LC_CTYPE` locale to the value of `LC_NUMERIC` in some cases. (Contributed by Victor Stinner in bpo-31900.)
- `pkgutil.walk_packages()` now raises a `ValueError` if *path* is a string. Previously an empty list was returned. (Contributed by Sanyam Khurana in bpo-24744.)
- A format string argument for `string.Formatter.format()` is now positional-only. Passing it as a keyword argument was deprecated in Python 3.5. (Contributed by Serhiy Storchaka in bpo-29193.)
- Attributes `key`, `value` and `coded_value` of class `http.cookies.Morsel` are now read-only. Assigning to them was deprecated in Python 3.5. Use the `set()` method for setting them. (Contributed by Serhiy Storchaka in bpo-29192.)
- The *mode* argument of `os.makedirs()` no longer affects the file permission bits of newly-created intermediate-level directories. To set their file permission bits you can set the umask before invoking `makedirs()`. (Contributed by Serhiy Storchaka in bpo-19930.)
- The `struct.Struct.format` type is now `str` instead of `bytes`. (Contributed by Victor Stinner in bpo-21071.)
- `parse_multipart()` now accepts the *encoding* and *errors* arguments and returns the same results as `FieldStorage`: for non-file fields, the value associated to a key is a list of strings, not bytes. (Contributed by Pierre Quentel in bpo-29979.)
- Due to internal changes in `socket`, calling `socket.fromshare()` on a socket created by `socket.share` in older Python versions is not supported.
- `repr` for `BaseException` has changed to not include the trailing comma. Most exceptions are affected by this change. (Contributed by Serhiy Storchaka in bpo-30399.)
- `repr` for `datetime.timedelta` has changed to include the keyword arguments in the output. (Contributed by Utkarsh Upadhyay in bpo-30302.)

- Because `shutil.rmtree()` is now implemented using the `os.scandir()` function, the user specified handler *onerror* is now called with the first argument `os.scandir` instead of `os.listdir` when listing the directory is failed.

- Support for nested sets and set operations in regular expressions as in Unicode Technical Standard #18 might be added in the future. This would change the syntax. To facilitate this future change a `FutureWarning` will be raised in ambiguous cases for the time being. That include sets starting with a literal `'['` or containing literal character sequences `'--'`, `'&&'`, `'~~'`, and `'||'`. To avoid a warning, escape them with a backslash. (Contributed by Serhiy Storchaka in bpo-30349.)

- The result of splitting a string on a `regular expression` that could match an empty string has been changed. For example splitting on `r'\s*'` will now split not only on whitespaces as it did previously, but also on empty strings before all non-whitespace characters and just before the end of the string. The previous behavior can be restored by changing the pattern to `r'\s+'`. A `FutureWarning` was emitted for such patterns since Python 3.5.

  For patterns that match both empty and non-empty strings, the result of searching for all matches may also be changed in other cases. For example in the string `'a\n\n'`, the pattern `r'(?m)^\s*?$'` will not only match empty strings at positions 2 and 3, but also the string `'\n'` at positions 2–3. To match only blank lines, the pattern should be rewritten as `r'(?m)^[^\S\n]*$'`.

  `re.sub()` now replaces empty matches adjacent to a previous non-empty match. For example `re.sub('x*', '-', 'abxd')` returns now `'-a-b--d-'` instead of `'-a-b-d-'` (the first minus between 'b' and 'd' replaces 'x', and the second minus replaces an empty string between 'x' and 'd').

  (Contributed by Serhiy Storchaka in bpo-25054 and bpo-32308.)

- Change `re.escape()` to only escape regex special characters instead of escaping all characters other than ASCII letters, numbers, and `'_'`. (Contributed by Serhiy Storchaka in bpo-29995.)

- `tracemalloc.Traceback` frames are now sorted from oldest to most recent to be more consistent with `traceback`. (Contributed by Jesse Bakker in bpo-32121.)

- On OSes that support `socket.SOCK_NONBLOCK` or `socket.SOCK_CLOEXEC` bit flags, the `socket.type` no longer has them applied. Therefore, checks like `if sock.type == socket.SOCK_STREAM` work as expected on all platforms. (Contributed by Yury Selivanov in bpo-32331.)

- On Windows the default for the *close_fds* argument of `subprocess.Popen` was changed from `False` to `True` when redirecting the standard handles. If you previously depended on handles being inherited when using `subprocess.Popen` with standard io redirection, you will have to pass `close_fds=False` to preserve the previous behaviour, or use `STARTUPINFO.lpAttributeList`.

- `importlib.machinery.PathFinder.invalidate_caches()` – which implicitly affects `importlib.invalidate_caches()` – now deletes entries in `sys.path_importer_cache` which are set to `None`. (Contributed by Brett Cannon in bpo-33169.)

- In `asyncio`, `loop.sock_recv()`, `loop.sock_sendall()`, `loop.sock_accept()`, `loop.getaddrinfo()`, `loop.getnameinfo()` have been changed to be proper coroutine methods to match their documentation. Previously, these methods returned `asyncio.Future` instances. (Contributed by Yury Selivanov in bpo-32327.)

- `asyncio.Server.sockets` now returns a copy of the internal list of server sockets, instead of returning it directly. (Contributed by Yury Selivanov in bpo-32662.)

- `Struct.format` is now a `str` instance instead of a `bytes` instance. (Contributed by Victor Stinner in bpo-21071.)

- `ast.literal_eval()` is now stricter. Addition and subtraction of arbitrary numbers are no longer allowed. (Contributed by Serhiy Storchaka in bpo-31778.)

- `Calendar.itermonthdates` will now consistently raise an exception when a date falls outside of the `0001-01-01` through `9999-12-31` range. To support applications that cannot tolerate such exceptions, the new `Calendar.itermonthdays3` and `Calendar.itermonthdays4` can be used. The new methods return tuples and are not restricted by the range supported by `datetime.date`. (Contributed by Alexander Belopolsky in bpo-28292.)
- `collections.ChainMap` now preserves the order of the underlying mappings. (Contributed by Raymond Hettinger in bpo-32792.)
- The `submit()` method of `concurrent.futures.ThreadPoolExecutor` and `concurrent.futures.ProcessPoolExecutor` now raises a `RuntimeError` if called during interpreter shutdown. (Contributed by Mark Nemec in bpo-33097.)
- The `configparser.ConfigParser` constructor now uses `read_dict()` to process the default values, making its behavior consistent with the rest of the parser. Non-string keys and values in the defaults dictionary are now being implicitly converted to strings. (Contributed by James Tocknell in bpo-23835.)
- Several undocumented internal imports were removed. One example is that `os.errno` is no longer available; use `import errno` directly instead. Note that such undocumented internal imports may be removed any time without notice, even in micro version releases.

17.3 C API

The function `PySlice_GetIndicesEx()` is considered unsafe for resizable sequences. If the slice indices are not instances of `int`, but objects that implement the `__index__()` method, the sequence can be resized after passing its length to `PySlice_GetIndicesEx()`. This can lead to returning indices out of the length of the sequence. For avoiding possible problems use new functions `PySlice_Unpack()` and `PySlice_AdjustIndices()`. (Contributed by Serhiy Storchaka in bpo-27867.)

17.4 CPython

There are two new opcodes: `LOAD_METHOD` and `CALL_METHOD`. (Contributed by Yury Selivanov and INADA Naoki in bpo-26110.)

The `STORE_ANNOTATION` opcode has been removed. (Contributed by Mark Shannon in bpo-32550.)

17.5 Windows-only Changes

The file used to override `sys.path` is now called `<python-executable>._pth` instead of `'sys.path'`. See finding_modules for more information. (Contributed by Steve Dower in bpo-28137.)

17.6 CPython

In preparation for potential future changes to the public CPython runtime initialization API (see **PEP 432** for an initial, but somewhat outdated, draft), CPython's internal startup and configuration management logic has been significantly refactored. While these updates are intended to be entirely transparent to both embedding applications and users of the regular CPython CLI, they're being mentioned here as the refactoring changes the internal order of various operations during interpreter startup, and hence may uncover previously latent defects, either in embedding applications, or in CPython itself. (Initially contributed by Nick Coghlan and Eric Snow as part of bpo-22257, and further updated by Nick, Eric, and Victor Stinner in a number of other issues). Some known details affected:

- `PySys_AddWarnOptionUnicode()` is not currently usable by embedding applications due to the requirement to create a Unicode object prior to calling *Py_Initialize*. Use `PySys_AddWarnOption()` instead.
- warnings filters added by an embedding application with `PySys_AddWarnOption()` should now more consistently take precedence over the default filters set by the interpreter

Due to changes in the way the default warnings filters are configured, setting `Py_BytesWarningFlag` to a value greater than one is no longer sufficient to both emit `BytesWarning` messages and have them converted to exceptions. Instead, the flag must be set (to cause the warnings to be emitted in the first place), and an explicit `error::BytesWarning` warnings filter added to convert them to exceptions.

Due to a change in the way docstrings are handled by the compiler, the implicit `return None` in a function body consisting solely of a docstring is now marked as occurring on the same line as the docstring, not on the function's header line.

The current exception state has been moved from the frame object to the co-routine. This simplified the interpreter and fixed a couple of obscure bugs caused by having swap exception state when entering or exiting a generator. (Contributed by Mark Shannon in bpo-25612.)

18 Python 3.7.1

Starting in 3.7.1, `Py_Initialize()` now consistently reads and respects all of the same environment settings as `Py_Main()` (in earlier Python versions, it respected an ill-defined subset of those environment variables, while in Python 3.7.0 it didn't read any of them due to bpo-34247). If this behavior is unwanted, set `Py_IgnoreEnvironmentFlag` to 1 before calling `Py_Initialize()`.

In 3.7.1 the C API for Context Variables was updated to use `PyObject` pointers. See also bpo-34762.

`xml.dom.minidom` and `xml.sax` modules no longer process external entities by default. See also bpo-17239.

In 3.7.1 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.)

19 Python 3.7.2

In 3.7.2, `venv` on Windows no longer copies the original binaries, but creates redirector scripts named `python.exe` and `pythonw.exe` instead. This resolves a long standing issue where all virtual environments would have to be upgraded or recreated with each Python update. However, note that this release will still require recreation of virtual environments in order to get the new scripts.

Non-alphabetical

PYTHONBREAKPOINT, 6
PYTHONCOERCECLOCALE, 5
PYTHONDEVMODE, 8
PYTHONPROFILEIMPORTTIME, 8
PYTHONUTF8, 6
PYTHONWARNINGS, 19
SOURCE_DATE_EPOCH, 15

P

Python
PEP 11, 5
PEP 432, 30
PEP 479, 27
PEP 484, 7
PEP 526, 5
PEP 538, 5, 6
PEP 539, 6
PEP 540, 6
PEP 545, 8
PEP 552, 7
PEP 553, 6
PEP 557, 9
PEP 560, 7, 18
PEP 562, 6
PEP 563, 5
PEP 564, 6, 7, 17
PEP 565, 7
PEP 567, 8, 9
PEP 3107, 5
PYTHONBREAKPOINT, 6
PYTHONCOERCECLOCALE, 5
PYTHONDEVMODE, 8
PYTHONPROFILEIMPORTTIME, 8
PYTHONUTF8, 6
PYTHONWARNINGS, 19

R

RFC
RFC 2396, 18
RFC 3986, 18

S

SOURCE_DATE_EPOCH, 15