
What's New in Python

Release 3.5.0a3

A. M. Kuchling

March 30, 2015

Python Software Foundation

Email: docs@python.org

Contents

1	Summary – Release highlights	2
1.1	PEP 471 - <code>os.scandir()</code> function – a better and faster directory iterator	3
1.2	PEP 475: Retry system calls failing with <code>EINTR</code>	3
1.3	PEP 486: Make the Python Launcher aware of virtual environments	3
2	Other Language Changes	3
3	New Modules	3
3.1	<code>zipapp</code>	3
4	Improved Modules	4
4.1	<code>argparse</code>	4
4.2	<code>cgi</code>	4
4.3	<code>code</code>	4
4.4	<code>compileall</code>	4
4.5	<code>contextlib</code>	4
4.6	<code>difflib</code>	4
4.7	<code>distutils</code>	4
4.8	<code>doctest</code>	4
4.9	<code>glob</code>	5
4.10	<code>imaplib</code>	5
4.11	<code>imghdr</code>	5
4.12	<code>importlib</code>	5
4.13	<code>inspect</code>	5
4.14	<code>ipaddress</code>	5
4.15	<code>json</code>	5
4.16	<code>os</code>	6
4.17	<code>re</code>	6
4.18	<code>math</code>	6
4.19	<code>shutil</code>	6
4.20	<code>signal</code>	6
4.21	<code>smtplib</code>	6
4.22	<code>smtplib</code>	6
4.23	<code>sndhdr</code>	6
4.24	<code>socket</code>	7
4.25	<code>sysconfig</code>	7
4.26	<code>tarfile</code>	7
4.27	<code>time</code>	7

4.28	<code>urllib</code>	7
4.29	<code>wsgiref</code>	7
4.30	<code>xmlrpc</code>	7
4.31	<code>faulthandler</code>	7
4.32	<code>zipfile</code>	7
5	Optimizations	8
6	Build and C API Changes	8
7	Deprecated	8
7.1	Unsupported Operating Systems	8
7.2	Deprecated Python modules, functions and methods	8
7.3	Deprecated functions and types of the C API	9
7.4	Deprecated features	9
8	Removed	9
8.1	API and Feature Removals	9
9	Porting to Python 3.5	9
9.1	Changes in the Python API	9
9.2	Changes in the C API	10
	Index	11

Release 3.5.0a3

Date March 30, 2015

This article explains the new features in Python 3.5, compared to 3.4.

For full details, see the [Misc/NEWS](#) file.

Note: Prerelease users should be aware that this document is currently in draft form. It will be updated substantially as Python 3.5 moves towards release, so it's worth checking back even after reading earlier versions.

See also:

[PEP 478](#) - Python 3.5 Release Schedule

1 Summary – Release highlights

New syntax features:

- None yet.

New library modules:

- `zipapp`: *Improving Python ZIP Application Support* ([PEP 441](#)).

New built-in features:

- None yet.

Implementation improvements:

- When the `LC_TYPE` locale is the POSIX locale (C locale), `sys.stdin` and `sys.stdout` are now using the `surrogateescape` error handler, instead of the `strict` error handler ([issue 19977](#)).

Significantly Improved Library Modules:

- None yet.

Security improvements:

- None yet.

Please read on for a comprehensive list of user-facing changes.

1.1 PEP 471 - `os.scandir()` function – a better and faster directory iterator

PEP 471 adds a new directory iteration function, `os.scandir()`, to the standard library. Additionally, `os.walk()` is now implemented using `os.scandir()`, which speeds it up by 3-5 times on POSIX systems and by 7-20 times on Windows systems.

PEP and implementation written by Ben Hoyt with the help of Victor Stinner.

See also:

PEP 471 – `os.scandir()` function – a better and faster directory iterator

1.2 PEP 475: Retry system calls failing with EINTR

PEP 475 adds support for automatic retry of system calls failing with EINTR: this means that user code doesn't have to deal with EINTR or InterruptedError manually, and should make it more robust against asynchronous signal reception.

See also:

PEP 475 – Retry system calls failing with EINTR

1.3 PEP 486: Make the Python Launcher aware of virtual environments

PEP 486 makes the Windows launcher (see **PEP 397**) aware of an active virtual environment. When the default interpreter would be used and the `VIRTUAL_ENV` environment variable is set, the interpreter in the virtual environment will be used.

See also:

PEP 486 – Make the Python Launcher aware of virtual environments

2 Other Language Changes

Some smaller changes made to the core Python language are:

- Added the `'namereplace'` error handlers. The `'backslashreplace'` error handlers now works with decoding and translating. (Contributed by Serhiy Storchaka in [issue 19676](#) and [issue 22286](#).)
- The `-b` option now affects comparisons of `bytes` with `int`. (Contributed by Serhiy Storchaka in [issue 23681](#))

3 New Modules

3.1 zipapp

The new `zipapp` module (specified in **PEP 441**) provides an API and command line tool for creating executable Python Zip Applications, which were introduced in Python 2.6 in [issue 1739468](#) but which were not well publicised, either at the time or since.

With the new module, bundling your application is as simple as putting all the files, including a `__main__.py` file, into a directory `myapp` and running:

```
$ python -m zipapp myapp
$ python myapp.pyz
```

4 Improved Modules

4.1 argparse

- `ArgumentParser` now allows to disable *abbreviated usage* of long options by setting `allow_abbrev` to `False`. (Contributed by Jonathan Paugh, Steven Bethard, paul j3 and Daniel Eriksson.)

4.2 cgi

- `FieldStorage` now supports the context management protocol. (Contributed by Berker Peksag in [issue 20289](#).)

4.3 code

- The `code.InteractiveInterpreter.showtraceback()` method now prints the full chained traceback, just like the interactive interpreter. (Contributed by Claudiu Popa in [issue 17442](#).)

4.4 compileall

- `compileall.compile_dir()` and `compileall`'s command-line interface can now do parallel byte-code compilation. (Contributed by Claudiu Popa in [issue 16104](#).)

4.5 contextlib

- The new `contextlib.redirect_stderr()` context manager (similar to `contextlib.redirect_stdout()`) makes it easier for utility scripts to handle inflexible APIs that write their output to `sys.stderr` and don't provide any options to redirect it. (Contributed by Berker Peksag in [issue 22389](#).)

4.6 difflib

- The charset of the HTML document generated by `difflib.HtmlDiff.make_file()` can now be customized by using `charset` keyword-only parameter. The default charset of HTML document changed from `'ISO-8859-1'` to `'utf-8'`. (Contributed by Berker Peksag in [issue 2052](#).)

4.7 distutils

- The `build` and `build_ext` commands now accept a `-j` option to enable parallel building of extension modules. (Contributed by Antoine Pitrou in [issue 5309](#).)

4.8 doctest

- `doctest.DocTestSuite()` returns an empty `unittest.TestSuite` if *module* contains no docstrings instead of raising `ValueError`. (Contributed by Glenn Jones in [issue 15916](#).)

4.9 glob

- `iglob()` and `glob()` now support recursive search in subdirectories using the “**” pattern. (Contributed by Serhiy Storchaka in [issue 13968](#).)

4.10 imaplib

- IMAP4 now supports the context management protocol. When used in a `with` statement, the IMAP4 `LOGOUT` command will be called automatically at the end of the block. (Contributed by Tarek Ziadé and Serhiy Storchaka in [issue 4972](#).)

4.11 imghdr

- `what()` now recognizes the [OpenEXR](#) format. (Contributed by Martin Vignali and Claudiu Popa in [issue 20295](#).)

4.12 importlib

- `importlib.util.LazyLoader` allows for the lazy loading of modules in applications where startup time is paramount. (Contributed by Brett Cannon in [issue 17621](#).)
- `importlib.abc.InspectLoader.source_to_code()` is now a static method to make it easier to work with source code in a string. With a module object that you want to initialize you can then use `exec(code, module.__dict__)` to execute the code in the module.
- `importlib.util.module_from_spec()` is now the preferred way to create a new module. Compared to `types.ModuleType`, this new function will set the various import-controlled attributes based on the passed-in spec object.

4.13 inspect

- `inspect.Signature` and `inspect.Parameter` are now picklable and hashable. (Contributed by Yuri Selivanov in [issue 20726](#) and [issue 20334](#).)
- New class method `inspect.Signature.from_callable()`, which makes subclassing of `Signature` easier. (Contributed by Yuri Selivanov and Eric Snow in [issue 17373](#).)

4.14 ipaddress

- `ipaddress.IPv4Network` and `ipaddress.IPv6Network` now accept an `(address, netmask)` tuple argument, so as to easily construct network objects from existing addresses. (Contributed by Peter Moody and Antoine Pitrou in [issue 16531](#).)

4.15 json

- The output of `json.tool` command line interface is now in the same order as the input. Use the `--sort-keys` option to sort the output of dictionaries alphabetically by key. (Contributed by Berker Peksag in [issue 21650](#).)
- JSON decoder now raises `json.JSONDecodeError` instead of `ValueError`. (Contributed by Serhiy Storchaka in [issue 19361](#).)

4.16 os

- New `os.scandir()` function that exposes file information from the operating system when listing a directory. `os.scandir()` returns an iterator of `os.DirEntry` objects corresponding to the entries in the directory given by *path*. (Contributed by Ben Hoyt with the help of Victor Stinner in [issue 22524](#).)
- `os.stat_result` now has a `st_file_attributes` attribute on Windows. (Contributed by Ben Hoyt in [issue 21719](#).)

4.17 re

- Number of capturing groups in regular expression is no longer limited by 100. (Contributed by Serhiy Storchaka in [issue 22437](#).)
- Now unmatched groups are replaced with empty strings in `re.sub()` and `re.subn()`. (Contributed by Serhiy Storchaka in [issue 1519638](#).)

4.18 math

- `math.inf` and `math.nan` constants added. (Contributed by Mark Dickinson in [issue 23185](#).)

4.19 shutil

- `move()` now accepts a *copy_function* argument, allowing, for example, `copy()` to be used instead of the default `copy2()` if there is a need to ignore metadata. (Contributed by Claudiu Popa in [issue 19840](#).)

4.20 signal

- Different constants of `signal` module are now enumeration values using the `enum` module. This allows meaningful names to be printed during debugging, instead of integer “magic numbers”. (Contributed by Giampaolo Rodola’ in [issue 21076](#).)

4.21 smtpd

- Both `SMTPServer` and `smtpd.SMTPChannel` now accept a *decode_data* keyword to determine if the DATA portion of the SMTP transaction is decoded using the `utf-8` codec or is instead provided to `process_message()` as a byte string. The default is `True` for backward compatibility reasons, but will change to `False` in Python 3.6. (Contributed by Maciej Szulik in [issue 19662](#).)
- It is now possible to provide, directly or via name resolution, IPv6 addresses in the `SMTPServer` constructor, and have it successfully connect. (Contributed by Milan Oberkirch in [issue 14758](#).)
- `SMTPServer` now supports **RFC 6531** via the *enable_SMTPUTF8* constructor argument and a user-provided `process_smtputf8_message()` method.

4.22 smtplib

- A new `auth()` method provides a convenient way to implement custom authentication mechanisms. (Contributed by Milan Oberkirch in [issue 15014](#).)

4.23 sndhdr

- `what()` and `whathdr()` now return `namedtuple()`. (Contributed by Claudiu Popa in [issue 18615](#).)

4.24 socket

- New `socket.socket.sendfile()` method allows to send a file over a socket by using high-performance `os.sendfile()` function on UNIX resulting in uploads being from 2x to 3x faster than when using plain `socket.socket.send()`. (Contributed by Giampaolo Rodola' in [issue 17552](#).)

4.25 sysconfig

- The user scripts directory on Windows is now versioned. (Contributed by Paul Moore in [issue 23437](#).)

4.26 tarfile

- The `tarfile.open()` function now supports 'x' (exclusive creation) mode. (Contributed by Berker Peksag in [issue 21717](#).)

4.27 time

- The `time.monotonic()` function is now always available. (Contributed by Victor Stinner in [issue 22043](#).)

4.28 urllib

- A new `urllib.request.HTTPBasicPriorAuthHandler` allows HTTP Basic Authentication credentials to be sent unconditionally with the first HTTP request, rather than waiting for a HTTP 401 Unauthorized response from the server. (Contributed by Matej Cepl in [issue 19494](#).)

4.29 wsgiref

- `headers` parameter of `wsgiref.headers.Headers` is now optional. (Contributed by Pablo Torres Navarrete and SilentGhost in [issue 5800](#).)

4.30 xmlrpc

- `xmlrpc.client.ServerProxy` is now a *context manager*. (Contributed by Claudiu Popa in [issue 20627](#).)

4.31 faulthandler

- `enable()`, `register()`, `dump_traceback()` and `dump_traceback_later()` functions now accept file descriptors. (Contributed by Wei Wu in [issue 23566](#).)

4.32 zipfile

- Added support for writing ZIP files to unseekable streams. (Contributed by Serhiy Storchaka in [issue 23252](#).)
- The `zipfile.ZipFile.open()` function now supports 'x' (exclusive creation) mode. (Contributed by Serhiy Storchaka in [issue 21717](#).)

5 Optimizations

The following performance enhancements have been added:

- `os.walk()` has been sped up by 3-5x on POSIX systems and 7-20x on Windows. This was done using the new `os.scandir()` function, which exposes file information from the underlying `readdir` and `FindFirstFile/FindNextFile` system calls. (Contributed by Ben Hoyt with help from Victor Stinner in [issue 23605](#).)
- Construction of `bytes(int)` (filled by zero bytes) is faster and use less memory for large objects. `calloc()` is used instead of `malloc()` to allocate memory for these objects.
- Some operations on `IPv4Network` and `IPv6Network` have been massively sped up, such as `subnets()`, `supernet()`, `summarize_address_range()`, `collapse_addresses()`. The speed up can range from 3x to 15x. ([issue 21486](#), [issue 21487](#), [issue 20826](#))
- Many operations on `io.BytesIO` are now 50% to 100% faster. (Contributed by Serhiy Storchaka in [issue 15381](#) and David Wilson in [issue 22003](#).)
- `marshal.dumps()` is now faster (65%-85% with versions 3-4, 20-25% with versions 0-2 on typical data, and up to 5x in best cases). (Contributed by Serhiy Storchaka in [issue 20416](#) and [issue 23344](#).)

6 Build and C API Changes

Changes to Python's build process and to the C API include:

- New `calloc` functions:
 - `PyMem_RawCalloc()`
 - `PyMem_Calloc()`
 - `PyObject_Calloc()`
 - `_PyObject_GC_Calloc()`

7 Deprecated

7.1 Unsupported Operating Systems

- None yet.

7.2 Deprecated Python modules, functions and methods

- The `formatter` module has now graduated to full deprecation and is still slated for removal in Python 3.6.
- `smtpd` has in the past always decoded the `DATA` portion of email messages using the `utf-8` codec. This can now be controlled by the new `decode_data` keyword to `SMTPServer`. The default value is `True`, but this default is deprecated. Specify the `decode_data` keyword with an appropriate value to avoid the deprecation warning.
- Directly assigning values to the `key`, `value` and `coded_value` of `Morsel` objects is deprecated. Use the `set()` method instead. In addition, the undocumented `LegalChars` parameter of `set()` is deprecated, and is now ignored.
- Passing a format string as keyword argument `format_string` to the `format()` method of the `string.Formatter` class has been deprecated.

7.3 Deprecated functions and types of the C API

- None yet.

7.4 Deprecated features

- None yet.

8 Removed

8.1 API and Feature Removals

The following obsolete and previously deprecated APIs and features have been removed:

- The `__version__` attribute has been dropped from the `email` package. The email code hasn't been shipped separately from the `stdlib` for a long time, and the `__version__` string was not updated in the last few releases.
- The internal `Netrc` class in the `ftplib` module was deprecated in 3.4, and has now been removed. (Contributed by Matt Chaput in [issue 6623](#).)

9 Porting to Python 3.5

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

9.1 Changes in the Python API

- **PEP 475:** the following functions are now retried when interrupted instead of raising `InterruptedError` if the signal handler does not raise an exception:
 - `os.open()`, `open()`
 - `os.read()`, `os.write()`
 - `time.sleep()`
- Before Python 3.5, a `datetime.time` object was considered to be false if it represented midnight in UTC. This behavior was considered obscure and error-prone and has been removed in Python 3.5. See [issue 13936](#) for full details.
- `ssl.SSLSocket.send()` now raises either `ssl.SSLWantReadError` or `ssl.SSLWantWriteError` on a non-blocking socket if the operation would block. Previously, it would return 0. See [issue 20951](#).
- The `__name__` attribute of generator is now set from the function name, instead of being set from the code name. Use `gen.gi_code.co_name` to retrieve the code name. Generators also have a new `__qualname__` attribute, the qualified name, which is now used for the representation of a generator (`repr(gen)`). See [issue 21205](#).
- The deprecated “strict” mode and argument of `HTMLParser`, `HTMLParser.error()`, and the `HTMLParserError` exception have been removed. (Contributed by Ezio Melotti in [issue 15114](#).) The `convert_charrefs` argument of `HTMLParser` is now `True` by default. (Contributed by Berker Peksag in [issue 21047](#).)
- Although it is not formally part of the API, it is worth noting for porting purposes (ie: fixing tests) that error messages that were previously of the form “‘sometype’ does not support the buffer protocol” are now of the form “a bytes-like object is required, not ‘sometype’”. (Contributed by Ezio Melotti in [issue 16518](#).)

- If the current directory is set to a directory that no longer exists then `FileNotFoundError` will no longer be raised and instead `find_spec()` will return `None` **without** caching `None` in `sys.path_importer_cache` which is different than the typical case ([issue 22834](#)).
- HTTP status code and messages from `http.client` and `http.server` were refactored into a common `HTTPStatus` enum. The values in `http.client` and `http.server` remain available for backwards compatibility. (Contributed by Demian Brecht in [issue 21793](#).)
- When an import loader defines `exec_module()` it is now expected to also define `create_module()` (raises a `DeprecationWarning` now, will be an error in Python 3.6). If the loader inherits from `importlib.abc.Loader` then there is nothing to do, else simply define `create_module()` to return `None` ([issue 23014](#)).
- `re.split()` always ignored empty pattern matches, so the `'x*'` pattern worked the same as `'x+'`, and the `'\b'` pattern never worked. Now `re.split()` raises a warning if the pattern could match an empty string. For compatibility use patterns that never match an empty string (e.g. `'x+'` instead of `'x*'`). Patterns that could only match an empty string (such as `'\b'`) now raise an error.
- The `Morsel` dict-like interface has been made self consistent: `morsel` comparison now takes the key and value into account, `copy()` now results in a `Morsel` instance rather than a *dict*, and `update()` will no longer raise an exception if any of the keys in the update dictionary are invalid. In addition, the undocumented `LegalChars` parameter of `set()` is deprecated and is now ignored. ([issue 2211](#))

9.2 Changes in the C API

- The undocumented `format` member of the (non-public) `PyMemoryViewObject` structure has been removed.
All extensions relying on the relevant parts in `memoryobject.h` must be rebuilt.
- The `PyMemAllocator` structure was renamed to `PyMemAllocatorEx` and a new `calloc` field was added.
- Removed non-documented macro `PyObject_REPR` which leaked references. Use format character `%R` in `PyUnicode_FromFormat()`-like functions to format the `repr()` of the object.
- Because the lack of the `__module__` attribute breaks pickling and introspection, a deprecation warning now is raised for builtin type without the `__module__` attribute. Would be an `AttributeError` in future. ([issue 20204](#))

Index

P

Python Enhancement Proposals

PEP 397, 3

PEP 441, 2, 3

PEP 471, 3

PEP 475, 3, 9

PEP 478, 2

PEP 486, 3

R

RFC

RFC 6531, 6