What's New in Python

Release 3.10.0a1

A. M. Kuchling

November 03, 2020

Python Software Foundation
Email: docs@python.org

Contents

Release 3.10.0a1

Date November 03, 2020

This article explains the new features in Python 3.10, compared to 3.9.

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 3.10 moves towards release, so it's worth checking back even after reading earlier versions.

---

1 Summary – Release highlights

2 New Features

2.1 PEP 563: Postponed Evaluation of Annotations Becomes Default

In Python 3.7, postponed evaluation of annotations was added, to be enabled with a `from __future__ import annotations` directive. In 3.10 this became the default behavior, even without that future directive. With this being default, all annotations stored in `__annotations__` will be strings. If needed, annotations can be resolved at runtime using `typing.get_type_hints()`. See **PEP 563** for a full description. Also, the `inspect.signature()` will try to resolve types from now on, and when it fails it will fall back to showing the string annotations. (Contributed by Batuhan Taskaya in bpo-38605.)

- The `int` type has a new method `int.bit_count()`, returning the number of ones in the binary expansion of a given integer, also known as the population count. (Contributed by Niklas Fiekas in bpo-29882.)
- The views returned by `dict.keys()`, `dict.values()` and `dict.items()` now all have a `mapping` attribute that gives a `types.MappingProxyType` object wrapping the original dictionary. (Contributed by Dennis Sweeney in bpo-40890.)
- **PEP 618**: The `zip()` function now has an optional `strict` flag, used to require that all the iterables have an equal length.

2.2 PEP 613: TypeAlias Annotation

PEP 484 introduced the concept of type aliases, only requiring them to be top-level unannotated assignments. This simplicity sometimes made it difficult for type checkers to distinguish between type aliases and ordinary assignments, especially when forward references or invalid types were involved. Compare:

```
StrCache = 'Cache[str]'  # a type alias
LOG_PREFIX = 'LOG[DEBUG]'  # a module constant
```

Now the `typing` module has a special annotation `TypeAlias` to declare type aliases more explicitly:

```
StrCache: TypeAlias = 'Cache[str]'  # a type alias
LOG_PREFIX = 'LOG[DEBUG]'  # a module constant
```

See **PEP 613** for more details.

(Contributed by Mikhail Golubev in bpo-41923.)

2.3 PEP604: New Type Union Operator

A new type union operator was introduced which enables the syntax `X | Y`. This provides a cleaner way of expressing 'either type X or type Y' instead of using `typing.Union`, especially in type hints (annotations).

In previous versions of Python, to apply a type hint for functions accepting arguments of multiple types, `typing.Union` was used:

```
def square(number: Union[int, float]) -> Union[int, float]:
    return number ** 2
```

Type hints can now be written in a more succinct manner:

```
def square(number: int | float) -> int | float:
    return number ** 2
```

See **PEP 604** for more details.

(Contributed by Maggie Moss and Philippe Prados in bpo-41428.)

3 Other Language Changes

- Builtin and extension functions that take integer arguments no longer accept `Decimals`, `Fractions` and 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). (Contributed by Serhiy Storchaka in bpo-37999.)

4 New Modules

- None yet.

5 Improved Modules

5.1 base64

Add `base64.b32hexencode()` and `base64.b32hexdecode()` to support the Base32 Encoding with Extended Hex Alphabet.

5.2 codecs

Add a `codecs.unregister()` function to unregister a codec search function. (Contributed by Hai Shi in bpo-41842.)

5.3 curses

The extended color functions added in ncurses 6.1 will be used transparently by `curses.color_content()`, `curses.init_color()`, `curses.init_pair()`, and `curses.pair_content()`. A new function, `curses.has_extended_color_support()`, indicates whether extended color support is provided by the underlying ncurses library. (Contributed by Jeffrey Kintscher and Hans Petter Jansson in bpo-36982.)

5.4 encodings

`encodings.normalize_encoding()` now ignores non-ASCII characters. (Contributed by Hai Shi in bpo-39337.)

5.5 glob

Added the *root_dir* and *dir_fd* parameters in `glob()` and `iglob()` which allow to specify the root directory for searching. (Contributed by Serhiy Storchaka in bpo-38144.)

5.6 os

Added `os.cpu_count()` support for VxWorks RTOS. (Contributed by Peixing Xin in bpo-41440.)

5.7 py_compile

Added `--quiet` option to command-line interface of `py_compile`. (Contributed by Gregory Schevchenko in bpo-38731.)

5.8 shelve

The `shelve` module now uses `pickle.DEFAULT_PROTOCOL` by default instead of `pickle` protocol `3` when creating shelves. (Contributed by Zackery Spytz in bpo-34204.)

5.9 sys

Add `sys.orig_argv` attribute: the list of the original command line arguments passed to the Python executable. (Contributed by Victor Stinner in bpo-23427.)

5.10 types

Reintroduced the `types.EllipsisType`, `types.NoneType` and `types.NotImplementedType` classes, providing a new set of types readily interpretable by type checkers. (Contributed by Bas van Beek in bpo-41810.)

5.11 unittest

Add new method `assertNoLogs()` to complement the existing `assertLogs()`. (Contributed by Kit Yan Choi in bpo-39385.)

5.12 xml

Add a `LexicalHandler` class to the `xml.sax.handler` module. (Contributed by Jonathan Gossage and Zackery Spytz in bpo-35018.)

6 Optimizations

- Constructors `str()`, `bytes()` and `bytearray()` are now faster (around 30–40% for small objects). (Contributed by Serhiy Storchaka in bpo-41334.)
- The `runpy` module now imports fewer modules. The `python3 -m module-name` command startup time is 1.3x faster in average. (Contributed by Victor Stinner in bpo-41006.)
- The `LOAD_ATTR` instruction now uses new "per opcode cache" mechanism. It is about 36% faster now. (Contributed by Pablo Galindo and Yury Selivanov in bpo-42093, based on ideas implemented originally in PyPy and MicroPython.)
- When building Python with `--enable-optimizations` now `-fno-semantic-interposition` is added to both the compile and link line. This speeds builds of the Python interpreter created with `--enable-shared` with `gcc` by up to 30%. See this article for more details. (Contributed by Victor Stinner and Pablo Galindo in bpo-38980)

7 Deprecated

- Starting in this release, there will be a concerted effort to begin cleaning up old import semantics that were kept for Python 2.7 compatibility. Specifically, `find_loader()`/`find_module()` (superseded by `find_spec()`), `load_module()` (superseded by `exec_module()`), `module_repr()` (which the import system takes care of for you), the `__package__` attribute (superseded by `__spec__.parent`), the `__loader__` attribute (superseded by `__spec__.loader`), and the `__cached__` attribute (superseded by `__spec__.cached`) will slowly be removed (as well as other classes and methods in `importlib`). `ImportWarning` and/or `DeprecationWarning` will be raised as appropriate to help identify code which needs updating during this transition.

8 Removed

- Removed special methods `__int__`, `__float__`, `__floordiv__`, `__mod__`, `__divmod__`, `__rfloordiv__`, `__rmod__` and `__rdivmod__` of the `complex` class. They always raised a `TypeError`. (Contributed by Serhiy Storchaka in bpo-41974.)
- The `ParserBase.error()` method from the private and undocumented `_markupbase` module has been removed. `html.parser.HTMLParser` is the only subclass of `ParserBase` and its `error()` implementation has already been removed in Python 3.5. (Contributed by Berker Peksag in bpo-31844.)
- Removed the `unicodedata.ucnhash_CAPI` attribute which was an internal PyCapsule object. The related private `_PyUnicode_Name_CAPI` structure was moved to the internal C API. (Contributed by Victor Stinner in bpo-42157.)

9 Porting to Python 3.10

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

10 Build Changes

- The C99 functions `snprintf()` and `vsnprintf()` are now required to build Python. (Contributed by Victor Stinner in bpo-36020.)
- `sqlite3` requires SQLite 3.7.3 or higher. (Contributed by Sergey Fedoseev and Erlend E. Aasland bpo-40744.)

11 C API Changes

11.1 New Features

- The result of `PyNumber_Index()` now always has exact type `int`. Previously, the result could have been an instance of a subclass of `int`. (Contributed by Serhiy Storchaka in bpo-40792.)
- Add a new `orig_argv` member to the `PyConfig` structure: the list of the original command line arguments passed to the Python executable. (Contributed by Victor Stinner in bpo-23427.)

- The `PyDateTime_DATE_GET_TZINFO()` and `PyDateTime_TIME_GET_TZINFO()` macros have been added for accessing the `tzinfo` attributes of `datetime.datetime` and `datetime.time` objects. (Contributed by Zackery Spytz in bpo-30155.)
- Add a `PyCodec_Unregister()` function to unregister a codec search function. (Contributed by Hai Shi in bpo-41842.)
- The `PyIter_Send()` function was added to allow sending value into iterator without raising `StopIteration` exception. (Contributed by Vladimir Matveev in bpo-41756.)
- Added `PyUnicode_AsUTF8AndSize()` to the limited C API. (Contributed by Alex Gaynor in bpo-41784.)

11.2 Porting to Python 3.10

- The `PY_SSIZE_T_CLEAN` macro must now be defined to use `PyArg_ParseTuple()` and `Py_BuildValue()` formats which use `#`: `es#`, `et#`, `s#`, `u#`, `y#`, `z#`, `U#` and `Z#`. See Parsing arguments and building values and the **PEP 353**. (Contributed by Victor Stinner in bpo-40943.)
- Since `Py_TYPE()` is changed to the inline static function, `Py_TYPE(obj) = new_type` must be replaced with `Py_SET_TYPE(obj, new_type)`: see `Py_SET_TYPE()` (available since Python 3.9). For backward compatibility, this macro can be used:

  ```
  #if PY_VERSION_HEX < 0x030900A4
  #  define Py_SET_TYPE(obj, type) ((Py_TYPE(obj) = (type)), (void)0)
  #endif
  ```

  (Contributed by Dong-hee Na in bpo-39573.)
- Since `Py_REFCNT()` is changed to the inline static function, `Py_REFCNT(obj) = new_refcnt` must be replaced with `Py_SET_REFCNT(obj, new_refcnt)`: see `Py_SET_REFCNT()` (available since Python 3.9). For backward compatibility, this macro can be used:

  ```
  #if PY_VERSION_HEX < 0x030900A4
  #  define Py_SET_REFCNT(obj, refcnt) ((Py_REFCNT(obj) = (refcnt)), (void)0)
  #endif
  ```

  (Contributed by Victor Stinner in bpo-39573.)
- Since `Py_SIZE()` is changed to the inline static function, `Py_SIZE(obj) = new_size` must be replaced with `Py_SET_SIZE(obj, new_size)`: see `Py_SET_SIZE()` (available since Python 3.9). For backward compatibility, this macro can be used:

  ```
  #if PY_VERSION_HEX < 0x030900A4
  #  define Py_SET_SIZE(obj, size) ((Py_SIZE(obj) = (size)), (void)0)
  #endif
  ```

  (Contributed by Victor Stinner in bpo-39573.)
- Calling `PyDict_GetItem()` without GIL held had been allowed for historical reason. It is no longer allowed. (Contributed by Victor Stinner in bpo-40839.)
- `PyUnicode_FromUnicode(NULL, size)` and `PyUnicode_FromStringAndSize(NULL, size)` raise `DeprecationWarning` now. Use `PyUnicode_New()` to allocate Unicode object without initial data. (Contributed by Inada Naoki in bpo-36346.)
- The private `_PyUnicode_Name_CAPI` structure of the PyCapsule API `unicodedata.ucnhash_CAPI` has been moved to the internal C API. (Contributed by Victor Stinner in bpo-42157.)

11.3 Deprecated

- The `PyUnicode_InternImmortal()` function is now deprecated and will be removed in Python 3.12: use `PyUnicode_InternInPlace()` instead. (Contributed by Victor Stinner in bpo-41692.)

11.4 Removed

- `PyObject_AsCharBuffer()`, `PyObject_AsReadBuffer()`, `PyObject_CheckReadBuffer()`, and `PyObject_AsWriteBuffer()` are removed. Please migrate to new buffer protocol; `PyObject_GetBuffer()` and `PyBuffer_Release()`. (Contributed by Inada Naoki in bpo-41103.)
- Removed `Py_UNICODE_str*` functions manipulating `Py_UNICODE*` strings. (Contributed by Inada Naoki in bpo-41123.)
  - `Py_UNICODE_strlen`: use `PyUnicode_GetLength()` or `PyUnicode_GET_LENGTH`
  - `Py_UNICODE_strcat`: use `PyUnicode_CopyCharacters()` or `PyUnicode_FromFormat()`
  - `Py_UNICODE_strcpy`, `Py_UNICODE_strncpy`: use `PyUnicode_CopyCharacters()` or `PyUnicode_Substring()`
  - `Py_UNICODE_strcmp`: use `PyUnicode_Compare()`
  - `Py_UNICODE_strncmp`: use `PyUnicode_Tailmatch()`
  - `Py_UNICODE_strchr`, `Py_UNICODE_strrchr`: use `PyUnicode_FindChar()`
- Removed `PyUnicode_GetMax()`. Please migrate to new (**PEP 393**) APIs. (Contributed by Inada Naoki in bpo-41103.)
- Removed `PyLong_FromUnicode()`. Please migrate to `PyLong_FromUnicodeObject()`. (Contributed by Inada Naoki in bpo-41103.)
- Removed `PyUnicode_AsUnicodeCopy()`. Please use `PyUnicode_AsUCS4Copy()` or `PyUnicode_AsWideCharString()` (Contributed by Inada Naoki in bpo-41103.)
- Removed `_Py_CheckRecursionLimit` variable: it has been replaced by `ceval.recursion_limit` of the `PyInterpreterState` structure. (Contributed by Victor Stinner in bpo-41834.)
- Removed undocumented macros `Py_ALLOW_RECURSION` and `Py_END_ALLOW_RECURSION` and the `recursion_critical` field of the `PyInterpreterState` structure. (Contributed by Serhiy Storchaka in bpo-41936.)

Index