What's New In Python 3.13

Editors:

Adam Turner and Thomas Wouters

This article explains the new features in Python 3.13, compared to 3.12. Python 3.13 was released on October 7, 2024. For full details, see the changelog.

See also

PEP 719 – Python 3.13 Release Schedule

Summary – Release Highlights

Python 3.13 is the latest stable release of the Python programming language, with a mix of changes to the language, the implementation and the standard library. The biggest changes include a new interactive interpreter, experimental support for running in a free-threaded mode (**PEP 703**), and a Just-In-Time compiler (**PEP 744**).

Error messages continue to improve, with tracebacks now highlighted in color by default. The locals() builtin now has defined semantics for changing the returned mapping, and type parameters now support default values.

The library changes contain removal of deprecated APIs and modules, as well as the usual improvements in user-friendliness and correctness. Several legacy standard library modules have now been removed following their deprecation in Python 3.11 (**PEP 594**).

This article doesn't attempt to provide a complete specification of all new features, but instead gives a convenient overview. For full details refer to the documentation, such as the Library Reference and Language Reference. To understand the complete implementation and design rationale for a change, refer to the PEP for a particular new feature; but note that PEPs usually are not kept up-to-date once a feature has been fully implemented. See Porting to Python 3.13 for guidance on upgrading from earlier versions of Python.

Interpreter improvements:

- A greatly improved interactive interpreter and improved error messages.
- **PEP 667**: The locals() builtin now has defined semantics when mutating the returned mapping. Python debuggers and similar tools may now more reliably update local variables in optimized scopes even during concurrent code execution.
- **PEP 703**: CPython 3.13 has experimental support for running with the global interpreter lock disabled. See Free-threaded CPython for more details.

- **PEP 744**: A basic JIT compiler was added. It is currently disabled by default (though we may turn it on later). Performance improvements are modest – we expect to improve this over the next few releases.
- Color support in the new interactive interpreter, as well as in tracebacks and doctest output. This can be disabled through the PYTHON_COLORS and NO_COLOR environment variables.

Python data model improvements:

- __static_attributes__ stores the names of attributes accessed through self.X in any function in a class body.
- __firstlineno__ records the first line number of a class definition.

Significant improvements in the standard library:

- Add a new PythonFinalizationError exception, raised when an operation is blocked during finalization.
- The argparse module now supports deprecating command-line options, positional arguments, and subcommands.
- The new functions base64.z85encode() and base64.z85decode() support encoding and decoding Z85 data.
- The copy module now has a copy.replace() function, with support for many builtin types and any class defining the __replace__() method.
- The new dbm.sqlite3 module is now the default dbm backend.
- The os module has a suite of new functions for working with Linux's timer notification file descriptors.
- The random module now has a command-line interface.

Security improvements:

- ssl.create_default_context() sets ssl.VERIFY_X509_PARTIAL_CHAIN and ssl.VERIFY_X509_STRICT as default flags.

C API improvements:

- The Py_mod_gil slot is now used to indicate that an extension module supports running with the GIL disabled.
- The PyTime C API has been added, providing access to system clocks.
- PyMutex is a new lightweight mutex that occupies a single byte.
- There is a new suite of functions for generating **PEP 669** monitoring events in the C API.

New typing features:

- **PEP 696**: Type parameters (typing.TypeVar, typing.ParamSpec, and typing.TypeVarTuple) now support defaults.
- **PEP 702**: The new warnings.deprecated() decorator adds support for marking deprecations in the type system and at runtime.
- **PEP 705**: typing.ReadOnly can be used to mark an item of a typing.TypedDict as read-only for type checkers.
- **PEP 742**: typing.TypeIs provides more intuitive type narrowing behavior, as an alternative to typing.TypeGuard.

Platform support:

- **PEP 730**: Apple's iOS is now an officially supported platform, at **tier 3**.
- **PEP 738**: Android is now an officially supported platform, at **tier 3**.
- wasm32-wasi is now supported as a **tier 2** platform.
- wasm32-emscripten is no longer an officially supported platform.

Important removals:

- PEP 594: The remaining 19 "dead batteries" (legacy stdlib modules) have been removed from the standard
  library: aifc, audioop, cgi, cgitb, chunk, crypt, imghdr, mailcap, msilib, nis, nntplib, ossaudio dev, pipes, sndhdr, spwd, sunau, telnetlib, uu and xdrlib.
- Remove the **2to3** tool and lib2to3 module (deprecated in Python 3.11).
- Remove the tkinter.tix module (deprecated in Python 3.6).
- Remove the locale.resetlocale() function.
- Remove the typing.io and typing.re namespaces.
- Remove chained classmethod descriptors.

Release schedule changes:

PEP 602 ("Annual Release Cycle for Python") has been updated to extend the full support ('bugfix') period for new releases to two years. This updated policy means that:

- Python 3.9–3.12 have one and a half years of full support, followed by three and a half years of security fixes.
- Python 3.13 and later have two years of full support, followed by three years of security fixes.

New Features

A better interactive interpreter

Python now uses a new interactive shell by default, based on code from the PyPy project. When the user starts the REPL from an interactive terminal, the following new features are now supported:

- Multiline editing with history preservation.
- Direct support for REPL-specific commands like help, exit, and quit, without the need to call them as functions.
- Prompts and tracebacks with color enabled by default.
- Interactive help browsing using F1 with a separate command history.
- History browsing using F2 that skips output as well as the >>> and ... prompts.
- “Paste mode” with F3 that makes pasting larger blocks of code easier (press F3 again to return to the regular prompt).

To disable the new interactive shell, set the PYTHON_BASIC_REPL environment variable. For more on interactive mode, see Interactive Mode.

(Contributed by Pablo Galindo Salgado, Łukasz Langa, and Lysandros Nikolaou in gh-111201 based on code from the PyPy project. Windows support contributed by Dino Viehland and Anthony Shaw.)

Improved error messages

- The interpreter now uses color by default when displaying tracebacks in the terminal. This feature can be controlled via the new PYTHON_COLORS environment variable as well as the canonical NO_COLOR and FORCE_COLOR environment variables. (Contributed by Pablo Galindo Salgado in gh-112730.)
- A common mistake is to write a script with the same name as a standard library module. When this results in errors, we now display a more helpful error message:
- $ python random.py
- Traceback (most recent call last):
- File "/home/me/random.py", line 1, in <module>
- **import random**
- File "/home/me/random.py", line 3, in <module>
- print(random.randint(5))
- ^^^^^^^^^^^^^^
- AttributeError: module 'random' has no attribute 'randint' (consider renaming '/home/me/random.py' since it has the same name as the standard library module named 'random' and prevents importing that standard library module)

Similarly, if a script has the same name as a third-party module that it attempts to import and this results in errors, we also display a more helpful error message:

$ python numpy.py

Traceback (most recent call last):

File "/home/me/numpy.py", line 1, in <module>

import numpy as np

File "/home/me/numpy.py", line 3, in <module>

np.array([1, 2, 3])

^^^^^^^^

AttributeError: module 'numpy' has no attribute 'array' (consider renaming '/home/me/numpy.py' if it has the same name as a library you intended to import)

(Contributed by Shantanu Jain in gh-95754.)

- The error message now tries to suggest the correct keyword argument when an incorrect keyword argument is passed to a function.
- **>>>** "Better error messages!".split(max_split=1)
- Traceback (most recent call last):
- File "<python-input-0>", line 1, in <module>
- "Better error messages!".split(max_split=1)
- ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~^^^^^^^^^^^^^
- TypeError: split() got an unexpected keyword argument 'max_split'. Did you mean 'maxsplit'?

(Contributed by Pablo Galindo Salgado and Shantanu Jain in gh-107944.)