PyPy v8.0.0: release of python 2.7, 3.11, and 3.12 beta released 2026-09-19

The PyPy team is proud to release version 8.0.0 of PyPy after the previous release on May 26, 2026. This is a major new version, hence the bump to 8.0.0. It is our first release of Python 3.12, which may still have some bugs so we are calling it “beta” quality.

Why the move to 8.0.0

glibc2.28

We have updated our linux buildbots (linux64, linux32, aarch64) to use manylinux_2_28 images based on AlmaLinux 8 and glibc 2.28. These use gcc14 instead of the gcc5 previously used. So our compiled tarballs will require at least glibc2.28, which should be universally supported by now (Ubuntu 24.04 uses glibc2.39). In order to prevent confusion, we felt bumping the major version would be prudent.

cp12-abi3 support

PyPy’s Python 3.12 support comes with a new model for the C layer `PyObject`. In order to link the C object to the internal RPython one, we have an extra field in the object `ob_pypy_link`, as described in-depth in Rawrefcount and the GC. In previous versions, this field was visible in a way that makes the `PyObject` struct different from the CPython one. From v8.0.0, we “hide” the PyPy-only extension in a prefix before the pointer we hand off to C-extension modules. The goal of this work is to allow PyPy to use cp312-abi3 wheels produced for CPython 3.12 and up, using the limited ABI. The required pieces have all been put in place:

- PyPy’s C headers, including struct definitions like `PyObject`, are compatible with CPython’s C headers when defining `Py_LIMITED_API=0x030C0000`
- PyPy no longer mangles exported function names from the limited API. In PyPy3.11 and earlier, functions like `PyTuple_New` were exported as `PyPyTupleNew`.

Still missing: the import machinery must be taught that abi3.so shared objects are valid for PyPy, and the larger ecosystem (pip, uv) must also accept that cp312-abi3 wheels are valid candidates for installation.

Yes, this is a big step. We are working with Cython and PyO3 to make sure it all will Just Work™. Hopefully this will make it easier for packages to support PyPy.

What is new in RPython code generation

PyPy is written in RPython, and has code generation to translate RPython into C as part of the VM build process. We have made some improvements to code generation in attempts to speed up the base interpreter. While the speedups have not been that impressive, we have made some steps forward:

- We now use computed gotos and more aggressively inline code. While this produces more compact sources, it does not boost performance as much as we wished.
- The source code includes comments mappng the source back to the RPython code that generated the block. This is very helpful to see exactly what is going on, and may enable further improvements.

Dropping HPy

We have dropped the internal HPy backend for PyPy. The HPy project's understanding of how to use handles instead of pointers was a good prototype, but the project did not attract enough supporters to become a new standard. The code is still in the PyPy codebase, and can be toggled on with a build option

A revived tool comparing headers and exported functions

We revived the clang-based pyhdrdump from #3397#issuecomment-1872091878 to compare PyPy's header files to CPython's header files. See the README for more information on how it works and how to use it.

Interpreters

The release includes three different interpreters:

- PyPy2.7, supporting the syntax and the features of Python 2.7 including the stdlib for CPython 2.7.18+ (the `+` is for backported security updates)
- PyPy3.11, supporting the syntax and the features of Python 3.11, including the stdlib for CPython 3.11.16. Barring security issues, this will be the last release to support 3.11.
- PyPy3.12, supporting the syntax and features of Python 3.12, including the stdlib for CPython 3.12.14.

The interpreters are based on much the same codebase, thus the triple release.

We recommend updating. You can find links to download the releases here:

https://pypy.org/download.html

We would like to thank our donors for the continued support of the PyPy project. If PyPy is not quite good enough for your needs, we are available for direct consulting work. If PyPy is helping you out, we would love to hear about it and encourage submissions to our blog via a pull request to https://github.com/pypy/pypy.org

We would also like to thank our contributors and encourage new people to join the project. PyPy has many layers and we need help with all of them: bug fixes, PyPy and RPython documentation improvements, or general help with making RPython's JIT even better.

If you are a python library maintainer and use C-extensions, please consider making a CFFI version of your library that would be performant on PyPy. Failing that, PyPy will soon support the cp312-abi3 tag for limited ABI wheels. In any case, cibuildwheel supports building wheels for PyPy.

Footnotes

What is PyPy?

PyPy is a Python interpreter, a drop-in replacement for CPython. It's fast (PyPy and CPython performance comparison) due to its integrated tracing JIT compiler.

We also welcome developers of other dynamic languages to see what RPython can do for them.

We provide binary builds for:

- **x86** machines on most common operating systems (Linux 32/64 bits, Mac OS 64 bits, Windows 64 bits)
- 64-bit **ARM** machines running Linux (`aarch64`) and macos (`macos_arm64`).

PyPy supports Windows 32-bit, Linux PPC64 big- and little-endian, Linux ARM 32 bit, RISC-V RV64IMAFD Linux, and s390x Linux but does not release binaries. Please reach out to us if you wish to sponsor binary releases for those platforms. Downstream packagers provide binary builds for debian, Fedora, conda, OpenBSD, FreeBSD, Gentoo, and more.

Changelog

The changelog does not describe the hundreds of changes made for PyPy3.12.

For all versions

- Add missing constants to `_socket` (#5340)
- Add implementation of `os.unshare()` where supported (Linux)
- Make `tar.gz` compressed binaries instead of `tar.bz2` (#5353)
- Quiet compilation warnings from gcc14, which is more strict around pointer use than gcc5.
- Cleaned up RPython test failures on platforms other than linux64
- Add an RPython `RPY_NORETURN __attribute__((noreturn))` and use it for aborts
- Teach `lltype` to handle variadic pointer arguments for untranslated macos arm64 tests
- Add docstrings and python2 typing to public functions and methods in `rpython.rlib.parsing.deterministic`
- Expose unstable DAWG packing layouts, and stabilize DAWG offset encoding. Remove the unicodedata txt files, download them on demand. Regenerate all the unicodedbs with a new optimized DAWG, and add unicodedb 15.0.0 (#5581)
- Fix bitfield handling in cffi on macos
- Backport cpython's `_multibytecodec` C code from 3.12.14

Bugfixes

- Fix `rmdir` failure on windows to return the correct `errno`
- Implement `inheritable=False` on windows `socket.dup`
- Fix index of `nsec` time on windows
- Check `WSAGetLastError` on windows when deciding if errors occurred
- Prevent fixpoint length occilation in unicodedata dawg encoding
- Do not allow `sys.setrecursionlimit` to go over 75% of stack size.
- Reverse logic in *os.linkat* to fix longstanding bug (#5545)
- When creating portable builds, compile `_openssl` without tests
- `rstruct.pack` with `'0p'` format no longer errors
- Teach the JIT about variadic ffi calls for macos-arm64

Speedups and enhancements

- Use computed-goto on GCC/Clang instead of a big switch statement
- Explicitly inline stack checks at the beginning of each `PyFrame`
- Do not create PDB files on windows when testing and translating
- Turn app level `lst = [None] * n` into a `malloc + memset`
- Port Crochemore-Perrin two-way string search from cpython3.12 into RPython
- Add some integer operation simplification rules to the optimizer (#5512, #5521)
- Make `dict.update` consume an iterable argument lazily (#5518)
- Implement remote debugger protocol for Windows (#5531)

- `str.count` an empty needle in the code point domain (#5535)
- Probe the gc for the value of `nolarge_max` in `rffi.alloc_buffer` rather than wasting a gc allocation on an unpinnable buffer (#5566)
- Rather than allocate `VMPROFSTACK` nodes at each `enter_code` call, use a freelist and remove an indirection from `_vmprof.execute_frame`, which is called very often (#5573)
- Add new peephole rewriting rules to the JIT optimizer to better optimize integer operations.
- Teach the JIT to reason about non-negative fields (#5577). In particular, the JIT now knows that the length of a list is non-negative. This can often make indexing access bounds checks more efficient.

JIT cleanups and simplifications (do not affect Python performance)

- x86 jit: break `cvtsi2sd` false dependency with a zeroing idiom (#5508)
- aarch64: emit shorter `MOVN/MOVK` sequences for negative immediates (#5509)
- repair the stale `rlist.ll_getitem()` call in the codewriter builtin table (#5538)
- Slice the `kind` in `rewrite_op_getarrayitem` so the `raw_r` guard can fire (#5541)
- Check the assembler's register count before encoding the instructions (#5553)
- Thread `handling_ovf` into `generate_last_exc()` so a re-raised OverflowError survives (#5552)
- Normalize a `force_cast` to Bool from an `unsigned` byte type (#5551)
- Add the missing `opimpl_record_known_result_r_ir_v` for completeness (#5548)

Python 2.7

- opt-out of `_cppyy`, `micronumpy`, for 8.0.0
- Improve `ctype.find_library` like on python3

Python 3.11 and 3.12

- Opt-out of `_cppyy`, `micronumpy`, `_hpy_universal` for 8.0.0
- Update CFFI to v2.1.0
- Update stdlib to 3.11.16 and vendored expat to 2.8.4
- On portable builds, update lzma to 5.8.4

Bugfixes including missing compatibility with CPython 3.11

- Fix `unwind_fblock(F_HANDLER_CLEANUP)` for bare except block (#5490)
- Implement `nt._getdiskusage` (#5250)
- Implement `_winapi.CreateJunction`
- Add `AF_ALG` and `AF_QICRTR` support to _socket when supported, add `sendmsg_afalg`

- Refactor support for the buffer protocol and fix places where buffers were not released.
- Fix crash in `__build_class__` (#5510)
- Fix crash when pattern-matching a mapping with a `**rest` argument (#5506)
- Fix `_pickle` for the `dill` package (uqfoundation/dill#756)
- Explicitly forbid iteration of typing `UnionType` (#5511)
- Prevent `os.urandom` from returning too short results (#5516)
- `malloc` failure in cpyext should return `NULL`, not raise
- `faulthandler.cancel_dump_traceback_later` must call `setup()` to initialize thread locks
- Fix `AttributeError` edge case when `AttributeError.__init__` has not been called (#5514)
- Add `PyUnicode_READY` before returning in *PyUnicode_FromFormat`* (#5524)
- Fix `PyUnicode_FromKindAndData` to only copy data (#5525)
- Fix `__getitem__` with `index<0` (#5526)
- Fix `PyObject_Format` for a subclass that overrides `__format__`
- Fix exception table propagation after non-fallthrough blocks (#5539)
- Fix `_json.encode` encoding (#5547)
- Fix `tp_dealloc` for exception types (#5555)
- Accept `-1` for color in `_curses` (#5557)
- Properly inherit `Py_TPFLAGS_HAVE_GC`, `tp_traverse`, `tp_clear` (#5556)
- Avoid deprecated `sysconfig` check_home arguments (#5554)
- Remove `itertools.tee` optimization (#5563)
- Add `fcntl` constants on macos (#5543)
- Fix `hashlib` to accept non-byte messages (#5544)
- Track down and fix buffer refcnt problems (#5546)
- Fix `bytes()`, `bytearray()` and `codecs` accepting `encoding=None` (#5558)
- Fix `inspect.isbuiltin`/`ismethodwrapper` for slot-wrapper bound methods. Do not call `repr()` on `Method.descr_method_repr` (which could recurse) (#5503)
- Fix `int()` raising `TypeError` instead of `ValueError` for a bad base (#5559)
- Fill cpyext's `make_sq_set_item` for either `__setitem__` or `__delitem__` (#5564)
- Add `os.readv`, `os.writev` and similar functions (#5565)
- Fix undefined behaviour when digit spec is not 1 char in `format`
- optimize `bytearray(N)` (#5567)
- Implement `_sre validation` and prevent `re.Pattern` instantiation (#5571)
- Add a lock for `textio` (#5575)
- Fix typo in `decode_never_raise` around BOM handling
- Accept bytes in `os._add_dll_directory` on windows
- Error on unexpected keyword errors for builtins like `"".split(max_split=1)`
- Make `id()` posonly, more `__builtin__` functions should also have this
- Add implementations of `os.sched_{g,s}etaffinity` (#5582)
- Mark the signal module `_fired_bitmask` non-JITable (#4946)

- Make `_pickle.Unpickler.find_class` available to app-level and check for overrides (#5583)
- Fix parser for incomplete single line input (#5505)
- Add `_winapi.UnmapViewOfFile` (#5361)
- Fix locale encode/decode on windows (`sizeof(wchar_t)` is 2 not 4)
- Get `termios` struct layouts from the compiler, fixes macos failures

Speedups and enhancements

- Refactor multiprocessing `resource_tracker` to work with PyPy
- Cache hash of tuples to only calculate them once
- Simplify `getattr` since PyPy's strings and bytes are internally always utf8