This is Python version 3.11.0 alpha 7

Tests passing

discourse join chat

See the end of this file for further copyright and license information.

Contents

General Information

- Website: https://www.python.org
- Source code: https://github.com/python/cpython
- Issue tracker: https://bugs.python.org
- Documentation: https://docs.python.org
- Developer's Guide: https://devguide.python.org/

Contributing to CPython

For more complete instructions on contributing to CPython development, see the Developer Guide.

Using Python

Installable Python kits, and information about using Python, are available at python.org.

Build Instructions

On Unix, Linux, BSD, macOS, and Cygwin:

```
./configure
make
make test
sudo make install
```

This will install Python as `python3`.

You can pass many options to the configure script; run `./configure --help` to find out more. On macOS case-insensitive file systems and on Cygwin, the executable is called `python.exe`; elsewhere it's just `python`.

Building a complete Python installation requires the use of various additional third-party libraries, depending on your build platform and configure options. Not all standard library modules are buildable or useable on all platforms. Refer to the Install dependencies section of the Developer Guide for current detailed information on dependencies for various Linux distributions and macOS.

On macOS, there are additional configure and build options related to macOS framework and universal builds. Refer to Mac/README.rst.

On Windows, see PCbuild/readme.txt.

If you wish, you can create a subdirectory and invoke configure from there. For example:

```
mkdir debug
```

```
cd debug
../configure --with-pydebug
make
make test
```

(This will fail if you *also* built at the top-level directory. You should do a `make clean` at the top-level first.)

To get an optimized build of Python, `configure --enable-optimizations` before you run `make`. This sets the default make targets up to enable Profile Guided Optimization (PGO) and may be used to auto-enable Link Time Optimization (LTO) on some platforms. For more details, see the sections below.

Profile Guided Optimization

PGO takes advantage of recent versions of the GCC or Clang compilers. If used, either via `configure --enable-optimizations` or by manually running `make profile-opt` regardless of configure flags, the optimized build process will perform the following steps:

The entire Python directory is cleaned of temporary files that may have resulted from a previous compilation.

An instrumented version of the interpreter is built, using suitable compiler flags for each flavor. Note that this is just an intermediary step. The binary resulting from this step is not good for real-life workloads as it has profiling instructions embedded inside.

After the instrumented interpreter is built, the Makefile will run a training workload. This is necessary in order to profile the interpreter's execution. Note also that any output, both stdout and stderr, that may appear at this step is suppressed.

The final step is to build the actual interpreter, using the information collected from the instrumented one. The end result will be a Python binary that is optimized; suitable for distribution or production installation.

Link Time Optimization

Enabled via configure's `--with-lto` flag. LTO takes advantage of the ability of recent compiler toolchains to optimize across the otherwise arbitrary `.o` file boundary when building final executables or shared libraries for additional performance gains.

What's New

We have a comprehensive overview of the changes in the What's New in Python 3.11 document. For a more detailed change log, read Misc/NEWS, but a full accounting of changes can only be gleaned from the commit history.

If you want to install multiple versions of Python, see the section below entitled "Installing multiple versions".

Documentation

Documentation for Python 3.11 is online, updated daily.

It can also be downloaded in many formats for faster access. The documentation is downloadable in HTML, PDF, and reStructuredText formats; the latter version is primarily for documentation authors, translators, and people with special formatting requirements.

For information about building Python's documentation, refer to Doc/README.rst.

Converting From Python 2.x to 3.x

Significant backward incompatible changes were made for the release of Python 3.0, which may cause programs written for Python 2 to fail when run with Python 3. For more information about porting your code from Python 2 to Python 3, see the Porting HOWTO.

Testing

To test the interpreter, type `make test` in the top-level directory. The test set produces some output. You can generally ignore the messages about skipped tests due to optional features which can't be imported. If a message is printed about a failed test or a traceback or core dump is produced, something is wrong.

By default, tests are prevented from overusing resources like disk space and memory. To enable these tests, run `make testall`.

If any tests fail, you can re-run the failing test(s) in verbose mode. For example, if `test_os` and `test_gdb` failed, you can run:

```
make test TESTOPTS="-v test_os test_gdb"
```

If the failure persists and appears to be a problem with Python rather than your environment, you can file a bug report and include relevant output from that command to show the issue.

See Running & Writing Tests for more on running tests.

Installing multiple versions

On Unix and Mac systems if you intend to install multiple versions of Python using the same installation prefix (`--prefix` argument to the configure script) you must take care that your primary python executable is not overwritten by the installation of a different version. All files and directories installed using `make altinstall` contain the major and minor version and can thus live side-by-side. `make install` also creates `${prefix}/bin/python3` which refers to `${prefix}/bin/pythonX.Y`. If you intend to install multiple versions using the same prefix you must decide which version (if any) is your "primary" version. Install that version using `make install`. Install all other versions using `make altinstall`.

For example, if you want to install Python 2.7, 3.6, and 3.11 with 3.11 being the primary version, you would execute `make install` in your 3.11 build directory and `make altinstall` in the others.

Issue Tracker and Mailing List

Bug reports are welcome! You can use the issue tracker to report bugs, and/or submit pull requests on GitHub.

You can also follow development discussion on the python-dev mailing list.

Proposals for enhancement

If you have a proposal to change Python, you may want to send an email to the comp.lang.python or python-ideas mailing lists for initial feedback. A Python Enhancement Proposal (PEP) may be submitted if your idea gains ground. All current PEPs, as well as guidelines for submitting a new PEP, are listed at peps.python.org.

Release Schedule

See PEP 664 for Python 3.11 release details.

Copyright and License Information