

# Command line and environment

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 1)**

Unknown directive type "highlight".

```
.. highlight:: sh
```

The CPython interpreter scans the command line and the environment for various settings.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 14)**

Unknown directive type "impl-detail".

```
.. impl-detail::
```

```
Other implementations' command line schemes may differ. See
:ref:`implementations` for further resources.
```

## Command line

When invoking Python, you may specify any of these options:

```
python [-bBdEhiIOqsSuvVWx?] [-c command | -m module-name | script | - ] [args]
```

The most common use case is, of course, a simple invocation of a script:

```
python myscript.py
```

## Interface options

The interpreter interface resembles that of the UNIX shell, but provides some additional methods of invocation:

- When called with standard input connected to a tty device, it prompts for commands and executes them until an EOF (an end-of-file character, you can produce that with `:kbd:Ctrl-D` on UNIX or `:kbd:Ctrl-Z, Enter` on Windows) is read.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 42);**  
[backlink](#)

Unknown interpreted text role "kbd".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 42);**  
[backlink](#)

Unknown interpreted text role "kbd".

- When called with a file name argument or with a file as standard input, it reads and executes a script from that file.
- When called with a directory name argument, it reads and executes an appropriately named script from that directory.
- When called with `-c command`, it executes the Python statement(s) given as *command*. Here *command* may contain multiple statements separated by newlines. Leading whitespace is significant in Python statements!
- When called with `-m module-name`, the given module is located on the Python module path and executed as a script.

In non-interactive mode, the entire input is parsed before it is executed.

An interface option terminates the list of options consumed by the interpreter, all consecutive arguments will end up in `:data:'sys.argv'` -- note that the first element, subscript zero (`sys.argv[0]`), is a string reflecting the program's source.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 57);** [backlink](#)

Unknown interpreted text role "data".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\cpython-main) (Doc) (using cmdline.rst, line 62)**

Unknown directive type "cmdoption".

```
.. cmdoption:: -c <command>
```

Execute the Python code in *\*command\**. *\*command\** can be one or more statements separated by newlines, with significant leading whitespace as in normal module code.

If this option is given, the first element of `:data:`sys.argv`` will be ```"-c"``` and the current directory will be added to the start of `:data:`sys.path`` (allowing modules in that directory to be imported as top level modules).

```
.. audit-event:: cpython.run_command command cmdoption-c
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\cpython-main) (Doc) (using cmdline.rst, line 75)**

Unknown directive type "cmdoption".

```
.. cmdoption:: -m <module-name>
```

Search `:data:`sys.path`` for the named module and execute its contents as the `:mod:`__main__`` module.

Since the argument is a *\*module\** name, you must not give a file extension (```.py```). The module name should be a valid absolute Python module name, but the implementation may not always enforce this (e.g. it may allow you to use a name that includes a hyphen).

Package names (including namespace packages) are also permitted. When a package name is supplied instead of a normal module, the interpreter will execute ```<pkg>.__main__``` as the main module. This behaviour is deliberately similar to the handling of directories and zipfiles that are passed to the interpreter as the script argument.

```
.. note::
```

This option cannot be used with built-in modules and extension modules written in C, since they do not have Python module files. However, it can still be used for precompiled modules, even if the original source file is not available.

If this option is given, the first element of `:data:`sys.argv`` will be the full path to the module file (while the module file is being located, the first element will be set to ```"-m"```). As with the `:option:`-c`` option, the current directory will be added to the start of `:data:`sys.path``.

`:option:`-I`` option can be used to run the script in isolated mode where `:data:`sys.path`` contains neither the current directory nor the user's site-packages directory. All `:envvar:`PYTHON*`` environment variables are ignored, too.

Many standard library modules contain code that is invoked on their execution as a script. An example is the `:mod:`timeit`` module::

```
python -m timeit -s 'setup here' 'benchmarked code here'
python -m timeit -h # for details
```

```
.. audit-event:: cpython.run_module module-name cmdoption-m
```

```
.. seealso::
```

```
:func:`runpy.run_module`
```

Equivalent functionality directly available to Python code

```
:pep:`338` -- Executing modules as scripts
```

```
.. versionchanged:: 3.1
```

Supply the package name to run a ```__main__``` submodule.

```
.. versionchanged:: 3.4
```

namespace packages are also supported

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 131)**

Unknown directive type "describe".

```
.. describe:: -

    Read commands from standard input (:data:`sys.stdin`). If standard input is
    a terminal, :option:`-i` is implied.

    If this option is given, the first element of :data:`sys.argv` will be
    ``"-``" and the current directory will be added to the start of
    :data:`sys.path`.

.. audit-event:: cpython.run_stdin "" ""
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 144)**

Unknown directive type "describe".

```
.. describe:: <script>

    Execute the Python code contained in *script*, which must be a filesystem
    path (absolute or relative) referring to either a Python file, a directory
    containing a ``__main__.py`` file, or a zipfile containing a
    ``__main__.py`` file.

    If this option is given, the first element of :data:`sys.argv` will be the
    script name as given on the command line.

    If the script name refers directly to a Python file, the directory
    containing that file is added to the start of :data:`sys.path`, and the
    file is executed as the :mod:`__main__` module.

    If the script name refers to a directory or zipfile, the script name is
    added to the start of :data:`sys.path` and the ``__main__.py`` file in
    that location is executed as the :mod:`__main__` module.

    :option:`-I` option can be used to run the script in isolated mode where
    :data:`sys.path` contains neither the script's directory nor the user's
    site-packages directory. All :envvar:`PYTHON*` environment variables are
    ignored, too.

.. audit-event:: cpython.run_file filename

.. seealso::
    :func:`runpy.run_path`
    Equivalent functionality directly available to Python code
```

If no interface option is given, :option:`-i` is implied, `sys.argv[0]` is an empty string (``""``) and the current directory will be added to the start of `:data:`sys.path``. Also, tab-completion and history editing is automatically enabled, if available on your platform (see [ref`rcompleter-config`](#)).

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 174); [backlink](#)**

Unknown interpreted text role "option".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 174); [backlink](#)**

Unknown interpreted text role "data".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 174); [backlink](#)**

Unknown interpreted text role "ref".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-**

**main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 180)**

Unknown directive type "seealso".

```
.. seealso:: :ref:`tut-invoking`
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 182)**

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.4
   Automatic enabling of tab-completion and history editing.
```

## Generic options

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 189)**

Unknown directive type "cmdoption".

```
.. cmdoption:: -?
               -h
               --help
```

Print a short description of all command line options.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 196)**

Unknown directive type "cmdoption".

```
.. cmdoption:: -V
               --version
```

Print the Python version number and exit. Example output could be:

```
.. code-block:: none
```

```
Python 3.8.0b2+
```

When given twice, print more information about the build, like:

```
.. code-block:: none
```

```
Python 3.8.0b2+ (3.8.0c076caaa8, Apr 20 2019, 21:55:00)
[GCC 6.2.0 20161005]
```

```
.. versionadded:: 3.6
   The ``-VV`` option.
```

## Miscellaneous options

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 220)**

Unknown directive type "cmdoption".

```
.. cmdoption:: -b
```

Issue a warning when comparing :class:`bytes` or :class:`bytearray` with :class:`str` or :class:`bytes` with :class:`int`. Issue an error when the option is given twice (:option:`!-bb`).

```
.. versionchanged:: 3.5
   Affects comparisons of :class:`bytes` with :class:`int`.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-**

**main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 229)**

Unknown directive type "cmdoption".

```
.. cmdoption:: -B
```

If given, Python won't try to write ``.pyc`` files on the import of source modules. See also :envvar:`PYTHONDONTWRITEBYTECODE`.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 235)**

Unknown directive type "cmdoption".

```
.. cmdoption:: --check-hash-based-pycs default|always|never
```

Control the validation behavior of hash-based ``.pyc`` files. See :ref:`pyc-invalidation`. When set to ``default``, checked and unchecked hash-based bytecode cache files are validated according to their default semantics. When set to ``always``, all hash-based ``.pyc`` files, whether checked or unchecked, are validated against their corresponding source file. When set to ``never``, hash-based ``.pyc`` files are not validated against their corresponding source files.

The semantics of timestamp-based ``.pyc`` files are unaffected by this option.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 249)**

Unknown directive type "cmdoption".

```
.. cmdoption:: -d
```

Turn on parser debugging output (for expert only, depending on compilation options). See also :envvar:`PYTHONDEBUG`.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 255)**

Unknown directive type "cmdoption".

```
.. cmdoption:: -E
```

Ignore all :envvar:`PYTHON\*` environment variables, e.g. :envvar:`PYTHONPATH` and :envvar:`PYTHONHOME`, that might be set.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 261)**

Unknown directive type "cmdoption".

```
.. cmdoption:: -i
```

When a script is passed as first argument or the :option:`-c` option is used, enter interactive mode after executing the script or the command, even when :data:`sys.stdin` does not appear to be a terminal. The :envvar:`PYTHONSTARTUP` file is not read.

This can be useful to inspect global variables or a stack trace when a script raises an exception. See also :envvar:`PYTHONINSPECT`.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 272)**

Unknown directive type "cmdoption".

```
.. cmdoption:: -I
```

Run Python in isolated mode. This also implies `-E` and `-s`.  
In isolated mode `:data:`sys.path`` contains neither the script's directory nor the user's site-packages directory. All `:envvar:`PYTHON*`` environment variables are ignored, too. Further restrictions may be imposed to prevent the user from injecting malicious code.

```
.. versionadded:: 3.4
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 283)**

Unknown directive type "cmdoption".

```
.. cmdoption:: -O
```

Remove assert statements and any code conditional on the value of `:const:`__debug__``. Augment the filename for compiled `(:term:`bytecode`)` files by adding ```.opt-1``` before the ```.pyc``` extension (see `:pep:`488``). See also `:envvar:`PYTHONOPTIMIZE``.

```
.. versionchanged:: 3.5
    Modify ``.pyc`` filenames according to :pep:`488`.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 294)**

Unknown directive type "cmdoption".

```
.. cmdoption:: -OO
```

Do `:option:`-O`` and also discard docstrings. Augment the filename for compiled `(:term:`bytecode`)` files by adding ```.opt-2``` before the ```.pyc``` extension (see `:pep:`488``).

```
.. versionchanged:: 3.5
    Modify ``.pyc`` filenames according to :pep:`488`.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 304)**

Unknown directive type "cmdoption".

```
.. cmdoption:: -q
```

Don't display the copyright and version messages even in interactive mode.

```
.. versionadded:: 3.2
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 311)**

Unknown directive type "cmdoption".

```
.. cmdoption:: -R
```

Turn on hash randomization. This option only has an effect if the `:envvar:`PYTHONHASHSEED`` environment variable is set to ```0```, since hash randomization is enabled by default.

On previous versions of Python, this option turns on hash randomization, so that the `:meth:`__hash__`` values of str and bytes objects are "salted" with an unpredictable random value. Although they remain constant within an individual Python process, they are not predictable between repeated invocations of Python.

Hash randomization is intended to provide protection against a denial-of-service caused by carefully-chosen inputs that exploit the worst

case performance of a dict construction,  $O(n^{\sup{2}})$  complexity. See <http://www.ocert.org/advisories/ocert-2011-003.html> for details.

`:envvar:PYTHONHASHSEED` allows you to set a fixed value for the hash seed secret.

.. versionchanged:: 3.7  
The option is no longer ignored.

.. versionadded:: 3.2.3

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\cpython-main) (Doc) (using cmdline.rst, line 337)**

Unknown directive type "cmdoption".

.. cmdoption:: -s

Don't add the `:data:~\user site-packages` directory `<site.USER_SITE>` to `:data:~\sys.path`.

.. seealso::

`:pep:370` -- Per user site-packages directory

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\cpython-main) (Doc) (using cmdline.rst, line 347)**

Unknown directive type "cmdoption".

.. cmdoption:: -S

Disable the import of the module `:mod:site` and the site-dependent manipulations of `:data:~\sys.path` that it entails. Also disable these manipulations if `:mod:site` is explicitly imported later (call `:func:site.main` if you want them to be triggered).

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\cpython-main) (Doc) (using cmdline.rst, line 355)**

Unknown directive type "cmdoption".

.. cmdoption:: -u

Force the stdout and stderr streams to be unbuffered. This option has no effect on the stdin stream.

See also `:envvar:PYTHONUNBUFFERED`.

.. versionchanged:: 3.7  
The text layer of the stdout and stderr streams now is unbuffered.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\cpython-main) (Doc) (using cmdline.rst, line 366)**

Unknown directive type "cmdoption".

.. cmdoption:: -v

Print a message each time a module is initialized, showing the place (filename or built-in module) from which it is loaded. When given twice (`:option:!!-vv`), print a message for each file that is checked for when searching for a module. Also provides information on module cleanup at exit.

.. versionchanged:: 3.10  
The `:mod:site` module reports the site-specific paths and `:file:~\.pth` files being processed.

See also `:envvar:PYTHONVERBOSE`.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 381)**

Unknown directive type "cmdoption".

```
.. cmdoption:: -W arg
```

Warning control. Python's warning machinery by default prints warning messages to `:data:`sys.stderr``.

The simplest settings apply a particular action unconditionally to all warnings emitted by a process (even those that are otherwise ignored by default)::

```
-Wdefault  # Warn once per call location
-Werror    # Convert to exceptions
-Walways   # Warn every time
-Wmodule   # Warn once per calling module
-Wonce     # Warn once per Python process
-Wignore   # Never warn
```

The action names can be abbreviated as desired and the interpreter will resolve them to the appropriate action name. For example, ```-Wi``` is the same as ```-Wignore```.

The full form of argument is::

```
action:message:category:module:lineno
```

Empty fields match all values; trailing empty fields may be omitted. For example ```-W ignore::DeprecationWarning``` ignores all `DeprecationWarning` warnings.

The `*action*` field is as explained above but only applies to warnings that match the remaining fields.

The `*message*` field must match the whole warning message; this match is case-insensitive.

The `*category*` field matches the warning category (ex: ```DeprecationWarning```). This must be a class name; the match test whether the actual warning category of the message is a subclass of the specified warning category.

The `*module*` field matches the (fully-qualified) module name; this match is case-sensitive.

The `*lineno*` field matches the line number, where zero matches all line numbers and is thus equivalent to an omitted line number.

Multiple `:option:`-W`` options can be given; when a warning matches more than one option, the action for the last matching option is performed. Invalid `:option:`-W`` options are ignored (though, a warning message is printed about invalid options when the first warning is issued).

Warnings can also be controlled using the `:envvar:`PYTHONWARNINGS`` environment variable and from within a Python program using the `:mod:`warnings`` module. For example, the `:func:`warnings.filterwarnings`` function can be used to use a regular expression on the warning message.

See `:ref:`warning-filter`` and `:ref:`describing-warning-filters`` for more details.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 439)**

Unknown directive type "cmdoption".

```
.. cmdoption:: -x
```

Skip the first line of the source, allowing use of non-Unix forms of ```#!cmd```. This is intended for a DOS specific hack only.



Unknown directive type "cmdoption".

```
.. cmdoption:: -X
```

Reserved for various implementation-specific options. CPython currently defines the following possible values:

- \* ``-X faulthandler`` to enable :mod:`faulthandler`;
- \* ``-X showrefcount`` to output the total reference count and number of used memory blocks when the program finishes or after each statement in the interactive interpreter. This only works on :ref:`debug builds <debug-build>`.
- \* ``-X tracemalloc`` to start tracing Python memory allocations using the :mod:`tracemalloc` module. By default, only the most recent frame is stored in a traceback of a trace. Use ``-X tracemalloc=NFRAME`` to start tracing with a traceback limit of \*NFRAME\* frames. See the :func:`tracemalloc.start` for more information.
- \* ``-X importtime`` to show how long each import takes. It shows module name, cumulative time (including nested imports) and self time (excluding nested imports). Note that its output may be broken in multi-threaded application. Typical usage is ``python3 -X importtime -c 'import asyncio'``. See also :envvar:`PYTHONPROFILEIMPORTTIME`.
- \* ``-X dev``: enable :ref:`Python Development Mode <devmode>`, introducing additional runtime checks that are too expensive to be enabled by default.
- \* ``-X utf8`` enables the :ref:`Python UTF-8 Mode <utf8-mode>`.
- \* ``-X utf8=0`` explicitly disables :ref:`Python UTF-8 Mode <utf8-mode>` (even when it would otherwise activate automatically).
- \* ``-X pycache\_prefix=PATH`` enables writing ``.pyc`` files to a parallel tree rooted at the given directory instead of to the code tree. See also :envvar:`PYTHONPYCACHEPREFIX`.
- \* ``-X warn\_default\_encoding`` issues a :class:`EncodingWarning` when the locale-specific default encoding is used for opening files. See also :envvar:`PYTHONWARNDEFAULTENCODING`.
- \* ``-X no\_debug\_ranges`` disables the inclusion of the tables mapping extra location information (end line, start column offset and end column offset) to every instruction in code objects. This is useful when smaller code objects and pyc files are desired as well as suppressing the extra visual location indicators when the interpreter displays tracebacks. See also :envvar:`PYTHONNODEBUGRANGES`.
- \* ``-X frozen\_modules`` determines whether or not frozen modules are ignored by the import machinery. A value of "on" means they get imported and "off" means they are ignored. The default is "on" if this is an installed Python (the normal case). If it's under development (running from the source tree) then the default is "off". Note that the "importlib\_bootstrap" and "importlib\_bootstrap\_external" frozen modules are always used, even if this flag is set to "off".

It also allows passing arbitrary values and retrieving them through the :data:`sys.\_xoptions` dictionary.

- ```
.. versionchanged:: 3.2
    The :option:`-X` option was added.

.. versionadded:: 3.3
    The ``-X faulthandler`` option.

.. versionadded:: 3.4
    The ``-X showrefcount`` and ``-X tracemalloc`` options.

.. versionadded:: 3.6
    The ``-X showalloccount`` option.

.. versionadded:: 3.7
    The ``-X importtime``, ``-X dev`` and ``-X utf8`` options.

.. versionadded:: 3.8
    The ``-X pycache_prefix`` option. The ``-X dev`` option now logs
    ``close()`` exceptions in :class:`io.IOBase` destructor.

.. versionchanged:: 3.9
    Using ``-X dev`` option, check *encoding* and *errors* arguments on
    string encoding and decoding operations.

    The ``-X showalloccount`` option has been removed.

.. versionadded:: 3.10
    The ``-X warn_default_encoding`` option.
```

```
.. deprecated-removed:: 3.9 3.10
   The ``-X oldparser`` option.

.. versionadded:: 3.11
   The ``-X no_debug_ranges`` option.

.. versionadded:: 3.11
   The ``-X frozen_modules`` option.
```

## Options you shouldn't use

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 535)**

Unknown directive type "cmdoption".

```
.. cmdoption:: -J

   Reserved for use by Jython_.
```

## Environment variables

These environment variables influence Python's behavior, they are processed before the command-line switches other than -E or -I. It is customary that command-line switches override environmental variables where there is a conflict.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 552)**

Unknown directive type "envvar".

```
.. envvar:: PYTHONHOME

   Change the location of the standard Python libraries. By default, the
   libraries are searched in :file:`{prefix}/lib/python{version}` and
   :file:`{exec_prefix}/lib/python{version}`, where :file:`{prefix}` and
   :file:`{exec_prefix}` are installation-dependent directories, both defaulting
   to :file:`/usr/local`.

   When :envvar:`PYTHONHOME` is set to a single directory, its value replaces
   both :file:`{prefix}` and :file:`{exec_prefix}`. To specify different values
   for these, set :envvar:`PYTHONHOME` to :file:`{prefix}:{exec_prefix}`.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 565)**

Unknown directive type "envvar".

```
.. envvar:: PYTHONPATH

   Augment the default search path for module files. The format is the same as
   the shell's :envvar:`PATH`: one or more directory pathnames separated by
   :data:`os.pathsep` (e.g. colons on Unix or semicolons on Windows).
   Non-existent directories are silently ignored.

   In addition to normal directories, individual :envvar:`PYTHONPATH` entries
   may refer to zipfiles containing pure Python modules (in either source or
   compiled form). Extension modules cannot be imported from zipfiles.

   The default search path is installation dependent, but generally begins with
   :file:`{prefix}/lib/python{version}` (see :envvar:`PYTHONHOME` above). It
   is *always* appended to :envvar:`PYTHONPATH`.

   An additional directory will be inserted in the search path in front of
   :envvar:`PYTHONPATH` as described above under
   :ref:`using-on-interface-options`. The search path can be manipulated from
   within a Python program as the variable :data:`sys.path`.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 586)**

Unknown directive type "envvar".

```
.. envvar:: PYTHONPLATLIBDIR
```

If this is set to a non-empty string, it overrides the `:data:`sys.platlibdir`` value.

```
.. versionadded:: 3.9
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 594)**

Unknown directive type "envvar".

```
.. envvar:: PYTHONSTARTUP
```

If this is the name of a readable file, the Python commands in that file are executed before the first prompt is displayed in interactive mode. The file is executed in the same namespace where interactive commands are executed so that objects defined or imported in it can be used without qualification in the interactive session. You can also change the prompts `:data:`sys.ps1`` and `:data:`sys.ps2`` and the hook `:data:`sys.__interactivehook__`` in this file.

```
.. audit-event:: cpython.run_startup filename envvar-PYTHONSTARTUP
```

Raises an `:ref:`auditing event <auditing>`` ```cpython.run_startup``` with the filename as the argument when called on startup.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 609)**

Unknown directive type "envvar".

```
.. envvar:: PYTHONOPTIMIZE
```

If this is set to a non-empty string it is equivalent to specifying the `:option:`-O`` option. If set to an integer, it is equivalent to specifying `:option:`-O`` multiple times.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 616)**

Unknown directive type "envvar".

```
.. envvar:: PYTHONBREAKPOINT
```

If this is set, it names a callable using dotted-path notation. The module containing the callable will be imported and then the callable will be run by the default implementation of `:func:`sys.breakpointhook`` which itself is called by built-in `:func:`breakpoint``. If not set, or set to the empty string, it is equivalent to the value `"pdb.set_trace"`. Setting this to the string `"0"` causes the default implementation of `:func:`sys.breakpointhook`` to do nothing but return immediately.

```
.. versionadded:: 3.7
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 628)**

Unknown directive type "envvar".

```
.. envvar:: PYTHONDEBUG
```

If this is set to a non-empty string it is equivalent to specifying the `:option:`-d`` option. If set to an integer, it is equivalent to specifying `:option:`-d`` multiple times.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 635)**

Unknown directive type "envvar".

```
.. envvar:: PYTHONINSPECT
```

If this is set to a non-empty string it is equivalent to specifying the :option:`-i` option.

This variable can also be modified by Python code using :data:`os.environ` to force inspect mode on program termination.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 644)**

Unknown directive type "envvar".

```
.. envvar:: PYTHONUNBUFFERED
```

If this is set to a non-empty string it is equivalent to specifying the :option:`-u` option.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 650)**

Unknown directive type "envvar".

```
.. envvar:: PYTHONVERBOSE
```

If this is set to a non-empty string it is equivalent to specifying the :option:`-v` option. If set to an integer, it is equivalent to specifying :option:`-v` multiple times.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 657)**

Unknown directive type "envvar".

```
.. envvar:: PYTHONCASEOK
```

If this is set, Python ignores case in :keyword:`import` statements. This only works on Windows and macOS.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 663)**

Unknown directive type "envvar".

```
.. envvar:: PYTHONDONTWRITEBYTECODE
```

If this is set to a non-empty string, Python won't try to write ``.pyc`` files on the import of source modules. This is equivalent to specifying the :option:`-B` option.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 670)**

Unknown directive type "envvar".

```
.. envvar:: PYTHONPYCACHEPREFIX
```

If this is set, Python will write ``.pyc`` files in a mirror directory tree

at this path, instead of in ``\_\_pycache\_\_`` directories within the source tree. This is equivalent to specifying the :option:`-X` ``pycache\_prefix=PATH`` option.

.. versionadded:: 3.8

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 680)**

Unknown directive type "envvar".

.. envvar:: PYTHONHASHSEED

If this variable is not set or set to ``random``, a random value is used to seed the hashes of str and bytes objects.

If :envvar:`PYTHONHASHSEED` is set to an integer value, it is used as a fixed seed for generating the hash() of the types covered by the hash randomization.

Its purpose is to allow repeatable hashing, such as for selftests for the interpreter itself, or to allow a cluster of python processes to share hash values.

The integer must be a decimal number in the range [0,4294967295]. Specifying the value 0 will disable hash randomization.

.. versionadded:: 3.2.3

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 699)**

Unknown directive type "envvar".

.. envvar:: PYTHONIOENCODING

If this is set before running the interpreter, it overrides the encoding used for stdin/stdout/stderr, in the syntax ``encodingname:errorhandler``. Both the ``encodingname`` and the ``:errorhandler`` parts are optional and have the same meaning as in :func:`str.encode`.

For stderr, the ``:errorhandler`` part is ignored; the handler will always be ``'backslashreplace'``.

.. versionchanged:: 3.4  
The ``encodingname`` part is now optional.

.. versionchanged:: 3.6  
On Windows, the encoding specified by this variable is ignored for interactive console buffers unless :envvar:`PYTHONLEGACYWINDOWSSTDIO` is also specified. Files and pipes redirected through the standard streams are not affected.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 717)**

Unknown directive type "envvar".

.. envvar:: PYTHONNOUSERSITE

If this is set, Python won't add the :data:`user site-packages` directory <site.USER\_SITE> to :data:`sys.path`.

.. seealso::

:pep:`370` -- Per user site-packages directory

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 727)**

Unknown directive type "envvar".

```
.. envvar:: PYTHONUSERBASE
```

Defines the :data:`user base directory <site.USER\_BASE>`, which is used to compute the path of the :data:`user site-packages directory <site.USER\_SITE>` and :ref:`Distutils installation paths <inst-alt-install-user>` for ``python setup.py install --user``.

```
.. seealso::
```

```
:pep:`370` -- Per user site-packages directory
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 739)**

Unknown directive type "envvar".

```
.. envvar:: PYTHONEXECUTABLE
```

If this environment variable is set, ``sys.argv[0]`` will be set to its value instead of the value got through the C runtime. Only works on macOS.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 745)**

Unknown directive type "envvar".

```
.. envvar:: PYTHONWARNINGS
```

This is equivalent to the :option:`-W` option. If set to a comma separated string, it is equivalent to specifying :option:`-W` multiple times, with filters later in the list taking precedence over those earlier in the list.

The simplest settings apply a particular action unconditionally to all warnings emitted by a process (even those that are otherwise ignored by default)::

```
PYTHONWARNINGS=default # Warn once per call location
PYTHONWARNINGS=error   # Convert to exceptions
PYTHONWARNINGS=always  # Warn every time
PYTHONWARNINGS=module  # Warn once per calling module
PYTHONWARNINGS=once    # Warn once per Python process
PYTHONWARNINGS=ignore  # Never warn
```

See :ref:`warning-filter` and :ref:`describing-warning-filters` for more details.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 767)**

Unknown directive type "envvar".

```
.. envvar:: PYTHONFAULTHANDLER
```

If this environment variable is set to a non-empty string, :func:`faulthandler.enable` is called at startup: install a handler for :const:`SIGSEGV`, :const:`SIGFPE`, :const:`SIGABRT`, :const:`SIGBUS` and :const:`SIGILL` signals to dump the Python traceback. This is equivalent to :option:`-X` ``faulthandler`` option.

```
.. versionadded:: 3.3
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 778)**

Unknown directive type "envvar".

```
.. envvar:: PYTHONTRACEMALLOC
```

If this environment variable is set to a non-empty string, start tracing Python memory allocations using the :mod:`tracemalloc` module. The value of the variable is the maximum number of frames stored in a traceback of a trace. For example, ``PYTHONTRACEMALLOC=1`` stores only the most recent frame. See the :func:`tracemalloc.start` for more information.

.. versionadded:: 3.4

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 789)**

Unknown directive type "envvar".

.. envvar:: PYTHONPROFILEIMPORTTIME

If this environment variable is set to a non-empty string, Python will show how long each import takes. This is exactly equivalent to setting ``-X importtime`` on the command line.

.. versionadded:: 3.7

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 798)**

Unknown directive type "envvar".

.. envvar:: PYTHONASYNCIODEBUG

If this environment variable is set to a non-empty string, enable the :ref:`debug mode <asyncio-debug-mode>` of the :mod:`asyncio` module.

.. versionadded:: 3.4

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 806)**

Unknown directive type "envvar".

.. envvar:: PYTHONMALLOC

Set the Python memory allocators and/or install debug hooks.

Set the family of memory allocators used by Python:

- \* ``default``: use the :ref:`default memory allocators <default-memory-allocators>`.
- \* ``malloc``: use the :c:func:`malloc` function of the C library for all domains (:c:data:`PYMEM\_DOMAIN\_RAW`, :c:data:`PYMEM\_DOMAIN\_MEM`, :c:data:`PYMEM\_DOMAIN\_OBJ`).
- \* ``pymalloc``: use the :ref:`pymalloc allocator <pymalloc>` for :c:data:`PYMEM\_DOMAIN\_MEM` and :c:data:`PYMEM\_DOMAIN\_OBJ` domains and use the :c:func:`malloc` function for the :c:data:`PYMEM\_DOMAIN\_RAW` domain.

Install :ref:`debug hooks <pymem-debug-hooks>`:

- \* ``debug``: install debug hooks on top of the :ref:`default memory allocators <default-memory-allocators>`.
- \* ``malloc\_debug``: same as ``malloc`` but also install debug hooks.
- \* ``pymalloc\_debug``: same as ``pymalloc`` but also install debug hooks.

.. versionchanged:: 3.7  
Added the ``"default"`` allocator.

.. versionadded:: 3.6

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 834)**

Unknown directive type "envvar".

```
.. envvar:: PYTHONMALLOCSTATS
```

If set to a non-empty string, Python will print statistics of the :ref:`pymalloc` memory allocator <pymalloc>` every time a new pymalloc object arena is created, and on shutdown.

This variable is ignored if the :envvar:`PYTHONMALLOC` environment variable is used to force the :c:func:`malloc` allocator of the C library, or if Python is configured without ``pymalloc`` support.

```
.. versionchanged:: 3.6
```

This variable can now also be used on Python compiled in release mode. It now has no effect if set to an empty string.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 849)**

Unknown directive type "envvar".

```
.. envvar:: PYTHONLEGACYWINDOWSFSENCODING
```

If set to a non-empty string, the default :term:`filesystem encoding and error handler` mode will revert to their pre-3.6 values of 'mbcs' and 'replace', respectively. Otherwise, the new defaults 'utf-8' and 'surrogatepass' are used.

This may also be enabled at runtime with :func:`sys.\_enablelegacywindowsfsencoding()`.

```
.. availability:: Windows.
```

```
.. versionadded:: 3.6
```

See :pep:`529` for more details.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 864)**

Unknown directive type "envvar".

```
.. envvar:: PYTHONLEGACYWINDOWSSSTDIO
```

If set to a non-empty string, does not use the new console reader and writer. This means that Unicode characters will be encoded according to the active console code page, rather than using utf-8.

This variable is ignored if the standard streams are redirected (to files or pipes) rather than referring to console buffers.

```
.. availability:: Windows.
```

```
.. versionadded:: 3.6
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 878)**

Unknown directive type "envvar".

```
.. envvar:: PYTHONCOERCECLOCALE
```

If set to the value ``0``, causes the main Python command line application to skip coercing the legacy ASCII-based C and POSIX locales to a more capable UTF-8 based alternative.

If this variable is *not* set (or is set to a value other than ``0``), the ``LC\_ALL`` locale override environment variable is also not set, and the current locale reported for the ``LC\_CTYPE`` category is either the default ``C`` locale, or else the explicitly ASCII-based ``POSIX`` locale, then the Python CLI will attempt to configure the following locales for the ``LC\_CTYPE`` category in the order listed before loading the interpreter runtime:

```
* ``C.UTF-8``  
* ``C.utf8``
```



```
* ``UTF-8``
```

If setting one of these locale categories succeeds, then the ``LC\_CTYPE`` environment variable will also be set accordingly in the current process environment before the Python runtime is initialized. This ensures that in addition to being seen by both the interpreter itself and other locale-aware components running in the same process (such as the GNU ``readline`` library), the updated setting is also seen in subprocesses (regardless of whether or not those processes are running a Python interpreter), as well as in operations that query the environment rather than the current C locale (such as Python's own `:func:`locale.getdefaultlocale``).

Configuring one of these locales (either explicitly or via the above implicit locale coercion) automatically enables the ``surrogateescape`` `:ref:`error handler <error-handlers>`` for `:data:`sys.stdin`` and `:data:`sys.stdout`` (`:data:`sys.stderr`` continues to use ``backslashreplace`` as it does in any other locale). This stream handling behavior can be overridden using `:envvar:`PYTHONIOENCODING`` as usual.

For debugging purposes, setting ``PYTHONCOERCECLOCALE=warn`` will cause Python to emit warning messages on ``stderr`` if either the locale coercion activates, or else if a locale that *would* have triggered coercion is still active when the Python runtime is initialized.

Also note that even when locale coercion is disabled, or when it fails to find a suitable target locale, `:envvar:`PYTHONUTF8`` will still activate by default in legacy ASCII-based locales. Both features must be disabled in order to force the interpreter to use ``ASCII`` instead of ``UTF-8`` for system interfaces.

```
.. availability:: \*nix.  
  
.. versionadded:: 3.7  
   See :pep:`538` for more details.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 930)**

Unknown directive type "envvar".

```
.. envvar:: PYTHONDEVMODE
```

If this environment variable is set to a non-empty string, enable `:ref:`Python Development Mode <devmode>``, introducing additional runtime checks that are too expensive to be enabled by default.

```
.. versionadded:: 3.7
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 938)**

Unknown directive type "envvar".

```
.. envvar:: PYTHONUTF8
```

If set to ``1``, enable the `:ref:`Python UTF-8 Mode <utf8-mode>``.

If set to ``0``, disable the `:ref:`Python UTF-8 Mode <utf8-mode>``.

Setting any other non-empty string causes an error during interpreter initialisation.

```
.. versionadded:: 3.7
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 949)**

Unknown directive type "envvar".

```
.. envvar:: PYTHONWARNDEFAULTENCODING
```

If this environment variable is set to a non-empty string, issue a `:class:`EncodingWarning`` when the locale-specific default encoding is used.

See :ref:`io-encoding-warning` for details.

.. versionadded:: 3.10

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 958)**

Unknown directive type "envvar".

.. envvar:: PYTHONNODEBUGRANGES

If this variable is set, it disables the inclusion of the tables mapping extra location information (end line, start column offset and end column offset) to every instruction in code objects. This is useful when smaller code objects and pyc files are desired as well as suppressing the extra visual location indicators when the interpreter displays tracebacks.

.. versionadded:: 3.11

## Debug-mode variables

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 973)**

Unknown directive type "envvar".

.. envvar:: PYTHONTHREADDEBUG

If set, Python will print threading debug info into stdout.

Need a :ref:`debug build of Python <debug-build>`.

.. deprecated-removed:: 3.10 3.12

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 982)**

Unknown directive type "envvar".

.. envvar:: PYTHONDUMPPREFS

If set, Python will dump objects and reference counts still alive after shutting down the interpreter.

Need Python configured with the :option:`--with-trace-refs` build option.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\using\ (cpython-main) (Doc) (using) cmdline.rst, line 989)**

Unknown directive type "envvar".

.. envvar:: PYTHONDUMPPREFSFILE=FILENAME

If set, Python will dump objects and reference counts still alive after shutting down the interpreter into a file called \*FILENAME\*.

Need Python configured with the :option:`--with-trace-refs` build option.

.. versionadded:: 3.11