Initialization, Finalization, and Threads

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1)

Unknown directive type "highlight".

```
.. highlight:: c
```

See also :ref:`Python Initialization Configuration <init-config>`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 10); *backlink*

Unknown interpreted text role "ref".

Before Python Initialization

In an application embedding Python, the :c:func:`Py_Initialize` function must be called before using any other Python/C API functions; with the exception of a few functions and the :ref:`global configuration variables <global-conf-vars>`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 17); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 17); *backlink*

Unknown interpreted text role "ref".

The following functions can be safely called before Python is initialized:

- Configuration functions:
  - :c:func:`PyImport_AppendInittab`

    System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 26); *backlink*

    Unknown interpreted text role "c:func".

  - :c:func:`PyImport_ExtendInittab`

    System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 27); *backlink*

    Unknown interpreted text role "c:func".

  - :c:func:`PyInitFrozenExtensions`

    System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 28); *backlink*

    Unknown interpreted text role "c:func".

  - :c:func:`PyMem_SetAllocator`

    System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 29); *backlink*

    Unknown interpreted text role "c:func".

  - :c:func:`PyMem_SetupDebugHooks`

    System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 30); *backlink*

    Unknown interpreted text role "c:func".

  - :c:func:`PyObject_SetArenaAllocator`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 31); *backlink*

Unknown interpreted text role "c:func".

- :c:func:`Py_SetPath`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 32); *backlink*

Unknown interpreted text role "c:func".

- :c:func:`Py_SetProgramName`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 33); *backlink*

Unknown interpreted text role "c:func".

- :c:func:`Py_SetPythonHome`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 34); *backlink*

Unknown interpreted text role "c:func".

- :c:func:`Py_SetStandardStreamEncoding`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 35); *backlink*

Unknown interpreted text role "c:func".

- :c:func:`PySys_AddWarnOption`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 36); *backlink*

Unknown interpreted text role "c:func".

- :c:func:`PySys_AddXOption`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 37); *backlink*

Unknown interpreted text role "c:func".

- :c:func:`PySys_ResetWarnOptions`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 38); *backlink*

Unknown interpreted text role "c:func".

- Informative functions:
  - :c:func:`Py_IsInitialized`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 42); *backlink*

Unknown interpreted text role "c:func".

  - :c:func:`PyMem_GetAllocator`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 43); *backlink*

Unknown interpreted text role "c:func".

- :c:func:`PyObject_GetArenaAllocator`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 44); *backlink*

  Unknown interpreted text role "c:func".

- :c:func:`Py_GetBuildInfo`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 45); *backlink*

  Unknown interpreted text role "c:func".

- :c:func:`Py_GetCompiler`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 46); *backlink*

  Unknown interpreted text role "c:func".

- :c:func:`Py_GetCopyright`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 47); *backlink*

  Unknown interpreted text role "c:func".

- :c:func:`Py_GetPlatform`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 48); *backlink*

  Unknown interpreted text role "c:func".

- :c:func:`Py_GetVersion`

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 49); *backlink*

  Unknown interpreted text role "c:func".

- Utilities:
  - :c:func:`Py_DecodeLocale`

    System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 53); *backlink*

    Unknown interpreted text role "c:func".

- Memory allocators:
  - :c:func:`PyMem_RawMalloc`

    System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 57); *backlink*

    Unknown interpreted text role "c:func".

  - :c:func:`PyMem_RawRealloc`

    System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 58); *backlink*

    Unknown interpreted text role "c:func".

  - :c:func:`PyMem_RawCalloc`

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

resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 59); *backlink*

Unknown interpreted text role "c:func".

- :c:func:`PyMem_RawFree`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 60); *backlink*

Unknown interpreted text role "c:func".

Note

The following functions **should not be called** before :c:func:`Py_Initialize`: :c:func:`Py_EncodeLocale`, :c:func:`Py_GetPath`, :c:func:`Py_GetPrefix`, :c:func:`Py_GetExecPrefix`, :c:func:`Py_GetProgramFullPath`, :c:func:`Py_GetPythonHome`, :c:func:`Py_GetProgramName` and :c:func:`PyEval_InitThreads`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 64); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 64); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 64); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 64); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 64); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 64); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 64); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 64); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 64); *backlink*

Unknown interpreted text role "c:func".

Global configuration variables

Python has variables for the global configuration to control different features and options. By default, these flags are controlled by

:ref:`command line options <using-on-interface-options>`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 76); *backlink*

Unknown interpreted text role "ref".

When a flag is set by an option, the value of the flag is the number of times that the option was set. For example, ``-b`` sets :c:data:`Py_BytesWarningFlag` to 1 and ``-bb`` sets :c:data:`Py_BytesWarningFlag` to 2.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 80); *backlink*

Unknown interpreted text role "c:data".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 80); *backlink*

Unknown interpreted text role "c:data".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 84)

Unknown directive type "c:var".

```
.. c:var:: int Py_BytesWarningFlag

   Issue a warning when comparing :class:`bytes` or :class:`bytearray` with
   :class:`str` or :class:`bytes` with :class:`int`.  Issue an error if greater
   or equal to ``2``.

   Set by the :option:`-b` option.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 92)

Unknown directive type "c:var".

```
.. c:var:: int Py_DebugFlag

   Turn on parser debugging output (for expert only, depending on compilation
   options).

   Set by the :option:`-d` option and the :envvar:`PYTHONDEBUG` environment
   variable.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 100)

Unknown directive type "c:var".

```
.. c:var:: int Py_DontWriteBytecodeFlag

   If set to non-zero, Python won't try to write ``.pyc`` files on the
   import of source modules.

   Set by the :option:`-B` option and the :envvar:`PYTHONDONTWRITEBYTECODE`
   environment variable.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 108)

Unknown directive type "c:var".

```
.. c:var:: int Py_FrozenFlag

   Suppress error messages when calculating the module search path in
   :c:func:`Py_GetPath`.

   Private flag used by ``_freeze_module`` and ``frozenmain`` programs.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 115)

Unknown directive type "c:var".

```
.. c:var:: int Py_HashRandomizationFlag

   Set to ``1`` if the :envvar:`PYTHONHASHSEED` environment variable is set to
   a non-empty string.

   If the flag is non-zero, read the :envvar:`PYTHONHASHSEED` environment
   variable to initialize the secret hash seed.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 123)

Unknown directive type "c:var".

```
.. c:var:: int Py_IgnoreEnvironmentFlag

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

   Set by the :option:`-E` and :option:`-I` options.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 130)

Unknown directive type "c:var".

```
.. c:var:: int Py_InspectFlag

   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.

   Set by the :option:`-i` option and the :envvar:`PYTHONINSPECT` environment
   variable.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 139)

Unknown directive type "c:var".

```
.. c:var:: int Py_InteractiveFlag

   Set by the :option:`-i` option.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 143)

Unknown directive type "c:var".

```
.. c:var:: int Py_IsolatedFlag

   Run Python in isolated mode. In isolated mode :data:`sys.path` contains
   neither the script's directory nor the user's site-packages directory.

   Set by the :option:`-I` option.

   .. versionadded:: 3.4
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 152)

Unknown directive type "c:var".

```
.. c:var:: int Py_LegacyWindowsFSEncodingFlag

   If the flag is non-zero, use the ``mbcs`` encoding with ``replace`` error
   handler, instead of the UTF-8 encoding with ``surrogatepass`` error handler,
   for the :term:`filesystem encoding and error handler`.

   Set to ``1`` if the :envvar:`PYTHONLEGACYWINDOWSFSENCODING` environment
   variable is set to a non-empty string.

   See :pep:`529` for more details.

   .. availability:: Windows.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 165)

Unknown directive type "c:var".

```
.. c:var:: int Py_LegacyWindowsStdioFlag

   If the flag is non-zero, use :class:`io.FileIO` instead of
   :class:`WindowsConsoleIO` for :mod:`sys` standard streams.

   Set to ``1`` if the :envvar:`PYTHONLEGACYWINDOWSSTDIO` environment
   variable is set to a non-empty string.

   See :pep:`528` for more details.

   .. availability:: Windows.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 177)

Unknown directive type "c:var".

```
.. c:var:: int Py_NoSiteFlag

   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).

   Set by the :option:`-S` option.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 186)

Unknown directive type "c:var".

```
.. c:var:: int Py_NoUserSiteDirectory

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

   Set by the :option:`-s` and :option:`-I` options, and the
   :envvar:`PYTHONNOUSERSITE` environment variable.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 194)

Unknown directive type "c:var".

```
.. c:var:: int Py_OptimizeFlag

   Set by the :option:`-O` option and the :envvar:`PYTHONOPTIMIZE` environment
   variable.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 199)

Unknown directive type "c:var".

```
.. c:var:: int Py_QuietFlag

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

   Set by the :option:`-q` option.

   .. versionadded:: 3.2
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 207)

Unknown directive type "c:var".

```
.. c:var:: int Py_UnbufferedStdioFlag

   Force the stdout and stderr streams to be unbuffered.

   Set by the :option:`-u` option and the :envvar:`PYTHONUNBUFFERED`
   environment variable.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 214)

Unknown directive type "c:var".

```
.. c:var:: int Py_VerboseFlag

   Print a message each time a module is initialized, showing the place
   (filename or built-in module) from which it is loaded.  If greater or equal
   to ``2``, print a message for each file that is checked for when
   searching for a module. Also provides information on module cleanup at exit.

   Set by the :option:`-v` option and the :envvar:`PYTHONVERBOSE` environment
   variable.
```

Initializing and finalizing the interpreter

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 229)

Unknown directive type "c:function".

```
.. c:function:: void Py_Initialize()

   .. index::
      single: Py_SetProgramName()
      single: PyEval_InitThreads()
      single: modules (in module sys)
      single: path (in module sys)
      module: builtins
      module: __main__
      module: sys
      triple: module; search; path
      single: PySys_SetArgv()
      single: PySys_SetArgvEx()
      single: Py_FinalizeEx()

   Initialize the Python interpreter.  In an application embedding  Python,
   this should be called before using any other Python/C API functions; see
   :ref:`Before Python Initialization <pre-init-safe>` for the few exceptions.

   This initializes
   the table of loaded modules (``sys.modules``), and creates the fundamental
   modules :mod:`builtins`, :mod:`__main__` and :mod:`sys`.  It also initializes
   the module search path (``sys.path``). It does not set ``sys.argv``; use
   :c:func:`PySys_SetArgvEx` for that.  This is a no-op when called for a second time
   (without calling :c:func:`Py_FinalizeEx` first).  There is no return value; it is a
   fatal error if the initialization fails.

   .. note::
      On Windows, changes the console mode from ``O_TEXT`` to ``O_BINARY``, which will
      also affect non-Python uses of the console using the C Runtime.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 261)

Unknown directive type "c:function".

```
.. c:function:: void Py_InitializeEx(int initsigs)

   This function works like :c:func:`Py_Initialize` if *initsigs* is ``1``. If
   *initsigs* is ``0``, it skips initialization registration of signal handlers, which
   might be useful when Python is embedded.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 268)

Unknown directive type "c:function".

```
.. c:function:: int Py_IsInitialized()

   Return true (nonzero) when the Python interpreter has been initialized, false
   (zero) if not.  After :c:func:`Py_FinalizeEx` is called, this returns false until
   :c:func:`Py_Initialize` is called again.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 275)

Unknown directive type "c:function".

```
.. c:function:: int Py_FinalizeEx()

   Undo all initializations made by :c:func:`Py_Initialize` and subsequent use of
   Python/C API functions, and destroy all sub-interpreters (see
   :c:func:`Py_NewInterpreter` below) that were created and not yet destroyed since
   the last call to :c:func:`Py_Initialize`.  Ideally, this frees all memory
   allocated by the Python interpreter.  This is a no-op when called for a second
   time (without calling :c:func:`Py_Initialize` again first).  Normally the
   return value is ``0``.  If there were errors during finalization
   (flushing buffered data), ``-1`` is returned.

   This function is provided for a number of reasons.  An embedding application
   might want to restart Python without having to restart the application itself.
   An application that has loaded the Python interpreter from a dynamically
   loadable library (or DLL) might want to free all memory allocated by Python
   before unloading the DLL. During a hunt for memory leaks in an application a
   developer might want to free all memory allocated by Python before exiting from
   the application.

   **Bugs and caveats:** The destruction of modules and objects in modules is done
   in random order; this may cause destructors (:meth:`__del__` methods) to fail
   when they depend on other objects (even functions) or modules.  Dynamically
   loaded extension modules loaded by Python are not unloaded.  Small amounts of
   memory allocated by the Python interpreter may not be freed (if you find a leak,
   please report it).  Memory tied up in circular references between objects is not
   freed.  Some memory allocated by extension modules may not be freed.  Some
   extensions may not work properly if their initialization routine is called more
   than once; this can happen if an application calls :c:func:`Py_Initialize` and
```

```
   :c:func:`Py_FinalizeEx` more than once.

   .. audit-event:: cpython._PySys_ClearAuditHooks "" c.Py_FinalizeEx

   .. versionadded:: 3.6
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 309)

Unknown directive type "c:function".

```
.. c:function:: void Py_Finalize()

   This is a backwards-compatible version of :c:func:`Py_FinalizeEx` that
   disregards the return value.
```

Process-wide parameters

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 319)

Unknown directive type "c:function".

```
.. c:function:: int Py_SetStandardStreamEncoding(const char *encoding, const char *errors)

   .. index::
      single: Py_Initialize()
      single: main()
      triple: stdin; stdout; sdterr

   This API is kept for backward compatibility: setting
   :c:member:`PyConfig.stdio_encoding` and :c:member:`PyConfig.stdio_errors`
   should be used instead, see :ref:`Python Initialization Configuration
   <init-config>`.

   This function should be called before :c:func:`Py_Initialize`, if it is
   called at all. It specifies which encoding and error handling to use
   with standard IO, with the same meanings as in :func:`str.encode`.

   It overrides :envvar:`PYTHONIOENCODING` values, and allows embedding code
   to control IO encoding when the environment variable does not work.

   *encoding* and/or *errors* may be ``NULL`` to use
   :envvar:`PYTHONIOENCODING` and/or default values (depending on other
   settings).

   Note that :data:`sys.stderr` always uses the "backslashreplace" error
   handler, regardless of this (or any other) setting.

   If :c:func:`Py_FinalizeEx` is called, this function will need to be called
   again in order to affect subsequent calls to :c:func:`Py_Initialize`.

   Returns ``0`` if successful, a nonzero value on error (e.g. calling after the
   interpreter has already been initialized).

   .. versionadded:: 3.4

   .. deprecated:: 3.11
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 356)

Unknown directive type "c:function".

```
.. c:function:: void Py_SetProgramName(const wchar_t *name)

   .. index::
      single: Py_Initialize()
      single: main()
      single: Py_GetPath()

   This API is kept for backward compatibility: setting
   :c:member:`PyConfig.program_name` should be used instead, see :ref:`Python
   Initialization Configuration <init-config>`.

   This function should be called before :c:func:`Py_Initialize` is called for
   the first time, if it is called at all.  It tells the interpreter the value
   of the ``argv[0]`` argument to the :c:func:`main` function of the program
   (converted to wide characters).
   This is used by :c:func:`Py_GetPath` and some other functions below to find
   the Python run-time libraries relative to the interpreter executable.  The
   default value is ``'python'``.  The argument should point to a
   zero-terminated wide character string in static storage whose contents will not
   change for the duration of the program's execution.  No code in the Python
   interpreter will change the contents of this storage.

   Use :c:func:`Py_DecodeLocale` to decode a bytes string to get a
   :c:type:`wchar_*` string.
```

```
   .. deprecated:: 3.11
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 384)

Unknown directive type "c:function".

```
.. c:function:: wchar* Py_GetProgramName()

   .. index:: single: Py_SetProgramName()

   Return the program name set with :c:func:`Py_SetProgramName`, or the default.
   The returned string points into static storage; the caller should not modify its
   value.

   This function should not be called before :c:func:`Py_Initialize`, otherwise
   it returns ``NULL``.

   .. versionchanged:: 3.10
      It now returns ``NULL`` if called before :c:func:`Py_Initialize`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 399)

Unknown directive type "c:function".

```
.. c:function:: wchar_t* Py_GetPrefix()

   Return the *prefix* for installed platform-independent files. This is derived
   through a number of complicated rules from the program name set with
   :c:func:`Py_SetProgramName` and some environment variables; for example, if the
   program name is ``'/usr/local/bin/python'``, the prefix is ``'/usr/local'``. The
   returned string points into static storage; the caller should not modify its
   value.  This corresponds to the :makevar:`prefix` variable in the top-level
   :file:`Makefile` and the ``--prefix`` argument to the :program:`configure`
   script at build time.  The value is available to Python code as ``sys.prefix``.
   It is only useful on Unix.  See also the next function.

   This function should not be called before :c:func:`Py_Initialize`, otherwise
   it returns ``NULL``.

   .. versionchanged:: 3.10
      It now returns ``NULL`` if called before :c:func:`Py_Initialize`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 418)

Unknown directive type "c:function".

```
.. c:function:: wchar_t* Py_GetExecPrefix()

   Return the *exec-prefix* for installed platform-*dependent* files.  This is
   derived through a number of complicated rules from the program name set with
   :c:func:`Py_SetProgramName` and some environment variables; for example, if the
   program name is ``'/usr/local/bin/python'``, the exec-prefix is
   ``'/usr/local'``.  The returned string points into static storage; the caller
   should not modify its value.  This corresponds to the :makevar:`exec_prefix`
   variable in the top-level :file:`Makefile` and the ``--exec-prefix``
   argument to the :program:`configure` script at build  time.  The value is
   available to Python code as ``sys.exec_prefix``.  It is only useful on Unix.

   Background: The exec-prefix differs from the prefix when platform dependent
   files (such as executables and shared libraries) are installed in a different
   directory tree.  In a typical installation, platform dependent files may be
   installed in the :file:`/usr/local/plat` subtree while platform independent may
   be installed in :file:`/usr/local`.

   Generally speaking, a platform is a combination of hardware and software
   families, e.g.  Sparc machines running the Solaris 2.x operating system are
   considered the same platform, but Intel machines running Solaris 2.x are another
   platform, and Intel machines running Linux are yet another platform.  Different
   major revisions of the same operating system generally also form different
   platforms.  Non-Unix operating systems are a different story; the installation
   strategies on those systems are so different that the prefix and exec-prefix are
   meaningless, and set to the empty string. Note that compiled Python bytecode
   files are platform independent (but not independent from the Python version by
   which they were compiled!).

   System administrators will know how to configure the :program:`mount` or
   :program:`automount` programs to share :file:`/usr/local` between platforms
   while having :file:`/usr/local/plat` be a different filesystem for each
   platform.

   This function should not be called before :c:func:`Py_Initialize`, otherwise
   it returns ``NULL``.

   .. versionchanged:: 3.10
```

```
   It now returns ``NULL`` if called before :c:func:`Py_Initialize`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 459)

Unknown directive type "c:function".

```
.. c:function:: wchar_t* Py_GetProgramFullPath()

   .. index::
      single: Py_SetProgramName()
      single: executable (in module sys)

   Return the full program name of the Python executable; this is  computed as a
   side-effect of deriving the default module search path  from the program name
   (set by :c:func:`Py_SetProgramName` above). The returned string points into
   static storage; the caller should not modify its value.  The value is available
   to Python code as ``sys.executable``.

   This function should not be called before :c:func:`Py_Initialize`, otherwise
   it returns ``NULL``.

   .. versionchanged:: 3.10
      It now returns ``NULL`` if called before :c:func:`Py_Initialize`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 478)

Unknown directive type "c:function".

```
.. c:function:: wchar_t* Py_GetPath()

   .. index::
      triple: module; search; path
      single: path (in module sys)
      single: Py_SetPath()

   Return the default module search path; this is computed from the program name
   (set by :c:func:`Py_SetProgramName` above) and some environment variables.
   The returned string consists of a series of directory names separated by a
   platform dependent delimiter character.  The delimiter character is ``':'``
   on Unix and macOS, ``';'`` on Windows.  The returned string points into
   static storage; the caller should not modify its value.  The list
   :data:`sys.path` is initialized with this value on interpreter startup; it
   can be (and usually is) modified later to change the search path for loading
   modules.

   This function should not be called before :c:func:`Py_Initialize`, otherwise
   it returns ``NULL``.

   .. XXX should give the exact rules

   .. versionchanged:: 3.10
      It now returns ``NULL`` if called before :c:func:`Py_Initialize`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 504)

Unknown directive type "c:function".

```
.. c:function::  void Py_SetPath(const wchar_t *)

   .. index::
      triple: module; search; path
      single: path (in module sys)
      single: Py_GetPath()

   This API is kept for backward compatibility: setting
   :c:member:`PyConfig.module_search_paths` and
   :c:member:`PyConfig.module_search_paths_set` should be used instead, see
   :ref:`Python Initialization Configuration <init-config>`.

   Set the default module search path.  If this function is called before
   :c:func:`Py_Initialize`, then :c:func:`Py_GetPath` won't attempt to compute a
   default search path but uses the one provided instead.  This is useful if
   Python is embedded by an application that has full knowledge of the location
   of all modules.  The path components should be separated by the platform
   dependent delimiter character, which is ``':'`` on Unix and macOS, ``';'``
   on Windows.

   This also causes :data:`sys.executable` to be set to the program
   full path (see :c:func:`Py_GetProgramFullPath`) and for :data:`sys.prefix` and
   :data:`sys.exec_prefix` to be empty.  It is up to the caller to modify these
   if required after calling :c:func:`Py_Initialize`.

   Use :c:func:`Py_DecodeLocale` to decode a bytes string to get a
   :c:type:`wchar_*` string.
```

```
   The path argument is copied internally, so the caller may free it after the
   call completes.

   .. versionchanged:: 3.8
      The program full path is now used for :data:`sys.executable`, instead
      of the program name.

   .. deprecated:: 3.11
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 542)

Unknown directive type "c:function".

```
.. c:function:: const char* Py_GetVersion()

   Return the version of this Python interpreter.  This is a string that looks
   something like ::

      "3.0a5+ (py3k:63103M, May 12 2008, 00:53:55) \n[GCC 4.2.3]"

   .. index:: single: version (in module sys)

   The first word (up to the first space character) is the current Python version;
   the first three characters are the major and minor version separated by a
   period.  The returned string points into static storage; the caller should not
   modify its value.  The value is available to Python code as :data:`sys.version`.

   See also the :data:`Py_Version` constant.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 559)

Unknown directive type "c:function".

```
.. c:function:: const char* Py_GetPlatform()

   .. index:: single: platform (in module sys)

   Return the platform identifier for the current platform.  On Unix, this is
   formed from the "official" name of the operating system, converted to lower
   case, followed by the major revision number; e.g., for Solaris 2.x, which is
   also known as SunOS 5.x, the value is ``'sunos5'``.  On macOS, it is
   ``'darwin'``.  On Windows, it is ``'win'``.  The returned string points into
   static storage; the caller should not modify its value.  The value is available
   to Python code as ``sys.platform``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 572)

Unknown directive type "c:function".

```
.. c:function:: const char* Py_GetCopyright()

   Return the official copyright string for the current Python version, for example

   ``'Copyright 1991-1995 Stichting Mathematisch Centrum, Amsterdam'``

   .. index:: single: copyright (in module sys)

   The returned string points into static storage; the caller should not modify its
   value.  The value is available to Python code as ``sys.copyright``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 584)

Unknown directive type "c:function".

```
.. c:function:: const char* Py_GetCompiler()

   Return an indication of the compiler used to build the current Python version,
   in square brackets, for example::

      "[GCC 2.7.2.2]"

   .. index:: single: version (in module sys)

   The returned string points into static storage; the caller should not modify its
   value.  The value is available to Python code as part of the variable
   ``sys.version``.
```

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

main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 598)

Unknown directive type "c:function".

```
.. c:function:: const char* Py_GetBuildInfo()

   Return information about the sequence number and build date and time  of the
   current Python interpreter instance, for example ::

      "#67, Aug  1 1997, 22:34:28"

   .. index:: single: version (in module sys)

   The returned string points into static storage; the caller should not modify its
   value.  The value is available to Python code as part of the variable
   ``sys.version``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 612)

Unknown directive type "c:function".

```
.. c:function:: void PySys_SetArgvEx(int argc, wchar_t **argv, int updatepath)

   .. index::
      single: main()
      single: Py_FatalError()
      single: argv (in module sys)

   Set :data:`sys.argv` based on *argc* and *argv*.  These parameters are
   similar to those passed to the program's :c:func:`main` function with the
   difference that the first entry should refer to the script file to be
   executed rather than the executable hosting the Python interpreter.  If there
   isn't a script that will be run, the first entry in *argv* can be an empty
   string.  If this function fails to initialize :data:`sys.argv`, a fatal
   condition is signalled using :c:func:`Py_FatalError`.

   If *updatepath* is zero, this is all the function does.  If *updatepath*
   is non-zero, the function also modifies :data:`sys.path` according to the
   following algorithm:

   - If the name of an existing script is passed in ``argv[0]``, the absolute
     path of the directory where the script is located is prepended to
     :data:`sys.path`.
   - Otherwise (that is, if *argc* is ``0`` or ``argv[0]`` doesn't point
     to an existing file name), an empty string is prepended to
     :data:`sys.path`, which is the same as prepending the current working
     directory (``"."``).

   Use :c:func:`Py_DecodeLocale` to decode a bytes string to get a
   :c:type:`wchar_*` string.

   See also :c:member:`PyConfig.orig_argv` and :c:member:`PyConfig.argv`
   members of the :ref:`Python Initialization Configuration <init-config>`.

   .. note::
      It is recommended that applications embedding the Python interpreter
      for purposes other than executing a single script pass ``0`` as *updatepath*,
      and update :data:`sys.path` themselves if desired.
      See `CVE-2008-5983 <https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2008-5983>`_.

      On versions before 3.1.3, you can achieve the same effect by manually
      popping the first :data:`sys.path` element after having called
      :c:func:`PySys_SetArgv`, for example using::

         PyRun_SimpleString("import sys; sys.path.pop(0)\n");

   .. versionadded:: 3.1.3

   .. XXX impl. doesn't seem consistent in allowing ``0``/``NULL`` for the params;
      check w/ Guido.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 663)

Unknown directive type "c:function".

```
.. c:function:: void PySys_SetArgv(int argc, wchar_t **argv)

   This function works like :c:func:`PySys_SetArgvEx` with *updatepath* set
   to ``1`` unless the :program:`python` interpreter was started with the
   :option:`-I`.

   Use :c:func:`Py_DecodeLocale` to decode a bytes string to get a
   :c:type:`wchar_*` string.

   See also :c:member:`PyConfig.orig_argv` and :c:member:`PyConfig.argv`
   members of the :ref:`Python Initialization Configuration <init-config>`.

   .. versionchanged:: 3.4 The *updatepath* value depends on :option:`-I`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 678)

Unknown directive type "c:function".

```
.. c:function:: void Py_SetPythonHome(const wchar_t *home)

   This API is kept for backward compatibility: setting
   :c:member:`PyConfig.home` should be used instead, see :ref:`Python
   Initialization Configuration <init-config>`.

   Set the default "home" directory, that is, the location of the standard
   Python libraries.  See :envvar:`PYTHONHOME` for the meaning of the
   argument string.

   The argument should point to a zero-terminated character string in static
   storage whose contents will not change for the duration of the program's
   execution.  No code in the Python interpreter will change the contents of
   this storage.

   Use :c:func:`Py_DecodeLocale` to decode a bytes string to get a
   :c:type:`wchar_*` string.

   .. deprecated:: 3.11
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 699)

Unknown directive type "c:function".

```
.. c:function:: w_char* Py_GetPythonHome()

   Return the default "home", that is, the value set by a previous call to
   :c:func:`Py_SetPythonHome`, or the value of the :envvar:`PYTHONHOME`
   environment variable if it is set.

   This function should not be called before :c:func:`Py_Initialize`, otherwise
   it returns ``NULL``.

   .. versionchanged:: 3.10
      It now returns ``NULL`` if called before :c:func:`Py_Initialize`.
```

Thread State and the Global Interpreter Lock

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 717)

Unknown directive type "index".

```
.. index::
   single: global interpreter lock
   single: interpreter lock
   single: lock, interpreter
```

The Python interpreter is not fully thread-safe. In order to support multi-threaded Python programs, there's a global lock, called the :term:`global interpreter lock` or :term:`GIL`, that must be held by the current thread before it can safely access Python objects. Without the lock, even the simplest operations could cause problems in a multi-threaded program: for example, when two threads simultaneously increment the reference count of the same object, the reference count could end up being incremented only once instead of twice.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 722); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 722); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 730)

Unknown directive type "index".

```
.. index:: single: setswitchinterval() (in module sys)
```

Therefore, the rule exists that only the thread that has acquired the :term:`GIL` may operate on Python objects or call Python/C API functions. In order to emulate concurrency of execution, the interpreter regularly tries to switch threads (see

:func:`sys.setswitchinterval`). The lock is also released around potentially blocking I/O operations like reading or writing a file, so that other Python threads can run in the meantime.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 732); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 732); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 739)

Unknown directive type "index".

```
.. index::
   single: PyThreadState
   single: PyThreadState
```

The Python interpreter keeps some thread-specific bookkeeping information inside a data structure called :c:type:`PyThreadState`. There's also one global variable pointing to the current :c:type:`PyThreadState`: it can be retrieved using :c:func:`PyThreadState_Get`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 743); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 743); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 743); *backlink*

Unknown interpreted text role "c:func".

Releasing the GIL from extension code

Most extension code manipulating the :term:`GIL` has the following simple structure:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 751); *backlink*

Unknown interpreted text role "term".

```
Save the thread state in a local variable.
Release the global interpreter lock.
... Do some blocking I/O operation ...
Reacquire the global interpreter lock.
Restore the thread state from the local variable.
```

This is so common that a pair of macros exists to simplify it:

```
Py_BEGIN_ALLOW_THREADS
... Do some blocking I/O operation ...
Py_END_ALLOW_THREADS
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 766)

Unknown directive type "index".

```
.. index::
   single: Py_BEGIN_ALLOW_THREADS
   single: Py_END_ALLOW_THREADS
```

The :c:macro:`Py_BEGIN_ALLOW_THREADS` macro opens a new block and declares a hidden local variable; the :c:macro:`Py_END_ALLOW_THREADS` macro closes the block.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 770); *backlink*

Unknown interpreted text role "c:macro".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 770); *backlink*

Unknown interpreted text role "c:macro".

The block above expands to the following code:

```
PyThreadState *_save;

_save = PyEval_SaveThread();
... Do some blocking I/O operation ...
PyEval_RestoreThread(_save);
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 782)

Unknown directive type "index".

```
.. index::
   single: PyEval_RestoreThread()
   single: PyEval_SaveThread()
```

Here is how these functions work: the global interpreter lock is used to protect the pointer to the current thread state. When releasing the lock and saving the thread state, the current thread state pointer must be retrieved before the lock is released (since another thread could immediately acquire the lock and store its own thread state in the global variable). Conversely, when acquiring the lock and restoring the thread state, the lock must be acquired before storing the thread state pointer.

Note

Calling system I/O functions is the most common use case for releasing the GIL, but it can also be useful before calling long-running computations which don't need access to Python objects, such as compression or cryptographic functions operating over memory buffers. For example, the standard :mod:`zlib` and :mod:`hashlib` modules release the GIL when compressing or hashing data.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 795); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 795); *backlink*

Unknown interpreted text role "mod".

Non-Python created threads

When threads are created using the dedicated Python APIs (such as the :mod:`threading` module), a thread state is automatically associated to them and the code showed above is therefore correct. However, when threads are created from C (for example by a third-party library with its own thread management), they don't hold the GIL, nor is there a thread state structure for them.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 808); *backlink*

Unknown interpreted text role "mod".

If you need to call Python code from these threads (often this will be part of a callback API provided by the aforementioned third-party library), you must first register these threads with the interpreter by creating a thread state data structure, then acquiring the GIL, and finally storing their thread state pointer, before you can start using the Python/C API. When you are done, you should reset the thread state pointer, release the GIL, and finally free the thread state data structure.

The :c:func:`PyGILState_Ensure` and :c:func:`PyGILState_Release` functions do all of the above automatically. The typical idiom for calling into Python from a C thread is:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 823); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 823); *backlink*

Unknown interpreted text role "c:func".

```
PyGILState_STATE gstate;
gstate = PyGILState_Ensure();

/* Perform Python actions here. */
result = CallSomeFunction();
/* evaluate result or handle exception */

/* Release the thread. No Python API allowed beyond this point. */
PyGILState_Release(gstate);
```

Note that the :c:func:`PyGILState_\*` functions assume there is only one global interpreter (created automatically by :c:func:`Py_Initialize`). Python supports the creation of additional interpreters (using :c:func:`Py_NewInterpreter`), but mixing multiple interpreters and the :c:func:`PyGILState_\*` API is unsupported.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 837); backlink

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 837); backlink

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 837); backlink

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 837); backlink

Unknown interpreted text role "c:func".

Cautions about fork()

Another important thing to note about threads is their behaviour in the face of the C :c:func:`fork` call. On most systems with :c:func:`fork`, after a process forks only the thread that issued the fork will exist. This has a concrete impact both on how locks must be handled and on all stored state in CPython's runtime.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 849); backlink

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 849); backlink

Unknown interpreted text role "c:func".

The fact that only the "current" thread remains means any locks held by other threads will never be released. Python solves this for :func:`os.fork` by acquiring the locks it uses internally before the fork, and releasing them afterwards. In addition, it resets any :ref:`lock-objects` in the child. When extending or embedding Python, there is no way to inform Python of additional (non-Python) locks that need to be acquired before or reset after a fork. OS facilities such as :c:func:`pthread_atfork` would need to be used to accomplish the same thing. Additionally, when extending or embedding Python, calling :c:func:`fork` directly rather than through :func:`os.fork` (and returning to or calling into Python) may result in a deadlock by one of Python's internal locks being held by a thread that is defunct after the fork. :c:func:`PyOS_AfterFork_Child` tries to reset the necessary locks, but is not always able to.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 855); backlink

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 855); backlink

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 855); backlink

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 855); backlink

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 855); backlink

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 855); backlink

Unknown interpreted text role "c:func".

The fact that all other threads go away also means that CPython's runtime state there must be cleaned up properly, which :func:`os.fork` does. This means finalizing all other :c:type:`PyThreadState` objects belonging to the current interpreter and all other :c:type:`PyInterpreterState` objects. Due to this and the special nature of the :ref:`"main" interpreter <sub-interpreter-support>`, :c:func:`fork` should only be called in that interpreter's "main" thread, where the CPython global runtime was originally initialized. The only exception is if :c:func:`exec` will be called immediately after.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 870); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 870); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 870); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 870); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 870); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 870); *backlink*

Unknown interpreted text role "c:func".

High-level API

These are the most commonly used types and functions when writing C extension code, or when embedding the Python interpreter:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 888)

Unknown directive type "c:type".

```
.. c:type:: PyInterpreterState

   This data structure represents the state shared by a number of cooperating
   threads.  Threads belonging to the same interpreter share their module
   administration and a few other internal items. There are no public members in
   this structure.

   Threads belonging to different interpreters initially share nothing, except
   process state like available memory, open file descriptors and such.  The global
   interpreter lock is also shared by all threads, regardless of to which
   interpreter they belong.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 901)

Unknown directive type "c:type".

```
.. c:type:: PyThreadState

   This data structure represents the state of a single thread.  The only public
   data member is :attr:`interp` (:c:type:`PyInterpreterState *`), which points to
   this thread's interpreter state.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 908)

Unknown directive type "c:function".

```
.. c:function:: void PyEval_InitThreads()

   .. index::
      single: PyEval_AcquireThread()
      single: PyEval_ReleaseThread()
      single: PyEval_SaveThread()
      single: PyEval_RestoreThread()
```

```
   Deprecated function which does nothing.

   In Python 3.6 and older, this function created the GIL if it didn't exist.

   .. versionchanged:: 3.9
      The function now does nothing.

   .. versionchanged:: 3.7
      This function is now called by :c:func:`Py_Initialize()`, so you don't
      have to call it yourself anymore.

   .. versionchanged:: 3.2
      This function cannot be called before :c:func:`Py_Initialize()` anymore.

   .. deprecated-removed:: 3.9 3.11

   .. index:: module: _thread
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 935)

Unknown directive type "c:function".

```
.. c:function:: int PyEval_ThreadsInitialized()

   Returns a non-zero value if :c:func:`PyEval_InitThreads` has been called.  This
   function can be called without holding the GIL, and therefore can be used to
   avoid calls to the locking API when running single-threaded.

   .. versionchanged:: 3.7
      The :term:`GIL` is now initialized by :c:func:`Py_Initialize()`.

   .. deprecated-removed:: 3.9 3.11
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 947)

Unknown directive type "c:function".

```
.. c:function:: PyThreadState* PyEval_SaveThread()

   Release the global interpreter lock (if it has been created) and reset the
   thread state to ``NULL``, returning the previous thread state (which is not
   ``NULL``).  If the lock has been created, the current thread must have
   acquired it.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 955)

Unknown directive type "c:function".

```
.. c:function:: void PyEval_RestoreThread(PyThreadState *tstate)

   Acquire the global interpreter lock (if it has been created) and set the
   thread state to *tstate*, which must not be ``NULL``.  If the lock has been
   created, the current thread must not have acquired it, otherwise deadlock
   ensues.

   .. note::
      Calling this function from a thread when the runtime is finalizing
      will terminate the thread, even if the thread was not created by Python.
      You can use :c:func:`_Py_IsFinalizing` or :func:`sys.is_finalizing` to
      check if the interpreter is in process of being finalized before calling
      this function to avoid unwanted termination.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 969)

Unknown directive type "c:function".

```
.. c:function:: PyThreadState* PyThreadState_Get()

   Return the current thread state.  The global interpreter lock must be held.
   When the current thread state is ``NULL``, this issues a fatal error (so that
   the caller needn't check for ``NULL``).
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 976)

Unknown directive type "c:function".

```
.. c:function:: PyThreadState* PyThreadState_Swap(PyThreadState *tstate)
```

```
   Swap the current thread state with the thread state given by the argument
   *tstate*, which may be ``NULL``.  The global interpreter lock must be held
   and is not released.
```

The following functions use thread-local storage, and are not compatible with sub-interpreters:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 986)

Unknown directive type "c:function".

```
.. c:function:: PyGILState_STATE PyGILState_Ensure()

   Ensure that the current thread is ready to call the Python C API regardless
   of the current state of Python, or of the global interpreter lock. This may
   be called as many times as desired by a thread as long as each call is
   matched with a call to :c:func:`PyGILState_Release`. In general, other
   thread-related APIs may be used between :c:func:`PyGILState_Ensure` and
   :c:func:`PyGILState_Release` calls as long as the thread state is restored to
   its previous state before the Release().  For example, normal usage of the
   :c:macro:`Py_BEGIN_ALLOW_THREADS` and :c:macro:`Py_END_ALLOW_THREADS` macros is
   acceptable.

   The return value is an opaque "handle" to the thread state when
   :c:func:`PyGILState_Ensure` was called, and must be passed to
   :c:func:`PyGILState_Release` to ensure Python is left in the same state. Even
   though recursive calls are allowed, these handles *cannot* be shared - each
   unique call to :c:func:`PyGILState_Ensure` must save the handle for its call
   to :c:func:`PyGILState_Release`.

   When the function returns, the current thread will hold the GIL and be able
   to call arbitrary Python code.  Failure is a fatal error.

   .. note::
      Calling this function from a thread when the runtime is finalizing
      will terminate the thread, even if the thread was not created by Python.
      You can use :c:func:`_Py_IsFinalizing` or :func:`sys.is_finalizing` to
      check if the interpreter is in process of being finalized before calling
      this function to avoid unwanted termination.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1015)

Unknown directive type "c:function".

```
.. c:function:: void PyGILState_Release(PyGILState_STATE)

   Release any resources previously acquired.  After this call, Python's state will
   be the same as it was prior to the corresponding :c:func:`PyGILState_Ensure` call
   (but generally this state will be unknown to the caller, hence the use of the
   GILState API).

   Every call to :c:func:`PyGILState_Ensure` must be matched by a call to
   :c:func:`PyGILState_Release` on the same thread.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1026)

Unknown directive type "c:function".

```
.. c:function:: PyThreadState* PyGILState_GetThisThreadState()

   Get the current thread state for this thread.  May return ``NULL`` if no
   GILState API has been used on the current thread.  Note that the main thread
   always has such a thread-state, even if no auto-thread-state call has been
   made on the main thread.  This is mainly a helper/diagnostic function.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1034)

Unknown directive type "c:function".

```
.. c:function:: int PyGILState_Check()

   Return ``1`` if the current thread is holding the GIL and ``0`` otherwise.
   This function can be called from any thread at any time.
   Only if it has had its Python thread state initialized and currently is
   holding the GIL will it return ``1``.
   This is mainly a helper/diagnostic function.  It can be useful
   for example in callback contexts or memory allocation functions when
   knowing that the GIL is locked can allow the caller to perform sensitive
   actions or otherwise behave differently.

   .. versionadded:: 3.4
```

The following macros are normally used without a trailing semicolon; look for example usage in the Python source distribution.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1052)

Unknown directive type "c:macro".

```
.. c:macro:: Py_BEGIN_ALLOW_THREADS

   This macro expands to ``{ PyThreadState *_save; _save = PyEval_SaveThread();``.
   Note that it contains an opening brace; it must be matched with a following
   :c:macro:`Py_END_ALLOW_THREADS` macro.  See above for further discussion of this
   macro.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1060)

Unknown directive type "c:macro".

```
.. c:macro:: Py_END_ALLOW_THREADS

   This macro expands to ``PyEval_RestoreThread(_save); }``. Note that it contains
   a closing brace; it must be matched with an earlier
   :c:macro:`Py_BEGIN_ALLOW_THREADS` macro.  See above for further discussion of
   this macro.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1068)

Unknown directive type "c:macro".

```
.. c:macro:: Py_BLOCK_THREADS

   This macro expands to ``PyEval_RestoreThread(_save);``: it is equivalent to
   :c:macro:`Py_END_ALLOW_THREADS` without the closing brace.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1074)

Unknown directive type "c:macro".

```
.. c:macro:: Py_UNBLOCK_THREADS

   This macro expands to ``_save = PyEval_SaveThread();``: it is equivalent to
   :c:macro:`Py_BEGIN_ALLOW_THREADS` without the opening brace and variable
   declaration.
```

Low-level API

All of the following functions must be called after :c:func:`Py_Initialize`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1084); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1086)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.7
   :c:func:`Py_Initialize()` now initializes the :term:`GIL`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1090)

Unknown directive type "c:function".

```
.. c:function:: PyInterpreterState* PyInterpreterState_New()

   Create a new interpreter state object.  The global interpreter lock need not
   be held, but may be held if it is necessary to serialize calls to this
   function.

   .. audit-event:: cpython.PyInterpreterState_New "" c.PyInterpreterState_New
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1099)

Unknown directive type "c:function".

```
.. c:function:: void PyInterpreterState_Clear(PyInterpreterState *interp)

   Reset all information in an interpreter state object.  The global interpreter
   lock must be held.

   .. audit-event:: cpython.PyInterpreterState_Clear "" c.PyInterpreterState_Clear
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1107)

Unknown directive type "c:function".

```
.. c:function:: void PyInterpreterState_Delete(PyInterpreterState *interp)

   Destroy an interpreter state object.  The global interpreter lock need not be
   held.  The interpreter state must have been reset with a previous call to
   :c:func:`PyInterpreterState_Clear`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1114)

Unknown directive type "c:function".

```
.. c:function:: PyThreadState* PyThreadState_New(PyInterpreterState *interp)

   Create a new thread state object belonging to the given interpreter object.
   The global interpreter lock need not be held, but may be held if it is
   necessary to serialize calls to this function.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1121)

Unknown directive type "c:function".

```
.. c:function:: void PyThreadState_Clear(PyThreadState *tstate)

   Reset all information in a thread state object.  The global interpreter lock
   must be held.

   .. versionchanged:: 3.9
      This function now calls the :c:member:`PyThreadState.on_delete` callback.
      Previously, that happened in :c:func:`PyThreadState_Delete`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1131)

Unknown directive type "c:function".

```
.. c:function:: void PyThreadState_Delete(PyThreadState *tstate)

   Destroy a thread state object.  The global interpreter lock need not be held.
   The thread state must have been reset with a previous call to
   :c:func:`PyThreadState_Clear`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1138)

Unknown directive type "c:function".

```
.. c:function:: void PyThreadState_DeleteCurrent(void)

   Destroy the current thread state and release the global interpreter lock.
   Like :c:func:`PyThreadState_Delete`, the global interpreter lock need not
   be held. The thread state must have been reset with a previous call
   to :c:func:`PyThreadState_Clear`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1146)

Unknown directive type "c:function".

```
.. c:function:: PyFrameObject* PyThreadState_GetFrame(PyThreadState *tstate)

   Get the current frame of the Python thread state *tstate*.

   Return a :term:`strong reference`. Return ``NULL`` if no frame is currently
   executing.

   See also :c:func:`PyEval_GetFrame`.

   *tstate* must not be ``NULL``.

   .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1160)

Unknown directive type "c:function".

```
.. c:function:: uint64_t PyThreadState_GetID(PyThreadState *tstate)

   Get the unique thread state identifier of the Python thread state *tstate*.

   *tstate* must not be ``NULL``.

   .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1169)

Unknown directive type "c:function".

```
.. c:function:: PyInterpreterState* PyThreadState_GetInterpreter(PyThreadState *tstate)

   Get the interpreter of the Python thread state *tstate*.

   *tstate* must not be ``NULL``.

   .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1178)

Unknown directive type "c:function".

```
.. c:function:: void PyThreadState_EnterTracing(PyThreadState *tstate)

   Suspend tracing and profiling in the Python thread state *tstate*.

   Resume them using the :c:func:`PyThreadState_LeaveTracing` function.

   .. versionadded:: 3.11
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1187)

Unknown directive type "c:function".

```
.. c:function:: void PyThreadState_LeaveTracing(PyThreadState *tstate)

   Resume tracing and profiling in the Python thread state *tstate* suspended
   by the :c:func:`PyThreadState_EnterTracing` function.

   See also :c:func:`PyEval_SetTrace` and :c:func:`PyEval_SetProfile`
   functions.

   .. versionadded:: 3.11
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1198)

Unknown directive type "c:function".

```
.. c:function:: PyInterpreterState* PyInterpreterState_Get(void)

   Get the current interpreter.

   Issue a fatal error if there no current Python thread state or no current
   interpreter. It cannot return NULL.

   The caller must hold the GIL.

   .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1210)

Unknown directive type "c:function".

```
.. c:function:: int64_t PyInterpreterState_GetID(PyInterpreterState *interp)

   Return the interpreter's unique ID.  If there was any error in doing
   so then ``-1`` is returned and an error is set.

   The caller must hold the GIL.

   .. versionadded:: 3.7
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1220)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyInterpreterState_GetDict(PyInterpreterState *interp)

   Return a dictionary in which interpreter-specific data may be stored.
   If this function returns ``NULL`` then no exception has been raised and
   the caller should assume no interpreter-specific dict is available.

   This is not a replacement for :c:func:`PyModule_GetState()`, which
   extensions should use to store interpreter-specific state information.

   .. versionadded:: 3.8
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1231)

Unknown directive type "c:type".

```
.. c:type:: PyObject* (*_PyFrameEvalFunction)(PyThreadState *tstate, PyFrameObject *frame, int throwflag)

   Type of a frame evaluation function.

   The *throwflag* parameter is used by the ``throw()`` method of generators:
   if non-zero, handle the current exception.

   .. versionchanged:: 3.9
      The function now takes a *tstate* parameter.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1241)

Unknown directive type "c:function".

```
.. c:function:: _PyFrameEvalFunction _PyInterpreterState_GetEvalFrameFunc(PyInterpreterState *interp)

   Get the frame evaluation function.

   See the :pep:`523` "Adding a frame evaluation API to CPython".

   .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1249)

Unknown directive type "c:function".

```
.. c:function:: void _PyInterpreterState_SetEvalFrameFunc(PyInterpreterState *interp, _PyFrameEvalFunction e

   Set the frame evaluation function.

   See the :pep:`523` "Adding a frame evaluation API to CPython".

   .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1258)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyThreadState_GetDict()

   Return a dictionary in which extensions can store thread-specific state
   information.  Each extension should use a unique key to use to store state in
   the dictionary.  It is okay to call this function when no current thread state
```

```
   is available. If this function returns ``NULL``, no exception has been raised and
   the caller should assume no current thread state is available.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1267)

Unknown directive type "c:function".

```
.. c:function:: int PyThreadState_SetAsyncExc(unsigned long id, PyObject *exc)

   Asynchronously raise an exception in a thread. The *id* argument is the thread
   id of the target thread; *exc* is the exception object to be raised. This
   function does not steal any references to *exc*. To prevent naive misuse, you
   must write your own C extension to call this.  Must be called with the GIL held.
   Returns the number of thread states modified; this is normally one, but will be
   zero if the thread id isn't found.  If *exc* is :const:`NULL`, the pending
   exception (if any) for the thread is cleared. This raises no exceptions.

   .. versionchanged:: 3.7
      The type of the *id* parameter changed from :c:type:`long` to
      :c:type:`unsigned long`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1281)

Unknown directive type "c:function".

```
.. c:function:: void PyEval_AcquireThread(PyThreadState *tstate)

   Acquire the global interpreter lock and set the current thread state to
   *tstate*, which must not be ``NULL``.  The lock must have been created earlier.
   If this thread already has the lock, deadlock ensues.

   .. note::
      Calling this function from a thread when the runtime is finalizing
      will terminate the thread, even if the thread was not created by Python.
      You can use :c:func:`_Py_IsFinalizing` or :func:`sys.is_finalizing` to
      check if the interpreter is in process of being finalized before calling
      this function to avoid unwanted termination.

   .. versionchanged:: 3.8
      Updated to be consistent with :c:func:`PyEval_RestoreThread`,
      :c:func:`Py_END_ALLOW_THREADS`, and :c:func:`PyGILState_Ensure`,
      and terminate the current thread if called while the interpreter is finalizing.

   :c:func:`PyEval_RestoreThread` is a higher-level function which is always
   available (even when threads have not been initialized).
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1303)

Unknown directive type "c:function".

```
.. c:function:: void PyEval_ReleaseThread(PyThreadState *tstate)

   Reset the current thread state to ``NULL`` and release the global interpreter
   lock.  The lock must have been created earlier and must be held by the current
   thread.  The *tstate* argument, which must not be ``NULL``, is only used to check
   that it represents the current thread state --- if it isn't, a fatal error is
   reported.

   :c:func:`PyEval_SaveThread` is a higher-level function which is always
   available (even when threads have not been initialized).
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1315)

Unknown directive type "c:function".

```
.. c:function:: void PyEval_AcquireLock()

   Acquire the global interpreter lock.  The lock must have been created earlier.
   If this thread already has the lock, a deadlock ensues.

   .. deprecated:: 3.2
      This function does not update the current thread state.  Please use
      :c:func:`PyEval_RestoreThread` or :c:func:`PyEval_AcquireThread`
      instead.

   .. note::
      Calling this function from a thread when the runtime is finalizing
      will terminate the thread, even if the thread was not created by Python.
      You can use :c:func:`_Py_IsFinalizing` or :func:`sys.is_finalizing` to
      check if the interpreter is in process of being finalized before calling
      this function to avoid unwanted termination.
```

```
   .. versionchanged:: 3.8
      Updated to be consistent with :c:func:`PyEval_RestoreThread`,
      :c:func:`Py_END_ALLOW_THREADS`, and :c:func:`PyGILState_Ensure`,
      and terminate the current thread if called while the interpreter is finalizing.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1338)

Unknown directive type "c:function".

```
.. c:function:: void PyEval_ReleaseLock()

   Release the global interpreter lock.  The lock must have been created earlier.

   .. deprecated:: 3.2
      This function does not update the current thread state.  Please use
      :c:func:`PyEval_SaveThread` or :c:func:`PyEval_ReleaseThread`
      instead.
```

Sub-interpreter support

While in most uses, you will only embed a single Python interpreter, there are cases where you need to create several independent interpreters in the same process and perhaps even in the same thread. Sub-interpreters allow you to do that.

The "main" interpreter is the first one created when the runtime initializes. It is usually the only Python interpreter in a process. Unlike sub-interpreters, the main interpreter has unique process-global responsibilities like signal handling. It is also responsible for execution during runtime initialization and is usually the active interpreter during runtime finalization. The :c:func:`PyInterpreterState_Main` function returns a pointer to its state.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1358); *backlink*

Unknown interpreted text role "c:func".

You can switch between sub-interpreters using the :c:func:`PyThreadState_Swap` function. You can create and destroy them using the following functions:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1365); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1369)

Unknown directive type "c:function".

```
.. c:function:: PyThreadState* Py_NewInterpreter()

   .. index::
      module: builtins
      module: __main__
      module: sys
      single: stdout (in module sys)
      single: stderr (in module sys)
      single: stdin (in module sys)

   Create a new sub-interpreter.  This is an (almost) totally separate environment
   for the execution of Python code.  In particular, the new interpreter has
   separate, independent versions of all imported modules, including the
   fundamental modules :mod:`builtins`, :mod:`__main__` and :mod:`sys`.  The
   table of loaded modules (``sys.modules``) and the module search path
   (``sys.path``) are also separate.  The new environment has no ``sys.argv``
   variable.  It has new standard I/O stream file objects ``sys.stdin``,
   ``sys.stdout`` and ``sys.stderr`` (however these refer to the same underlying
   file descriptors).

   The return value points to the first thread state created in the new
   sub-interpreter.  This thread state is made in the current thread state.
   Note that no actual thread is created; see the discussion of thread states
   below.  If creation of the new interpreter is unsuccessful, ``NULL`` is
   returned; no exception is set since the exception state is stored in the
   current thread state and there may not be a current thread state.  (Like all
   other Python/C API functions, the global interpreter lock must be held before
   calling this function and is still held when it returns; however, unlike most
   other Python/C API functions, there needn't be a current thread state on
   entry.)

   .. index::
      single: Py_FinalizeEx()
      single: Py_Initialize()

   Extension modules are shared between (sub-)interpreters as follows:

   *  For modules using multi-phase initialization,
      e.g. :c:func:`PyModule_FromDefAndSpec`, a separate module object is
```

```
      created and initialized for each interpreter.
      Only C-level static and global variables are shared between these
      module objects.

   *  For modules using single-phase initialization,
      e.g. :c:func:`PyModule_Create`, the first time a particular extension
      is imported, it is initialized normally, and a (shallow) copy of its
      module's dictionary is squirreled away.
      When the same extension is imported by another (sub-)interpreter, a new
      module is initialized and filled with the contents of this copy; the
      extension's ``init`` function is not called.
      Objects in the module's dictionary thus end up shared across
      (sub-)interpreters, which might cause unwanted behavior (see
      `Bugs and caveats`_ below).

      Note that this is different from what happens when an extension is
      imported after the interpreter has been completely re-initialized by
      calling :c:func:`Py_FinalizeEx` and :c:func:`Py_Initialize`; in that
      case, the extension's ``initmodule`` function *is* called again.
      As with multi-phase initialization, this means that only C-level static
      and global variables are shared between these modules.

   .. index:: single: close() (in module os)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1433)

Unknown directive type "c:function".

```
.. c:function:: void Py_EndInterpreter(PyThreadState *tstate)

   .. index:: single: Py_FinalizeEx()

   Destroy the (sub-)interpreter represented by the given thread state. The given
   thread state must be the current thread state.  See the discussion of thread
   states below.  When the call returns, the current thread state is ``NULL``.  All
   thread states associated with this interpreter are destroyed.  (The global
   interpreter lock must be held before calling this function and is still held
   when it returns.)  :c:func:`Py_FinalizeEx` will destroy all sub-interpreters that
   haven't been explicitly destroyed at that point.
```

Bugs and caveats

Because sub-interpreters (and the main interpreter) are part of the same process, the insulation between them isn't perfect --- for example, using low-level file operations like :func:`os.close` they can (accidentally or maliciously) affect each other's open files. Because of the way extensions are shared between (sub-)interpreters, some extensions may not work properly; this is especially likely when using single-phase initialization or (static) global variables. It is possible to insert objects created in one sub-interpreter into a namespace of another (sub-)interpreter; this should be avoided if possible.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1449); *backlink*

Unknown interpreted text role "func".

Special care should be taken to avoid sharing user-defined functions, methods, instances or classes between sub-interpreters, since import operations executed by such objects may affect the wrong (sub-)interpreter's dictionary of loaded modules. It is equally important to avoid sharing objects from which the above are reachable.

Also note that combining this functionality with :c:func:`PyGILState_*` APIs is delicate, because these APIs assume a bijection between Python thread states and OS-level threads, an assumption broken by the presence of sub-interpreters. It is highly recommended that you don't switch sub-interpreters between a pair of matching :c:func:`PyGILState_Ensure` and :c:func:`PyGILState_Release` calls. Furthermore, extensions (such as :mod:`ctypes`) using these APIs to allow calling of Python code from non-Python created threads will probably be broken when using sub-interpreters.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1465); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1465); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1465); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1465); *backlink*

Unknown interpreted text role "mod".

Asynchronous Notifications

A mechanism is provided to make asynchronous notifications to the main interpreter thread. These notifications take the form of a function pointer and a void pointer argument.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1483)

Unknown directive type "c:function".

```
.. c:function:: int Py_AddPendingCall(int (*func)(void *), void *arg)

   .. index:: single: Py_AddPendingCall()

   Schedule a function to be called from the main interpreter thread.  On
   success, ``0`` is returned and *func* is queued for being called in the
   main thread.  On failure, ``-1`` is returned without setting any exception.

   When successfully queued, *func* will be *eventually* called from the
   main interpreter thread with the argument *arg*.  It will be called
   asynchronously with respect to normally running Python code, but with
   both these conditions met:

   * on a :term:`bytecode` boundary;
   * with the main thread holding the :term:`global interpreter lock`
     (*func* can therefore use the full C API).

   *func* must return ``0`` on success, or ``-1`` on failure with an exception
   set.  *func* won't be interrupted to perform another asynchronous
   notification recursively, but it can still be interrupted to switch
   threads if the global interpreter lock is released.

   This function doesn't need a current thread state to run, and it doesn't
   need the global interpreter lock.

   To call this function in a subinterpreter, the caller must hold the GIL.
   Otherwise, the function *func* can be scheduled to be called from the wrong
   interpreter.

   .. warning::
      This is a low-level function, only useful for very special cases.
      There is no guarantee that *func* will be called as quick as
      possible.  If the main thread is busy executing a system call,
      *func* won't be called before the system call returns.  This
      function is generally **not** suitable for calling Python code from
      arbitrary C threads.  Instead, use the :ref:`PyGILState API<gilstate>`.

   .. versionchanged:: 3.9
      If this function is called in a subinterpreter, the function *func* is
      now scheduled to be called from the subinterpreter, rather than being
      called from the main interpreter. Each subinterpreter now has its own
      list of scheduled calls.

   .. versionadded:: 3.1
```

Profiling and Tracing

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1533)

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Fred L. Drake, Jr. <fdrake@acm.org>
```

The Python interpreter provides some low-level support for attaching profiling and execution tracing facilities. These are used for profiling, debugging, and coverage analysis tools.

This C interface allows the profiling or tracing code to avoid the overhead of calling through Python-level callable objects, making a direct C function call instead. The essential attributes of the facility have not changed; the interface allows trace functions to be installed per-thread, and the basic events reported to the trace function are the same as had been reported to the Python-level trace functions in previous versions.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1548)

Unknown directive type "c:type".

```
.. c:type:: int (*Py_tracefunc)(PyObject *obj, PyFrameObject *frame, int what, PyObject *arg)

   The type of the trace function registered using :c:func:`PyEval_SetProfile` and
   :c:func:`PyEval_SetTrace`. The first parameter is the object passed to the
   registration function as *obj*, *frame* is the frame object to which the event
   pertains, *what* is one of the constants :const:`PyTrace_CALL`,
   :const:`PyTrace_EXCEPTION`, :const:`PyTrace_LINE`, :const:`PyTrace_RETURN`,
   :const:`PyTrace_C_CALL`, :const:`PyTrace_C_EXCEPTION`, :const:`PyTrace_C_RETURN`,
```

```
   or :const:`PyTrace_OPCODE`, and *arg* depends on the value of *what*:

   +------------------------------+--------------------------------------+
   | Value of *what*              | Meaning of *arg*                     |
   +==============================+======================================+
   | :const:`PyTrace_CALL`        | Always :c:data:`Py_None`.            |
   +------------------------------+--------------------------------------+
   | :const:`PyTrace_EXCEPTION`   | Exception information as returned by |
   |                              | :func:`sys.exc_info`.                |
   +------------------------------+--------------------------------------+
   | :const:`PyTrace_LINE`        | Always :c:data:`Py_None`.            |
   +------------------------------+--------------------------------------+
   | :const:`PyTrace_RETURN`      | Value being returned to the caller,  |
   |                              | or ``NULL`` if caused by an exception. |
   +------------------------------+--------------------------------------+
   | :const:`PyTrace_C_CALL`      | Function object being called.        |
   +------------------------------+--------------------------------------+
   | :const:`PyTrace_C_EXCEPTION` | Function object being called.        |
   +------------------------------+--------------------------------------+
   | :const:`PyTrace_C_RETURN`    | Function object being called.        |
   +------------------------------+--------------------------------------+
   | :const:`PyTrace_OPCODE`      | Always :c:data:`Py_None`.            |
   +------------------------------+--------------------------------------+
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1580)

Unknown directive type "c:var".

```
.. c:var:: int PyTrace_CALL

   The value of the *what* parameter to a :c:type:`Py_tracefunc` function when a new
   call to a function or method is being reported, or a new entry into a generator.
   Note that the creation of the iterator for a generator function is not reported
   as there is no control transfer to the Python bytecode in the corresponding
   frame.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1589)

Unknown directive type "c:var".

```
.. c:var:: int PyTrace_EXCEPTION

   The value of the *what* parameter to a :c:type:`Py_tracefunc` function when an
   exception has been raised.  The callback function is called with this value for
   *what* when after any bytecode is processed after which the exception becomes
   set within the frame being executed.  The effect of this is that as exception
   propagation causes the Python stack to unwind, the callback is called upon
   return to each frame as the exception propagates.  Only trace functions receives
   these events; they are not needed by the profiler.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1600)

Unknown directive type "c:var".

```
.. c:var:: int PyTrace_LINE

   The value passed as the *what* parameter to a :c:type:`Py_tracefunc` function
   (but not a profiling function) when a line-number event is being reported.
   It may be disabled for a frame by setting :attr:`f_trace_lines` to *0* on that frame.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1607)

Unknown directive type "c:var".

```
.. c:var:: int PyTrace_RETURN

   The value for the *what* parameter to :c:type:`Py_tracefunc` functions when a
   call is about to return.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1613)

Unknown directive type "c:var".

```
.. c:var:: int PyTrace_C_CALL

   The value for the *what* parameter to :c:type:`Py_tracefunc` functions when a C
   function is about to be called.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1619)

Unknown directive type "c:var".

```
.. c:var:: int PyTrace_C_EXCEPTION

   The value for the *what* parameter to :c:type:`Py_tracefunc` functions when a C
   function has raised an exception.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1625)

Unknown directive type "c:var".

```
.. c:var:: int PyTrace_C_RETURN

   The value for the *what* parameter to :c:type:`Py_tracefunc` functions when a C
   function has returned.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1631)

Unknown directive type "c:var".

```
.. c:var:: int PyTrace_OPCODE

   The value for the *what* parameter to :c:type:`Py_tracefunc` functions (but not
   profiling functions) when a new opcode is about to be executed.  This event is
   not emitted by default: it must be explicitly requested by setting
   :attr:`f_trace_opcodes` to *1* on the frame.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1639)

Unknown directive type "c:function".

```
.. c:function:: void PyEval_SetProfile(Py_tracefunc func, PyObject *obj)

   Set the profiler function to *func*.  The *obj* parameter is passed to the
   function as its first parameter, and may be any Python object, or ``NULL``.  If
   the profile function needs to maintain state, using a different value for *obj*
   for each thread provides a convenient and thread-safe place to store it.  The
   profile function is called for all monitored events except :const:`PyTrace_LINE`
   :const:`PyTrace_OPCODE` and :const:`PyTrace_EXCEPTION`.

   See also the :func:`sys.setprofile` function.

   The caller must hold the :term:`GIL`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1653)

Unknown directive type "c:function".

```
.. c:function:: void PyEval_SetTrace(Py_tracefunc func, PyObject *obj)

   Set the tracing function to *func*.  This is similar to
   :c:func:`PyEval_SetProfile`, except the tracing function does receive line-number
   events and per-opcode events, but does not receive any event related to C function
   objects being called.  Any trace function registered using :c:func:`PyEval_SetTrace`
   will not receive :const:`PyTrace_C_CALL`, :const:`PyTrace_C_EXCEPTION` or
   :const:`PyTrace_C_RETURN` as a value for the *what* parameter.

   See also the :func:`sys.settrace` function.

   The caller must hold the :term:`GIL`.
```

Advanced Debugger Support

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1672)

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Fred L. Drake, Jr. <fdrake@acm.org>
```

These functions are only intended to be used by advanced debugging tools.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1678)

Unknown directive type "c:function".

```
.. c:function:: PyInterpreterState* PyInterpreterState_Head()

   Return the interpreter state object at the head of the list of all such objects.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1683)

Unknown directive type "c:function".

```
.. c:function:: PyInterpreterState* PyInterpreterState_Main()

   Return the main interpreter state object.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1688)

Unknown directive type "c:function".

```
.. c:function:: PyInterpreterState* PyInterpreterState_Next(PyInterpreterState *interp)

   Return the next interpreter state object after *interp* from the list of all
   such objects.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1694)

Unknown directive type "c:function".

```
.. c:function:: PyThreadState * PyInterpreterState_ThreadHead(PyInterpreterState *interp)

   Return the pointer to the first :c:type:`PyThreadState` object in the list of
   threads associated with the interpreter *interp*.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1700)

Unknown directive type "c:function".

```
.. c:function:: PyThreadState* PyThreadState_Next(PyThreadState *tstate)

   Return the next thread state object after *tstate* from the list of all such
   objects belonging to the same :c:type:`PyInterpreterState` object.
```

Thread Local Storage Support

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1711)

Unknown directive type "sectionauthor".

```
.. sectionauthor:: Masayuki Yamamoto <ma3yuki.8mamo10@gmail.com>
```

The Python interpreter provides low-level support for thread-local storage (TLS) which wraps the underlying native TLS implementation to support the Python-level thread local storage API (:class:`threading.local`). The CPython C level APIs are similar to those offered by pthreads and Windows: use a thread key and functions to associate a :c:type:`void*` value per thread.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1713); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1713); *backlink*

Unknown interpreted text role "c:type".

The GIL does *not* need to be held when calling these functions; they supply their own locking.

Note that :file:`Python.h` does not include the declaration of the TLS APIs, you need to include :file:`pythread.h` to use thread-local storage.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1723); *backlink*

Unknown interpreted text role "file".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1723); *backlink*

Unknown interpreted text role "file".

Note

None of these API functions handle memory management on behalf of the :c:type:`void*` values. You need to allocate and deallocate them yourself. If the :c:type:`void*` values happen to be :c:type:`PyObject*`, these functions don't do refcount operations on them either.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1727); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1727); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1727); *backlink*

Unknown interpreted text role "c:type".

Thread Specific Storage (TSS) API

TSS API is introduced to supersede the use of the existing TLS API within the CPython interpreter. This API uses a new type :c:type:`Py_tss_t` instead of :c:type:`int` to represent thread keys.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1737); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1737); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1741)

Unknown directive type "versionadded".

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

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1743)

Unknown directive type "seealso".

```
.. seealso:: "A New C-API for Thread-Local Storage in CPython" (:pep:`539`)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1746)

Unknown directive type "c:type".

```
.. c:type:: Py_tss_t

   This data structure represents the state of a thread key, the definition of
   which may depend on the underlying TLS implementation, and it has an
   internal field representing the key's initialization state.  There are no
   public members in this structure.
```

```
When :ref:`Py_LIMITED_API <stable>` is not defined, static allocation of
this type by :c:macro:`Py_tss_NEEDS_INIT` is allowed.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1757)

Unknown directive type "c:macro".

```
.. c:macro:: Py_tss_NEEDS_INIT

   This macro expands to the initializer for :c:type:`Py_tss_t` variables.
   Note that this macro won't be defined with :ref:`Py_LIMITED_API <stable>`.
```

Dynamic Allocation

Dynamic allocation of the :c:type:`Py_tss_t`, required in extension modules built with :ref:`Py_LIMITED_API <stable>`, where static allocation of this type is not possible due to its implementation being opaque at build time.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1766); *backlink*

Unknown interpreted text role "c:type".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1766); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1771)

Unknown directive type "c:function".

```
.. c:function:: Py_tss_t* PyThread_tss_alloc()

   Return a value which is the same state as a value initialized with
   :c:macro:`Py_tss_NEEDS_INIT`, or ``NULL`` in the case of dynamic allocation
   failure.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1778)

Unknown directive type "c:function".

```
.. c:function:: void PyThread_tss_free(Py_tss_t *key)

   Free the given *key* allocated by :c:func:`PyThread_tss_alloc`, after
   first calling :c:func:`PyThread_tss_delete` to ensure any associated
   thread locals have been unassigned. This is a no-op if the *key*
   argument is `NULL`.

   .. note::
      A freed key becomes a dangling pointer. You should reset the key to
      `NULL`.
```

Methods

The parameter *key* of these functions must not be `NULL`. Moreover, the behaviors of :c:func:`PyThread_tss_set` and :c:func:`PyThread_tss_get` are undefined if the given :c:type:`Py_tss_t` has not been initialized by :c:func:`PyThread_tss_create`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1793); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1793); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1793); *backlink*

Unknown interpreted text role "c:type".

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

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1793); *backlink*

Unknown interpreted text role "c:func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1799)

Unknown directive type "c:function".

```
.. c:function:: int PyThread_tss_is_created(Py_tss_t *key)

   Return a non-zero value if the given :c:type:`Py_tss_t` has been initialized
   by :c:func:`PyThread_tss_create`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1805)

Unknown directive type "c:function".

```
.. c:function:: int PyThread_tss_create(Py_tss_t *key)

   Return a zero value on successful initialization of a TSS key.  The behavior
   is undefined if the value pointed to by the *key* argument is not
   initialized by :c:macro:`Py_tss_NEEDS_INIT`.  This function can be called
   repeatedly on the same key -- calling it on an already initialized key is a
   no-op and immediately returns success.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1814)

Unknown directive type "c:function".

```
.. c:function:: void PyThread_tss_delete(Py_tss_t *key)

   Destroy a TSS key to forget the values associated with the key across all
   threads, and change the key's initialization state to uninitialized.  A
   destroyed key is able to be initialized again by
   :c:func:`PyThread_tss_create`. This function can be called repeatedly on
   the same key -- calling it on an already destroyed key is a no-op.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1823)

Unknown directive type "c:function".

```
.. c:function:: int PyThread_tss_set(Py_tss_t *key, void *value)

   Return a zero value to indicate successfully associating a :c:type:`void*`
   value with a TSS key in the current thread.  Each thread has a distinct
   mapping of the key to a :c:type:`void*` value.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1830)

Unknown directive type "c:function".

```
.. c:function:: void* PyThread_tss_get(Py_tss_t *key)

   Return the :c:type:`void*` value associated with a TSS key in the current
   thread.  This returns ``NULL`` if no value is associated with the key in the
   current thread.
```

Thread Local Storage (TLS) API

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1842)

Unknown directive type "deprecated".

```
.. deprecated:: 3.7
   This API is superseded by
   :ref:`Thread Specific Storage (TSS) API <thread-specific-storage-api>`.
```

Note

This version of the API does not support platforms where the native TLS key is defined in a way that cannot be safely cast to `int`. On such platforms, :c:func:`PyThread_create_key` will return immediately with a failure status, and the other TLS functions will all be no-ops on such platforms.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1847); *backlink*

Unknown interpreted text role "c:func".

Due to the compatibility problem noted above, this version of the API should not be used in new code.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1855)

Unknown directive type "c:function".

```
.. c:function:: int PyThread_create_key()
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1856)

Unknown directive type "c:function".

```
.. c:function:: void PyThread_delete_key(int key)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1857)

Unknown directive type "c:function".

```
.. c:function:: int PyThread_set_key_value(int key, void *value)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1858)

Unknown directive type "c:function".

```
.. c:function:: void* PyThread_get_key_value(int key)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1859)

Unknown directive type "c:function".

```
.. c:function:: void PyThread_delete_key_value(int key)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)init.rst, line 1860)

Unknown directive type "c:function".

```
.. c:function:: void PyThread_ReInitTLS()
```