The Very High Level Layer

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

Unknown directive type "highlight".

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

The functions in this chapter will let you execute Python source code given in a file or a buffer, but they will not let you interact in a more detailed way with the interpreter.

Several of these functions accept a start symbol from the grammar as a parameter. The available start symbols are :const:`Py_eval_input`, :const:`Py_file_input`, and :const:`Py_single_input`. These are described following the functions which accept them as parameters.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst`, line 14); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst`, line 14); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst`, line 14); *backlink*

Unknown interpreted text role "const".

Note also that several of these functions take :c:type:`FILE*` parameters. One particular issue which needs to be handled carefully is that the :c:type:`FILE` structure for different C libraries can be different and incompatible. Under Windows (at least), it is possible for dynamically linked extensions to actually use different libraries, so care should be taken that :c:type:`FILE*` parameters are only passed to these functions if it is certain that they were created by the same library that the Python runtime is using.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst`, line 19); *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)veryhigh.rst`, line 19); *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)veryhigh.rst`, line 19); *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)veryhigh.rst`, line 28)

Unknown directive type "c:function".

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

   The main program for the standard interpreter.  This is made available for
   programs which embed Python.  The *argc* and *argv* parameters should be
   prepared exactly as those which are passed to a C program's :c:func:`main`
   function (converted to wchar_t according to the user's locale).  It is
   important to note that the argument list may be modified (but the contents of
   the strings pointed to by the argument list are not). The return value will
   be ``0`` if the interpreter exits normally (i.e., without an exception),
   ``1`` if the interpreter exits due to an exception, or ``2`` if the parameter
   list does not represent a valid Python command line.

   Note that if an otherwise unhandled :exc:`SystemExit` is raised, this
   function will not return ``1``, but exit the process, as long as
   ``Py_InspectFlag`` is not set.
```

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

Unknown directive type "c:function".

```
.. c:function:: int Py_BytesMain(int argc, char **argv)

   Similar to :c:func:`Py_Main` but *argv* is an array of bytes strings.

   .. versionadded:: 3.8
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst`, line 52)

Unknown directive type "c:function".

```
.. c:function:: int PyRun_AnyFile(FILE *fp, const char *filename)

   This is a simplified interface to :c:func:`PyRun_AnyFileExFlags` below, leaving
   *closeit* set to ``0`` and *flags* set to ``NULL``.
```

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

Unknown directive type "c:function".

```
.. c:function:: int PyRun_AnyFileFlags(FILE *fp, const char *filename, PyCompilerFlags *flags)

   This is a simplified interface to :c:func:`PyRun_AnyFileExFlags` below, leaving
   the *closeit* argument set to ``0``.
```

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

Unknown directive type "c:function".

```
.. c:function:: int PyRun_AnyFileEx(FILE *fp, const char *filename, int closeit)

   This is a simplified interface to :c:func:`PyRun_AnyFileExFlags` below, leaving
   the *flags* argument set to ``NULL``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 70)

Unknown directive type "c:function".

```
.. c:function:: int PyRun_AnyFileExFlags(FILE *fp, const char *filename, int closeit, PyCompilerFlags *flags)

   If *fp* refers to a file associated with an interactive device (console or
   terminal input or Unix pseudo-terminal), return the value of
   :c:func:`PyRun_InteractiveLoop`, otherwise return the result of
   :c:func:`PyRun_SimpleFile`.  *filename* is decoded from the filesystem
   encoding (:func:`sys.getfilesystemencoding`).  If *filename* is ``NULL``, this
   function uses ``"???"`` as the filename.
   If *closeit* is true, the file is closed before
   ``PyRun_SimpleFileExFlags()`` returns.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 82)

Unknown directive type "c:function".

```
.. c:function:: int PyRun_SimpleString(const char *command)

   This is a simplified interface to :c:func:`PyRun_SimpleStringFlags` below,
   leaving the :c:type:`PyCompilerFlags`\* argument set to ``NULL``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 88)

Unknown directive type "c:function".

```
.. c:function:: int PyRun_SimpleStringFlags(const char *command, PyCompilerFlags *flags)

   Executes the Python source code from *command* in the :mod:`__main__` module
   according to the *flags* argument. If :mod:`__main__` does not already exist, it
   is created.  Returns ``0`` on success or ``-1`` if an exception was raised.  If
   there was an error, there is no way to get the exception information. For the
   meaning of *flags*, see below.

   Note that if an otherwise unhandled :exc:`SystemExit` is raised, this
   function will not return ``-1``, but exit the process, as long as
   ``Py_InspectFlag`` is not set.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 101)

Unknown directive type "c:function".

```
.. c:function:: int PyRun_SimpleFile(FILE *fp, const char *filename)

   This is a simplified interface to :c:func:`PyRun_SimpleFileExFlags` below,
   leaving *closeit* set to ``0`` and *flags* set to ``NULL``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 107)

Unknown directive type "c:function".

```
.. c:function:: int PyRun_SimpleFileEx(FILE *fp, const char *filename, int closeit)

   This is a simplified interface to :c:func:`PyRun_SimpleFileExFlags` below,
   leaving *flags* set to ``NULL``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 113)

Unknown directive type "c:function".

```
.. c:function:: int PyRun_SimpleFileExFlags(FILE *fp, const char *filename, int closeit, PyCompilerFlags *flags)

   Similar to :c:func:`PyRun_SimpleStringFlags`, but the Python source code is read
   from *fp* instead of an in-memory string. *filename* should be the name of
   the file, it is decoded from :term:`filesystem encoding and error handler`.
   If *closeit* is true, the file is closed before
   ``PyRun_SimpleFileExFlags()`` returns.

   .. note::
      On Windows, *fp* should be opened as binary mode (e.g. ``fopen(filename, "rb")``).
      Otherwise, Python may not handle script file with LF line ending correctly.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 126)

Unknown directive type "c:function".

```
.. c:function:: int PyRun_InteractiveOne(FILE *fp, const char *filename)

   This is a simplified interface to :c:func:`PyRun_InteractiveOneFlags` below,
   leaving *flags* set to ``NULL``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 132)

Unknown directive type "c:function".

```
.. c:function:: int PyRun_InteractiveOneFlags(FILE *fp, const char *filename, PyCompilerFlags *flags)
```

```
   Read and execute a single statement from a file associated with an
   interactive device according to the *flags* argument.  The user will be
   prompted using ``sys.ps1`` and ``sys.ps2``.  *filename* is decoded from the
   :term:`filesystem encoding and error handler`.

   Returns ``0`` when the input was
   executed successfully, ``-1`` if there was an exception, or an error code
   from the :file:`errcode.h` include file distributed as part of Python if
   there was a parse error.  (Note that :file:`errcode.h` is not included by
   :file:`Python.h`, so must be included specifically if needed.)
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 146)

Unknown directive type "c:function".

```
.. c:function:: int PyRun_InteractiveLoop(FILE *fp, const char *filename)

   This is a simplified interface to :c:func:`PyRun_InteractiveLoopFlags` below,
   leaving *flags* set to ``NULL``.
```

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

Unknown directive type "c:function".

```
.. c:function:: int PyRun_InteractiveLoopFlags(FILE *fp, const char *filename, PyCompilerFlags *flags)

   Read and execute statements from a file associated with an interactive device
   until EOF is reached.  The user will be prompted using ``sys.ps1`` and
   ``sys.ps2``.  *filename* is decoded from the :term:`filesystem encoding and
   error handler`.  Returns ``0`` at EOF or a negative number upon failure.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 160)

Unknown directive type "c:var".

```
.. c:var:: int (*PyOS_InputHook)(void)

   Can be set to point to a function with the prototype
   ``int func(void)``.  The function will be called when Python's
   interpreter prompt is about to become idle and wait for user input
   from the terminal.  The return value is ignored.  Overriding this
   hook can be used to integrate the interpreter's prompt with other
   event loops, as done in the :file:`Modules/_tkinter.c` in the
   Python source code.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 171)

Unknown directive type "c:var".

```
.. c:var:: char* (*PyOS_ReadlineFunctionPointer)(FILE *, FILE *, const char *)

   Can be set to point to a function with the prototype
   ``char *func(FILE *stdin, FILE *stdout, char *prompt)``,
   overriding the default function used to read a single line of input
   at the interpreter's prompt.  The function is expected to output
   the string *prompt* if it's not ``NULL``, and then read a line of
   input from the provided standard input file, returning the
   resulting string.  For example, The :mod:`readline` module sets
   this hook to provide line-editing and tab-completion features.

   The result must be a string allocated by :c:func:`PyMem_RawMalloc` or
   :c:func:`PyMem_RawRealloc`, or ``NULL`` if an error occurred.

   .. versionchanged:: 3.4
      The result must be allocated by :c:func:`PyMem_RawMalloc` or
      :c:func:`PyMem_RawRealloc`, instead of being allocated by
      :c:func:`PyMem_Malloc` or :c:func:`PyMem_Realloc`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 190)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyRun_String(const char *str, int start, PyObject *globals, PyObject *locals)

   This is a simplified interface to :c:func:`PyRun_StringFlags` below, leaving
   *flags* set to ``NULL``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 196)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyRun_StringFlags(const char *str, int start, PyObject *globals, PyObject *locals, PyCompilerFlags *flags

   Execute Python source code from *str* in the context specified by the
   objects *globals* and *locals* with the compiler flags specified by
   *flags*.  *globals* must be a dictionary; *locals* can be any object
   that implements the mapping protocol.  The parameter *start* specifies
   the start token that should be used to parse the source code.

   Returns the result of executing the code as a Python object, or ``NULL`` if an
   exception was raised.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 208)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyRun_File(FILE *fp, const char *filename, int start, PyObject *globals, PyObject *locals)

   This is a simplified interface to :c:func:`PyRun_FileExFlags` below, leaving
   *closeit* set to ``0`` and *flags* set to ``NULL``.
```

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

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyRun_FileEx(FILE *fp, const char *filename, int start, PyObject *globals, PyObject *locals, int closeit

   This is a simplified interface to :c:func:`PyRun_FileExFlags` below, leaving
   *flags* set to ``NULL``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 220)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyRun_FileFlags(FILE *fp, const char *filename, int start, PyObject *globals, PyObject *locals, PyCompil

   This is a simplified interface to :c:func:`PyRun_FileExFlags` below, leaving
   *closeit* set to ``0``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 226)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyRun_FileExFlags(FILE *fp, const char *filename, int start, PyObject *globals, PyObject *locals, int cl

   Similar to :c:func:`PyRun_StringFlags`, but the Python source code is read from
   *fp* instead of an in-memory string. *filename* should be the name of the file,
   it is decoded from the :term:`filesystem encoding and error handler`.
   If *closeit* is true, the file is closed before :c:func:`PyRun_FileExFlags`
   returns.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 235)

Unknown directive type "c:function".

```
.. c:function:: PyObject* Py_CompileString(const char *str, const char *filename, int start)

   This is a simplified interface to :c:func:`Py_CompileStringFlags` below, leaving
   *flags* set to ``NULL``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 241)

Unknown directive type "c:function".

```
.. c:function:: PyObject* Py_CompileStringFlags(const char *str, const char *filename, int start, PyCompilerFlags *flags)

   This is a simplified interface to :c:func:`Py_CompileStringExFlags` below, with
   *optimize* set to ``-1``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 247)

Unknown directive type "c:function".

```
.. c:function:: PyObject* Py_CompileStringObject(const char *str, PyObject *filename, int start, PyCompilerFlags *flags, int optim

   Parse and compile the Python source code in *str*, returning the resulting code
   object.  The start token is given by *start*; this can be used to constrain the
   code which can be compiled and should be :const:`Py_eval_input`,
   :const:`Py_file_input`, or :const:`Py_single_input`.  The filename specified by
   *filename* is used to construct the code object and may appear in tracebacks or
   :exc:`SyntaxError` exception messages.  This returns ``NULL`` if the code
   cannot be parsed or compiled.

   The integer *optimize* specifies the optimization level of the compiler; a
   value of ``-1`` selects the optimization level of the interpreter as given by
   :option:`-O` options.  Explicit levels are ``0`` (no optimization;
   ``__debug__`` is true), ``1`` (asserts are removed, ``__debug__`` is false)
   or ``2`` (docstrings are removed too).

   .. versionadded:: 3.4
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 266)

Unknown directive type "c:function".

```
.. c:function:: PyObject* Py_CompileStringExFlags(const char *str, const char *filename, int start, PyCompilerFlags *flags, int op

   Like :c:func:`Py_CompileStringObject`, but *filename* is a byte string
   decoded from the :term:`filesystem encoding and error handler`.

   .. versionadded:: 3.2
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 273)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyEval_EvalCode(PyObject *co, PyObject *globals, PyObject *locals)

   This is a simplified interface to :c:func:`PyEval_EvalCodeEx`, with just
   the code object, and global and local variables.  The other arguments are
   set to ``NULL``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 280)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyEval_EvalCodeEx(PyObject *co, PyObject *globals, PyObject *locals, PyObject *const *args, int argcount,

   Evaluate a precompiled code object, given a particular environment for its
```

```
   evaluation.  This environment consists of a dictionary of global variables,
   a mapping object of local variables, arrays of arguments, keywords and
   defaults, a dictionary of default values for :ref:`keyword-only
   <keyword-only_parameter>` arguments and a closure tuple of cells.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 289)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyEval_EvalFrame(PyFrameObject *f)

   Evaluate an execution frame.  This is a simplified interface to
   :c:func:`PyEval_EvalFrameEx`, for backward compatibility.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 295)

Unknown directive type "c:function".

```
.. c:function:: PyObject* PyEval_EvalFrameEx(PyFrameObject *f, int throwflag)

   This is the main, unvarnished function of Python interpretation.  The code
   object associated with the execution frame *f* is executed, interpreting
   bytecode and executing calls as needed.  The additional *throwflag*
   parameter can mostly be ignored - if true, then it causes an exception
   to immediately be thrown; this is used for the :meth:`~generator.throw`
   methods of generator objects.

   .. versionchanged:: 3.4
      This function now includes a debug assertion to help ensure that it
      does not silently discard an active exception.
```

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

Unknown directive type "c:function".

```
.. c:function:: int PyEval_MergeCompilerFlags(PyCompilerFlags *cf)

   This function changes the flags of the current evaluation frame, and returns
   true on success, false on failure.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 315)

Unknown directive type "c:var".

```
.. c:var:: int Py_eval_input

   .. index:: single: Py_CompileString()

   The start symbol from the Python grammar for isolated expressions; for use with
   :c:func:`Py_CompileString`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 323)

Unknown directive type "c:var".

```
.. c:var:: int Py_file_input

   .. index:: single: Py_CompileString()

   The start symbol from the Python grammar for sequences of statements as read
   from a file or other source; for use with :c:func:`Py_CompileString`.  This is
   the symbol to use when compiling arbitrarily long Python source code.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 332)

Unknown directive type "c:var".

```
.. c:var:: int Py_single_input

   .. index:: single: Py_CompileString()

   The start symbol from the Python grammar for a single statement; for use with
   :c:func:`Py_CompileString`. This is the symbol used for the interactive
   interpreter loop.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main) (Doc) (c-api)veryhigh.rst, line 341)

Unknown directive type "c:type".

```
.. c:type:: struct PyCompilerFlags

   This is the structure used to hold compiler flags.  In cases where code is only
   being compiled, it is passed as ``int flags``, and in cases where code is being
   executed, it is passed as ``PyCompilerFlags *flags``.  In this case, ``from
   __future__ import`` can modify *flags*.

   Whenever ``PyCompilerFlags *flags`` is ``NULL``, :attr:`cf_flags` is treated as
   equal to ``0``, and any modification due to ``from __future__ import`` is
   discarded.

   .. c:member:: int cf_flags

      Compiler flags.

   .. c:member:: int cf_feature_version

      *cf_feature_version* is the minor Python version. It should be
      initialized to ``PY_MINOR_VERSION``.

      The field is ignored by default, it is used if and only if
      ``PyCF_ONLY_AST`` flag is set in *cf_flags*.
```

```
   .. versionchanged:: 3.8
      Added *cf_feature_version* field.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\c-api\(cpython-main)(Doc)(c-api)veryhigh.rst, line 368)

Unknown directive type "c:var".

```
   .. c:var:: int CO_FUTURE_DIVISION

      This bit can be set in *flags* to cause division operator ``/`` to be
      interpreted as "true division" according to :pep:`238`.
```