:mod:`signal` --- Set handlers for asynchronous events

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 1); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 4)

Unknown directive type "module".

```
.. module:: signal
   :synopsis: Set handlers for asynchronous events.
```

This module provides mechanisms to use signal handlers in Python.

General rules

The :func:`signal.signal` function allows defining custom handlers to be executed when a signal is received. A small number of default handlers are installed: :const:`SIGPIPE` is ignored (so write errors on pipes and sockets can be reported as ordinary Python exceptions) and :const:`SIGINT` is translated into a :exc:`KeyboardInterrupt` exception if the parent process has not changed it.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 15); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 15); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 15); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 15); *backlink*

Unknown interpreted text role "exc".

A handler for a particular signal, once set, remains installed until it is explicitly reset (Python emulates the BSD style interface regardless of the underlying implementation), with the exception of the handler for :const:`SIGCHLD`, which follows the underlying implementation.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 22); *backlink*

Unknown interpreted text role "const".

Execution of Python signal handlers

A Python signal handler does not get executed inside the low-level (C) signal handler. Instead, the low-level signal handler sets a flag which tells the :term:`virtual machine` to execute the corresponding Python signal handler at a later point(for example at the next :term:`bytecode` instruction). This has consequences:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 31); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 31); *backlink*

Unknown interpreted text role "term".

- It makes little sense to catch synchronous errors like :const:`SIGFPE` or :const:`SIGSEGV` that are caused by an invalid operation in C code. Python will return from the signal handler to the C code, which is likely to raise the same signal again, causing Python to apparently hang. From Python 3.3 onwards, you can use the :mod:`faulthandler` module to report on synchronous errors.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 37); *backlink*

  Unknown interpreted text role "const".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 37); *backlink*

  Unknown interpreted text role "const".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 37); *backlink*

  Unknown interpreted text role "mod".

- A long-running calculation implemented purely in C (such as regular expression matching on a large body of text) may run uninterrupted for an arbitrary amount of time, regardless of any signals received. The Python signal handlers will be called when the calculation finishes.
- If the handler raises an exception, it will be raised "out of thin air" in the main thread. See the :ref:`note below <handlers-and-exceptions>` for a discussion.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 49); *backlink*

  Unknown interpreted text role "ref".

Signals and threads

Python signal handlers are always executed in the main Python thread of the main interpreter, even if the signal was received in another thread. This means that signals can't be used as a means of inter-thread communication. You can use the synchronization primitives from the :mod:`threading` module instead.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 59); *backlink*

Unknown interpreted text role "mod".

Besides, only the main thread of the main interpreter is allowed to set a new signal handler.

Module contents

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 70)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.5
   signal (SIG*), handler (:const:`SIG_DFL`, :const:`SIG_IGN`) and sigmask
   (:const:`SIG_BLOCK`, :const:`SIG_UNBLOCK`, :const:`SIG_SETMASK`)
   related constants listed below were turned into
   :class:`enums <enum.IntEnum>` (:class:`Signals`, :class:`Handlers` and :class:`Sigmasks` respective
   :func:`getsignal`, :func:`pthread_sigmask`, :func:`sigpending` and
   :func:`sigwait` functions return human-readable
   :class:`enums <enum.IntEnum>` as :class:`Signals` objects.
```

The signal module defines three enums:

:class:`enum.IntEnum` collection of SIG* constants and the CTRL_* constants.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 84); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 86)

Unknown directive type "versionadded".

```
.. versionadded:: 3.5
```

:class:`enum.IntEnum` collection the constants :const:`SIG_DFL` and :const:`SIG_IGN`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 90); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 90); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 90); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 92)

Unknown directive type "versionadded".

```
.. versionadded:: 3.5
```

:class:`enum.IntEnum` collection the constants :const:`SIG_BLOCK`, :const:`SIG_UNBLOCK` and :const:`SIG_SETMASK`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 96); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 96); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 96); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 96); *backlink*

Unknown interpreted text role "const".

Availability: Unix. See the man page :manpage:`sigprocmask(3)` and :manpage:`pthread_sigmask(3)` for further information.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 98); *backlink*

Unknown interpreted text role "manpage".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 98); *backlink*

Unknown interpreted text role "manpage".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 101)

Unknown directive type "versionadded".

```
.. versionadded:: 3.5
```

The variables defined in the :mod:`signal` module are:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 104); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 107)

Unknown directive type "data".

```
.. data:: SIG_DFL

   This is one of two standard signal handling options; it will simply perform
   the default function for the signal.  For example, on most systems the
   default action for :const:`SIGQUIT` is to dump core and exit, while the
   default action for :const:`SIGCHLD` is to simply ignore it.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 115)

Unknown directive type "data".

```
.. data:: SIG_IGN

   This is another standard signal handler, which will simply ignore the given
   signal.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 121)

Unknown directive type "data".

```
.. data:: SIGABRT

   Abort signal from :manpage:`abort(3)`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 125)

Unknown directive type "data".

```
.. data:: SIGALRM

   Timer signal from :manpage:`alarm(2)`.

   .. availability:: Unix.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 131)

Unknown directive type "data".

```
.. data:: SIGBREAK

   Interrupt from keyboard (CTRL + BREAK).

   .. availability:: Windows.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 137)

Unknown directive type "data".

```
.. data:: SIGBUS

   Bus error (bad memory access).
```

```
   .. availability:: Unix.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 143)

Unknown directive type "data".

```
.. data:: SIGCHLD

   Child process stopped or terminated.

   .. availability:: Unix.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 149)

Unknown directive type "data".

```
.. data:: SIGCLD

   Alias to :data:`SIGCHLD`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 153)

Unknown directive type "data".

```
.. data:: SIGCONT

   Continue the process if it is currently stopped

   .. availability:: Unix.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 159)

Unknown directive type "data".

```
.. data:: SIGFPE

   Floating-point exception. For example, division by zero.

   .. seealso::
      :exc:`ZeroDivisionError` is raised when the second argument of a division
      or modulo operation is zero.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 167)

Unknown directive type "data".

```
.. data:: SIGHUP

   Hangup detected on controlling terminal or death of controlling process.

   .. availability:: Unix.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 173)

Unknown directive type "data".

```
.. data:: SIGILL

   Illegal instruction.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 177)

Unknown directive type "data".

```
.. data:: SIGINT
```

```
   Interrupt from keyboard (CTRL + C).

   Default action is to raise :exc:`KeyboardInterrupt`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 183)

Unknown directive type "data".

```
.. data:: SIGKILL

   Kill signal.

   It cannot be caught, blocked, or ignored.

   .. availability:: Unix.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 191)

Unknown directive type "data".

```
.. data:: SIGPIPE

   Broken pipe: write to pipe with no readers.

   Default action is to ignore the signal.

   .. availability:: Unix.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 199)

Unknown directive type "data".

```
.. data:: SIGSEGV

   Segmentation fault: invalid memory reference.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 203)

Unknown directive type "data".

```
.. data:: SIGSTKFLT

    Stack fault on coprocessor. The Linux kernel does not raise this signal: it
    can only be raised in user space.

   .. availability:: Linux, on architectures where the signal is available. See
      the man page :manpage:`signal(7)` for further information.

   .. versionadded:: 3.11
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 213)

Unknown directive type "data".

```
.. data:: SIGTERM

   Termination signal.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 217)

Unknown directive type "data".

```
.. data:: SIGUSR1

   User-defined signal 1.

   .. availability:: Unix.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 223)

Unknown directive type "data".

```
.. data:: SIGUSR2

   User-defined signal 2.

   .. availability:: Unix.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 229)

Unknown directive type "data".

```
.. data:: SIGWINCH

   Window resize signal.

   .. availability:: Unix.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 235)

Unknown directive type "data".

```
.. data:: SIG*

   All the signal numbers are defined symbolically.  For example, the hangup signal
   is defined as :const:`signal.SIGHUP`; the variable names are identical to the
   names used in C programs, as found in ``<signal.h>``.  The Unix man page for
   ':c:func:`signal`' lists the existing signals (on some systems this is
   :manpage:`signal(2)`, on others the list is in :manpage:`signal(7)`). Note that
   not all systems define the same set of signal names; only those names defined by
   the system are defined by this module.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 246)

Unknown directive type "data".

```
.. data:: CTRL_C_EVENT

   The signal corresponding to the :kbd:`Ctrl+C` keystroke event. This signal can
   only be used with :func:`os.kill`.

   .. availability:: Windows.

   .. versionadded:: 3.2
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 256)

Unknown directive type "data".

```
.. data:: CTRL_BREAK_EVENT

   The signal corresponding to the :kbd:`Ctrl+Break` keystroke event. This signal can
   only be used with :func:`os.kill`.

   .. availability:: Windows.

   .. versionadded:: 3.2
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 266)

Unknown directive type "data".

```
.. data:: NSIG

   One more than the number of the highest signal number.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 271)

Unknown directive type "data".

```
.. data:: ITIMER_REAL

   Decrements interval timer in real time, and delivers :const:`SIGALRM` upon
   expiration.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 277)

Unknown directive type "data".

```
.. data:: ITIMER_VIRTUAL

   Decrements interval timer only when the process is executing, and delivers
   SIGVTALRM upon expiration.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 283)

Unknown directive type "data".

```
.. data:: ITIMER_PROF

   Decrements interval timer both when the process executes and when the
   system is executing on behalf of the process. Coupled with ITIMER_VIRTUAL,
   this timer is usually used to profile the time spent by the application
   in user and kernel space. SIGPROF is delivered upon expiration.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 291)

Unknown directive type "data".

```
.. data:: SIG_BLOCK

   A possible value for the *how* parameter to :func:`pthread_sigmask`
   indicating that signals are to be blocked.

   .. versionadded:: 3.3
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 298)

Unknown directive type "data".

```
.. data:: SIG_UNBLOCK

   A possible value for the *how* parameter to :func:`pthread_sigmask`
   indicating that signals are to be unblocked.

   .. versionadded:: 3.3
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 305)

Unknown directive type "data".

```
.. data:: SIG_SETMASK

   A possible value for the *how* parameter to :func:`pthread_sigmask`
   indicating that the signal mask is to be replaced.

   .. versionadded:: 3.3
```

The :mod:`signal` module defines one exception:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 313); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 315)

Unknown directive type "exception".

```
.. exception:: ItimerError

   Raised to signal an error from the underlying :func:`setitimer` or
   :func:`getitimer` implementation. Expect this error if an invalid
   interval timer or a negative time is passed to :func:`setitimer`.
   This error is a subtype of :exc:`OSError`.

   .. versionadded:: 3.3
      This error used to be a subtype of :exc:`IOError`, which is now an
      alias of :exc:`OSError`.
```

The :mod:`signal` module defines the following functions:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 327); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 330)

Unknown directive type "function".

```
.. function:: alarm(time)

   If *time* is non-zero, this function requests that a :const:`SIGALRM` signal be
   sent to the process in *time* seconds. Any previously scheduled alarm is
   canceled (only one alarm can be scheduled at any time).  The returned value is
   then the number of seconds before any previously set alarm was to have been
   delivered. If *time* is zero, no alarm is scheduled, and any scheduled alarm is
   canceled.  If the return value is zero, no alarm is currently scheduled.

   .. availability:: Unix.  See the man page :manpage:`alarm(2)` for further
      information.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 343)

Unknown directive type "function".

```
.. function:: getsignal(signalnum)

   Return the current signal handler for the signal *signalnum*. The returned value
   may be a callable Python object, or one of the special values
   :const:`signal.SIG_IGN`, :const:`signal.SIG_DFL` or :const:`None`.  Here,
   :const:`signal.SIG_IGN` means that the signal was previously ignored,
   :const:`signal.SIG_DFL` means that the default way of handling the signal was
   previously in use, and ``None`` means that the previous signal handler was not
   installed from Python.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 354)

Unknown directive type "function".

```
.. function:: strsignal(signalnum)

   Return the system description of the signal *signalnum*, such as
   "Interrupt", "Segmentation fault", etc. Returns :const:`None` if the signal
   is not recognized.

   .. versionadded:: 3.8
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 363)

Unknown directive type "function".

```
.. function:: valid_signals()

   Return the set of valid signal numbers on this platform.  This can be
   less than ``range(1, NSIG)`` if some signals are reserved by the system
   for internal use.

   .. versionadded:: 3.8
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 372)

Unknown directive type "function".

```
.. function:: pause()

   Cause the process to sleep until a signal is received; the appropriate handler
   will then be called.  Returns nothing.

   .. availability:: Unix.  See the man page :manpage:`signal(2)` for further
      information.

   See also :func:`sigwait`, :func:`sigwaitinfo`, :func:`sigtimedwait` and
   :func:`sigpending`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 384)

Unknown directive type "function".

```
.. function:: raise_signal(signum)

   Sends a signal to the calling process. Returns nothing.

   .. versionadded:: 3.8
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 391)

Unknown directive type "function".

```
.. function:: pidfd_send_signal(pidfd, sig, siginfo=None, flags=0)

   Send signal *sig* to the process referred to by file descriptor *pidfd*.
   Python does not currently support the *siginfo* parameter; it must be
   ``None``.  The *flags* argument is provided for future extensions; no flag
   values are currently defined.

   See the :manpage:`pidfd_send_signal(2)` man page for more information.

   .. availability:: Linux 5.1+
   .. versionadded:: 3.9
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 404)

Unknown directive type "function".

```
.. function:: pthread_kill(thread_id, signalnum)

   Send the signal *signalnum* to the thread *thread_id*, another thread in the
   same process as the caller.  The target thread can be executing any code
   (Python or not).  However, if the target thread is executing the Python
   interpreter, the Python signal handlers will be :ref:`executed by the main
   thread of the main interpreter <signals-and-threads>`.  Therefore, the only point of sending a
   signal to a particular Python thread would be to force a running system call
   to fail with :exc:`InterruptedError`.

   Use :func:`threading.get_ident()` or the :attr:`~threading.Thread.ident`
   attribute of :class:`threading.Thread` objects to get a suitable value
   for *thread_id*.
```

```
   If *signalnum* is 0, then no signal is sent, but error checking is still
   performed; this can be used to check if the target thread is still running.

   .. audit-event:: signal.pthread_kill thread_id,signalnum signal.pthread_kill

   .. availability:: Unix.  See the man page :manpage:`pthread_kill(3)` for further
      information.

   See also :func:`os.kill`.

   .. versionadded:: 3.3
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 431)

Unknown directive type "function".

```
.. function:: pthread_sigmask(how, mask)

   Fetch and/or change the signal mask of the calling thread.  The signal mask
   is the set of signals whose delivery is currently blocked for the caller.
   Return the old signal mask as a set of signals.

   The behavior of the call is dependent on the value of *how*, as follows.

   * :data:`SIG_BLOCK`: The set of blocked signals is the union of the current
     set and the *mask* argument.
   * :data:`SIG_UNBLOCK`: The signals in *mask* are removed from the current
     set of blocked signals.  It is permissible to attempt to unblock a
     signal which is not blocked.
   * :data:`SIG_SETMASK`: The set of blocked signals is set to the *mask*
     argument.

   *mask* is a set of signal numbers (e.g. {:const:`signal.SIGINT`,
   :const:`signal.SIGTERM`}). Use :func:`~signal.valid_signals` for a full
   mask including all signals.

   For example, ``signal.pthread_sigmask(signal.SIG_BLOCK, [])`` reads the
   signal mask of the calling thread.

   :data:`SIGKILL` and :data:`SIGSTOP` cannot be blocked.

   .. availability:: Unix.  See the man page :manpage:`sigprocmask(2)` and
      :manpage:`pthread_sigmask(3)` for further information.

   See also :func:`pause`, :func:`sigpending` and :func:`sigwait`.

   .. versionadded:: 3.3
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 464)

Unknown directive type "function".

```
.. function:: setitimer(which, seconds, interval=0.0)

   Sets given interval timer (one of :const:`signal.ITIMER_REAL`,
   :const:`signal.ITIMER_VIRTUAL` or :const:`signal.ITIMER_PROF`) specified
   by *which* to fire after *seconds* (float is accepted, different from
   :func:`alarm`) and after that every *interval* seconds (if *interval*
   is non-zero). The interval timer specified by *which* can be cleared by
   setting *seconds* to zero.

   When an interval timer fires, a signal is sent to the process.
   The signal sent is dependent on the timer being used;
   :const:`signal.ITIMER_REAL` will deliver :const:`SIGALRM`,
   :const:`signal.ITIMER_VIRTUAL` sends :const:`SIGVTALRM`,
   and :const:`signal.ITIMER_PROF` will deliver :const:`SIGPROF`.

   The old values are returned as a tuple: (delay, interval).

   Attempting to pass an invalid interval timer will cause an
   :exc:`ItimerError`.

   .. availability:: Unix.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 487)

Unknown directive type "function".

```
.. function:: getitimer(which)

   Returns current value of a given interval timer specified by *which*.

   .. availability:: Unix.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 494)

Unknown directive type "function".

```
.. function:: set_wakeup_fd(fd, *, warn_on_full_buffer=True)

   Set the wakeup file descriptor to *fd*.  When a signal is received, the
   signal number is written as a single byte into the fd.  This can be used by
   a library to wakeup a poll or select call, allowing the signal to be fully
   processed.

   The old wakeup fd is returned (or -1 if file descriptor wakeup was not
   enabled).  If *fd* is -1, file descriptor wakeup is disabled.
   If not -1, *fd* must be non-blocking.  It is up to the library to remove
   any bytes from *fd* before calling poll or select again.

   When threads are enabled, this function can only be called
   from :ref:`the main thread of the main interpreter <signals-and-threads>`;
   attempting to call it from other threads will cause a :exc:`ValueError`
   exception to be raised.

   There are two common ways to use this function. In both approaches,
   you use the fd to wake up when a signal arrives, but then they
   differ in how they determine *which* signal or signals have
   arrived.

   In the first approach, we read the data out of the fd's buffer, and
   the byte values give you the signal numbers. This is simple, but in
   rare cases it can run into a problem: generally the fd will have a
   limited amount of buffer space, and if too many signals arrive too
   quickly, then the buffer may become full, and some signals may be
   lost. If you use this approach, then you should set
   ``warn_on_full_buffer=True``, which will at least cause a warning
   to be printed to stderr when signals are lost.

   In the second approach, we use the wakeup fd *only* for wakeups,
   and ignore the actual byte values. In this case, all we care about
   is whether the fd's buffer is empty or non-empty; a full buffer
   doesn't indicate a problem at all. If you use this approach, then
   you should set ``warn_on_full_buffer=False``, so that your users
   are not confused by spurious warning messages.

   .. versionchanged:: 3.5
      On Windows, the function now also supports socket handles.

   .. versionchanged:: 3.7
      Added ``warn_on_full_buffer`` parameter.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 538)

Unknown directive type "function".

```
.. function:: siginterrupt(signalnum, flag)

   Change system call restart behaviour: if *flag* is :const:`False`, system
   calls will be restarted when interrupted by signal *signalnum*, otherwise
   system calls will be interrupted.  Returns nothing.

   .. availability:: Unix.  See the man page :manpage:`siginterrupt(3)`
      for further information.

   Note that installing a signal handler with :func:`signal` will reset the
   restart behaviour to interruptible by implicitly calling
   :c:func:`siginterrupt` with a true *flag* value for the given signal.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 552)

Unknown directive type "function".

```
.. function:: signal(signalnum, handler)

   Set the handler for signal *signalnum* to the function *handler*.  *handler* can
   be a callable Python object taking two arguments (see below), or one of the
   special values :const:`signal.SIG_IGN` or :const:`signal.SIG_DFL`.  The previous
   signal handler will be returned (see the description of :func:`getsignal`
   above).  (See the Unix man page :manpage:`signal(2)` for further information.)

   When threads are enabled, this function can only be called
   from :ref:`the main thread of the main interpreter <signals-and-threads>`;
   attempting to call it from other threads will cause a :exc:`ValueError`
   exception to be raised.

   The *handler* is called with two arguments: the signal number and the current
   stack frame (``None`` or a frame object; for a description of frame objects,
   see the :ref:`description in the type hierarchy <frame-objects>` or see the
   attribute descriptions in the :mod:`inspect` module).

   On Windows, :func:`signal` can only be called with :const:`SIGABRT`,
   :const:`SIGFPE`, :const:`SIGILL`, :const:`SIGINT`, :const:`SIGSEGV`,
   :const:`SIGTERM`, or :const:`SIGBREAK`.
   A :exc:`ValueError` will be raised in any other case.
   Note that not all systems define the same set of signal names; an
   :exc:`AttributeError` will be raised if a signal name is not defined as
   ``SIG*`` module level constant.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 579)

Unknown directive type "function".

```
.. function:: sigpending()

   Examine the set of signals that are pending for delivery to the calling
   thread (i.e., the signals which have been raised while blocked).  Return the
   set of the pending signals.

   .. availability:: Unix.  See the man page :manpage:`sigpending(2)` for further
      information.

   See also :func:`pause`, :func:`pthread_sigmask` and :func:`sigwait`.

   .. versionadded:: 3.3
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 593)

Unknown directive type "function".

```
.. function:: sigwait(sigset)

   Suspend execution of the calling thread until the delivery of one of the
   signals specified in the signal set *sigset*.  The function accepts the signal
   (removes it from the pending list of signals), and returns the signal number.

   .. availability:: Unix.  See the man page :manpage:`sigwait(3)` for further
      information.

   See also :func:`pause`, :func:`pthread_sigmask`, :func:`sigpending`,
   :func:`sigwaitinfo` and :func:`sigtimedwait`.

   .. versionadded:: 3.3
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 608)

Unknown directive type "function".

```
.. function:: sigwaitinfo(sigset)

   Suspend execution of the calling thread until the delivery of one of the
   signals specified in the signal set *sigset*.  The function accepts the
   signal and removes it from the pending list of signals. If one of the
   signals in *sigset* is already pending for the calling thread, the function
   will return immediately with information about that signal. The signal
   handler is not called for the delivered signal. The function raises an
   :exc:`InterruptedError` if it is interrupted by a signal that is not in
   *sigset*.
```

```
The return value is an object representing the data contained in the
:c:type:`siginfo_t` structure, namely: :attr:`si_signo`, :attr:`si_code`,
:attr:`si_errno`, :attr:`si_pid`, :attr:`si_uid`, :attr:`si_status`,
:attr:`si_band`.

.. availability:: Unix.  See the man page :manpage:`sigwaitinfo(2)` for further
   information.

See also :func:`pause`, :func:`sigwait` and :func:`sigtimedwait`.

.. versionadded:: 3.3

.. versionchanged:: 3.5
   The function is now retried if interrupted by a signal not in *sigset*
   and the signal handler does not raise an exception (see :pep:`475` for
   the rationale).
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 637)

Unknown directive type "function".

```
.. function:: sigtimedwait(sigset, timeout)

   Like :func:`sigwaitinfo`, but takes an additional *timeout* argument
   specifying a timeout. If *timeout* is specified as :const:`0`, a poll is
   performed. Returns :const:`None` if a timeout occurs.

   .. availability:: Unix.  See the man page :manpage:`sigtimedwait(2)` for further
      information.

   See also :func:`pause`, :func:`sigwait` and :func:`sigwaitinfo`.

   .. versionadded:: 3.3

   .. versionchanged:: 3.5
      The function is now retried with the recomputed *timeout* if interrupted
      by a signal not in *sigset* and the signal handler does not raise an
      exception (see :pep:`475` for the rationale).
```

Examples

Here is a minimal example program. It uses the :func:`alarm` function to limit the time spent waiting to open a file; this is useful if the file is for a serial device that may not be turned on, which would normally cause the :func:`os.open` to hang indefinitely. The solution is to set a 5-second alarm before opening the file; if the operation takes too long, the alarm signal will be sent, and the handler raises an exception.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 661); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 661); *backlink*

Unknown interpreted text role "func".

```
import signal, os

def handler(signum, frame):
    signame = signal.Signals(signum).name
    print(f'Signal handler called with signal {signame} ({signum})')
    raise OSError("Couldn't open device!")

# Set the signal handler and a 5-second alarm
signal.signal(signal.SIGALRM, handler)
signal.alarm(5)

# This open() may hang indefinitely
fd = os.open('/dev/ttyS0', os.O_RDWR)

signal.alarm(0)          # Disable the alarm
```

Note on SIGPIPE

Piping output of your program to tools like :manpage:`head(1)` will cause a :const:`SIGPIPE` signal to be sent to your process when the receiver of its standard output closes early. This results in an exception like `BrokenPipeError: [Errno 32] Broken pipe`.

To handle this case, wrap your entry point to catch this exception as follows:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 687); *backlink*

Unknown interpreted text role "manpage".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 687); *backlink*

Unknown interpreted text role "const".

```
import os
import sys

def main():
    try:
        # simulate large output (your code replaces this loop)
        for x in range(10000):
            print("y")
        # flush output here to force SIGPIPE to be triggered
        # while inside this try block.
        sys.stdout.flush()
    except BrokenPipeError:
        # Python flushes standard streams on exit; redirect remaining output
        # to devnull to avoid another BrokenPipeError at shutdown
        devnull = os.open(os.devnull, os.O_WRONLY)
        os.dup2(devnull, sys.stdout.fileno())
        sys.exit(1)  # Python exits with error code 1 on EPIPE

if __name__ == '__main__':
    main()
```

Do not set :const:`SIGPIPE`'s disposition to :const:`SIG_DFL` in order to avoid :exc:`BrokenPipeError`. Doing that would cause your program to exit unexpectedly also whenever any socket connection is interrupted while your program is still writing to it.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 714); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 714); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 714); *backlink*

Unknown interpreted text role "exc".

Note on Signal Handlers and Exceptions

If a signal handler raises an exception, the exception will be propagated to the main thread and may be raised after any :term:`bytecode` instruction. Most notably, a :exc:`KeyboardInterrupt` may appear at any point during execution. Most Python code, including the standard library, cannot be made robust against this, and so a :exc:`KeyboardInterrupt` (or any other exception resulting from a signal handler) may on rare occasions put the program in an unexpected state.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 724); *backlink*

Unknown interpreted text role "term".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 724); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 724); *backlink*

Unknown interpreted text role "exc".

To illustrate this issue, consider the following code:

```
class SpamContext:
    def __init__(self):
        self.lock = threading.Lock()

    def __enter__(self):
        # If KeyboardInterrupt occurs here, everything is fine
        self.lock.acquire()
        # If KeyboardInterrupt occcurs here, __exit__ will not be called
        ...
        # KeyboardInterrupt could occur just before the function returns

    def __exit__(self, exc_type, exc_val, exc_tb):
        ...
        self.lock.release()
```

For many programs, especially those that merely want to exit on :exc:`KeyboardInterrupt`, this is not a problem, but applications that are complex or require high reliability should avoid raising exceptions from signal handlers. They should also avoid catching :exc:`KeyboardInterrupt` as a means of gracefully shutting down. Instead, they should install their own :const:`SIGINT` handler. Below is an example of an HTTP server that avoids :exc:`KeyboardInterrupt`:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 748); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 748); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 748); *backlink*

Unknown interpreted text role "const".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]signal.rst, line 748); *backlink*

Unknown interpreted text role "exc".

```
import signal
import socket
from selectors import DefaultSelector, EVENT_READ
from http.server import HTTPServer, SimpleHTTPRequestHandler

interrupt_read, interrupt_write = socket.socketpair()

def handler(signum, frame):
    print('Signal handler called with signal', signum)
    interrupt_write.send(b'\0')
signal.signal(signal.SIGINT, handler)

def serve_forever(httpd):
    sel = DefaultSelector()
    sel.register(interrupt_read, EVENT_READ)
    sel.register(httpd, EVENT_READ)

    while True:
        for key, _ in sel.select():
            if key.fileobj == interrupt_read:
                interrupt_read.recv(1)
                return
            if key.fileobj == httpd:
                httpd.handle_request()

print("Serving on port 8000")
httpd = HTTPServer(('', 8000), SimpleHTTPRequestHandler)
serve_forever(httpd)
print("Shutdown...")
```