:mod:`concurrent.futures` --- Launching parallel tasks

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

Unknown directive type "module".

```
.. module:: concurrent.futures
   :synopsis: Execute computations concurrently using threads or processes.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 7)

Unknown directive type "versionadded".

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

Source code: :source:`Lib/concurrent/futures/thread.py` and :source:`Lib/concurrent/futures/process.py`

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 9); *backlink*

Unknown interpreted text role "source".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 9); *backlink*

Unknown interpreted text role "source".

The :mod:`concurrent.futures` module provides a high-level interface for asynchronously executing callables.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 14); *backlink*

Unknown interpreted text role "mod".

The asynchronous execution can be performed with threads, using :class:`ThreadPoolExecutor`, or separate processes, using :class:`ProcessPoolExecutor`. Both implement the same interface, which is defined by the abstract :class:`Executor` class.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 17); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 17); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 17); *backlink*

Unknown interpreted text role "class".

Executor Objects

An abstract class that provides methods to execute calls asynchronously. It should not be used directly, but through its concrete

subclasses.

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

Unknown directive type "method".

```
.. method:: submit(fn, /, *args, **kwargs)

   Schedules the callable, *fn*, to be executed as ``fn(*args, **kwargs)``
   and returns a :class:`Future` object representing the execution of the
   callable. ::

      with ThreadPoolExecutor(max_workers=1) as executor:
          future = executor.submit(pow, 323, 1235)
          print(future.result())
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 41)

Unknown directive type "method".

```
.. method:: map(func, *iterables, timeout=None, chunksize=1)

   Similar to :func:`map(func, *iterables) <map>` except:

   * the *iterables* are collected immediately rather than lazily;

   * *func* is executed asynchronously and several calls to
     *func* may be made concurrently.

   The returned iterator raises a :exc:`TimeoutError`
   if :meth:`~iterator.__next__` is called and the result isn't available
   after *timeout* seconds from the original call to :meth:`Executor.map`.
   *timeout* can be an int or a float.  If *timeout* is not specified or
   ``None``, there is no limit to the wait time.

   If a *func* call raises an exception, then that exception will be
   raised when its value is retrieved from the iterator.

   When using :class:`ProcessPoolExecutor`, this method chops *iterables*
   into a number of chunks which it submits to the pool as separate
   tasks.  The (approximate) size of these chunks can be specified by
   setting *chunksize* to a positive integer.  For very long iterables,
   using a large value for *chunksize* can significantly improve
   performance compared to the default size of 1.  With
   :class:`ThreadPoolExecutor`, *chunksize* has no effect.

   .. versionchanged:: 3.5
      Added the *chunksize* argument.
```

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

Unknown directive type "method".

```
.. method:: shutdown(wait=True, *, cancel_futures=False)

   Signal the executor that it should free any resources that it is using
   when the currently pending futures are done executing.  Calls to
   :meth:`Executor.submit` and :meth:`Executor.map` made after shutdown will
   raise :exc:`RuntimeError`.

   If *wait* is ``True`` then this method will not return until all the
   pending futures are done executing and the resources associated with the
   executor have been freed.  If *wait* is ``False`` then this method will
   return immediately and the resources associated with the executor will be
   freed when all pending futures are done executing.  Regardless of the
   value of *wait*, the entire Python program will not exit until all
   pending futures are done executing.

   If *cancel_futures* is ``True``, this method will cancel all pending
   futures that the executor has not started running. Any futures that
```

```
        are completed or running won't be cancelled, regardless of the value
        of *cancel_futures*.

        If both *cancel_futures* and *wait* are ``True``, all futures that the
        executor has started running will be completed prior to this method
        returning. The remaining futures are cancelled.

        You can avoid having to call this method explicitly if you use the
        :keyword:`with` statement, which will shutdown the :class:`Executor`
        (waiting as if :meth:`Executor.shutdown` were called with *wait* set to
        ``True``)::

           import shutil
           with ThreadPoolExecutor(max_workers=4) as e:
               e.submit(shutil.copy, 'src1.txt', 'dest1.txt')
               e.submit(shutil.copy, 'src2.txt', 'dest2.txt')
               e.submit(shutil.copy, 'src3.txt', 'dest3.txt')
               e.submit(shutil.copy, 'src4.txt', 'dest4.txt')

        .. versionchanged:: 3.9
           Added *cancel_futures*.
```

ThreadPoolExecutor

:class:`ThreadPoolExecutor` is an :class:`Executor` subclass that uses a pool of threads to execute calls asynchronously.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 113); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 113); *backlink*

Unknown interpreted text role "class".

Deadlocks can occur when the callable associated with a :class:`Future` waits on the results of another :class:`Future`. For example:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 116); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 116); *backlink*

Unknown interpreted text role "class".

```
import time
def wait_on_b():
    time.sleep(5)
    print(b.result())  # b will never complete because it is waiting on a.
    return 5

def wait_on_a():
    time.sleep(5)
    print(a.result())  # a will never complete because it is waiting on b.
    return 6


executor = ThreadPoolExecutor(max_workers=2)
a = executor.submit(wait_on_b)
b = executor.submit(wait_on_a)
```

And:

```
def wait_on_future():
    f = executor.submit(pow, 5, 2)
    # This will never complete because there is only one worker thread and
    # it is executing this function.
    print(f.result())

executor = ThreadPoolExecutor(max_workers=1)
```

```
executor.submit(wait_on_future)
```

An :class:`Executor` subclass that uses a pool of at most *max_workers* threads to execute calls asynchronously.

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

Unknown interpreted text role "class".

initializer is an optional callable that is called at the start of each worker thread; *initargs* is a tuple of arguments passed to the initializer. Should *initializer* raise an exception, all currently pending jobs will raise a :exc:`~concurrent.futures.thread.BrokenThreadPool`, as well as any attempt to submit more jobs to the pool.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 152); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 158)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.5
   If *max_workers* is ``None`` or
   not given, it will default to the number of processors on the machine,
   multiplied by ``5``, assuming that :class:`ThreadPoolExecutor` is often
   used to overlap I/O instead of CPU work and the number of workers
   should be higher than the number of workers
   for :class:`ProcessPoolExecutor`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 166)

Unknown directive type "versionadded".

```
.. versionadded:: 3.6
   The *thread_name_prefix* argument was added to allow users to
   control the :class:`threading.Thread` names for worker threads created by
   the pool for easier debugging.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 171)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.7
   Added the *initializer* and *initargs* arguments.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 174)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.8
   Default value of *max_workers* is changed to ``min(32, os.cpu_count() + 4)``.
   This default value preserves at least 5 workers for I/O bound tasks.
   It utilizes at most 32 CPU cores for CPU bound tasks which release the GIL.
   And it avoids using very large resources implicitly on many-core machines.

   ThreadPoolExecutor now reuses idle worker threads before starting
   *max_workers* worker threads too.
```

ThreadPoolExecutor Example

```
import concurrent.futures
import urllib.request

URLS = ['http://www.foxnews.com/',
```

```
            'http://www.cnn.com/',
            'http://europe.wsj.com/',
            'http://www.bbc.co.uk/',
            'http://some-made-up-domain.com/']

    # Retrieve a single page and report the URL and contents
    def load_url(url, timeout):
        with urllib.request.urlopen(url, timeout=timeout) as conn:
            return conn.read()

    # We can use a with statement to ensure threads are cleaned up promptly
    with concurrent.futures.ThreadPoolExecutor(max_workers=5) as executor:
        # Start the load operations and mark each future with its URL
        future_to_url = {executor.submit(load_url, url, 60): url for url in URLS}
        for future in concurrent.futures.as_completed(future_to_url):
            url = future_to_url[future]
            try:
                data = future.result()
            except Exception as exc:
                print('%r generated an exception: %s' % (url, exc))
            else:
                print('%r page is %d bytes' % (url, len(data)))
```

ProcessPoolExecutor

The :class:`ProcessPoolExecutor` class is an :class:`Executor` subclass that uses a pool of processes to execute calls asynchronously. :class:`ProcessPoolExecutor` uses the :mod:`multiprocessing` module, which allows it to side-step the :term:`Global Interpreter Lock <global interpreter lock>` but also means that only picklable objects can be executed and returned.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 221); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 221); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 221); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 221); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 221); *backlink*

Unknown interpreted text role "term".

The `__main__` module must be importable by worker subprocesses. This means that :class:`ProcessPoolExecutor` will not work in the interactive interpreter.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 228); *backlink*

Unknown interpreted text role "class".

Calling :class:`Executor` or :class:`Future` methods from a callable submitted to a :class:`ProcessPoolExecutor` will result in deadlock.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 231); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 231); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 231); *backlink*

Unknown interpreted text role "class".

An :class:`Executor` subclass that executes calls asynchronously using a pool of at most *max_workers* processes. If *max_workers* is `None` or not given, it will default to the number of processors on the machine. If *max_workers* is less than or equal to `0`, then a :exc:`ValueError` will be raised. On Windows, *max_workers* must be less than or equal to `61`. If it is not then :exc:`ValueError` will be raised. If *max_workers* is `None`, then the default chosen will be at most `61`, even if more processors are available. *mp_context* can be a multiprocessing context or None. It will be used to launch the workers. If *mp_context* is `None` or not given, the default multiprocessing context is used.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 236); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 236); *backlink*

Unknown interpreted text role "exc".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 236); *backlink*

Unknown interpreted text role "exc".

initializer is an optional callable that is called at the start of each worker process; *initargs* is a tuple of arguments passed to the initializer. Should *initializer* raise an exception, all currently pending jobs will raise a :exc:`~concurrent.futures.process.BrokenProcessPool`, as well as any attempt to submit more jobs to the pool.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 249); *backlink*

Unknown interpreted text role "exc".

max_tasks_per_child is an optional argument that specifies the maximum number of tasks a single process can execute before it will exit and be replaced with a fresh worker process. The default *max_tasks_per_child* is `None` which means worker processes will live as long as the pool.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 260)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.3
   When one of the worker processes terminates abruptly, a
   :exc:`BrokenProcessPool` error is now raised.  Previously, behaviour
   was undefined but operations on the executor or its futures would often
   freeze or deadlock.
```

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

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.7
   The *mp_context* argument was added to allow users to control the
   start_method for worker processes created by the pool.

   Added the *initializer* and *initargs* arguments.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 272)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.11
   The *max_tasks_per_child* argument was added to allow users to
   control the lifetime of workers in the pool.
```

ProcessPoolExecutor Example

```
import concurrent.futures
import math

PRIMES = [
    112272535095293,
    112582705942171,
    112272535095293,
    115280095190773,
    115797848077099,
    1099726899285419]

def is_prime(n):
    if n < 2:
        return False
    if n == 2:
        return True
    if n % 2 == 0:
        return False

    sqrt_n = int(math.floor(math.sqrt(n)))
    for i in range(3, sqrt_n + 1, 2):
        if n % i == 0:
            return False
    return True

def main():
    with concurrent.futures.ProcessPoolExecutor() as executor:
        for number, prime in zip(PRIMES, executor.map(is_prime, PRIMES)):
            print('%d is prime: %s' % (number, prime))

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

Future Objects

The :class:`Future` class encapsulates the asynchronous execution of a callable. :class:`Future` instances are created by :meth:`Executor.submit`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 320); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 320); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 320); *backlink*

Unknown interpreted text role "meth".

Encapsulates the asynchronous execution of a callable. :class:`Future` instances are created by :meth:`Executor.submit` and should not be created directly except for testing.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 325); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 325); *backlink*

Unknown interpreted text role "meth".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 329)

Unknown directive type "method".

```
.. method:: cancel()

   Attempt to cancel the call.  If the call is currently being executed or
   finished running and cannot be cancelled then the method will return
   ``False``, otherwise the call will be cancelled and the method will
   return ``True``.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 336)

Unknown directive type "method".

```
.. method:: cancelled()

   Return ``True`` if the call was successfully cancelled.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 340)

Unknown directive type "method".

```
.. method:: running()

   Return ``True`` if the call is currently being executed and cannot be
   cancelled.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 345)

Unknown directive type "method".

```
.. method:: done()

   Return ``True`` if the call was successfully cancelled or finished
   running.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 350)

Unknown directive type "method".

```
.. method:: result(timeout=None)

   Return the value returned by the call. If the call hasn't yet completed
   then this method will wait up to *timeout* seconds.  If the call hasn't
   completed in *timeout* seconds, then a
   :exc:`TimeoutError` will be raised. *timeout* can be
   an int or float.  If *timeout* is not specified or ``None``, there is no
   limit to the wait time.

   If the future is cancelled before completing then :exc:`.CancelledError`
   will be raised.

   If the call raised an exception, this method will raise the same exception.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 364)

Unknown directive type "method".

```
.. method:: exception(timeout=None)

   Return the exception raised by the call.  If the call hasn't yet
   completed then this method will wait up to *timeout* seconds.  If the
   call hasn't completed in *timeout* seconds, then a
   :exc:`TimeoutError` will be raised.  *timeout* can be
   an int or float.  If *timeout* is not specified or ``None``, there is no
   limit to the wait time.

   If the future is cancelled before completing then :exc:`.CancelledError`
   will be raised.

   If the call completed without raising, ``None`` is returned.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 378)

Unknown directive type "method".

```
.. method:: add_done_callback(fn)

   Attaches the callable *fn* to the future.  *fn* will be called, with the
   future as its only argument, when the future is cancelled or finishes
   running.

   Added callables are called in the order that they were added and are
   always called in a thread belonging to the process that added them.  If
   the callable raises an :exc:`Exception` subclass, it will be logged and
   ignored.  If the callable raises a :exc:`BaseException` subclass, the
   behavior is undefined.

   If the future has already completed or been cancelled, *fn* will be
   called immediately.
```

The following :class:`Future` methods are meant for use in unit tests and :class:`Executor` implementations.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 393); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 393); *backlink*

Unknown interpreted text role "class".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 396)

Unknown directive type "method".

```
.. method:: set_running_or_notify_cancel()

   This method should only be called by :class:`Executor` implementations
   before executing the work associated with the :class:`Future` and by unit
   tests.

   If the method returns ``False`` then the :class:`Future` was cancelled,
   i.e. :meth:`Future.cancel` was called and returned `True`.  Any threads
   waiting on the :class:`Future` completing (i.e. through
   :func:`as_completed` or :func:`wait`) will be woken up.

   If the method returns ``True`` then the :class:`Future` was not cancelled
   and has been put in the running state, i.e. calls to
   :meth:`Future.running` will return `True`.
```

```
   This method can only be called once and cannot be called after
   :meth:`Future.set_result` or :meth:`Future.set_exception` have been
   called.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 415)

Unknown directive type "method".

```
.. method:: set_result(result)

   Sets the result of the work associated with the :class:`Future` to
   *result*.

   This method should only be used by :class:`Executor` implementations and
   unit tests.

   .. versionchanged:: 3.8
      This method raises
      :exc:`concurrent.futures.InvalidStateError` if the :class:`Future` is
      already done.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 428)

Unknown directive type "method".

```
.. method:: set_exception(exception)

   Sets the result of the work associated with the :class:`Future` to the
   :class:`Exception` *exception*.

   This method should only be used by :class:`Executor` implementations and
   unit tests.

   .. versionchanged:: 3.8
      This method raises
      :exc:`concurrent.futures.InvalidStateError` if the :class:`Future` is
      already done.
```

Module Functions

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 444)

Unknown directive type "function".

```
.. function:: wait(fs, timeout=None, return_when=ALL_COMPLETED)

   Wait for the :class:`Future` instances (possibly created by different
   :class:`Executor` instances) given by *fs* to complete. Duplicate futures
   given to *fs* are removed and will be returned only once. Returns a named
   2-tuple of sets.  The first set, named ``done``, contains the futures that
   completed (finished or cancelled futures) before the wait completed.  The
   second set, named ``not_done``, contains the futures that did not complete
   (pending or running futures).

   *timeout* can be used to control the maximum number of seconds to wait before
   returning.  *timeout* can be an int or float.  If *timeout* is not specified
   or ``None``, there is no limit to the wait time.

   *return_when* indicates when this function should return.  It must be one of
   the following constants:

   .. tabularcolumns:: |l|L|

   +-----------------------------+----------------------------------------+
   | Constant                    | Description                            |
   +=============================+========================================+
   | :const:`FIRST_COMPLETED`    | The function will return when any      |
   |                             | future finishes or is cancelled.       |
```

```
+-----------------------------+----------------------------------------+
| :const:`FIRST_EXCEPTION`    | The function will return when any      |
|                             | future finishes by raising an          |
|                             | exception.  If no future raises an     |
|                             | exception then it is equivalent to     |
|                             | :const:`ALL_COMPLETED`.                |
+-----------------------------+----------------------------------------+
| :const:`ALL_COMPLETED`      | The function will return when all      |
|                             | futures finish or are cancelled.       |
+-----------------------------+----------------------------------------+
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 479)

Unknown directive type "function".

```
.. function:: as_completed(fs, timeout=None)

   Returns an iterator over the :class:`Future` instances (possibly created by
   different :class:`Executor` instances) given by *fs* that yields futures as
   they complete (finished or cancelled futures). Any futures given by *fs* that
   are duplicated will be returned once. Any futures that completed before
   :func:`as_completed` is called will be yielded first.  The returned iterator
   raises a :exc:`TimeoutError` if :meth:`~iterator.__next__`
   is called and the result isn't available after *timeout* seconds from the
   original call to :func:`as_completed`.  *timeout* can be an int or float. If
   *timeout* is not specified or ``None``, there is no limit to the wait time.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 492)

Unknown directive type "seealso".

```
.. seealso::

   :pep:`3148` -- futures - execute computations asynchronously
      The proposal which described this feature for inclusion in the Python
      standard library.
```

Exception classes

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 502)

Unknown directive type "currentmodule".

```
.. currentmodule:: concurrent.futures
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 504)

Unknown directive type "exception".

```
.. exception:: CancelledError

   Raised when a future is cancelled.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 508)

Unknown directive type "exception".

```
.. exception:: TimeoutError

   A deprecated alias of :exc:`TimeoutError`,
   raised when a future operation exceeds the given timeout.

   .. versionchanged:: 3.11
```

```
        This class was made an alias of :exc:`TimeoutError`.
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 518)

Unknown directive type "exception".

```
.. exception:: BrokenExecutor

   Derived from :exc:`RuntimeError`, this exception class is raised
   when an executor is broken for some reason, and cannot be used
   to submit or execute new tasks.

   .. versionadded:: 3.7
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 526)

Unknown directive type "exception".

```
.. exception:: InvalidStateError

   Raised when an operation is performed on a future that is not allowed
   in the current state.

   .. versionadded:: 3.8
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 533)

Unknown directive type "currentmodule".

```
.. currentmodule:: concurrent.futures.thread
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 535)

Unknown directive type "exception".

```
.. exception:: BrokenThreadPool

   Derived from :exc:`~concurrent.futures.BrokenExecutor`, this exception
   class is raised when one of the workers of a :class:`ThreadPoolExecutor`
   has failed initializing.

   .. versionadded:: 3.7
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 543)

Unknown directive type "currentmodule".

```
.. currentmodule:: concurrent.futures.process
```

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]concurrent.futures.rst, line 545)

Unknown directive type "exception".

```
.. exception:: BrokenProcessPool

   Derived from :exc:`~concurrent.futures.BrokenExecutor` (formerly
   :exc:`RuntimeError`), this exception class is raised when one of the
   workers of a :class:`ProcessPoolExecutor` has terminated in a non-clean
   fashion (for example, if it was killed from the outside).

   .. versionadded:: 3.3
```