

# Futures

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

Unknown directive type "currentmodule".

```
.. currentmodule:: asyncio
```

**Source code:** :source:`Lib/asyncio/futures.py`, :source:`Lib/asyncio/base\_futures.py`

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) asyncio-future.rst, line 10); [backlink](#)

Unknown interpreted text role "source".

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) asyncio-future.rst, line 10); [backlink](#)

Unknown interpreted text role "source".

---

*Future* objects are used to bridge **low-level callback-based code** with high-level `async/await` code.

## Future Functions

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) asyncio-future.rst, line 22)

Unknown directive type "function".

```
.. function:: isfuture(obj)

Return ``True`` if *obj* is either of:

* an instance of :class:`asyncio.Future`,
* an instance of :class:`asyncio.Task`,
* a Future-like object with a ``_asyncio_future_blocking``
  attribute.

.. versionadded:: 3.5
```

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) asyncio-future.rst, line 34)

Unknown directive type "function".

```
.. function:: ensure_future(obj, *, loop=None)

Return:

* *obj* argument as is, if *obj* is a :class:`Future`,
  a :class:`Task`, or a Future-like object (:func:`isfuture`
  is used for the test.)

* a :class:`Task` object wrapping *obj*, if *obj* is a
  coroutine (:func:`iscoroutine` is used for the test);
  in this case the coroutine will be scheduled by
  ``ensure_future()``.

* a :class:`Task` object that would await on *obj*, if *obj* is an
  awaitable (:func:`inspect.isawaitable` is used for the test.)

If *obj* is neither of the above a :exc:`TypeError` is raised.

.. important::
```

See also the `:func:`create_task`` function which is the preferred way for creating new Tasks.

Save a reference to the result of this function, to avoid a task disappearing mid execution.

```
.. versionchanged:: 3.5.1
    The function accepts any :term:`awaitable` object.

.. deprecated:: 3.10
    Deprecation warning is emitted if *obj* is not a Future-like object
    and *loop* is not specified and there is no running event loop.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) asyncio-future.rst, line 68)**

Unknown directive type "function".

```
.. function:: wrap_future(future, *, loop=None)
```

Wrap a `:class:`concurrent.futures.Future`` object in a `:class:`asyncio.Future`` object.

```
.. deprecated:: 3.10
    Deprecation warning is emitted if *future* is not a Future-like object
    and *loop* is not specified and there is no running event loop.
```

## Future Object

A Future represents an eventual result of an asynchronous operation. Not thread-safe.

Future is an `:term:`awaitable`` object. Coroutines can await on Future objects until they either have a result or an exception set, or until they are cancelled.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) asyncio-future.rst, line 86); [backlink](#)**

Unknown interpreted text role "term".

Typically Futures are used to enable low-level callback-based code (e.g. in protocols implemented using asyncio `:ref:`transports <asyncio-transports-protocols>`) to interoperate with high-level async/await code.

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

Unknown interpreted text role "ref".

The rule of thumb is to never expose Future objects in user-facing APIs, and the recommended way to create a Future object is to call `:meth:`loop.create_future``. This way alternative event loop implementations can inject their own optimized implementations of a Future object.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) asyncio-future.rst, line 95); [backlink](#)**

Unknown interpreted text role "meth".

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) asyncio-future.rst, line 101)**

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.7
    Added support for the :mod:`contextvars` module.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) asyncio-future.rst, line 104)**

Unknown directive type "deprecated".

```
.. deprecated:: 3.10
    Deprecation warning is emitted if *loop* is not specified
    and there is no running event loop.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) asyncio-future.rst, line 108)**

Unknown directive type "method".

```
.. method:: result()

    Return the result of the Future.

    If the Future is *done* and has a result set by the
    :meth:`set_result` method, the result value is returned.

    If the Future is *done* and has an exception set by the
    :meth:`set_exception` method, this method raises the exception.

    If the Future has been *cancelled*, this method raises
    a :exc:`CancelledError` exception.

    If the Future's result isn't yet available, this method raises
    a :exc:`InvalidStateError` exception.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) asyncio-future.rst, line 124)**

Unknown directive type "method".

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

    Mark the Future as *done* and set its result.

    Raises a :exc:`InvalidStateError` error if the Future is
    already *done*.
```

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

Unknown directive type "method".

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

    Mark the Future as *done* and set an exception.

    Raises a :exc:`InvalidStateError` error if the Future is
    already *done*.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) asyncio-future.rst, line 138)**

Unknown directive type "method".

```
.. method:: done()

    Return ``True`` if the Future is *done*.

    A Future is *done* if it was *cancelled* or if it has a result
    or an exception set with :meth:`set_result` or
    :meth:`set_exception` calls.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) asyncio-future.rst, line 146)**

Unknown directive type "method".

```
.. method:: cancelled()
```

Return ``True`` if the Future was \*cancelled\*.

The method is usually used to check if a Future is not \*cancelled\* before setting a result or an exception for it::

```
if not fut.cancelled():
    fut.set_result(42)
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) asyncio-future.rst, line 156)**

Unknown directive type "method".

```
.. method:: add_done_callback(callback, *, context=None)
```

Add a callback to be run when the Future is \*done\*.

The \*callback\* is called with the Future object as its only argument.

If the Future is already \*done\* when this method is called, the callback is scheduled with :meth:`loop.call\_soon`.

An optional keyword-only \*context\* argument allows specifying a custom :class:`contextvars.Context` for the \*callback\* to run in. The current context is used when no \*context\* is provided.

:func:`functools.partial` can be used to pass parameters to the callback, e.g.::

```
# Call 'print("Future:", fut)' when "fut" is done.
fut.add_done_callback(
    functools.partial(print, "Future:"))
```

```
.. versionchanged:: 3.7
```

The \*context\* keyword-only parameter was added.

See :pep:`567` for more details.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) asyncio-future.rst, line 181)**

Unknown directive type "method".

```
.. method:: remove_done_callback(callback)
```

Remove \*callback\* from the callbacks list.

Returns the number of callbacks removed, which is typically 1, unless a callback was added more than once.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) asyncio-future.rst, line 188)**

Unknown directive type "method".

```
.. method:: cancel(msg=None)
```

Cancel the Future and schedule callbacks.

If the Future is already \*done\* or \*cancelled\*, return ``False``. Otherwise, change the Future's state to \*cancelled\*, schedule the callbacks, and return ``True``.

```
.. versionchanged:: 3.9
    Added the *msg* parameter.
```

```
.. deprecated-removed:: 3.11 3.14
    *msg* parameter is ambiguous when multiple :meth:`cancel`
    are called with different cancellation messages.
    The argument will be removed.
```

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) asyncio-future.rst, line 204)**

Unknown directive type "method".

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

Return the exception that was set on this Future.

The exception (or ``None`` if no exception was set) is returned only if the Future is \*done\*.

If the Future has been \*cancelled\*, this method raises a :exc:`CancelledError` exception.

If the Future isn't \*done\* yet, this method raises an :exc:`InvalidStateError` exception.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) asyncio-future.rst, line 217)**

Unknown directive type "method".

```
.. method:: get_loop()
```

Return the event loop the Future object is bound to.

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

This example creates a Future object, creates and schedules an asynchronous Task to set result for the Future, and waits until the Future has a result:

```
async def set_after(fut, delay, value):
    # Sleep for *delay* seconds.
    await asyncio.sleep(delay)

    # Set *value* as a result of *fut* Future.
    fut.set_result(value)

async def main():
    # Get the current event loop.
    loop = asyncio.get_running_loop()

    # Create a new Future object.
    fut = loop.create_future()

    # Run "set_after()" coroutine in a parallel Task.
    # We are using the low-level "loop.create_task()" API here because
    # we already have a reference to the event loop at hand.
    # Otherwise we could have just used "asyncio.create_task()".
    loop.create_task(
        set_after(fut, 1, '... world'))

    print('hello ...')

    # Wait until *fut* has a result (1 second) and print it.
    print(await fut)

asyncio.run(main())
```

### Important

The Future object was designed to mimic :class:`concurrent.futures.Future`. Key differences include:

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) asyncio-future.rst, line 261); [backlink](#)**

Unknown interpreted text role "class".

- unlike asyncio Futures, :class:`concurrent.futures.Future` instances cannot be awaited.

**System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc) (library) asyncio-future.rst, line 264); [backlink](#)**

Unknown interpreted text role "class".

- `meth:'asyncio.Future.result'` and `meth:'asyncio.Future.exception'` do not accept the *timeout* argument.

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc)  
(library) asyncio-future.rst, line 267); [backlink](#)

Unknown interpreted text role "meth".

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc)  
(library) asyncio-future.rst, line 267); [backlink](#)

Unknown interpreted text role "meth".

- `meth:'asyncio.Future.result'` and `meth:'asyncio.Future.exception'` raise an `exc:'InvalidStateError'` exception when the Future is not *done*.

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc)  
(library) asyncio-future.rst, line 270); [backlink](#)

Unknown interpreted text role "meth".

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc)  
(library) asyncio-future.rst, line 270); [backlink](#)

Unknown interpreted text role "meth".

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc)  
(library) asyncio-future.rst, line 270); [backlink](#)

Unknown interpreted text role "exc".

- Callbacks registered with `meth:'asyncio.Future.add_done_callback'` are not called immediately. They are scheduled with `meth:'loop.call_soon'` instead.

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc)  
(library) asyncio-future.rst, line 274); [backlink](#)

Unknown interpreted text role "meth".

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc)  
(library) asyncio-future.rst, line 274); [backlink](#)

Unknown interpreted text role "meth".

- `asyncio Future` is not compatible with the `func:'concurrent.futures.wait'` and `func:'concurrent.futures.as_completed'` functions.

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc)  
(library) asyncio-future.rst, line 278); [backlink](#)

Unknown interpreted text role "func".

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\ (cpython-main) (Doc)  
(library) asyncio-future.rst, line 278); [backlink](#)

Unknown interpreted text role "func".

- `:meth:`asyncio.Future.cancel`` accepts an optional `msg` argument, but `:func:`concurrent.futures.cancel`` does not.

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main) (Doc)  
(library) `asyncio-future.rst`, line 282; [backlink](#)

Unknown interpreted text role "meth".

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main) (Doc)  
(library) `asyncio-future.rst`, line 282; [backlink](#)

Unknown interpreted text role "func".

**System Message: ERROR/3** (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\cpython-main) (Doc)  
(library) `asyncio-future.rst`, line 285

Unknown directive type "deprecated-removed".

```
.. deprecated-removed:: 3.11 3.14
   *msg* parameter is ambiguous when multiple :meth:`cancel`
   are called with different cancellation messages.
   The argument will be removed.
```