Queues

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

Unknown directive type "currentmodule".

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

Source code: :source:`Lib/asyncio/queues.py`

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

Unknown interpreted text role "source".

---

asyncio queues are designed to be similar to classes of the :mod:`queue` module. Although asyncio queues are not thread-safe, they are designed to be used specifically in async/await code.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]asyncio-queue.rst`, line 13); *backlink*

Unknown interpreted text role "mod".

Note that methods of asyncio queues don't have a *timeout* parameter; use :func:`asyncio.wait_for` function to do queue operations with a timeout.

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

Unknown interpreted text role "func".

See also the Examples section below.

Queue

A first in, first out (FIFO) queue.

If *maxsize* is less than or equal to zero, the queue size is infinite. If it is an integer greater than `0`, then `await put()` blocks when the queue reaches *maxsize* until an item is removed by :meth:`get`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]asyncio-queue.rst`, line 30); *backlink*

Unknown interpreted text role "meth".

Unlike the standard library threading :mod:`queue`, the size of the queue is always known and can be returned by calling the :meth:`qsize` method.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]asyncio-queue.rst`, line 35); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]asyncio-queue.rst`, line 35); *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-queue.rst`, line 39)

Unknown directive type "versionchanged".

```
.. versionchanged:: 3.10
   Removed the *loop* parameter.
```

This class is :ref:`not thread safe <asyncio-multithreading>`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]asyncio-queue.rst`, line 43); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]asyncio-queue.rst`, line 45)

Unknown directive type "attribute".

```
.. attribute:: maxsize

   Number of items allowed in the queue.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]asyncio-queue.rst`, line 49)

Unknown directive type "method".

```
.. method:: empty()

   Return ``True`` if the queue is empty, ``False`` otherwise.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]asyncio-queue.rst`, line 53)

Unknown directive type "method".

```
.. method:: full()

   Return ``True`` if there are :attr:`maxsize` items in the queue.

   If the queue was initialized with ``maxsize=0`` (the default),
   then :meth:`full()` never returns ``True``.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]asyncio-queue.rst`, line 60)

Unknown directive type "coroutinemethod".

```
.. coroutinemethod:: get()

   Remove and return an item from the queue. If queue is empty,
   wait until an item is available.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]asyncio-queue.rst`, line 65)

Unknown directive type "method".

```
.. method:: get_nowait()

   Return an item if one is immediately available, else raise
   :exc:`QueueEmpty`.
```

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

Unknown directive type "coroutinemethod".

```
.. coroutinemethod:: join()
```

```
      Block until all items in the queue have been received and processed.

      The count of unfinished tasks goes up whenever an item is added
      to the queue. The count goes down whenever a consumer coroutine calls
      :meth:`task_done` to indicate that the item was retrieved and all
      work on it is complete.  When the count of unfinished tasks drops
      to zero, :meth:`join` unblocks.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]asyncio-queue.rst`, line 80)

Unknown directive type "coroutinemethod".

```
.. coroutinemethod:: put(item)

   Put an item into the queue. If the queue is full, wait until a
   free slot is available before adding the item.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]asyncio-queue.rst`, line 85)

Unknown directive type "method".

```
.. method:: put_nowait(item)

   Put an item into the queue without blocking.

   If no free slot is immediately available, raise :exc:`QueueFull`.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]asyncio-queue.rst`, line 91)

Unknown directive type "method".

```
.. method:: qsize()

   Return the number of items in the queue.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]asyncio-queue.rst`, line 95)

Unknown directive type "method".

```
.. method:: task_done()

   Indicate that a formerly enqueued task is complete.

   Used by queue consumers. For each :meth:`~Queue.get` used to
   fetch a task, a subsequent call to :meth:`task_done` tells the
   queue that the processing on the task is complete.

   If a :meth:`join` is currently blocking, it will resume when all
   items have been processed (meaning that a :meth:`task_done`
   call was received for every item that had been :meth:`~Queue.put`
   into the queue).

   Raises :exc:`ValueError` if called more times than there were
   items placed in the queue.
```

Priority Queue

A variant of :class:`Queue`; retrieves entries in priority order (lowest first).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]asyncio-queue.rst`, line 117); *backlink*

Unknown interpreted text role "class".

Entries are typically tuples of the form `(priority_number, data)`.

LIFO Queue

A variant of :class:`Queue` that retrieves most recently added entries first (last in, first out).

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]asyncio-queue.rst`, line 129); *backlink*

Unknown interpreted text role "class".

Exceptions

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]asyncio-queue.rst`, line 136)

Unknown directive type "exception".

```
.. exception:: QueueEmpty

   This exception is raised when the :meth:`~Queue.get_nowait` method
   is called on an empty queue.
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\library\[cpython-main][Doc][library]asyncio-queue.rst`, line 142)

Unknown directive type "exception".

```
.. exception:: QueueFull

   Exception raised when the :meth:`~Queue.put_nowait` method is called
   on a queue that has reached its *maxsize*.
```

Examples

Queues can be used to distribute workload between several concurrent tasks:

```
import asyncio
import random
import time


async def worker(name, queue):
    while True:
        # Get a "work item" out of the queue.
        sleep_for = await queue.get()

        # Sleep for the "sleep_for" seconds.
        await asyncio.sleep(sleep_for)

        # Notify the queue that the "work item" has been processed.
        queue.task_done()

        print(f'{name} has slept for {sleep_for:.2f} seconds')


async def main():
    # Create a queue that we will use to store our "workload".
    queue = asyncio.Queue()

    # Generate random timings and put them into the queue.
    total_sleep_time = 0
    for _ in range(20):
        sleep_for = random.uniform(0.05, 1.0)
        total_sleep_time += sleep_for
        queue.put_nowait(sleep_for)

    # Create three worker tasks to process the queue concurrently.
    tasks = []
    for i in range(3):
        task = asyncio.create_task(worker(f'worker-{i}', queue))
        tasks.append(task)
    # Wait until the queue is fully processed.
    started_at = time.monotonic()
    await queue.join()
    total_slept_for = time.monotonic() - started_at

    # Cancel our worker tasks.
    for task in tasks:
        task.cancel()
    # Wait until all worker tasks are cancelled.
    await asyncio.gather(*tasks, return_exceptions=True)

    print('====')
    print(f'3 workers slept in parallel for {total_slept_for:.2f} seconds')
    print(f'total expected sleep time: {total_sleep_time:.2f} seconds')


asyncio.run(main())
```