

asyncio — Asynchronous I/O, event loop, and concurrency tools

Purpose: An asynchronous I/O and concurrency framework.

The `asyncio` module provides tools for building concurrent applications using coroutines. While the [threading](#) module implements concurrency through application threads and [multiprocessing](#) implements concurrency using system processes, `asyncio` uses a single-threaded, single-process approach in which parts of an application cooperate to switch tasks explicitly at optimal times. Most often this context switching occurs when the program would otherwise block waiting to read or write data, but `asyncio` also includes support for scheduling code to run at a specific future time, to enable one coroutine to wait for another to complete, for handling system signals, and for recognizing other events that may be reasons for an application to change what it is working on.

- [Asynchronous Concurrency Concepts](#)
- [Cooperative Multitasking with Coroutines](#)
 - [Starting a Coroutine](#)
 - [Returning Values from Coroutines](#)
 - [Chaining Coroutines](#)
 - [Generators Instead of Coroutines](#)
- [Scheduling Calls to Regular Functions](#)
 - [Scheduling a Callback “Soon”](#)
 - [Scheduling a Callback with a Delay](#)
 - [Scheduling a Callback for a Specific Time](#)
- [Producing Results Asynchronously](#)
 - [Waiting for a Future](#)
 - [Future Callbacks](#)
- [Executing Tasks Concurrently](#)
 - [Starting a Task](#)
 - [Canceling a Task](#)
 - [Creating Tasks from Coroutines](#)
- [Composing Coroutines with Control Structures](#)
 - [Waiting for Multiple Coroutines](#)
 - [Gathering Results from Coroutines](#)
 - [Handling Background Operations as They Finish](#)
- [Synchronization Primitives](#)
 - [Locks](#)
 - [Events](#)
 - [Conditions](#)
 - [Queues](#)
- [Asynchronous I/O with Protocol Class Abstractions](#)
 - [Echo Server](#)
 - [Echo Client](#)
 - [Output](#)
- [Asynchronous I/O Using Coroutines and Streams](#)
 - [Echo Server](#)
 - [Echo Client](#)
 - [Output](#)
- [Using SSL](#)
- [Interacting with Domain Name Services](#)
 - [Address Lookup by Name](#)

- Name Lookup by Address
- Working with Subprocesses
 - Using the Protocol Abstraction with Subprocesses
 - Calling Subprocesses with Coroutines and Streams
 - Sending Data to a Subprocess
- Receiving Unix Signals
- Combining Coroutines with Threads and Processes
 - Threads
 - Processes
- Debugging with `asyncio`

Note

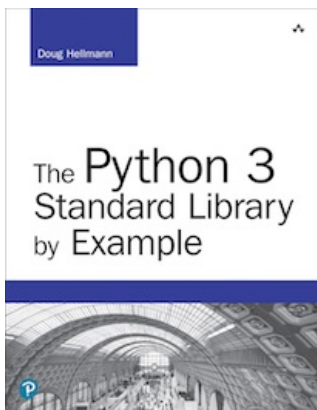
In Python 3.5, `asyncio` is still a *provisional* module. The API was stabilized in Python 3.6, and most of the changes were backported to later patch releases of Python 3.5. As a result, the module may work slightly differently under different versions of Python 3.5.

See also

- [Standard library documentation for `asyncio`](#)
- [PEP 3156](#) – *Asynchronous IO Support Rebooted: the “`asyncio`” Module*
- [PEP 380](#) – *Syntax for Delegating to a Subgenerator*
- [PEP 492](#) – *Coroutines with `async` and `await` syntax*
- [concurrent.futures](#) – Manage Pools of Concurrent Tasks
- [socket](#) – Low-level network communication
- [select](#) – Low-level asynchronous I/O tools
- [socketserver](#) – Framework for creating network servers
- [asyncio: What’s New in Python 3.6](#) – Summary of the changes to `asyncio` as the API stabilized in Python 3.6.
- [trollius](#) – A port of `Tulip`, the original version of `asyncio`, to Python 2.
- [The New `asyncio` Module in Python 3.4: Event Loops](#) – Article by Gastón Hillar in Dr. Dobbs’s
- [Exploring Python 3’s `Asyncio` by Example](#) – Blog post by Chat Lung
- [A Web Crawler With `asyncio` Coroutines](#) – An article in *The Architecture of Open Source Applications* by A. Jesse Jiryu Davis and Guido van Rossum
- [Playing with `asyncio`](#) – blog post by Nathan Hoad
- [Async I/O and Python](#) – blog post by Mark McLoughlin
- [A Curious Course on Coroutines and Concurrency](#) – PyCon 2009 tutorial by David Beazley
- [How the heck does `async/await` work in Python 3.5?](#) – blog post by Brett Cannon
- *Unix Network Programming, Volume 1: The Sockets Networking API, 3/E* By W. Richard Stevens, Bill Fenner, and Andrew M. Rudoff. Published by Addison-Wesley Professional, 2004. ISBN-10: 0131411151
- *Foundations of Python Network Programming, 3/E* By Brandon Rhodes and John Goerzen. Published by Apress, 2014. ISBN-10: 1430258543

Navigation

- ▶ Implementing MapReduce
- ▶ Asynchronous Concurrency Concepts



[Get the book](#)

The output from all the example programs from PyMOTW-3 has been generated with Python 3.7.1, unless otherwise noted. Some of the features described here may not be available in earlier versions of Python.

Looking for [examples for Python 2?](#)

This Site

☰ Module Index

I Index



© Copyright 2019, Doug Hellmann



Other Writing

✍ Blog

📖 The Python Standard Library By Example