Networking

Network communication is used to retrieve data needed for an algorithm running locally, share information for distributed processing, and to manage cloud services. Python’s standard library comes complete with modules for creating network services, as well as for accessing existing services remotely.

The `ipaddress` module includes classes for validating, comparing, and otherwise operating on IPv4 and IPv6 network addresses.

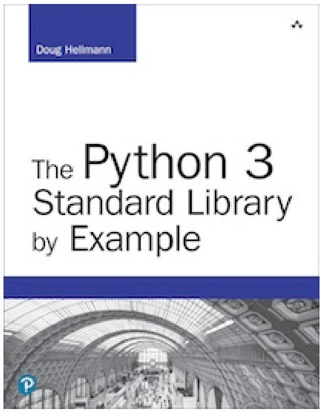
The low-level `socket` library provides direct access to the native C socket library, and can be used to communicate with any network service. `selectors` provides a high-level interface for watching multiple sockets simultaneously, and is useful for allowing network servers to communicate with multiple clients simultaneously. `select` provides the low-level APIs used by `selectors`.

The frameworks in `socketserver` abstract out a lot of the repetitive work necessary to create a new network server. The classes can be combined to create servers that fork or use threads and support TCP or UDP. Only the actual message handling needs to be provided by the application.

concurrent.futures — Manage Pools of Concurrent Tasks   ipaddress — Internet Addresses

This page was last updated 2016-10-09.

Navigation
concurrent.futures — Manage Pools of Concurrent Tasks
ipaddress — Internet Addresses



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?