The first series of these scripts describes the functionality of Python Concurrent Futures.

Python concurrent.futures allows launching parallel tasks. The concurrent.futures module provides a high-level interface for asynchronously executing callables. The asynchronous execution can be performed with threads, using ThreadPoolExecutor, or separate processes, using ProcessPoolExecutor. Both implement the same interface, which is defined by the abstract Executor class.

Also explored is the Python ‘asyncore’ module used as an asynchronous socket handler. This module exists for backwards compatibility only. For new code we recommend using asyncio. This module provides the basic infrastructure for writing asynchronous socket service clients and servers. There are only two ways to have a program on a single processor do “more than one thing at a time.”

Multi-threaded programming is the simplest and most popular way to do it, but there is another very different technique, that lets you have nearly all the advantages of multi-threading, without actually using multiple threads. It’s really only practical if your program is largely I/O bound. If your program is processor bound, then pre-emptive scheduled threads are probably what you really need. Network servers are rarely processor bound, however.

Compiled and presented by Vakindu Philliam.