Application Building Blocks

The strength of Python’s standard library is its size. It includes implementations of so many aspects of a program’s structure that developers can concentrate on what makes their application unique, instead of having to write all of the basic pieces over and over again. This chapter covers some of the more frequently reused building blocks that solve problems common to so many applications.

`argparse` is an interface for parsing and validating command line arguments. It supports converting arguments from strings to integers and other types, running callbacks when an option is encountered, setting default values for options not provided by the user, and automatically producing usage instructions for a program. `getopt` implements the low level argument processing model available to C programs and shell scripts. It has fewer features than other option parsing libraries, but that simplicity and familiarity make it a popular choice.

Interactive programs should use `readline` to give the user a command prompt. It includes tools for managing history, auto-completing parts of commands, and interactive editing with `emacs` and `vi` key-bindings. To securely prompt the user for a password or other secret value, without echoing the value to the screen as it is typed, use `getpass`.

The `cmd` module includes a framework for interactive, command-driven shell style programs. It provides the main loop and handles the interaction with the user so the application only needs to implement the processing callbacks for the individual commands.

`shlex` is a parser for shell-style syntax, with lines made up of tokens separated by white-space. It is smart about quotes and escape sequences, so text with embedded spaces is treated as a single token. `shlex` works well as the tokenizer for domain-specific languages such as configuration files or programming languages.

It is easy to manage application configuration files with `configparser`. It can save user preferences between program runs and read them the next time an application starts, or even serve as a simple data file format.

Applications being deployed in the real world need to give their users debugging information. Simple error messages and tracebacks are helpful, but when it is difficult to reproduce an issue a full activity log can point directly to the chain of events that leads to a failure. The `logging` module includes a full-featured API that manages log files, supports multiple threads, and even interfaces with remote logging daemons for centralized logging.

One of the most common patterns for programs in Unix environments is a line-by-line filter that reads data, modifies it, and writes it back out. Reading from files is simple enough, but there may not be an easier way to create a filter application than by using the `fileinput` module. Its API is a line iterator that yields each input line, so the main body of the program is a simple `for` loop. The module handles parsing command line arguments for filenames to be processed, or falling back to reading directly from standard input, so tools built on `fileinput` can be run directly on a file or as part of a pipeline.

Use `atexit` to schedule functions to be run as the interpreter is shutting down a program. Registering exit callbacks is useful for releasing resources by logging out of remote services, closing files, etc.

The `sched` module implements a scheduler for triggering events at set times in the future. The API does not dictate the definition of “time,” so anything from true clock time to interpreter steps can be used.

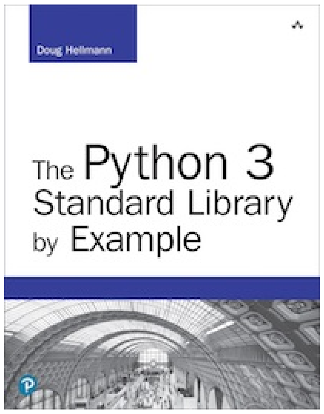
imaplib — IMAP4 Client Library

argparse — Command-Line Option and Argument Parsing

This page was last updated 2016-12-30.

Navigation

imaplib — IMAP4 Client Library
argparse — Command-Line Option and Argument Parsing



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
Index

Other Writing

Blog
The Python Standard Library By Example