

# The File System

Python's standard library includes a large range of tools for working with files on the file system, building and parsing filenames, and examining file contents.

The first step in working with files is to determine the name of the file to work on. Python represents filenames as simple strings, but provides tools for building them from standard, platform-independent, components in [os.path](#).

The [pathlib](#) module provides an object-oriented API for working with file system paths. Using it instead of [os.path](#) provides some conveniences because it operates at a higher level of abstraction.

List the contents of a directory with `listdir()` from [os](#), or use [glob](#) to build a list of filenames from a pattern.

The file name pattern matching used by [glob](#) is also exposed directly through [fnmatch](#) so it can be used in other contexts.

After the name of the file is identified, other characteristics, such as permissions or the file size, can be checked using `os.stat()` and the constants in `stat`.

When an application needs random access to files, [linecache](#) makes it easy to read lines by their line number. The contents of the file are maintained in a cache, so be careful of memory consumption.

[tempfile](#) is useful for cases that need to create scratch files to hold data temporarily, or before moving it to a permanent location. It provides classes to create temporary files and directories safely and securely. Names are guaranteed to be unique, and include random components so they are not easily guessable.

Frequently, programs need to work on files as a whole, without regard to their content. The [shutil](#) module includes high-level file operations such as copying files and directories, and creating or extracting archives of files.

The [filecmp](#) module compares files and directories by looking at the bytes they contain, but without any special knowledge about their format.

The built-in `file` class can be used to read and write files visible on local file systems. A program's performance can suffer when it accesses large files through the `read()` and `write()` interfaces, though, since they both involve copying the data multiple times as it is moved from the disk to memory the application can see. Using [mmap](#) tells the operating system to use its virtual memory subsystem to map a file's contents directly into memory accessible by a program, avoiding a copy step between the operating system and the internal buffer for the `file` object.

Text data using characters not available in ASCII is usually saved in a Unicode data format. Since the standard file handle assumes each byte of a text file represents one character, reading Unicode text with multi-byte encodings requires extra processing. The [codecs](#) module handles the encoding and decoding automatically, so that in many cases a non-ASCII file can be used without any other changes to the program.

The [io](#) module provides access to the classes used to implement Python's file-based input and output. For testing code that depends on reading or writing data from files, [io](#) provides an in-memory stream object that behaves like a file, but does not reside on disk.

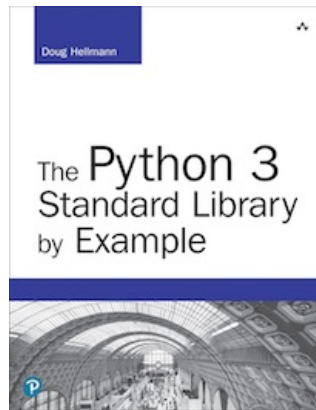
- [os.path](#) — Platform-independent Manipulation of Filenames
- [pathlib](#) — Filesystem Paths as Objects
- [glob](#) — Filename Pattern Matching
- [fnmatch](#) — Unix-style Glob Pattern Matching
- [linecache](#) — Read Text Files Efficiently
- [tempfile](#) — Temporary File System Objects
- [shutil](#) — High-level File Operations
- [filecmp](#) — Compare Files
- [mmap](#) — Memory-map Files
- [codecs](#) — String Encoding and Decoding
- [io](#) — Text, Binary, and Raw Stream I/O Tools

[← statistics — Statistical Calculations](#)

[os.path — Platform-independent Manipulation of Filenames →](#)

## Navigation

- Statistics — Statistical Calculations
- os.path — Platform-independent Manipulation of Filenames




[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](#)



© Copyright 2019, Doug Hellmann



## Other Writing

 [Blog](#)

 [The Python Standard Library By Example](#)