

Previous topic

[10. Full Grammar specification](#)

Next topic

[Introduction](#)

This Page

[Report a Bug](#)  
[Show Source](#)

# The Python Standard Library

While [The Python Language Reference](#) describes the exact syntax and semantics of the Python language, this library reference manual describes the standard library that is distributed with Python. It also describes some of the optional components that are commonly included in Python distributions.

Python’s standard library is very extensive, offering a wide range of facilities as indicated by the long table of contents listed below. The library contains built-in modules (written in C) that provide access to system functionality such as file I/O that would otherwise be inaccessible to Python programmers, as well as modules written in Python that provide standardized solutions for many problems that occur in everyday programming. Some of these modules are explicitly designed to encourage and enhance the portability of Python programs by abstracting away platform-specifics into platform-neutral APIs.

The Python installers for the Windows platform usually include the entire standard library and often also include many additional components. For Unix-like operating systems Python is normally provided as a collection of packages, so it may be necessary to use the packaging tools provided with the operating system to obtain some or all of the optional components.

In addition to the standard library, there is a growing collection of several thousand components (from individual programs and modules to packages and entire application development frameworks), available from the [Python Package Index](#).

- [Introduction](#)
  - [Notes on availability](#)
- [Built-in Functions](#)
- [Built-in Constants](#)
  - [Constants added by the `site` module](#)
- [Built-in Types](#)
  - [Truth Value Testing](#)
  - [Boolean Operations](#) — `and`, `or`, `not`
  - [Comparisons](#)
  - [Numeric Types](#) — `int`, `float`, `complex`
  - [Iterator Types](#)
  - [Sequence Types](#) — `list`, `tuple`, `range`
  - [Text Sequence Type](#) — `str`

- Binary Sequence Types — `bytes`, `bytearray`, `memoryview`
    - Set Types — `set`, `frozenset`
    - Mapping Types — `dict`
    - Context Manager Types
    - Generic Alias Type
    - Other Built-in Types
    - Special Attributes
  - Built-in Exceptions
    - Base classes
    - Concrete exceptions
    - Warnings
    - Exception hierarchy
  - Text Processing Services
    - `string` — Common string operations
    - `re` — Regular expression operations
    - `difflib` — Helpers for computing deltas
    - `textwrap` — Text wrapping and filling
    - `unicodedata` — Unicode Database
    - `stringprep` — Internet String Preparation
    - `readline` — GNU readline interface
    - `rlcompleter` — Completion function for GNU readline
  - Binary Data Services
    - `struct` — Interpret bytes as packed binary data
    - `codecs` — Codec registry and base classes
  - Data Types
    - `datetime` — Basic date and time types
    - `zoneinfo` — IANA time zone support
    - `calendar` — General calendar-related functions
    - `collections` — Container datatypes
    - `collections.abc` — Abstract Base Classes for Containers
    - `heapq` — Heap queue algorithm
    - `bisect` — Array bisection algorithm
    - `array` — Efficient arrays of numeric values
    - `weakref` — Weak references
    - `types` — Dynamic type creation and names for built-in types
    - `copy` — Shallow and deep copy operations
    - `pprint` — Data pretty printer
    - `reprlib` — Alternate `repr()` implementation
    - `enum` — Support for enumerations
    - `graphlib` — Functionality to operate with graph-like structures
  - Numeric and Mathematical Modules
    - `numbers` — Numeric abstract base classes
    - `math` — Mathematical functions
    - `cmath` — Mathematical functions for complex numbers
    - `decimal` — Decimal fixed point and floating point arithmetic

- [fractions](#) — Rational numbers
- [random](#) — Generate pseudo-random numbers
- [statistics](#) — Mathematical statistics functions
- **Functional Programming Modules**
  - [itertools](#) — Functions creating iterators for efficient looping
  - [functools](#) — Higher-order functions and operations on callable objects
  - [operator](#) — Standard operators as functions
- **File and Directory Access**
  - [pathlib](#) — Object-oriented filesystem paths
  - [os.path](#) — Common pathname manipulations
  - [fileinput](#) — Iterate over lines from multiple input streams
  - [stat](#) — Interpreting `stat()` results
  - [filecmp](#) — File and Directory Comparisons
  - [tempfile](#) — Generate temporary files and directories
  - [glob](#) — Unix style pathname pattern expansion
  - [fnmatch](#) — Unix filename pattern matching
  - [linecache](#) — Random access to text lines
  - [shutil](#) — High-level file operations
- **Data Persistence**
  - [pickle](#) — Python object serialization
  - [copyreg](#) — Register `pickle` support functions
  - [shelve](#) — Python object persistence
  - [marshal](#) — Internal Python object serialization
  - [dbm](#) — Interfaces to Unix “databases”
  - [sqlite3](#) — DB-API 2.0 interface for SQLite databases
- **Data Compression and Archiving**
  - [zlib](#) — Compression compatible with **gzip**
  - [gzip](#) — Support for **gzip** files
  - [bz2](#) — Support for **bzip2** compression
  - [lzma](#) — Compression using the LZMA algorithm
  - [zipfile](#) — Work with ZIP archives
  - [tarfile](#) — Read and write tar archive files
- **File Formats**
  - [csv](#) — CSV File Reading and Writing
  - [configparser](#) — Configuration file parser
  - [netrc](#) — `netrc` file processing
  - [xdrlib](#) — Encode and decode XDR data
  - [plistlib](#) — Generate and parse Apple `.plist` files
- **Cryptographic Services**
  - [hashlib](#) — Secure hashes and message digests
  - [hmac](#) — Keyed-Hashing for Message Authentication
  - [secrets](#) — Generate secure random numbers for managing secrets
- **Generic Operating System Services**
  - [os](#) — Miscellaneous operating system interfaces
  - [io](#) — Core tools for working with streams
  - [time](#) — Time access and conversions

- `argparse` — Parser for command-line options, arguments and sub-commands
- `getopt` — C-style parser for command line options
- `logging` — Logging facility for Python
- `logging.config` — Logging configuration
- `logging.handlers` — Logging handlers
- `getpass` — Portable password input
- `curses` — Terminal handling for character-cell displays
- `curses.textpad` — Text input widget for curses programs
- `curses.ascii` — Utilities for ASCII characters
- `curses.panel` — A panel stack extension for curses
- `platform` — Access to underlying platform's identifying data
- `errno` — Standard errno system symbols
- `ctypes` — A foreign function library for Python
- **Concurrent Execution**
  - `threading` — Thread-based parallelism
  - `multiprocessing` — Process-based parallelism
  - `multiprocessing.shared_memory` — Provides shared memory for direct access across processes
  - **The concurrent package**
  - `concurrent.futures` — Launching parallel tasks
  - `subprocess` — Subprocess management
  - `sched` — Event scheduler
  - `queue` — A synchronized queue class
  - `contextvars` — Context Variables
  - `_thread` — Low-level threading API
- **Networking and Interprocess Communication**
  - `asyncio` — Asynchronous I/O
  - `socket` — Low-level networking interface
  - `ssl` — TLS/SSL wrapper for socket objects
  - `select` — Waiting for I/O completion
  - `selectors` — High-level I/O multiplexing
  - `asyncore` — Asynchronous socket handler
  - `asynchat` — Asynchronous socket command/response handler
  - `signal` — Set handlers for asynchronous events
  - `mmap` — Memory-mapped file support
- **Internet Data Handling**
  - `email` — An email and MIME handling package
  - `json` — JSON encoder and decoder
  - `mailcap` — Mailcap file handling
  - `mailbox` — Manipulate mailboxes in various formats
  - `mimetypes` — Map filenames to MIME types
  - `base64` — Base16, Base32, Base64, Base85 Data Encodings
  - `binhex` — Encode and decode binhex4 files
  - `binascii` — Convert between binary and ASCII
  - `quopri` — Encode and decode MIME quoted-printable

«

- data
    - `uu` — Encode and decode uuencode files
- **Structured Markup Processing Tools**
  - `html` — HyperText Markup Language support
  - `html.parser` — Simple HTML and XHTML parser
  - `html.entities` — Definitions of HTML general entities
  - **XML Processing Modules**
  - `xml.etree.ElementTree` — The ElementTree XML API
  - `xml.dom` — The Document Object Model API
  - `xml.dom.minidom` — Minimal DOM implementation
  - `xml.dom.pulldom` — Support for building partial DOM trees
  - `xml.sax` — Support for SAX2 parsers
  - `xml.sax.handler` — Base classes for SAX handlers
  - `xml.sax.saxutils` — SAX Utilities
  - `xml.sax.xmlreader` — Interface for XML parsers
  - `xml.parsers.expat` — Fast XML parsing using Expat
- **Internet Protocols and Support**
  - `webbrowser` — Convenient Web-browser controller
  - `cgi` — Common Gateway Interface support
  - `cgitb` — Traceback manager for CGI scripts
  - `wsgiref` — WSGI Utilities and Reference Implementation
  - `urllib` — URL handling modules
  - `urllib.request` — Extensible library for opening URLs
  - `urllib.response` — Response classes used by urllib
  - `urllib.parse` — Parse URLs into components
  - `urllib.error` — Exception classes raised by `urllib.request`
  - `urllib.robotparser` — Parser for robots.txt
  - `http` — HTTP modules
  - `http.client` — HTTP protocol client
  - `ftplib` — FTP protocol client
  - `poplib` — POP3 protocol client
  - `imaplib` — IMAP4 protocol client
  - `nntplib` — NNTP protocol client
  - `smtplib` — SMTP protocol client
  - `smtpd` — SMTP Server
  - `telnetlib` — Telnet client
  - `uuid` — UUID objects according to **RFC 4122**
  - `socketserver` — A framework for network servers
  - `http.server` — HTTP servers
  - `http.cookies` — HTTP state management
  - `http.cookiejar` — Cookie handling for HTTP clients
  - `xmlrpc` — XMLRPC server and client modules
  - `xmlrpc.client` — XML-RPC client access
  - `xmlrpc.server` — Basic XML-RPC servers
  - `ipaddress` — IPv4/IPv6 manipulation library
- **Multimedia Services**

- [audioop](#) — Manipulate raw audio data
- [aifc](#) — Read and write AIFF and AIFC files
- [sunau](#) — Read and write Sun AU files
- [wave](#) — Read and write WAV files
- [chunk](#) — Read IFF chunked data
- [colorsys](#) — Conversions between color systems
- [imghdr](#) — Determine the type of an image
- [sndhdr](#) — Determine type of sound file
- [ossaudiodev](#) — Access to OSS-compatible audio devices
- **Internationalization**
  - [gettext](#) — Multilingual internationalization services
  - [locale](#) — Internationalization services
- **Program Frameworks**
  - [turtle](#) — Turtle graphics
  - [cmd](#) — Support for line-oriented command interpreters
  - [shlex](#) — Simple lexical analysis
- **Graphical User Interfaces with Tk**
  - [tkinter](#) — Python interface to Tcl/Tk
  - [tkinter.colorchooser](#) — Color choosing dialog
  - [tkinter.font](#) — Tkinter font wrapper
  - **Tkinter Dialogs**
  - [tkinter.messagebox](#) — Tkinter message prompts
  - [tkinter.scrolledtext](#) — Scrolled Text Widget
  - [tkinter.dnd](#) — Drag and drop support
  - [tkinter.ttk](#) — Tk themed widgets
  - [tkinter.tix](#) — Extension widgets for Tk
  - **IDLE**
- **Development Tools**
  - [typing](#) — Support for type hints
  - [pydoc](#) — Documentation generator and online help system
  - **Python Development Mode**
  - **Effects of the Python Development Mode**
  - **ResourceWarning Example**
  - **Bad file descriptor error example**
  - [doctest](#) — Test interactive Python examples
  - [unittest](#) — Unit testing framework
  - [unittest.mock](#) — mock object library
  - [unittest.mock](#) — getting started
  - **2to3 - Automated Python 2 to 3 code translation**
  - [test](#) — Regression tests package for Python
  - [test.support](#) — Utilities for the Python test suite
  - [test.support.socket\\_helper](#) — Utilities for socket tests
  - [test.support.script\\_helper](#) — Utilities for the Python execution tests
  - [test.support.bytecode\\_helper](#) — Support tools for testing correct bytecode generation
- **Debugging and Profiling**

- [Audit events table](#)
- [bdb](#) — Debugger framework
- [faulthandler](#) — Dump the Python traceback
- [pdb](#) — The Python Debugger
- [The Python Profilers](#)
- [timeit](#) — Measure execution time of small code snippets
- [trace](#) — Trace or track Python statement execution
- [tracemalloc](#) — Trace memory allocations
- [Software Packaging and Distribution](#)
  - [distutils](#) — Building and installing Python modules
  - [ensurepip](#) — Bootstrapping the `pip` installer
  - [venv](#) — Creation of virtual environments
  - [zipapp](#) — Manage executable Python zip archives
- [Python Runtime Services](#)
  - [sys](#) — System-specific parameters and functions
  - [sysconfig](#) — Provide access to Python’s configuration information
  - [builtins](#) — Built-in objects
  - [\\_\\_main\\_\\_](#) — Top-level script environment
  - [warnings](#) — Warning control
  - [dataclasses](#) — Data Classes
  - [contextlib](#) — Utilities for `with`-statement contexts
  - [abc](#) — Abstract Base Classes
  - [atexit](#) — Exit handlers
  - [traceback](#) — Print or retrieve a stack traceback
  - [\\_\\_future\\_\\_](#) — Future statement definitions
  - [gc](#) — Garbage Collector interface
  - [inspect](#) — Inspect live objects
  - [site](#) — Site-specific configuration hook
- [Custom Python Interpreters](#)
  - [code](#) — Interpreter base classes
  - [codeop](#) — Compile Python code
- [Importing Modules](#)
  - [zipimport](#) — Import modules from Zip archives
  - [pkgutil](#) — Package extension utility
  - [modulefinder](#) — Find modules used by a script
  - [runpy](#) — Locating and executing Python modules
  - [importlib](#) — The implementation of `import`
  - [Using importlib.metadata](#)
- [Python Language Services](#)
  - [parser](#) — Access Python parse trees
  - [ast](#) — Abstract Syntax Trees
  - [symtable](#) — Access to the compiler’s symbol tables
  - [symbol](#) — Constants used with Python parse trees
  - [token](#) — Constants used with Python parse trees
  - [keyword](#) — Testing for Python keywords
  - [tokenize](#) — Tokenizer for Python source
  - [tabnanny](#) — Detection of ambiguous indentation
  - [pyclbr](#) — Python module browser support



- `py_compile` — Compile Python source files
- `compileall` — Byte-compile Python libraries
- `dis` — Disassembler for Python bytecode
- `pickletools` — Tools for pickle developers
- **Miscellaneous Services**
  - `formatter` — Generic output formatting
- **MS Windows Specific Services**
  - `msilib` — Read and write Microsoft Installer files
  - `msvcrt` — Useful routines from the MS VC++ runtime
  - `winreg` — Windows registry access
  - `winsound` — Sound-playing interface for Windows
- **Unix Specific Services**
  - `posix` — The most common POSIX system calls
  - `pwd` — The password database
  - `spwd` — The shadow password database
  - `grp` — The group database
  - `crypt` — Function to check Unix passwords
  - `termios` — POSIX style tty control
  - `tty` — Terminal control functions
  - `pty` — Pseudo-terminal utilities
  - `fcntl` — The `fcntl` and `ioctl` system calls
  - `pipes` — Interface to shell pipelines
  - `resource` — Resource usage information
  - `nis` — Interface to Sun's NIS (Yellow Pages)
  - `syslog` — Unix syslog library routines
- **Superseded Modules**
  - `optparse` — Parser for command line options
  - `imp` — Access the import internals
- **Undocumented Modules**
  - Platform specific modules
- **Security Considerations**



