



Doug Hellmann

The Python 3 Standard Library by Example





Musée d'Orsay, Paris, France

Located on the Seine's left bank, the Musée d'Orsay is housed in a breathtaking Beaux-Arts building originally designed as the world's first electrified urban railway station. The original "Gare d'Orsay" was built on the site of the old Palais d'Orsay, which had lain empty since it burned in 1871 during the Paris Commune. The building opened on Bastille Day, July 14, 1900, to help celebrate Paris's Fifth Universal Exhibition. Designated a Historical Monument in 1978, it was then recreated as a museum by Renaud Bardon, Pierre Colboc, and Jean-Paul Philippon of the ACT architecture group. Per the museum's official history, the new architects "highlighted the great hall, using it as the main artery of the visit, and transformed the magnificent glass awning into the museum's entrance." Inside, Gae Aulenti adapted the enormous station into museum spaces, unified via consistent stone wall and floor surfaces. Opened in 1986, the new museum brought together three major art collections from the era 1848-1914. More than three million visitors now come every year to see works from artists including Cézanne, Courbet, Degas, Gauguin, Manet, Monet, and Renoir.

Python 3 Standard Library by Example, The

Table of Contents

Cover

Title Page

Copyright Page

Contents

Introduction

Acknowledgments

About the Author

Chapter 1 Text

1.1 string: Text Constants and Templates

1.1.1 Functions

1.1.2 Templates

1.1.3 Advanced Templates

1.1.4 Formatter

1.1.5 Constants

1.2 textwrap: Formatting Text Paragraphs

1.2.1 Example Data

1.2.2 Filling Paragraphs

1.2.3 Removing Existing Indentation

1.2.4 Combining Dedent and Fill

1.2.5 Indenting Blocks

1.2.6 Hanging Indents

1.2.7 Truncating Long Text

Table of Contents

1.3 re: Regular Expressions

- 1.3.1 Finding Patterns in Text
- 1.3.2 Compiling Expressions
- 1.3.3 Multiple Matches
- 1.3.4 Pattern Syntax
- 1.3.5 Constraining the Search
- 1.3.6 Dissecting Matches with Groups
- 1.3.7 Search Options
- 1.3.8 Looking Ahead or Behind
- 1.3.9 Self-referencing Expressions
- 1.3.10 Modifying Strings with Patterns
- 1.3.11 Splitting with Patterns

1.4 difflib: Compare Sequences

- 1.4.1 Comparing Bodies of Text
- 1.4.2 Junk Data
- 1.4.3 Comparing Arbitrary Types

Chapter 2 Data Structures

2.1 enum: Enumeration Type

- 2.1.1 Creating Enumerations
- 2.1.2 Iteration
- 2.1.3 Comparing Enums
- 2.1.4 Unique Enumeration Values
- 2.1.5 Creating Enumerations Programmatically
- 2.1.6 Non-integer Member Values

2.2 collections: Container Data Types

- 2.2.1 ChainMap: Search Multiple Dictionaries
- 2.2.2 Counter: Count Hashable Objects
- 2.2.3 defaultdict: Missing Keys Return a Default Value
- 2.2.4 deque: Double-Ended Queue

Table of Contents

2.2.5 namedtuple: Tuple Subclass with Named Fields

2.2.6 OrderedDict: Remember the Order Keys Are Added to a Dictionary

2.2.7 collections.abc: Abstract Base Classes for Containers

2.3 array: Sequence of Fixed-Type Data

2.3.1 Initialization

2.3.2 Manipulating Arrays

2.3.3 Arrays and Files

2.3.4 Alternative Byte Ordering

2.4 heapq: Heap Sort Algorithm

2.4.1 Example Data

2.4.2 Creating a Heap

2.4.3 Accessing the Contents of a Heap

2.4.4 Data Extremes from a Heap

2.4.5 Efficiently Merging Sorted Sequences

2.5 bisect: Maintain Lists in Sorted Order

2.5.1 Inserting in Sorted Order

2.5.2 Handling Duplicates

2.6 queue: Thread-Safe FIFO Implementation

2.6.1 Basic FIFO Queue

2.6.2 LIFO Queue

2.6.3 Priority Queue

2.6.4 Building a Threaded Podcast Client

2.7 struct: Binary Data Structures

2.7.1 Functions Versus Struct Class

2.7.2 Packing and Unpacking

2.7.3 Endianness

2.7.4 Buffers

2.8 weakref: Impermanent References to Objects

Table of Contents

2.8.1 References

2.8.2 Reference Callbacks

2.8.3 Finalizing Objects

2.8.4 Proxies

2.8.5 Caching Objects

2.9 copy: Duplicate Objects

2.9.1 Shallow Copies

2.9.2 Deep Copies

2.9.3 Customizing Copy Behavior

2.9.4 Recursion in Deep Copy

2.10 pprint: Pretty-Print Data Structures

2.10.1 Printing

2.10.2 Formatting

2.10.3 Arbitrary Classes

2.10.4 Recursion

2.10.5 Limiting Nested Output

2.10.6 Controlling Output Width

Chapter 3 Algorithms

3.1 functools: Tools for Manipulating Functions

3.1.1 Decorators

3.1.2 Comparison

3.1.3 Caching

3.1.4 Reducing a Data Set

3.1.5 Generic Functions

3.2 itertools: Iterator Functions

3.2.1 Merging and Splitting Iterators

3.2.2 Converting Inputs

3.2.3 Producing New Values

3.2.4 Filtering

Table of Contents

3.2.5 Grouping Data

3.2.6 Combining Inputs

3.3 operator: Functional Interface to Built-in Operators

3.3.1 Logical Operations

3.3.2 Comparison Operators

3.3.3 Arithmetic Operators

3.3.4 Sequence Operators

3.3.5 In-Place Operators

3.3.6 Attribute and Item Getters

3.3.7 Combining Operators and Custom Classes

3.4 contextlib: Context Manager Utilities

3.4.1 Context Manager API

3.4.2 Context Managers as Function Decorators

3.4.3 From Generator to Context Manager

3.4.4 Closing Open Handles

3.4.5 Ignoring Exceptions

3.4.6 Redirecting Output Streams

3.4.7 Dynamic Context Manager Stacks

Chapter 4 Dates and Times

4.1 time: Clock Time

4.1.1 Comparing Clocks

4.1.2 Wall Clock Time

4.1.3 Monotonic Clocks

4.1.4 Processor Clock Time

4.1.5 Performance Counter

4.1.6 Time Components

4.1.7 Working with Time Zones

4.1.8 Parsing and Formatting Times

4.2 datetime: Date and Time Value Manipulation

Table of Contents

4.2.1 Times

4.2.2 Dates

4.2.3 timedeltas

4.2.4 Date Arithmetic

4.2.5 Comparing Values

4.2.6 Combining Dates and Times

4.2.7 Formatting and Parsing

4.2.8 Time Zones

4.3 calendar: Work with Dates

4.3.1 Formatting Examples

4.3.2 Locales

4.3.3 Calculating Dates

Chapter 5 Mathematics

5.1 decimal: Fixed- and Floating-Point Math

5.1.1 Decimal

5.1.2 Formatting

5.1.3 Arithmetic

5.1.4 Special Values

5.1.5 Context

5.2 fractions: Rational Numbers

5.2.1 Creating Fraction Instances

5.2.2 Arithmetic

5.2.3 Approximating Values

5.3 random: Pseudorandom Number Generators

5.3.1 Generating Random Numbers

5.3.2 Seeding

5.3.3 Saving State

5.3.4 Random Integers

5.3.5 Picking Random Items

Table of Contents

5.3.6 Permutations

5.3.7 Sampling

5.3.8 Multiple Simultaneous Generators

5.3.9 SystemRandom

5.3.10 Non-uniform Distributions

5.4 math: Mathematical Functions

5.4.1 Special Constants

5.4.2 Testing for Exceptional Values

5.4.3 Comparing

5.4.4 Converting Floating-Point Values to Integers

5.4.5 Alternative Representations of Floating-Point Values

5.4.6 Positive and Negative Signs

5.4.7 Commonly Used Calculations

5.4.8 Exponents and Logarithms

5.4.9 Angles

5.4.10 Trigonometry

5.4.11 Hyperbolic Functions

5.4.12 Special Functions

5.5 statistics: Statistical Calculations

5.5.1 Averages

5.5.2 Variance

Chapter 6 The File System

6.1 os.path: Platform-Independent Manipulation of Filenames

6.1.1 Parsing Paths

6.1.2 Building Paths

6.1.3 Normalizing Paths

6.1.4 File Times

6.1.5 Testing Files

6.2 pathlib: File System Paths as Objects

Table of Contents

- 6.2.1 Path Representations
- 6.2.2 Building Paths
- 6.2.3 Parsing Paths
- 6.2.4 Creating Concrete Paths
- 6.2.5 Directory Contents
- 6.2.6 Reading and Writing Files
- 6.2.7 Manipulating Directories and Symbolic Links
- 6.2.8 File Types
- 6.2.9 File Properties
- 6.2.10 Permissions
- 6.2.11 Deleting
- 6.3 glob: Filename Pattern Matching
 - 6.3.1 Example Data
 - 6.3.2 Wildcards
 - 6.3.3 Single-Character Wildcard
 - 6.3.4 Character Ranges
 - 6.3.5 Escaping Meta-characters
- 6.4 fnmatch: Unix-Style Glob Pattern Matching
 - 6.4.1 Simple Matching
 - 6.4.2 Filtering
 - 6.4.3 Translating Patterns
- 6.5 linecache: Read Text Files Efficiently
 - 6.5.1 Test Data
 - 6.5.2 Reading Specific Lines
 - 6.5.3 Handling Blank Lines
 - 6.5.4 Error Handling
 - 6.5.5 Reading Python Source Files
- 6.6 tempfile: Temporary File System Objects
 - 6.6.1 Temporary Files
 - 6.6.2 Named Files

Table of Contents

- 6.6.3 Spooled Files
- 6.6.4 Temporary Directories
- 6.6.5 Predicting Names
- 6.6.6 Temporary File Location
- 6.7 shutil: High-Level File Operations
 - 6.7.1 Copying Files
 - 6.7.2 Copying File Metadata
 - 6.7.3 Working with Directory Trees
 - 6.7.4 Finding Files
 - 6.7.5 Archives
 - 6.7.6 File System Space
- 6.8 filecmp: Compare Files
 - 6.8.1 Example Data
 - 6.8.2 Comparing Files
 - 6.8.3 Comparing Directories
 - 6.8.4 Using Differences in a Program
- 6.9 mmap: Memory-Map Files
 - 6.9.1 Reading
 - 6.9.2 Writing
 - 6.9.3 Regular Expressions
- 6.10 codecs: String Encoding and Decoding
 - 6.10.1 Unicode Primer
 - 6.10.2 Working with Files
 - 6.10.3 Byte Order
 - 6.10.4 Error Handling
 - 6.10.5 Encoding Translation
 - 6.10.6 Non-Unicode Encodings
 - 6.10.7 Incremental Encoding
 - 6.10.8 Unicode Data and Network Communication

Table of Contents

6.10.9 Defining a Custom Encoding

6.11 io: Text, Binary, and Raw Stream I/O Tools

6.11.1 In-Memory Streams

6.11.2 Wrapping Byte Streams for Text Data

Chapter 7 Data Persistence and Exchange

7.1 pickle: Object Serialization

7.1.1 Encoding and Decoding Data in Strings

7.1.2 Working with Streams

7.1.3 Problems Reconstructing Objects

7.1.4 Unpicklable Objects

7.1.5 Circular References

7.2 shelf: Persistent Storage of Objects

7.2.1 Creating a New Shelf

7.2.2 Writeback

7.2.3 Specific Shelf Types

7.3 dbm: Unix Key/Value Databases

7.3.1 Database Types

7.3.2 Creating a New Database

7.3.3 Opening an Existing Database

7.3.4 Error Cases

7.4 sqlite3: Embedded Relational Database

7.4.1 Creating a Database

7.4.2 Retrieving Data

7.4.3 Query Metadata

7.4.4 Row Objects

7.4.5 Using Variables with Queries

7.4.6 Bulk Loading

7.4.7 Defining New Column Types

7.4.8 Determining Types for Columns

Table of Contents

7.4.9 Transactions

7.4.10 Isolation Levels

7.4.11 In-Memory Databases

7.4.12 Exporting the Contents of a Database

7.4.13 Using Python Functions in SQL

7.4.14 Querying with Regular Expressions

7.4.15 Custom Aggregation

7.4.16 Threading and Connection Sharing

7.4.17 Restricting Access to Data

7.5 xml.etree.ElementTree: XML Manipulation API

7.5.1 Parsing an XML Document

7.5.2 Traversing the Parsed Tree

7.5.3 Finding Nodes in a Document

7.5.4 Parsed Node Attributes

7.5.5 Watching Events While Parsing

7.5.6 Creating a Custom Tree Builder

7.5.7 Parsing Strings

7.5.8 Building Documents With Element Nodes

7.5.9 Pretty-Printing XML

7.5.10 Setting Element Properties

7.5.11 Building Trees from Lists of Nodes

7.5.12 Serializing XML to a Stream

7.6 csv: Comma-Separated Value Files

7.6.1 Reading

7.6.2 Writing

7.6.3 Dialects

7.6.4 Using Field Names

Chapter 8 Data Compression and Archiving

8.1 zlib: GNU zlib Compression

Table of Contents

8.1.1 Working with Data in Memory

8.1.2 Incremental Compression and Decompression

8.1.3 Mixed Content Streams

8.1.4 Checksums

8.1.5 Compressing Network Data

8.2 gzip: Read and Write GNU zip Files

8.2.1 Writing Compressed Files

8.2.2 Reading Compressed Data

8.2.3 Working with Streams

8.3 bz2: bzip2 Compression

8.3.1 One-Shot Operations in Memory

8.3.2 Incremental Compression and Decompression

8.3.3 Mixed-Content Streams

8.3.4 Writing Compressed Files

8.3.5 Reading Compressed Files

8.3.6 Reading and Writing Unicode Data

8.3.7 Compressing Network Data

8.4 tarfile: Tar Archive Access

8.4.1 Testing Tar Files

8.4.2 Reading Metadata from an Archive

8.4.3 Extracting Files from an Archive

8.4.4 Creating New Archives

8.4.5 Using Alternative Archive Member Names

8.4.6 Writing Data from Sources Other Than Files

8.4.7 Appending to Archives

8.4.8 Working with Compressed Archives

8.5 zipfile: ZIP Archive Access

8.5.1 Testing ZIP Files

8.5.2 Reading Metadata from an Archive

8.5.3 Extracting Archived Files From an Archive

Table of Contents

- 8.5.4 Creating New Archives
- 8.5.5 Using Alternative Archive Member Names
- 8.5.6 Writing Data from Sources Other Than Files
- 8.5.7 Writing with a ZipInfo Instance
- 8.5.8 Appending to Files
- 8.5.9 Python ZIP Archives
- 8.5.10 Limitations

Chapter 9 Cryptography

9.1 hashlib: Cryptographic Hashing

- 9.1.1 Hash Algorithms
- 9.1.2 Sample Data
- 9.1.3 MD5 Example
- 9.1.4 SHA1 Example
- 9.1.5 Creating a Hash by Name
- 9.1.6 Incremental Updates

9.2 hmac: Cryptographic Message Signing and Verification

- 9.2.1 Signing Messages
- 9.2.2 Alternative Digest Types
- 9.2.3 Binary Digests
- 9.2.4 Applications of Message Signatures

Chapter 10 Concurrency with Processes, Threads, and Coroutines

10.1 subprocess: Spawning Additional Processes

- 10.1.1 Running External Command
- 10.1.2 Working with Pipes Directly
- 10.1.3 Connecting Segments of a Pipe
- 10.1.4 Interacting with Another Command
- 10.1.5 Signaling Between Processes

10.2 signal: Asynchronous System Events

Table of Contents

- 10.2.1 Receiving Signals
- 10.2.2 Retrieving Registered Handlers
- 10.2.3 Sending Signals
- 10.2.4 Alarms
- 10.2.5 Ignoring Signals
- 10.2.6 Signals and Threads
- 10.3 threading: Manage Concurrent Operations Within a Process
 - 10.3.1 Thread Objects
 - 10.3.2 Determining the Current Thread
 - 10.3.3 Daemon Versus Non-daemon Threads
 - 10.3.4 Enumerating All Threads
 - 10.3.5 Subclassing Thread
 - 10.3.6 Timer Threads
 - 10.3.7 Signaling Between Threads
 - 10.3.8 Controlling Access to Resources
 - 10.3.9 Synchronizing Threads
 - 10.3.10 Limiting Concurrent Access to Resources
 - 10.3.11 Thread Specific Data
- 10.4 multiprocessing: Manage Processes Like Threads
 - 10.4.1 multiprocessing Basics
 - 10.4.2 Importable Target Functions
 - 10.4.3 Determining the Current Process
 - 10.4.4 Daemon Processes
 - 10.4.5 Waiting for Processes
 - 10.4.6 Terminating Processes
 - 10.4.7 Process Exit Status
 - 10.4.8 Logging
 - 10.4.9 Subclassing Process
 - 10.4.10 Passing Messages to Processes

Table of Contents

- 10.4.11 Signaling Between Processes
- 10.4.12 Controlling Access to Resources
- 10.4.13 Synchronizing Operations
- 10.4.14 Controlling Concurrent Access to Resources
- 10.4.15 Managing Shared State
- 10.4.16 Shared Namespaces
- 10.4.17 Process Pools
- 10.4.18 Implementing MapReduce

10.5 asyncio: Asynchronous I/O, Event Loop, and Concurrency Tools

- 10.5.1 Asynchronous Concurrency Concepts
- 10.5.2 Cooperative Multitasking with Coroutines
- 10.5.3 Scheduling Calls to Regular Functions
- 10.5.4 Producing Results Asynchronously
- 10.5.5 Executing Tasks Concurrently
- 10.5.6 Composing Coroutines with Control Structures
- 10.5.7 Synchronization Primitives
- 10.5.8 Asynchronous I/O with Protocol Class Abstractions
- 10.5.9 Asynchronous I/O Using Coroutines and Streams
- 10.5.10 Using SSL
- 10.5.11 Interacting with Domain Name Services
- 10.5.12 Working with Subprocesses
- 10.5.13 Receiving Unix Signals
- 10.5.14 Combining Coroutines with Threads and Processes
- 10.5.15 Debugging with asyncio

10.6 concurrent.futures: Manage Pools of Concurrent Tasks

- 10.6.1 Using map() with a Basic Thread Pool
- 10.6.2 Scheduling Individual Tasks
- 10.6.3 Waiting for Tasks in Any Order
- 10.6.4 Future Callbacks

Table of Contents

- 10.6.5 Canceling Tasks
- 10.6.6 Exceptions in Tasks
- 10.6.7 Context Manager
- 10.6.8 Process Pools

Chapter 11 Networking

11.1 ipaddress: Internet Addresses

- 11.1.1 Addresses
- 11.1.2 Networks
- 11.1.3 Interfaces

11.2 socket: Network Communication

- 11.2.1 Addressing, Protocol Families, and Socket Types
- 11.2.2 TCP/IP Client and Server
- 11.2.3 User Datagram Client and Server
- 11.2.4 Unix Domain Sockets
- 11.2.5 Multicast
- 11.2.6 Sending Binary Data
- 11.2.7 Non-blocking Communication and Timeouts

11.3 selectors: I/O Multiplexing Abstractions

- 11.3.1 Operating Model
- 11.3.2 Echo Server
- 11.3.3 Echo Client
- 11.3.4 Server and Client Together

11.4 select: Wait for I/O Efficiently

- 11.4.1 Using select()
- 11.4.2 Non-blocking I/O with Timeouts
- 11.4.3 Using poll()
- 11.4.4 Platform-Specific Options

11.5 socketserver: Creating Network Servers

- 11.5.1 Server Types

Table of Contents

- 11.5.2 Server Objects
- 11.5.3 Implementing a Server
- 11.5.4 Request Handlers
- 11.5.5 Echo Example
- 11.5.6 Threading and Forking

Chapter 12 The Internet

- 12.1 urllib.parse: Split URLs into Components
 - 12.1.1 Parsing
 - 12.1.2 Unparsing
 - 12.1.3 Joining
 - 12.1.4 Encoding Query Arguments
- 12.2 urllib.request: Network Resource Access
 - 12.2.1 HTTP GET
 - 12.2.2 Encoding Arguments
 - 12.2.3 HTTP POST
 - 12.2.4 Adding Outgoing Headers
 - 12.2.5 Posting Form Data from a Request
 - 12.2.6 Uploading Files
 - 12.2.7 Creating Custom Protocol Handlers
- 12.3 urllib.robotparser: Internet Spider Access Control
 - 12.3.1 robots.txt
 - 12.3.2 Testing Access Permissions
 - 12.3.3 Long-Lived Spiders
- 12.4 base64: Encode Binary Data with ASCII
 - 12.4.1 Base 64 Encoding
 - 12.4.2 Base64 Decoding
 - 12.4.3 URL-Safe Variations
 - 12.4.4 Other Encodings
- 12.5 http.server: Base Classes for Implementing Web Servers

Table of Contents

- 12.5.1 HTTP GET
- 12.5.2 HTTP POST
- 12.5.3 Threading and Forking
- 12.5.4 Handling Errors
- 12.5.5 Setting Headers
- 12.5.6 Command-Line Use
- 12.6 http.cookies: HTTP Cookies
 - 12.6.1 Creating and Setting a Cookie
 - 12.6.2 Morsels
 - 12.6.3 Encoded Values
 - 12.6.4 Receiving and Parsing Cookie Headers
 - 12.6.5 Alternative Output Formats
- 12.7 webbrowser: Displays Web Pages
 - 12.7.1 Simple Example
 - 12.7.2 Windows Versus Tabs
 - 12.7.3 Using a Specific Browser
 - 12.7.4 BROWSER Variable
 - 12.7.5 Command-Line Interface
- 12.8 uuid: Universally Unique Identifiers
 - 12.8.1 UUID 1: IEEE 802 MAC Address
 - 12.8.2 UUID 3 and 5: Name-Based Values
 - 12.8.3 UUID 4: Random Values
 - 12.8.4 Working with UUID Objects
- 12.9 json: JavaScript Object Notation
 - 12.9.1 Encoding and Decoding Simple Data Types
 - 12.9.2 Human-Consumable Versus Compact Output
 - 12.9.3 Encoding Dictionaries
 - 12.9.4 Working with Custom Types
 - 12.9.5 Encoder and Decoder Classes
 - 12.9.6 Working with Streams and Files

Table of Contents

12.9.7 Mixed Data Streams

12.9.8 JSON at the Command Line

12.10 xmlrpc.client: Client Library for XML-RPC

12.10.1 Connecting to a Server

12.10.2 Data Types

12.10.3 Passing Objects

12.10.4 Binary Data

12.10.5 Exception Handling

12.10.6 Combining Calls into One Message

12.11 xmlrpc.server: An XML-RPC Server

12.11.1 A Simple Server

12.11.2 Alternate API Names

12.11.3 Dotted API Names

12.11.4 Arbitrary API Names

12.11.5 Exposing Methods of Objects

12.11.6 Dispatching Calls

12.11.7 Introspection API

Chapter 13 Email

13.1 smtplib: Simple Mail Transfer Protocol Client

13.1.1 Sending an Email Message

13.1.2 Authentication and Encryption

13.1.3 Verifying an Email Address

13.2 smtpd: Sample Mail Servers

13.2.1 Mail Server Base Class

13.2.2 Debugging Server

13.2.3 Proxy Server

13.3 mailbox: Manipulate Email Archives

13.3.1 mbox

13.3.2 Maildir

Table of Contents

13.3.3 Message Flags

13.3.4 Other Formats

13.4 imaplib: IMAP4 Client Library

13.4.1 Variations

13.4.2 Connecting to a Server

13.4.3 Example Configuration

13.4.4 Listing Mailboxes

13.4.5 Mailbox Status

13.4.6 Selecting a Mailbox

13.4.7 Searching for Messages

13.4.8 Search Criteria

13.4.9 Fetching Messages

13.4.10 Whole Messages

13.4.11 Uploading Messages

13.4.12 Moving and Copying Messages

13.4.13 Deleting Messages

Chapter 14 Application Building Blocks

14.1 argparse: Command-Line Option and Argument Parsing

14.1.1 Setting Up a Parser

14.1.2 Defining Arguments

14.1.3 Parsing a Command Line

14.1.4 Simple Examples

14.1.5 Help Output

14.1.6 Parser Organization

14.1.7 Advanced Argument Processing

14.2 getopt: Command-Line Option Parsing

14.2.1 Function Arguments

14.2.2 Short-Form Options

14.2.3 Long-Form Options

Table of Contents

- 14.2.4 A Complete Example
- 14.2.5 Abbreviating Long-Form Options
- 14.2.6 GNU-Style Option Parsing
- 14.2.7 Ending Argument Processing
- 14.3 readline: The GNU readline Library
 - 14.3.1 Configuring readline
 - 14.3.2 Completing Text
 - 14.3.3 Accessing the Completion Buffer
 - 14.3.4 Input History
 - 14.3.5 Hooks
- 14.4 getpass: Secure Password Prompt
 - 14.4.1 Example
 - 14.4.2 Using getpass Without a Terminal
- 14.5 cmd: Line-Oriented Command Processors
 - 14.5.1 Processing Commands
 - 14.5.2 Command Arguments
 - 14.5.3 Live Help
 - 14.5.4 Auto-Completion
 - 14.5.5 Overriding Base Class Methods
 - 14.5.6 Configuring Cmd Through Attributes
 - 14.5.7 Running Shell Commands
 - 14.5.8 Alternative Inputs
 - 14.5.9 Commands from sys.argv
- 14.6 shlex: Parse Shell-Style Syntaxes
 - 14.6.1 Parsing Quoted Strings
 - 14.6.2 Making Safe Strings for Shells
 - 14.6.3 Embedded Comments
 - 14.6.4 Splitting Strings into Tokens
 - 14.6.5 Including Other Sources of Tokens

Table of Contents

- 14.6.6 Controlling the Parser
- 14.6.7 Error Handling
- 14.6.8 POSIX Versus Non-POSIX Parsing
- 14.7 configparser: Work with Configuration Files
 - 14.7.1 Configuration File Format
 - 14.7.2 Reading Configuration Files
 - 14.7.3 Accessing Configuration Settings
 - 14.7.4 Modifying Settings
 - 14.7.5 Saving Configuration Files
 - 14.7.6 Option Search Path
 - 14.7.7 Combining Values with Interpolation
- 14.8 logging: Report Status, Error, and Informational Messages
 - 14.8.1 Logging Components
 - 14.8.2 Logging in Applications Versus Libraries
 - 14.8.3 Logging to a File
 - 14.8.4 Rotating Log Files
 - 14.8.5 Verbosity Levels
 - 14.8.6 Naming Logger Instances
 - 14.8.7 The Logging Tree
 - 14.8.8 Integration with the warnings Module
- 14.9 fileinput: Command-Line Filter Framework
 - 14.9.1 Converting M3U Files to RSS
 - 14.9.2 Progress Metadata
 - 14.9.3 In-Place Filtering
- 14.10 atexit: Program Shutdown Callbacks
 - 14.10.1 Registering Exit Callbacks
 - 14.10.2 Decorator Syntax
 - 14.10.3 Canceling Callbacks
 - 14.10.4 When Are atexit Callbacks Not Called?

Table of Contents

14.10.5 Handling Exceptions

14.11 sched: Timed Event Scheduler

14.11.1 Running Events with a Delay

14.11.2 Overlapping Events

14.11.3 Event Priorities

14.11.4 Canceling Events

Chapter 15 Internationalization and Localization

15.1 gettext: Message Catalogs

15.1.1 Translation Workflow Overview

15.1.2 Creating Message Catalogs from Source Code

15.1.3 Finding Message Catalogs at Runtime

15.1.4 Plural Values

15.1.5 Application Versus Module Localization

15.1.6 Switching Translations

15.2 locale: Cultural Localization API

15.2.1 Probing the Current Locale

15.2.2 Currency

15.2.3 Formatting Numbers

15.2.4 Parsing Numbers

15.2.5 Dates and Times

Chapter 16 Developer Tools

16.1 pydoc: Online Help for Modules

16.1.1 Plain Text Help

16.1.2 HTML Help

16.1.3 Interactive Help

16.2 doctest: Testing Through Documentation

16.2.1 Getting Started

16.2.2 Handling Unpredictable Output

16.2.3 Tracebacks

Table of Contents

- 16.2.4 Working Around Whitespace
- 16.2.5 Test Locations
- 16.2.6 External Documentation
- 16.2.7 Running Tests
- 16.2.8 Test Context
- 16.3 unittest: Automated Testing Framework
 - 16.3.1 Basic Test Structure
 - 16.3.2 Running Tests
 - 16.3.3 Test Outcomes
 - 16.3.4 Asserting Truth
 - 16.3.5 Testing Equality
 - 16.3.6 Almost Equal?
 - 16.3.7 Containers
 - 16.3.8 Testing for Exceptions
 - 16.3.9 Test Fixtures
 - 16.3.10 Repeating Tests with Different Inputs
 - 16.3.11 Skipping Tests
 - 16.3.12 Ignoring Failing Tests
- 16.4 trace: Follow Program Flow
 - 16.4.1 Example Program
 - 16.4.2 Tracing Execution
 - 16.4.3 Code Coverage
 - 16.4.4 Calling Relationships
 - 16.4.5 Programming Interface
 - 16.4.6 Saving Result Data
 - 16.4.7 Options
- 16.5 traceback: Exceptions and Stack Traces
 - 16.5.1 Supporting Functions
 - 16.5.2 Examining the Stack
 - 16.5.3 TracebackException

Table of Contents

16.5.4 Low-Level Exception APIs

16.5.5 Low-Level Stack APIs

16.6 cgitb: Detailed Traceback Reports

16.6.1 Standard Traceback Dumps

16.6.2 Enabling Detailed Tracebacks

16.6.3 Local Variables in Tracebacks

16.6.4 Exception Properties

16.6.5 HTML Output

16.6.6 Logging Tracebacks

16.7 pdb: Interactive Debugger

16.7.1 Starting the Debugger

16.7.2 Controlling the Debugger

16.7.3 Breakpoints

16.7.4 Changing Execution Flow

16.7.5 Customizing the Debugger with Aliases

16.7.6 Saving Configuration Settings

16.8 profile and pstats: Performance Analysis

16.8.1 Running the Profiler

16.8.2 Running in a Context

16.8.3 pstats: Saving and Working with Statistics

16.8.4 Limiting Report Contents

16.8.5 Caller/Callee Graphs

16.9 timeit: Time the Execution of Small Bits of Python Code

16.9.1 Module Contents

16.9.2 Basic Example

16.9.3 Storing Values in a Dictionary

16.9.4 From the Command Line

16.10 tabnanny: Indentation Validator

16.10.1 Running from the Command Line

Table of Contents

16.11 compileall: Byte-Compile Source Files

16.11.1 Compiling One Directory

16.11.2 Ignoring Files

16.11.3 Compiling sys.path

16.11.4 Compiling Individual Files

16.11.5 From the Command Line

16.12 pycbr: Class Browser

16.12.1 Scanning for Classes

16.12.2 Scanning for Functions

16.13 venv: Create Virtual Environments

16.13.1 Creating Environments

16.13.2 Contents of a Virtual Environment

16.13.3 Using Virtual Environments

16.14 ensurepip: Install the Python Package Installer

16.14.1 Installing pip

Chapter 17 Runtime Features

17.1 site: Site-wide Configuration

17.1.1 Import Path

17.1.2 User Directories

17.1.3 Path Configuration Files

17.1.4 Customizing Site Configuration

17.1.5 Customizing User Configuration

17.1.6 Disabling the site Module

17.2 sys: System-Specific Configuration

17.2.1 Interpreter Settings

17.2.2 Runtime Environment

17.2.3 Memory Management and Limits

17.2.4 Exception Handling

17.2.5 Low-Level Thread Support

Table of Contents

17.2.6 Modules and Imports

17.2.7 Tracing a Program As It Runs

17.3 os: Portable Access to Operating System Specific Features

17.3.1 Examining the File System Contents

17.3.2 Managing File System Permissions

17.3.3 Creating and Deleting Directories

17.3.4 Working with Symbolic Links

17.3.5 Safely Replacing an Existing File

17.3.6 Detecting and Changing the Process Owner

17.3.7 Managing the Process Environment

17.3.8 Managing the Process Working Directory

17.3.9 Running External Commands

17.3.10 Creating Processes with `os.fork()`

17.3.11 Waiting for Child Processes

17.3.12 Spawning New Processes

17.3.13 Operating System Error Codes

17.4 platform: System Version Information

17.4.1 Interpreter

17.4.2 Platform

17.4.3 Operating System and Hardware Information

17.4.4 Executable Architecture

17.5 resource: System Resource Management

17.5.1 Current Usage

17.5.2 Resource Limits

17.6 gc: Garbage Collector

17.6.1 Tracing References

17.6.2 Forcing Garbage Collection

17.6.3 Finding References to Objects That Cannot Be Collected

17.6.4 Collection Thresholds and Generations

Table of Contents

17.6.5 Debugging

17.7 sysconfig: Interpreter Compile-Time Configuration

17.7.1 Configuration Variables

17.7.2 Installation Paths

17.7.3 Python Version and Platform

Chapter 18 Language Tools

18.1 warnings: Non-fatal Alerts

18.1.1 Categories and Filtering

18.1.2 Generating Warnings

18.1.3 Filtering with Patterns

18.1.4 Repeated Warnings

18.1.5 Alternative Message Delivery Functions

18.1.6 Formatting

18.1.7 Stack Level in Warnings

18.2 abc: Abstract Base Classes

18.2.1 How ABCs Work

18.2.2 Registering a Concrete Class

18.2.3 Implementation Through Subclassing

18.2.4 Helper Base Class

18.2.5 Incomplete Implementations

18.2.6 Concrete Methods in ABCs

18.2.7 Abstract Properties

18.2.8 Abstract Class and Static Methods

18.3 dis: Python Byte-Code Disassembler

18.3.1 Basic Disassembly

18.3.2 Disassembling Functions

18.3.3 Classes

18.3.4 Source Code

18.3.5 Using Disassembly to Debug

Table of Contents

18.3.6 Performance Analysis of Loops

18.3.7 Compiler Optimizations

18.4 inspect: Inspect Live Objects

18.4.1 Example Module

18.4.2 Inspecting Modules

18.4.3 Inspecting Classes

18.4.4 Inspecting Instances

18.4.5 Documentation Strings

18.4.6 Retrieving Source

18.4.7 Method and Function Signatures

18.4.8 Class Hierarchies

18.4.9 Method Resolution Order

18.4.10 The Stack and Frames

18.4.11 Command-Line Interface

Chapter 19 Modules and Packages

19.1 importlib: Python's Import Mechanism

19.1.1 Example Package

19.1.2 Module Types

19.1.3 Importing Modules

19.1.4 Loaders

19.2 pkgutil: Package Utilities

19.2.1 Package Import Paths

19.2.2 Development Versions of Packages

19.2.3 Managing Paths with PKG Files

19.2.4 Nested Packages

19.2.5 Package Data

19.3 zipimport: Load Python Code from ZIP Archives

19.3.1 Example

19.3.2 Finding a Module

Table of Contents

19.3.3 Accessing Code

19.3.4 Source

19.3.5 Packages

19.3.6 Data

Appendix A: Porting Notes

A.1 References

A.2 New Modules

A.3 Renamed Modules

A.4 Removed Modules

A.4.1 bsddb

A.4.2 commands

A.4.3 compiler

A.4.4 dircache

A.4.5 EasyDialogs

A.4.6 exceptions

A.4.7 htmllib

A.4.8 md5

A.4.9 mimetools, MimeWriter, mimify, multifile, and rfc822

A.4.10 popen2

A.4.11 posixfile

A.4.12 sets

A.4.13 sha

A.4.14 sre

A.4.15 statvfs

A.4.16 thread

A.4.17 user

A.5 Deprecated Modules

A.5.1 asyncore and asynchat

A.5.2 formatter

Table of Contents

A.5.3 imp

A.5.4 optparse

A.6 Summary of Changes to Modules

A.6.1 abc

A.6.2 anydbm

A.6.3 argparse

A.6.4 array

A.6.5 atexit

A.6.6 base64

A.6.7 bz2

A.6.8 collections

A.6.9 comands

A.6.10 configparser

A.6.11 contextlib

A.6.12 csv

A.6.13 datetime

A.6.14 decimal

A.6.15 fractions

A.6.16 gc

A.6.17 gettext

A.6.18 glob

A.6.19 http.cookies

A.6.20 imaplib

A.6.21 inspect

A.6.22 itertools

A.6.23 json

A.6.24 locale

A.6.25 logging

A.6.26 mailbox

A.6.27 mmap

Table of Contents

- A.6.28 operator
- A.6.29 os
- A.6.30 os.path
- A.6.31 pdb
- A.6.32 pickle
- A.6.33 pipes
- A.6.34 platform
- A.6.35 random
- A.6.36 re
- A.6.37 shelve
- A.6.38 signal
- A.6.39 socket
- A.6.40 socketserver
- A.6.41 string
- A.6.42 struct
- A.6.43 subprocess
- A.6.44 sys
- A.6.45 threading
- A.6.46 time
- A.6.47 unittest
- A.6.48 UserDict, UserList, and UserString
- A.6.49 uuid
- A.6.50 whichdb
- A.6.51 xml.etree.ElementTree
- A.6.52 zipimport

Appendix B: Outside of the Standard Library

B.1 Text

B.2 Algorithms

B.3 Dates and Times

Table of Contents

B.4 Mathematics

B.5 Data Persistence and Exchange

B.6 Cryptography

B.7 Concurrency with Processes, Threads, and Coroutines

B.8 The Internet

B.9 Email

B.10 Application Building Blocks

B.11 Developer Tools

Index of Python Modules

A

B

C

D

E

F

G

H

I

J

L

M

O

P

Q

R

S

T

Table of Contents

U

V

W

X

Z

Index