

Table of Contents

Preface.....	xi
1. Data Structures and Algorithms.....	1
1.1. Unpacking a Sequence into Separate Variables	1
1.2. Unpacking Elements from Iterables of Arbitrary Length	3
1.3. Keeping the Last N Items	5
1.4. Finding the Largest or Smallest N Items	7
1.5. Implementing a Priority Queue	8
1.6. Mapping Keys to Multiple Values in a Dictionary	11
1.7. Keeping Dictionaries in Order	12
1.8. Calculating with Dictionaries	13
1.9. Finding Commonalities in Two Dictionaries	15
1.10. Removing Duplicates from a Sequence while Maintaining Order	17
1.11. Naming a Slice	18
1.12. Determining the Most Frequently Occurring Items in a Sequence	20
1.13. Sorting a List of Dictionaries by a Common Key	21
1.14. Sorting Objects Without Native Comparison Support	23
1.15. Grouping Records Together Based on a Field	24
1.16. Filtering Sequence Elements	26
1.17. Extracting a Subset of a Dictionary	28
1.18. Mapping Names to Sequence Elements	29
1.19. Transforming and Reducing Data at the Same Time	32
1.20. Combining Multiple Mappings into a Single Mapping	33
2. Strings and Text.....	37
2.1. Splitting Strings on Any of Multiple Delimiters	37
2.2. Matching Text at the Start or End of a String	38
2.3. Matching Strings Using Shell Wildcard Patterns	40
2.4. Matching and Searching for Text Patterns	42

2.5. Searching and Replacing Text	45
2.6. Searching and Replacing Case-Insensitive Text	46
2.7. Specifying a Regular Expression for the Shortest Match	47
2.8. Writing a Regular Expression for Multiline Patterns	48
2.9. Normalizing Unicode Text to a Standard Representation	50
2.10. Working with Unicode Characters in Regular Expressions	52
2.11. Stripping Unwanted Characters from Strings	53
2.12. Sanitizing and Cleaning Up Text	54
2.13. Aligning Text Strings	57
2.14. Combining and Concatenating Strings	58
2.15. Interpolating Variables in Strings	61
2.16. Reformatting Text to a Fixed Number of Columns	64
2.17. Handling HTML and XML Entities in Text	65
2.18. Tokenizing Text	66
2.19. Writing a Simple Recursive Descent Parser	69
2.20. Performing Text Operations on Byte Strings	78
3. Numbers, Dates, and Times.	83
3.1. Rounding Numerical Values	83
3.2. Performing Accurate Decimal Calculations	84
3.3. Formatting Numbers for Output	87
3.4. Working with Binary, Octal, and Hexadecimal Integers	89
3.5. Packing and Unpacking Large Integers from Bytes	90
3.6. Performing Complex-Valued Math	92
3.7. Working with Infinity and NaNs	94
3.8. Calculating with Fractions	96
3.9. Calculating with Large Numerical Arrays	97
3.10. Performing Matrix and Linear Algebra Calculations	100
3.11. Picking Things at Random	102
3.12. Converting Days to Seconds, and Other Basic Time Conversions	104
3.13. Determining Last Friday's Date	106
3.14. Finding the Date Range for the Current Month	107
3.15. Converting Strings into Datetimes	109
3.16. Manipulating Dates Involving Time Zones	110
4. Iterators and Generators.	113
4.1. Manually Consuming an Iterator	113
4.2. Delegating Iteration	114
4.3. Creating New Iteration Patterns with Generators	115
4.4. Implementing the Iterator Protocol	117
4.5. Iterating in Reverse	119
4.6. Defining Generator Functions with Extra State	120

4.7. Taking a Slice of an Iterator	122
4.8. Skipping the First Part of an Iterable	123
4.9. Iterating Over All Possible Combinations or Permutations	125
4.10. Iterating Over the Index-Value Pairs of a Sequence	127
4.11. Iterating Over Multiple Sequences Simultaneously	129
4.12. Iterating on Items in Separate Containers	131
4.13. Creating Data Processing Pipelines	132
4.14. Flattening a Nested Sequence	135
4.15. Iterating in Sorted Order Over Merged Sorted Iterables	136
4.16. Replacing Infinite while Loops with an Iterator	138
5. Files and I/O.....	141
5.1. Reading and Writing Text Data	141
5.2. Printing to a File	144
5.3. Printing with a Different Separator or Line Ending	144
5.4. Reading and Writing Binary Data	145
5.5. Writing to a File That Doesn't Already Exist	147
5.6. Performing I/O Operations on a String	148
5.7. Reading and Writing Compressed Datafiles	149
5.8. Iterating Over Fixed-Sized Records	151
5.9. Reading Binary Data into a Mutable Buffer	152
5.10. Memory Mapping Binary Files	153
5.11. Manipulating Pathnames	156
5.12. Testing for the Existence of a File	157
5.13. Getting a Directory Listing	158
5.14. Bypassing Filename Encoding	160
5.15. Printing Bad Filenames	161
5.16. Adding or Changing the Encoding of an Already Open File	163
5.17. Writing Bytes to a Text File	165
5.18. Wrapping an Existing File Descriptor As a File Object	166
5.19. Making Temporary Files and Directories	167
5.20. Communicating with Serial Ports	170
5.21. Serializing Python Objects	171
6. Data Encoding and Processing.....	175
6.1. Reading and Writing CSV Data	175
6.2. Reading and Writing JSON Data	179
6.3. Parsing Simple XML Data	183
6.4. Parsing Huge XML Files Incrementally	186
6.5. Turning a Dictionary into XML	189
6.6. Parsing, Modifying, and Rewriting XML	191
6.7. Parsing XML Documents with Namespaces	193

6.8. Interacting with a Relational Database	195
6.9. Decoding and Encoding Hexadecimal Digits	197
6.10. Decoding and Encoding Base64	199
6.11. Reading and Writing Binary Arrays of Structures	199
6.12. Reading Nested and Variable-Sized Binary Structures	203
6.13. Summarizing Data and Performing Statistics	214
7. Functions.....	217
7.1. Writing Functions That Accept Any Number of Arguments	217
7.2. Writing Functions That Only Accept Keyword Arguments	219
7.3. Attaching Informational Metadata to Function Arguments	220
7.4. Returning Multiple Values from a Function	221
7.5. Defining Functions with Default Arguments	222
7.6. Defining Anonymous or Inline Functions	224
7.7. Capturing Variables in Anonymous Functions	225
7.8. Making an N-Argument Callable Work As a Callable with Fewer Arguments	227
7.9. Replacing Single Method Classes with Functions	231
7.10. Carrying Extra State with Callback Functions	232
7.11. Inlining Callback Functions	235
7.12. Accessing Variables Defined Inside a Closure	238
8. Classes and Objects.....	243
8.1. Changing the String Representation of Instances	243
8.2. Customizing String Formatting	245
8.3. Making Objects Support the Context-Management Protocol	246
8.4. Saving Memory When Creating a Large Number of Instances	248
8.5. Encapsulating Names in a Class	250
8.6. Creating Managed Attributes	251
8.7. Calling a Method on a Parent Class	256
8.8. Extending a Property in a Subclass	260
8.9. Creating a New Kind of Class or Instance Attribute	264
8.10. Using Lazily Computed Properties	267
8.11. Simplifying the Initialization of Data Structures	270
8.12. Defining an Interface or Abstract Base Class	274
8.13. Implementing a Data Model or Type System	277
8.14. Implementing Custom Containers	283
8.15. Delegating Attribute Access	287
8.16. Defining More Than One Constructor in a Class	291
8.17. Creating an Instance Without Invoking <i>init</i>	293
8.18. Extending Classes with Mixins	294
8.19. Implementing Stateful Objects or State Machines	299

8.20. Calling a Method on an Object Given the Name As a String	305
8.21. Implementing the Visitor Pattern	306
8.22. Implementing the Visitor Pattern Without Recursion	311
8.23. Managing Memory in Cyclic Data Structures	317
8.24. Making Classes Support Comparison Operations	321
8.25. Creating Cached Instances	323
9. Metaprogramming.....	329
9.1. Putting a Wrapper Around a Function	329
9.2. Preserving Function Metadata When Writing Decorators	331
9.3. Unwrapping a Decorator	333
9.4. Defining a Decorator That Takes Arguments	334
9.5. Defining a Decorator with User Adjustable Attributes	336
9.6. Defining a Decorator That Takes an Optional Argument	339
9.7. Enforcing Type Checking on a Function Using a Decorator	341
9.8. Defining Decorators As Part of a Class	345
9.9. Defining Decorators As Classes	347
9.10. Applying Decorators to Class and Static Methods	350
9.11. Writing Decorators That Add Arguments to Wrapped Functions	352
9.12. Using Decorators to Patch Class Definitions	355
9.13. Using a Metaclass to Control Instance Creation	356
9.14. Capturing Class Attribute Definition Order	359
9.15. Defining a Metaclass That Takes Optional Arguments	362
9.16. Enforcing an Argument Signature on *args and **kwargs	364
9.17. Enforcing Coding Conventions in Classes	367
9.18. Defining Classes Programmatically	370
9.19. Initializing Class Members at Definition Time	374
9.20. Implementing Multiple Dispatch with Function Annotations	376
9.21. Avoiding Repetitive Property Methods	382
9.22. Defining Context Managers the Easy Way	384
9.23. Executing Code with Local Side Effects	386
9.24. Parsing and Analyzing Python Source	389
9.25. Disassembling Python Byte Code	393
10. Modules and Packages.....	397
10.1. Making a Hierarchical Package of Modules	397
10.2. Controlling the Import of Everything	398
10.3. Importing Package Submodules Using Relative Names	399
10.4. Splitting a Module into Multiple Files	401
10.5. Making Separate Directories of Code Import Under a Common Namespace	404
10.6. Reloading Modules	406

10.7. Making a Directory or Zip File Runnable As a Main Script	407
10.8. Reading Datafiles Within a Package	408
10.9. Adding Directories to sys.path	409
10.10. Importing Modules Using a Name Given in a String	411
10.11. Loading Modules from a Remote Machine Using Import Hooks	412
10.12. Patching Modules on Import	428
10.13. Installing Packages Just for Yourself	431
10.14. Creating a New Python Environment	432
10.15. Distributing Packages	433
11. Network and Web Programming.....	437
11.1. Interacting with HTTP Services As a Client	437
11.2. Creating a TCP Server	441
11.3. Creating a UDP Server	445
11.4. Generating a Range of IP Addresses from a CIDR Address	447
11.5. Creating a Simple REST-Based Interface	449
11.6. Implementing a Simple Remote Procedure Call with XML-RPC	454
11.7. Communicating Simply Between Interpreters	456
11.8. Implementing Remote Procedure Calls	458
11.9. Authenticating Clients Simply	461
11.10. Adding SSL to Network Services	464
11.11. Passing a Socket File Descriptor Between Processes	470
11.12. Understanding Event-Driven I/O	475
11.13. Sending and Receiving Large Arrays	481
12. Concurrency.....	485
12.1. Starting and Stopping Threads	485
12.2. Determining If a Thread Has Started	488
12.3. Communicating Between Threads	491
12.4. Locking Critical Sections	497
12.5. Locking with Deadlock Avoidance	500
12.6. Storing Thread-Specific State	504
12.7. Creating a Thread Pool	505
12.8. Performing Simple Parallel Programming	509
12.9. Dealing with the GIL (and How to Stop Worrying About It)	513
12.10. Defining an Actor Task	516
12.11. Implementing Publish/Subscribe Messaging	520
12.12. Using Generators As an Alternative to Threads	524
12.13. Polling Multiple Thread Queues	532
12.14. Launching a Daemon Process on Unix	535
13. Utility Scripting and System Administration.....	539

13.1. Accepting Script Input via Redirection, Pipes, or Input Files	539
13.2. Terminating a Program with an Error Message	540
13.3. Parsing Command-Line Options	541
13.4. Prompting for a Password at Runtime	544
13.5. Getting the Terminal Size	545
13.6. Executing an External Command and Getting Its Output	545
13.7. Copying or Moving Files and Directories	547
13.8. Creating and Unpacking Archives	549
13.9. Finding Files by Name	550
13.10. Reading Configuration Files	552
13.11. Adding Logging to Simple Scripts	555
13.12. Adding Logging to Libraries	558
13.13. Making a Stopwatch Timer	559
13.14. Putting Limits on Memory and CPU Usage	561
13.15. Launching a Web Browser	563
14. Testing, Debugging, and Exceptions.	565
14.1. Testing Output Sent to stdout	565
14.2. Patching Objects in Unit Tests	567
14.3. Testing for Exceptional Conditions in Unit Tests	570
14.4. Logging Test Output to a File	572
14.5. Skipping or Anticipating Test Failures	573
14.6. Handling Multiple Exceptions	574
14.7. Catching All Exceptions	576
14.8. Creating Custom Exceptions	578
14.9. Raising an Exception in Response to Another Exception	580
14.10. Reraising the Last Exception	582
14.11. Issuing Warning Messages	583
14.12. Debugging Basic Program Crashes	585
14.13. Profiling and Timing Your Program	587
14.14. Making Your Programs Run Faster	590
15. C Extensions.	597
15.1. Accessing C Code Using ctypes	599
15.2. Writing a Simple C Extension Module	605
15.3. Writing an Extension Function That Operates on Arrays	609
15.4. Managing Opaque Pointers in C Extension Modules	612
15.5. Defining and Exporting C APIs from Extension Modules	614
15.6. Calling Python from C	619
15.7. Releasing the GIL in C Extensions	625
15.8. Mixing Threads from C and Python	625
15.9. Wrapping C Code with Swig	627

15.10. Wrapping Existing C Code with Cython	632
15.11. Using Cython to Write High-Performance Array Operations	638
15.12. Turning a Function Pointer into a Callable	643
15.13. Passing NULL-Terminated Strings to C Libraries	644
15.14. Passing Unicode Strings to C Libraries	648
15.15. Converting C Strings to Python	653
15.16. Working with C Strings of Dubious Encoding	654
15.17. Passing Filenames to C Extensions	657
15.18. Passing Open Files to C Extensions	658
15.19. Reading File-Like Objects from C	659
15.20. Consuming an Iterable from C	662
15.21. Diagnosing Segmentation Faults	663
A. Further Reading.....	665
Index.....	667