

Table of Contents

Preface.....	xix
--------------	-----

Part I. Data Structures

1. The Python Data Model.....	3
What's New in This Chapter	4
A Pythonic Card Deck	5
How Special Methods Are Used	8
Emulating Numeric Types	9
String Representation	12
Boolean Value of a Custom Type	13
Collection API	14
Overview of Special Methods	15
Why len Is Not a Method	17
Chapter Summary	18
Further Reading	18
2. An Array of Sequences.....	21
What's New in This Chapter	22
Overview of Built-In Sequences	22
List Comprehensions and Generator Expressions	25
List Comprehensions and Readability	25
Listcomps Versus map and filter	27
Cartesian Products	27
Generator Expressions	29
Tuples Are Not Just Immutable Lists	30
Tuples as Records	30

Table of Contents

Preface.....	xix
--------------	-----

Part I. Data Structures

1. The Python Data Model.....	3
What's New in This Chapter	4
A Pythonic Card Deck	5
How Special Methods Are Used	8
Emulating Numeric Types	9
String Representation	12
Boolean Value of a Custom Type	13
Collection API	14
Overview of Special Methods	15
Why len Is Not a Method	17
Chapter Summary	18
Further Reading	18
2. An Array of Sequences.....	21
What's New in This Chapter	22
Overview of Built-In Sequences	22
List Comprehensions and Generator Expressions	25
List Comprehensions and Readability	25
Listcomps Versus map and filter	27
Cartesian Products	27
Generator Expressions	29
Tuples Are Not Just Immutable Lists	30
Tuples as Records	30

Tuples as Immutable Lists	32
Comparing Tuple and List Methods	34
Unpacking Sequences and Iterables	35
Using * to Grab Excess Items	36
Unpacking with * in Function Calls and Sequence Literals	37
Nested Unpacking	37
Pattern Matching with Sequences	39
Pattern Matching Sequences in an Interpreter	43
Slicing	47
Why Slices and Ranges Exclude the Last Item	47
Slice Objects	48
Multidimensional Slicing and Ellipsis	49
Assigning to Slices	50
Using + and * with Sequences	50
Building Lists of Lists	51
Augmented Assignment with Sequences	53
A += Assignment Puzzle	54
list.sort Versus the sorted Built-In	56
When a List Is Not the Answer	59
Arrays	59
Memory Views	62
NumPy	64
Deque and Other Queues	67
Chapter Summary	70
Further Reading	71
3. Dictionaries and Sets.....	77
What's New in This Chapter	78
Modern dict Syntax	78
dict Comprehensions	79
Unpacking Mappings	80
Merging Mappings with	80
Pattern Matching with Mappings	81
Standard API of Mapping Types	83
What Is Hashable	84
Overview of Common Mapping Methods	85
Inserting or Updating Mutable Values	87
Automatic Handling of Missing Keys	90
defaultdict: Another Take on Missing Keys	90
The __missing__ Method	91
Inconsistent Usage of __missing__ in the Standard Library	94
Variations of dict	95

collections.OrderedDict	95
collections.ChainMap	95
collections.Counter	96
shelve.Shelf	97
Subclassing UserDict Instead of dict	97
Immutable Mappings	99
Dictionary Views	101
Practical Consequences of How dict Works	102
Set Theory	103
Set Literals	105
Set Comprehensions	106
Practical Consequences of How Sets Work	107
Set Operations	107
Set Operations on dict Views	110
Chapter Summary	112
Further Reading	113
4. Unicode Text Versus Bytes.	117
What's New in This Chapter	118
Character Issues	118
Byte Essentials	120
Basic Encoders/Decoders	123
Understanding Encode/Decode Problems	125
Coping with UnicodeEncodeError	125
Coping with UnicodeDecodeError	126
SyntaxError When Loading Modules with Unexpected Encoding	128
How to Discover the Encoding of a Byte Sequence	128
BOM: A Useful Gremlin	129
Handling Text Files	131
Beware of Encoding Defaults	134
Normalizing Unicode for Reliable Comparisons	140
Case Folding	142
Utility Functions for Normalized Text Matching	143
Extreme "Normalization": Taking Out Diacritics	144
Sorting Unicode Text	148
Sorting with the Unicode Collation Algorithm	150
The Unicode Database	151
Finding Characters by Name	151
Numeric Meaning of Characters	153
Dual-Mode str and bytes APIs	155
str Versus bytes in Regular Expressions	155
str Versus bytes in os Functions	157

Chapter Summary	157
Further Reading	158
5. Data Class Builders.....	163
What's New in This Chapter	164
Overview of Data Class Builders	164
Main Features	167
Classic Named Tuples	169
Typed Named Tuples	172
Type Hints 101	173
No Runtime Effect	173
Variable Annotation Syntax	174
The Meaning of Variable Annotations	175
More About @dataclass	179
Field Options	180
Post-init Processing	183
Typed Class Attributes	185
Initialization Variables That Are Not Fields	186
@dataclass Example: Dublin Core Resource Record	187
Data Class as a Code Smell	190
Data Class as Scaffolding	191
Data Class as Intermediate Representation	191
Pattern Matching Class Instances	192
Simple Class Patterns	192
Keyword Class Patterns	193
Positional Class Patterns	194
Chapter Summary	195
Further Reading	196
6. Object References, Mutability, and Recycling.....	201
What's New in This Chapter	202
Variables Are Not Boxes	202
Identity, Equality, and Aliases	204
Choosing Between == and is	206
The Relative Immutability of Tuples	207
Copies Are Shallow by Default	208
Deep and Shallow Copies of Arbitrary Objects	211
Function Parameters as References	213
Mutable Types as Parameter Defaults: Bad Idea	214
Defensive Programming with Mutable Parameters	216
del and Garbage Collection	219
Tricks Python Plays with Immutables	221

Chapter Summary	223
Further Reading	224

Part II. Functions as Objects

7. Functions as First-Class Objects.	231
What's New in This Chapter	232
Treating a Function Like an Object	232
Higher-Order Functions	234
Modern Replacements for map, filter, and reduce	235
Anonymous Functions	236
The Nine Flavors of Callable Objects	237
User-Defined Callable Types	239
From Positional to Keyword-Only Parameters	240
Positional-Only Parameters	242
Packages for Functional Programming	243
The operator Module	243
Freezing Arguments with functools.partial	247
Chapter Summary	249
Further Reading	250
8. Type Hints in Functions.	253
What's New in This Chapter	254
About Gradual Typing	254
Gradual Typing in Practice	255
Starting with Mypy	256
Making Mypy More Strict	257
A Default Parameter Value	258
Using None as a Default	260
Types Are Defined by Supported Operations	261
Types Usable in Annotations	266
The Any Type	266
Simple Types and Classes	269
Optional and Union Types	270
Generic Collections	271
Tuple Types	274
Generic Mappings	277
Abstract Base Classes	278
Iterable	280
Parameterized Generics and TypeVar	282
Static Protocols	287

Callable	292
NoReturn	295
Annotating Positional Only and Variadic Parameters	295
Imperfect Typing and Strong Testing	296
Chapter Summary	298
Further Reading	299
9. Decorators and Closures.....	305
What's New in This Chapter	306
Decorators 101	306
When Python Executes Decorators	308
Registration Decorators	310
Variable Scope Rules	310
Closures	313
The nonlocal Declaration	317
Variable Lookup Logic	318
Implementing a Simple Decorator	319
How It Works	320
Decorators in the Standard Library	322
Memoization with functools.cache	322
Using lru_cache	325
Single Dispatch Generic Functions	326
Parameterized Decorators.....	331
A Parameterized Registration Decorator	331
The Parameterized Clock Decorator	334
A Class-Based Clock Decorator	337
Chapter Summary	338
Further Reading	338
10. Design Patterns with First-Class Functions.....	343
What's New in This Chapter	344
Case Study: Refactoring Strategy	344
Classic Strategy	344
Function-Oriented Strategy	349
Choosing the Best Strategy: Simple Approach	352
Finding Strategies in a Module	353
Decorator-Enhanced Strategy Pattern	355
The Command Pattern	357
Chapter Summary	359
Further Reading	360

Part III. Classes and Protocols

11. A Pythonic Object.....	365
What's New in This Chapter	366
Object Representations	366
Vector Class Redux	367
An Alternative Constructor	370
classmethod Versus staticmethod	371
Formatted Displays	372
A Hashable Vector2d	376
Supporting Positional Pattern Matching	379
Complete Listing of Vector2d, Version 3	380
Private and "Protected" Attributes in Python	384
Saving Memory with <code>__slots__</code>	386
Simple Measure of <code>__slot__</code> Savings	389
Summarizing the Issues with <code>__slots__</code>	390
Overriding Class Attributes	391
Chapter Summary	393
Further Reading	394
12. Special Methods for Sequences.....	399
What's New in This Chapter	400
Vector: A User-Defined Sequence Type	400
Vector Take #1: Vector2d Compatible	401
Protocols and Duck Typing	404
Vector Take #2: A Sliceable Sequence	405
How Slicing Works	406
A Slice-Aware <code>__getitem__</code>	408
Vector Take #3: Dynamic Attribute Access	409
Vector Take #4: Hashing and a Faster <code>==</code>	413
Vector Take #5: Formatting	420
Chapter Summary	427
Further Reading	428
13. Interfaces, Protocols, and ABCs.....	433
The Typing Map	434
What's New in This Chapter	435
Two Kinds of Protocols	436
Programming Ducks	438
Python Digs Sequences	438
Monkey Patching: Implementing a Protocol at Runtime	440
Defensive Programming and "Fail Fast"	442

Goose Typing	444
Subclassing an ABC	449
ABCs in the Standard Library	451
Defining and Using an ABC	453
ABC Syntax Details	459
Subclassing an ABC	460
A Virtual Subclass of an ABC	462
Usage of register in Practice	465
Structural Typing with ABCs	466
Static Protocols	468
The Typed double Function	468
Runtime Checkable Static Protocols	470
Limitations of Runtime Protocol Checks	473
Supporting a Static Protocol	474
Designing a Static Protocol	476
Best Practices for Protocol Design	478
Extending a Protocol	479
The numbers ABCs and Numeric Protocols	480
Chapter Summary	483
Further Reading	484
14. Inheritance: For Better or for Worse.....	489
What's New in This Chapter	490
The super() Function	490
Subclassing Built-In Types Is Tricky	492
Multiple Inheritance and Method Resolution Order	496
Mixin Classes	502
Case-Insensitive Mappings	502
Multiple Inheritance in the Real World	504
ABCs Are Mixins Too	504
ThreadingMixIn and ForkingMixIn	505
Django Generic Views Mixins	506
Multiple Inheritance in Tkinter	509
Coping with Inheritance	512
Favor Object Composition over Class Inheritance	512
Understand Why Inheritance Is Used in Each Case	512
Make Interfaces Explicit with ABCs	513
Use Explicit Mixins for Code Reuse	513
Provide Aggregate Classes to Users	513
Subclass Only Classes Designed for Subclassing	514
Avoid Subclassing from Concrete Classes	515
Tkinter: The Good, the Bad, and the Ugly	515

Chapter Summary

Further Reading

15. More About Type Hints.....

What's New in This Chapter

Overloaded Signatures

Max Overload

Takeaways from Overloading max

TypedDict

Type Casting

Reading Type Hints at Runtime

Problems with Annotations at Runtime

Dealing with the Problem

Implementing a Generic Class

Basic Jargon for Generic Types

Variance

An Invariant Dispenser

A Covariant Dispenser

A Contravariant Trash Can

Variance Review

Implementing a Generic Static Protocol

Chapter Summary

Further Reading

16. Operator Overloading.....

What's New in This Chapter

Operator Overloading 101

Unary Operators

Overloading + for Vector Addition

Overloading * for Scalar Multiplication

Using @ as an Infix Operator

Wrapping-Up Arithmetic Operators

Rich Comparison Operators

Augmented Assignment Operators

Chapter Summary

Further Reading

Part IV. Control Flow

17. Iterators, Generators, and Classic Coroutines.....

What's New in This Chapter

516

517

523

523

524

525

529

530

538

541

542

544

545

548

548

549

550

551

553

556

558

559

565

566

566

567

570

576

578

580

581

584

589

591

597

598

Table of Contents

|

xiii

A Sequence of Words	598
Why Sequences Are Iterable: The iter Function	600
Using iter with a Callable	602
Iterables Versus Iterators	603
Sentence Classes with <code>__iter__</code>	607
Sentence Take #2: A Classic Iterator	607
Don't Make the Iterable an Iterator for Itself	609
Sentence Take #3: A Generator Function	610
How a Generator Works	611
Lazy Sentences	614
Sentence Take #4: Lazy Generator	614
Sentence Take #5: Lazy Generator Expression	616
When to Use Generator Expressions	617
An Arithmetic Progression Generator	619
Arithmetic Progression with <code>itertools</code>	622
Generator Functions in the Standard Library	623
Iterable Reducing Functions	634
Subgenerators with <code>yield from</code>	636
Reinventing chain	637
Traversing a Tree	638
Generic Iterable Types	643
Classic Coroutines	645
Example: Coroutine to Compute a Running Average	647
Returning a Value from a Coroutine	650
Generic Type Hints for Classic Coroutines	654
Chapter Summary	656
Further Reading	656
18. with, match, and else Blocks.....	661
What's New in This Chapter	662
Context Managers and <code>with</code> Blocks	662
The <code>contextlib</code> Utilities	667
Using <code>@contextmanager</code>	668
Pattern Matching in <code>lis.py</code> : A Case Study	673
Scheme Syntax	673
Imports and Types	675
The Parser	675
The Environment	677
The REPL	679
The Evaluator	680
Procedure: A Class Implementing a Closure	689
Using OR-patterns	690

Do This, Then That: else Blocks Beyond if	691
Chapter Summary	693
Further Reading	694
19. Concurrency Models in Python.....	699
What's New in This Chapter	700
The Big Picture	700
A Bit of Jargon	701
Processes, Threads, and Python's Infamous GIL	703
A Concurrent Hello World	705
Spinner with Threads	705
Spinner with Processes	708
Spinner with Coroutines	710
Supervisors Side-by-Side	715
The Real Impact of the GIL	717
Quick Quiz	717
A Homegrown Process Pool	720
Process-Based Solution	722
Understanding the Elapsed Times	722
Code for the Multicore Prime Checker	723
Experimenting with More or Fewer Processes	727
Thread-Based Nonsolution	728
Python in the Multicore World	729
System Administration	730
Data Science	731
Server-Side Web/Mobile Development	732
WSGI Application Servers	734
Distributed Task Queues	736
Chapter Summary	737
Further Reading	738
Concurrency with Threads and Processes	738
The GIL	740
Concurrency Beyond the Standard Library	740
Concurrency and Scalability Beyond Python	742
20. Concurrent Executors.....	747
What's New in This Chapter	747
Concurrent Web Downloads	748
A Sequential Download Script	750
Downloading with concurrent.futures	753
Where Are the Futures?	755
Launching Processes with concurrent.futures	758

Multicore Prime Checker Redux	759
Experimenting with Executor.map	762
Downloads with Progress Display and Error Handling	766
Error Handling in the flags2 Examples	770
Using futures.as_completed	773
Chapter Summary	776
Further Reading	776
21. Asynchronous Programming.....	779
What's New in This Chapter	780
A Few Definitions	781
An asyncio Example: Probing Domains	782
Guido's Trick to Read Asynchronous Code	784
New Concept: Awaitable	785
Downloading with asyncio and HTTPX	786
The Secret of Native Coroutines: Humble Generators	789
The All-or-Nothing Problem	790
Asynchronous Context Managers	790
Enhancing the asyncio Downloader	792
Using asyncio.as_completed and a Thread	793
Throttling Requests with a Semaphore	795
Making Multiple Requests for Each Download	799
Delegating Tasks to Executors	801
Writing asyncio Servers	803
A FastAPI Web Service	805
An asyncio TCP Server	808
Asynchronous Iteration and Asynchronous Iterables	815
Asynchronous Generator Functions	816
Async Comprehensions and Async Generator Expressions	822
async Beyond asyncio: Curio	825
Type Hinting Asynchronous Objects	828
How Async Works and How It Doesn't	829
Running Circles Around Blocking Calls	829
The Myth of I/O-Bound Systems	830
Avoiding CPU-Bound Traps	830
Chapter Summary	831
Further Reading	832

Part V. Metaprogramming

22. Dynamic Attributes and Properties. 839

 What's New in This Chapter 840

 Data Wrangling with Dynamic Attributes 840

 Exploring JSON-Like Data with Dynamic Attributes 842

 The Invalid Attribute Name Problem 846

 Flexible Object Creation with `__new__` 847

 Computed Properties 849

 Step 1: Data-Driven Attribute Creation 850

 Step 2: Property to Retrieve a Linked Record 852

 Step 3: Property Overriding an Existing Attribute 856

 Step 4: Bespoke Property Cache 857

 Step 5: Caching Properties with `functools` 859

 Using a Property for Attribute Validation 861

 LineItem Take #1: Class for an Item in an Order 861

 LineItem Take #2: A Validating Property 862

 A Proper Look at Properties 864

 Properties Override Instance Attributes 865

 Property Documentation 868

 Coding a Property Factory 869

 Handling Attribute Deletion 872

 Essential Attributes and Functions for Attribute Handling 873

 Special Attributes that Affect Attribute Handling 874

 Built-In Functions for Attribute Handling 874

 Special Methods for Attribute Handling 875

 Chapter Summary 877

 Further Reading 878

23. Attribute Descriptors. 883

 What's New in This Chapter 884

 Descriptor Example: Attribute Validation 884

 LineItem Take #3: A Simple Descriptor 884

 LineItem Take #4: Automatic Naming of Storage Attributes 891

 LineItem Take #5: A New Descriptor Type 893

 Overriding Versus Nonoverriding Descriptors 896

 Overriding Descriptors 898

 Overriding Descriptor Without `__get__` 899

 Nonoverriding Descriptor 900

 Overwriting a Descriptor in the Class 901

 Methods Are Descriptors 902

 Descriptor Usage Tips 904

Descriptor Docstring and Overriding Deletion	906
Chapter Summary	907
Further Reading	908
24. Class Metaprogramming.....	911
What's New in This Chapter	912
Classes as Objects	912
type: The Built-In Class Factory	913
A Class Factory Function	915
Introducing <code>__init_subclass__</code>	918
Why <code>__init_subclass__</code> Cannot Configure <code>__slots__</code>	925
Enhancing Classes with a Class Decorator	926
What Happens When: Import Time Versus Runtime	929
Evaluation Time Experiments	930
Metaclasses 101	935
How a Metaclass Customizes a Class	937
A Nice Metaclass Example	938
Metaclass Evaluation Time Experiment	941
A Metaclass Solution for Checked	946
Metaclasses in the Real World	951
Modern Features Simplify or Replace Metaclasses	951
Metaclasses Are Stable Language Features	952
A Class Can Only Have One Metaclass	952
Metaclasses Should Be Implementation Details	953
A Metaclass Hack with <code>__prepare__</code>	954
Wrapping Up	956
Chapter Summary	957
Further Reading	958
Afterword.....	963
Index.....	967