

Table of Contents

Preface	1
Chapter 1: Benchmarking and Profiling	9
Designing your application	10
Writing tests and benchmarks	16
Timing your benchmark	18
Better tests and benchmarks with pytest-benchmark	21
Finding bottlenecks with cProfile	24
Profile line by line with line_profiler	29
Optimizing our code	31
The dis module	33
Profiling memory usage with memory_profiler	34
Summary	37
Chapter 2: Pure Python Optimizations	39
Useful algorithms and data structures	40
Lists and deques	41
Dictionaries	43
Building an in-memory search index using a hash map	45
Sets	47
Heaps	48
Tries	50
Caching and memoization	52
Joblib	55
Comprehensions and generators	56
Summary	58
Chapter 3: Fast Array Operations with NumPy and Pandas	59
Getting started with NumPy	60
Creating arrays	60
Accessing arrays	62
Broadcasting	67
Mathematical operations	70
Calculating the norm	71
Rewriting the particle simulator in NumPy	71
Reaching optimal performance with numexpr	75
Pandas	77
Pandas fundamentals	77
Indexing Series and DataFrame objects	79
Database-style operations with Pandas	81

Mapping	82
Grouping, aggregations, and transforms	84
Joining	86
Summary	88
Chapter 4: C Performance with Cython	89
Compiling Cython extensions	89
Adding static types	92
Variables	92
Functions	94
Classes	95
Sharing declarations	97
Working with arrays	98
C arrays and pointers	98
NumPy arrays	101
Typed memoryviews	102
Particle simulator in Cython	104
Profiling Cython	108
Using Cython with Jupyter	112
Summary	115
Chapter 5: Exploring Compilers	117
Numba	118
First steps with Numba	118
Type specializations	120
Object mode versus native mode	121
Numba and NumPy	124
Universal functions with Numba	124
Generalized universal functions	126
JIT classes	129
Limitations in Numba	132
The PyPy project	133
Setting up PyPy	134
Running a particle simulator in PyPy	135
Other interesting projects	136
Summary	137
Chapter 6: Implementing Concurrency	139
Asynchronous programming	140
Waiting for I/O	140
Concurrency	141
Callbacks	143
Futures	146
Event loops	148
The asyncio framework	151
Coroutines	152

Converting blocking code into non-blocking code	156
Reactive programming	158
Observables	158
Useful operators	161
Hot and cold observables	165
Building a CPU monitor	168
Summary	171
Chapter 7: Parallel Processing	173
Introduction to parallel programming	174
Graphic processing units	176
Using multiple processes	177
The Process and Pool classes	178
The Executor interface	180
Monte Carlo approximation of pi	181
Synchronization and locks	184
Parallel Cython with OpenMP	187
Automatic parallelism	189
Getting started with Theano	190
Profiling Theano	195
Tensorflow	197
Running code on a GPU	199
Summary	203
Chapter 8: Advanced Introduction to Concurrent and Parallel Programming	205
Technical requirements	206
What is concurrency?	206
Concurrent versus sequential	206
Example 1 – checking whether a non-negative number is prime	207
Concurrent versus parallel	210
A quick metaphor	211
Not everything should be made concurrent	211
Embarrassingly parallel	212
Inherently sequential	212
Example 2 – inherently sequential tasks	213
I/O bound	215
The history, present, and future of concurrency	215
The history of concurrency	216
The present	217
The future	219
A brief overview of mastering concurrency in Python	221
Why Python?	222
Setting up your Python environment	224
General setup	224
Summary	225

Questions	226
Further reading	226
Chapter 9: Amdahl's Law	227
Technical requirements	227
Amdahl's Law	228
Terminology	228
Formula and interpretation	229
The formula for Amdahl's Law	229
A quick example	230
Implications	230
Amdahl's Law's relationship to the law of diminishing returns	231
How to simulate in Python	232
Practical applications of Amdahl's Law	236
Summary	237
Questions	238
Further reading	238
Chapter 10: Working with Threads in Python	239
Technical requirements	240
The concept of a thread	240
Threads versus processes	240
Multithreading	241
An example in Python	243
An overview of the threading module	247
The thread module in Python 2	247
The threading module in Python 3	247
Creating a new thread in Python	248
Starting a thread with the thread module	249
Starting a thread with the threading module	251
Synchronizing threads	254
The concept of thread synchronization	254
The threading.Lock class	255
An example in Python	255
Multithreaded priority queue	257
A connection between real-life and programmatic queues	257
The queue module	258
Queuing in concurrent programming	259
Multithreaded priority queue	263
Summary	264
Questions	265
Further reading	265
Chapter 11: Using the with Statement in Threads	267
Technical requirements	267

Context management	268
Starting from managing files	268
The with statement as a context manager	269
The syntax of the with statement	271
The with statement in concurrent programming	271
Example of deadlock handling	272
Summary	274
Questions	274
Further reading	275
Chapter 12: Concurrent Web Requests	277
Technical requirements	277
The basics of web requests	278
HTML	278
HTTP requests	280
HTTP status code	281
The requests module	282
Making a request in Python	283
Running a ping test	285
Concurrent web requests	286
Spawning multiple threads	287
Refactoring request logic	289
The problem of timeout	291
Support from httpstat.us and simulation in Python	291
Timeout specifications	292
Good practices in making web requests	296
Consider the terms of service and data-collecting policies	296
Error handling	296
Update your program regularly	297
Avoid making a large number of requests	297
Summary	299
Questions	299
Further reading	299
Chapter 13: Working with Processes in Python	301
Technical requirements	302
The concept of a process	302
Processes versus threads	304
Multiprocessing	305
Introductory example in Python	307
An overview of the multiprocessing module	309
The process class	309
The Pool class	310
Determining the current process, waiting, and terminating processes	311
Determining the current process	311

Waiting for processes	314
Terminating processes	317
Interprocess communication	317
Message passing for a single worker	318
Message passing between several workers	320
Summary	326
Questions	327
Further reading	327
Chapter 14: Reduction Operators in Processes	329
Technical requirements	329
The concept of reduction operators	330
Properties of a reduction operator	330
Examples and non-examples	331
Example implementation in Python	333
Real-life applications of concurrent reduction operators	338
Summary	338
Questions	339
Further reading	339
Chapter 15: Concurrent Image Processing	341
Technical requirements	341
Image processing fundamentals	342
Python as an image processing tool	342
Installing OpenCV and NumPy	343
Computer image basics	344
RGB values	344
Pixels and image files	345
Coordinates inside an image	345
OpenCV API	346
Image processing techniques	348
Grayscaleing	349
Thresholding	351
Applying concurrency to image processing	356
Good concurrent image processing practices	360
Choosing the correct way (out of many)	360
Spawning an appropriate number of processes	363
Processing input/output concurrently	363
Summary	363
Questions	364
Further reading	364
Chapter 16: Introduction to Asynchronous Programming	365
Technical requirements	365
A quick analogy	366
Asynchronous versus other programming models	367

Asynchronous versus synchronous programming	368
Asynchronous versus threading and multiprocessing	369
An example in Python	370
Summary	373
Questions	373
Further reading	374
Chapter 17: Implementing Asynchronous Programming in Python	375
Technical requirements	375
The asyncio module	376
Coroutines, event loops, and futures	376
Asyncio API	378
The asyncio framework in action	379
Asynchronously counting down	380
A note about blocking functions	384
Asynchronous prime-checking	385
Improvements from Python 3.7	389
Inherently blocking tasks	390
concurrent.futures as a solution for blocking tasks	391
Changes in the framework	392
Examples in Python	392
Summary	396
Questions	397
Further reading	398
Chapter 18: Building Communication Channels with asyncio	399
Technical requirements	400
The ecosystem of communication channels	400
Communication protocol layers	400
Asynchronous programming for communication channels	402
Transports and protocols in asyncio	403
The big picture of asyncio's server client	405
Python example	406
Starting a server	406
Installing Telnet	408
Simulating a connection channel	409
Sending messages back to clients	410
Closing the transports	411
Client-side communication with aiohttp	413
Installing aiohttp and aiofiles	414
Fetching a website's HTML code	414
Writing files asynchronously	416
Summary	418
Questions	419
Further reading	419

Chapter 19: Deadlocks	421
Technical requirements	421
The concept of deadlock	422
The Dining Philosophers problem	422
Deadlock in a concurrent system	425
Python simulation	426
Approaches to deadlock situations	430
Implementing ranking among resources	430
Ignoring locks and sharing resources	436
An additional note about locks	438
Concluding note on deadlock solutions	439
The concept of livelock	439
Summary	442
Questions	442
Further reading	442
Chapter 20: Starvation	443
Technical requirements	443
The concept of starvation	444
What is starvation?	444
Scheduling	445
Causes of starvation	446
Starvation's relationship to deadlock	447
The readers-writers problem	448
Problem statement	448
The first readers-writers problem	449
The second readers-writers problem	453
The third readers-writers problem	456
Solutions to starvation	458
Summary	459
Questions	460
Further reading	460
Chapter 21: Race Conditions	461
Technical requirements	461
The concept of race conditions	462
Critical sections	462
How race conditions occur	463
Simulating race conditions in Python	465
Locks as a solution to race conditions	467
The effectiveness of locks	467
Implementation in Python	469
The downside of locks	470
Turning a concurrent program sequential	471
Locks do not lock anything	473

Race conditions in real life	474
Security	474
Operating systems	475
Networking	476
Summary	477
Questions	477
Further reading	478
Chapter 22: The Global Interpreter Lock	479
Technical requirements	479
An introduction to the Global Interpreter Lock	480
An analysis of memory management in Python	480
The problem that the GIL addresses	483
Problems raised by the GIL	484
The potential removal of the GIL from Python	486
How to work with the GIL	486
Implementing multiprocessing, rather than multithreading	487
Getting around the GIL with native extensions	489
Utilizing a different Python interpreter	489
Summary	489
Questions	490
Further reading	490
Chapter 23: The Factory Pattern	491
The factory method	492
Real-world examples	493
Use cases	493
Implementing the factory method	494
The abstract factory	502
Real-world examples	502
Use cases	503
Implementing the abstract factory pattern	503
Summary	508
Chapter 24: The Builder Pattern	509
Real-world examples	510
Use cases	511
Implementation	515
Summary	521
Chapter 25: Other Creational Patterns	523
The prototype pattern	524
Real-world examples	524
Use cases	525
Implementation	525
Singleton	529

Real-world examples	529
Use cases	530
Implementation	530
Summary	535
Chapter 26: The Adapter Pattern	537
Real-world examples	537
Use cases	538
Implementation	538
Summary	541
Chapter 27: The Decorator Pattern	543
Real-world examples	544
Use cases	544
Implementation	545
Summary	550
Chapter 28: The Bridge Pattern	551
Real-world examples	551
Use cases	552
Implementation	552
Summary	556
Chapter 29: The Facade Pattern	557
Real-world examples	558
Use cases	558
Implementation	559
Summary	563
Chapter 30: Other Structural Patterns	565
The flyweight pattern	566
Real-world examples	567
Use cases	567
Implementation	568
The model-view-controller pattern	573
Real-world examples	574
Use cases	575
Implementation	576
The proxy pattern	580
Real-world examples	583
Use cases	583
Implementation	584
Summary	588
Chapter 31: The Chain of Responsibility Pattern	589
Real-world examples	590

Use cases	592
Implementation	593
Summary	598
Chapter 32: The Command Pattern	599
Real-world examples	600
Use cases	600
Implementation	601
Summary	609
Chapter 33: The Observer Pattern	611
Real-world examples	611
Use cases	612
Implementation	613
Summary	619
Appendix	621
Other Books You May Enjoy	641
Index	645