

# Table of Contents

|                                                       |           |
|-------------------------------------------------------|-----------|
| <b>Foreword.....</b>                                  | <b>ix</b> |
| <b>Preface.....</b>                                   | <b>xi</b> |
| <b>1. Understanding Performant Python.....</b>        | <b>1</b>  |
| The Fundamental Computer System                       | 2         |
| Computing Units                                       | 2         |
| Memory Units                                          | 6         |
| Communications Layers                                 | 8         |
| Idealized Computing Versus the Python Virtual Machine | 10        |
| Idealized Computing                                   | 11        |
| Python's Virtual Machine                              | 12        |
| So Why Use Python?                                    | 15        |
| How to Be a Highly Performant Programmer              | 17        |
| Good Working Practices                                | 18        |
| Optimizing for the Team Rather than the Code Block    | 21        |
| The Remote Performant Programmer                      | 23        |
| Some Thoughts on Good Notebook Practice               | 23        |
| Your Work                                             | 24        |
| The Future of Python                                  | 25        |
| Where Did the GIL Go?                                 | 25        |
| Does Python Have a JIT?                               | 25        |
| Wrap-Up                                               | 26        |
| <b>2. Profiling to Find Bottlenecks.....</b>          | <b>27</b> |
| Profiling Efficiently                                 | 28        |
| Introducing the Julia Set                             | 30        |
| Calculating the Full Julia Set                        | 33        |

|                                                          |            |
|----------------------------------------------------------|------------|
| Simple Approaches to Timing—print and a Decorator        | 36         |
| Simple Timing Using the Unix time Command                | 39         |
| Using the cProfile Module                                | 41         |
| Visualizing cProfile Output with SnakeViz                | 46         |
| Using line_profiler for Line-by-Line Measurements        | 48         |
| Using memory_profiler to Diagnose Memory Usage           | 53         |
| Combining CPU and Memory Profiling with Scalene          | 61         |
| Introspecting an Existing Process with PySpy             | 63         |
| VizTracer for an Interactive Time-Based Call Stack       | 65         |
| Bytecode: Under the Hood                                 | 67         |
| Using the dis Module to Examine CPython Bytecode         | 67         |
| Digging into Bytecode Specialization with Specialist     | 69         |
| Different Approaches, Different Complexity               | 71         |
| Unit Testing During Optimization to Maintain Correctness | 73         |
| No-op @profile Decorator                                 | 74         |
| Strategies to Profile Your Code Successfully             | 77         |
| Wrap-Up                                                  | 78         |
| <b>3. Lists and Tuples.....</b>                          | <b>79</b>  |
| A More Efficient Search                                  | 82         |
| Lists Versus Tuples                                      | 85         |
| Lists as Dynamic Arrays                                  | 86         |
| Tuples as Static Arrays                                  | 90         |
| Wrap-Up                                                  | 93         |
| <b>4. Dictionaries and Sets.....</b>                     | <b>95</b>  |
| How Do Dictionaries and Sets Work?                       | 99         |
| Inserting and Retrieving                                 | 99         |
| Deletion                                                 | 104        |
| Resizing                                                 | 104        |
| Hash Functions and Entropy                               | 106        |
| Wrap-Up                                                  | 111        |
| <b>5. Iterators and Generators.....</b>                  | <b>113</b> |
| Iterators for Infinite Series                            | 118        |
| Lazy Generator Evaluation                                | 120        |
| Wrap-Up                                                  | 124        |
| <b>6. Matrix and Vector Computation.....</b>             | <b>125</b> |
| Introduction to the Problem                              | 126        |
| Aren't Python Lists Good Enough?                         | 131        |
| Problems with Allocating Too Much                        | 133        |

---

|                                                               |            |
|---------------------------------------------------------------|------------|
| Memory Fragmentation                                          | 136        |
| Understanding perf                                            | 139        |
| Making Decisions with perf's Output                           | 142        |
| Enter numpy                                                   | 143        |
| Applying numpy to the Diffusion Problem                       | 146        |
| Memory Allocations and In-Place Operations                    | 149        |
| Selective Optimizations: Finding What Needs to Be Fixed       | 153        |
| numexpr: Making In-Place Operations Faster and Easier         | 156        |
| Graphics Processing Units (GPUs)                              | 158        |
| Dynamic Graphs: PyTorch                                       | 159        |
| GPU Speed and Numerical Precision                             | 162        |
| GPU-Specific Operations                                       | 165        |
| Basic GPU Profiling                                           | 168        |
| Performance Considerations of GPUs                            | 170        |
| When to Use GPUs                                              | 172        |
| Deep Learning Performance Considerations                      | 174        |
| A Cautionary Tale: Verify "Optimizations" (scipy)             | 179        |
| Lessons from Matrix Optimizations                             | 180        |
| Wrap-Up                                                       | 183        |
| <b>7. Pandas, Dask, and Polars.....</b>                       | <b>185</b> |
| Pandas                                                        | 186        |
| Pandas's Internal Model                                       | 187        |
| Arrow and NumPy                                               | 189        |
| Applying a Function to Many Rows of Data                      | 189        |
| Numba to Compile NumPy for Pandas                             | 199        |
| Building from Partial Results Rather than Concatenating       | 200        |
| There's More Than One (and Possibly a Faster) Way to Do a Job | 202        |
| Advice for Effective Pandas Development                       | 203        |
| Dask for Distributed Data Structures and DataFrames           | 205        |
| Diagnostics                                                   | 206        |
| Parallel Pandas with Dask                                     | 207        |
| Parallelized apply with Swifter on Dask                       | 209        |
| Polars for Fast DataFrames                                    | 210        |
| Wrap-Up                                                       | 211        |
| <b>8. Compiling to C.....</b>                                 | <b>213</b> |
| What Sort of Speed Gains Are Possible?                        | 214        |
| JIT Versus AOT Compilers                                      | 216        |
| Why Does Type Information Help the Code Run Faster?           | 216        |
| Using a C Compiler                                            | 217        |
| Reviewing the Julia Set Example                               | 218        |

|                                                       |            |
|-------------------------------------------------------|------------|
| Cython                                                | 219        |
| Compiling a Pure Python Version Using Cython          | 219        |
| pyximport                                             | 221        |
| Cython Annotations to Analyze a Block of Code         | 222        |
| Adding Some Type Annotations                          | 224        |
| Cython and numpy                                      | 229        |
| Parallelizing the Solution with OpenMP on One Machine | 232        |
| Numba                                                 | 234        |
| PyPy                                                  | 236        |
| Garbage Collection Differences                        | 238        |
| Running PyPy and Installing Modules                   | 238        |
| A Summary of Speed Improvements                       | 240        |
| When to Use Each Technology                           | 241        |
| Foreign Function Interfaces                           | 242        |
| ctypes                                                | 243        |
| cffi                                                  | 246        |
| f2py                                                  | 249        |
| CPython Extensions: C                                 | 252        |
| CPython Extensions: Rust                              | 256        |
| Wrap-Up                                               | 260        |
| <b>9. Asynchronous I/O.....</b>                       | <b>261</b> |
| Introduction to Asynchronous Programming              | 263        |
| How Does async/await Work?                            | 267        |
| Serial Web Crawler                                    | 268        |
| Asynchronous Web Crawler                              | 270        |
| Shared CPU-I/O Workload                               | 275        |
| Serial CPU Workload                                   | 276        |
| Batched CPU Workload                                  | 278        |
| Fully Asynchronous CPU Workload                       | 281        |
| Wrap-Up                                               | 286        |
| <b>10. The multiprocessing Module.....</b>            | <b>289</b> |
| An Overview of the multiprocessing Module             | 292        |
| Estimating Pi Using the Monte Carlo Method            | 294        |
| Estimating Pi Using Processes and Threads             | 296        |
| Using Python Objects                                  | 296        |
| Replacing multiprocessing with Joblib                 | 304        |
| Random Numbers in Parallel Systems                    | 308        |
| Using numpy                                           | 309        |
| Finding Prime Numbers                                 | 312        |
| Queues of Work                                        | 319        |

|                                                                      |            |
|----------------------------------------------------------------------|------------|
| Asynchronously Adding Jobs to the Queue                              | 323        |
| Verifying Primes Using Interprocess Communication                    | 324        |
| Serial Solution                                                      | 329        |
| Naive Pool Solution                                                  | 329        |
| A Less Naive Pool Solution                                           | 330        |
| Using manager.Value as a Flag                                        | 331        |
| Using Redis as a Flag                                                | 333        |
| Using RawValue as a Flag                                             | 336        |
| Using mmap as a Flag                                                 | 337        |
| Using mmap as a Flag Redux                                           | 338        |
| Sharing numpy Data with multiprocessing                              | 340        |
| Synchronizing File and Variable Access                               | 348        |
| File Locking                                                         | 348        |
| Locking a Value                                                      | 352        |
| Wrap-Up                                                              | 356        |
| <b>11. Clusters and Job Queues.....</b>                              | <b>357</b> |
| Benefits of Clustering                                               | 358        |
| Drawbacks of Clustering                                              | 359        |
| \$462 Million Wall Street Loss Through Poor Cluster Upgrade Strategy | 360        |
| Skype's 24-Hour Global Outage                                        | 361        |
| Common Cluster Designs                                               | 362        |
| How to Start a Clustered Solution                                    | 362        |
| Ways to Avoid Pain When Using Clusters                               | 363        |
| Two Clustering Solutions                                             | 365        |
| Using IPython Parallel to Support Research                           | 365        |
| Message Brokering for Cluster Efficiency                             | 368        |
| Other Clustering Tools to Look At                                    | 372        |
| Docker                                                               | 373        |
| Docker's Performance                                                 | 373        |
| Advantages of Docker                                                 | 377        |
| Wrap-Up                                                              | 378        |
| <b>12. Using Less RAM.....</b>                                       | <b>379</b> |
| Objects for Primitives Are Expensive                                 | 380        |
| The array Module Stores Many Primitive Objects Cheaply               | 382        |
| Using Less RAM in NumPy with NumExpr                                 | 385        |
| Understanding the RAM Used in a Collection                           | 389        |
| Bytes Versus Unicode                                                 | 390        |
| Efficiently Storing Lots of Text in RAM                              | 391        |
| Trying These Approaches on 11 Million Tokens                         | 392        |
| Modeling More Text with scikit-learn's FeatureHasher                 | 400        |

|                                                                       |            |
|-----------------------------------------------------------------------|------------|
| Introducing DictVectorizer and FeatureHasher                          | 401        |
| Comparing DictVectorizer and FeatureHasher on a Real Problem          | 404        |
| SciPy's Sparse Matrices                                               | 405        |
| Tips for Using Less RAM                                               | 408        |
| Probabilistic Data Structures                                         | 409        |
| Very Approximate Counting with a 1-Byte Morris Counter                | 410        |
| K-Minimum Values                                                      | 413        |
| Bloom Filters                                                         | 417        |
| LogLog Counter                                                        | 423        |
| Real-World Example                                                    | 427        |
| Wrap-Up                                                               | 430        |
| <b>13. Lessons from the Field.....</b>                                | <b>431</b> |
| Developing a High Performance Machine Learning Algorithm              | 432        |
| High Performance Computing in Journalism                              | 435        |
| Lessons from the Field of Cyber Reinsurance                           | 441        |
| Python in Quant Finance                                               | 451        |
| Maintain Flexibility to Achieve High Performance                      | 455        |
| Streamlining Feature Engineering Pipelines with Feature-engine (2020) | 458        |
| Highly Performant Data Science Teams (2020)                           | 464        |
| Numba (2020)                                                          | 468        |
| Optimizing Versus Thinking (2020)                                     | 474        |
| Making Deep Learning Fly with RadimRehurek.com (2014)                 | 477        |
| Large-Scale Social Media Analysis at Smesh (2014)                     | 483        |
| <b>Index.....</b>                                                     | <b>487</b> |