Python Concurrency with asyncio

Matthew Fowler

MANNING

inside front cover

I want to . . .	How?	Chapter(s)
Learn the basics of single-threaded concurrency	Understand how selectors and the event loop work	1, 3
Run two operations concurrently	Use asyncio tasks and API functions	2, 4
Add a timeout to a long-running task	Use the task's `wait_for` method	2
Cancel a long-running task	Use the task's `cancel` method	2
Learn how non-blocking sockets work	Create a non-blocking echo server with selectors	3
Handle graceful shutdowns of asyncio applications	Use the signals API (*nix systems only)	3

I want to . . .	How?	Chapter(s)
Run multiple tasks concurrently	Use asyncio API functions such as `gather`, `wait`, and `as_completed`	4
Run asynchronous web API requests	Use a library, such as aiohttp, with asyncio API Functions	4
Run multiple SQL queries concurrently	Use an asynchronous library like asyncpg with asyncio API functions and connection pools	5
Run CPU-intensive work concurrently	Use multiprocessing and process pools	6
Use shared state among multiple processes	Save the state to shared memory	6
Avoid race conditions in	Use multiprocessing	6

I want to . . .	How?	Chapter(s)
multiprocessing code	locks	
Run blocking I/O-based APIs, such as requests, concurrently	Use multithreading and thread pools	7
Avoid race conditions and deadlocks in multithreading code	Use multithreading locks and reentrant locks	7
Build a responsive GUI with asyncio	Use multithreading with threading queues	7
Run multiple CPU-intensive tasks, such as data analysis with NumPy, concurrently	Use multiprocessing, multithreading in certain circumstances	6, 7
Build a non-blocking command line application	Use streams to asynchronously read data (*nix systems only)	8

I want to . . .	How?	Chapter(s)
Build a web application with asyncio	Use a web framework with ASGI support such as Starlette or Django	9
Use WebSockets asynchronously	Use the WebSocket library	9
Build a resilient backend-for-frontend microservice architecture with async concepts	Use asyncio API functions with retries and the circuit breaker pattern	10
Prevent single-threaded race conditions	Use asyncio locks	11
Limit the number of tasks running concurrently	Use asyncio semaphores	11
Wait until an event occurs before performing an operation	Use asyncio events	11

I want to . . .	How?	Chapter(s)
Control access to a shared resource	Use asyncio condtions	11

Continued on inside back cover

Python Concurrency with asyncio

Matthew Fowler

To comment go to liveBook

Manning

Shelter Island

For more information on this and other Manning titles go to

www.manning.com

Copyright

Manning Publications Co.
20 Baldwin Road Technical
PO Box 761
Shelter Island, NY 11964

Development editor:	Doug Rudder
Technical development editor:	Robert Wenner
Review editor:	Mihaela Batinić
Production editor:	Andy Marinkovich
Copy editor:	Christian Berk
Proofreader:	Keri Hales
Technical proofreader:	Mayur Patil
Typesetter:	Dennis Dalinnik

Cover designer: Marija Tudor

ISBN: 9781617298660

dedication

To my beautiful wife Kathy, thank you for always being there.

contents

7 Handling blocking work with threads

front matter

e

Nearly 20 years ago, I got my start in professional software engineering writing a mashup of Matlab, C++, and VB.net code to control and analyze data from mass spectrometers and other laboratory devices. The thrill of seeing a line of code trigger a machine to move how I wanted always stuck with me, and ever since then, I knew software engineering was the career for me. Over the years, I gradually moved toward API development and distributed systems, mainly focusing on Java and Scala, learning a lot of Python along the way.

I got my start in Python around 2015, primarily by working on a machine learning pipeline that took sensor data and used it to make predictions—such as sleep tracking, step count, sit-to-stand transitions, and similar activities—about the sensor's wearer. At the time, this machine learning pipeline was slow to the point that it was becoming a customer issue. One of the ways I worked on alleviating the issue was utilizing concurrency. As I dug into the knowledge available for learning concurrent programming in Python, I found things hard to navigate and learn compared to what I was used to in the Java world. Why doesn't multithreading work the same way that it would in Java? Does it make more sense to use multiprocessing? What about the newly introduced asyncio? What is the global interpreter lock, and why does it exist? There weren't a lot of books on the topic of concurrency in Python, and most knowledge was scattered throughout documentation and a smattering of blogs with varying consistency of quality. Fast-forward to today, and things haven't changed much. While there are more resources, the landscape is still sparse, disjointed, and not as friendly for newcomers to concurrency as it should be.

Of course, a lot has changed in the past several years. Back then, asyncio was in its infancy and has since become an important module in Python. Now, single-threaded concurrency models and coroutines are a core component of concurrency in Python, in addition to multithreading and multiprocessing. This means the concurrency landscape in Python has gotten larger and more complex, while still not having comprehensive resources for those wanting to learn it.

My motivation for writing this book was to fill this gap that exists in the Python landscape on the topic of concurrency, specifically with asyncio and single-threaded concurrency. I wanted to make the complex and under-documented topic of single-threaded concurrency more accessible to developers of all skill levels. I also wanted to write a book that would enhance generic understanding of concurrency topics outside of Python. Frameworks such as Node.js and languages such as Kotlin have single-threaded concurrency models and coroutines, so knowledge gained here is helpful in those domains as well. My hope is that all who read it find this book useful in their day-to-day lives as developers—not only within the Python landscape but also within the domain of concurrent programming.

wledgments

First, I want to thank my wife, Kathy, who was always there for me to proofread when I wasn't sure if something made sense, and who was extremely supportive through the entire process. A close second goes to my dog, Dug, who was always around to drop his ball near me to remind me to take a break from writing to play.

Next, I'd like to thank my editor, Doug Rudder, and my technical reviewer, Robert Wenner. Your feedback was invaluable in helping keep this book on schedule and

high quality, ensuring that my code and explanations made sense and were easy to understand.

To all the reviewers: Alexey Vyskubov, Andy Miles, Charles M. Shelton, Chris Viner, Christopher Kottmyer, Clifford Thurber, Dan Sheikh, David Cabrero, Didier Garcia, Dimitrios Kouzis-Loukas, Eli Mayost, Gary Bake, Gonzalo Gabriel Jiménez Fuentes, Gregory A. Lussier, James Liu, Jeremy Chen, Kent R. Spillner, Lakshmi Narayanan Narasimhan, Leonardo Taccari, Matthias Busch, Pavel Filatov, Phillip Sorensen, Richard Vaughan, Sanjeev Kilarapu, Simeon Leyzerzon, Simon Tschöke, Simone Sguazza, Sumit K. Singh, Viron Dadala, William Jamir Silva, and Zoheb Ainapore, your suggestions helped make this a better book.

Finally, I want to thank the countless number of teachers, coworkers, and mentors I've had over the past years. I've learned and grown so much from all of you. The sum of the experiences we've had together has given me the tools needed to produce this work as well as succeed in my career. Without all of you, I wouldn't be where I am today. Thank you!

this book

Python Concurrency with asyncio was written to teach you how to utilize concurrency in Python to improve application performance, throughput, and responsiveness. We start by focusing on core concurrency topics, explaining how asyncio's model of single-threaded concurrency works as well as how coroutines and async/await syntax works. We then transition into practical applications of concurrency, such as making multiple web requests or database queries concurrently, managing threads and processes, building web applications, and handling synchronization issues.

hould read this book?

This book is for intermediate to advanced developers who are looking to better understand and utilize concurrency in their existing or new Python applications. One of the goals of this book is to explain complex concurrency topics in plain, easy-to-understand language. To that end, no prior experience with concurrency is needed, though of course, it is helpful. In this book we'll cover a wide range of uses, from web-based APIs to command-line applications, so this book should be applicable to many problems you'll need to solve as a developer.

his book is organized: A road map

This book is organized into 14 chapters, covering gradually more advanced topics that build on what you've learned in previous chapters.

- **Chapter 1** focuses on basic concurrency knowledge in Python. We learn what CPU-bound and I/O-bound work is and introduce how asyncio's single-threaded concurrency model works.
- **Chapter 2** focuses on the basics of asyncio coroutines and how to use async/await syntax to build applications utilizing concurrency.
- **Chapter 3** focuses on how non-blocking sockets and selectors work and how to build an echo server using asyncio.
- **Chapter 4** focuses on how to make multiple web requests concurrently. Doing this, we'll learn more about the core asyncio APIs for running coroutines concurrently.
- **Chapter 5** focuses on how to make multiple database queries concurrently using connection pools. We'll also learn about asynchronous context managers and asynchronous generators in the context of databases
- **Chapter 6** focuses on multiprocessing, specifically how to utilize it with asyncio to handle CPU-intensive work. We'll build a map/reduce application to demonstrate this.
- **Chapter 7** focuses on multithreading, specifically how to utilize it with asyncio to handle blocking I/O. This is useful for libraries that don't have native asyncio support but can still benefit from concurrency.
- **Chapter 8** focuses on network streams and protocols. We'll use this to create a chat server and client capable of handling multiple users concurrently.

- **Chapter 9** focuses on asyncio-powered web applications and the ASGI (asynchronous server gateway interface). We'll explore a few ASGI frameworks and discuss how to build web APIs with them. We'll also explore WebSockets.
- **Chapter 10** describes how to use asyncio-based web APIs to build a hypothetical microservice architecture.
- **Chapter 11** focuses on single-threaded concurrency synchronization issues and how to resolve them. We dive into locks, semaphores, events, and conditions.
- **Chapter 12** focuses on asynchronous queues. We'll use these to build a web application that responds to client requests instantly, despite doing time-consuming work in the background.
- **Chapter 13** focuses on creating and managing subprocesses, showing you how to read from and write data to them.
- **Chapter 14** focuses on advanced topics, such as forcing event loop iterations, context variables, and creating your own event loop. This information will be most useful to asyncio API designers and those interested in how the innards of the asyncio event loop function.

At minimum, you should read the first four chapters to get a full understanding of how asyncio works, how to build your first real application, and how to use the core asyncio APIs to run coroutines concurrently (covered in chapter 4). After this you should feel free to move around the book based on your interests.

the code

This book contains many code examples, both in numbered listings and in-line. Some code listings are reused as imports in later listings in the same chapter, and some are reused across multiple chapters. Code reused across multiple chapters will assume you've created a module named `util`; you'll create this in chapter 2. For each

individual code listing, we will assume you have created a module for that chapter named `chapter_ {chapter_number}` and then put the code in a file of the format `listing_{chapter_ number}_{listing_number}.py` within that module. For example, the code for listing 2.2 in chapter 2 will be in a module called `chapter_2` in a file named `listing_2_2.py`.

Several places in the book go through performance numbers, such as time for a program to complete or web requests completed per second. Code samples in this book were run and benchmarked on a 2019 MacBook Pro with a 2.4 GHz 8-Core Intel Core i9 processor and 32 GB 2667 MHz DDR4 RAM, using a gigabit wireless internet connection. Depending on the machine you run on, these numbers will be different, and factors of speedup or improvement will be different.

Executable snippets of code can be found in the liveBook (online) version of this book at https://livebook.manning.com/book/python-concurrency-with-asyncio. The complete source code can be downloaded free of charge from the Manning website at https://www.manning.com/books/python-concurrency-with-asyncio, and is also available on Github at https://github.com/concurrency-in-python-with-asyncio.

ok discussion forum

Purchase of *Python Concurrency with asyncio* includes free access to liveBook, Manning's online reading platform. Using liveBook's exclusive discussion features, you can attach comments to the book globally or to specific sections or paragraphs. It's a snap to make notes for yourself, ask and answer technical questions, and receive help from the author and other users. To access the forum, go to https://livebook.manning .com/#!/book/python-concurrency-with-asyncio/discussion. You can also learn more about Manning's forums and the rules of conduct at https://livebook.manning.com/#!/discussion.

Manning's commitment to our readers is to provide a venue where a meaningful dialogue between individual readers and between readers and the author can take place. It is not a commitment to any specific amount of participation on the part of the author, whose contribution to the forum remains voluntary (and unpaid). We suggest you try asking the author some challenging questions lest his interest stray! The forum and the archives of previous discussions will be accessible from the publisher's website for as long as the book is in print.

the author

MATTHEW FOWLER has nearly 20 years of software engineering experience in roles from software architect to engineering director. He started out writing software for scientific applications and moved into full-stack web development and distributed systems, eventually leading multiple teams of developers and managers to do the same for an e-commerce site with tens of millions of users. He lives in Lexington, Massachusetts with his wife, Kathy.

the cover illustration

The figure on the cover of *Python Concurrency with asyncio* is "Paysanne du Marquisat de Bade," or Peasant woman of the Marquisate of Baden, taken from a book by Jacques Grasset de Saint-Sauveur published in 1797. Each illustration is finely drawn and colored by hand.

In those days, it was easy to identify where people lived and what their trade or station in life was just by their dress. Manning celebrates the inventiveness and initiative of the computer business with book covers based on the rich diversity of regional culture centuries ago, brought back to life by pictures from collections such as this one.