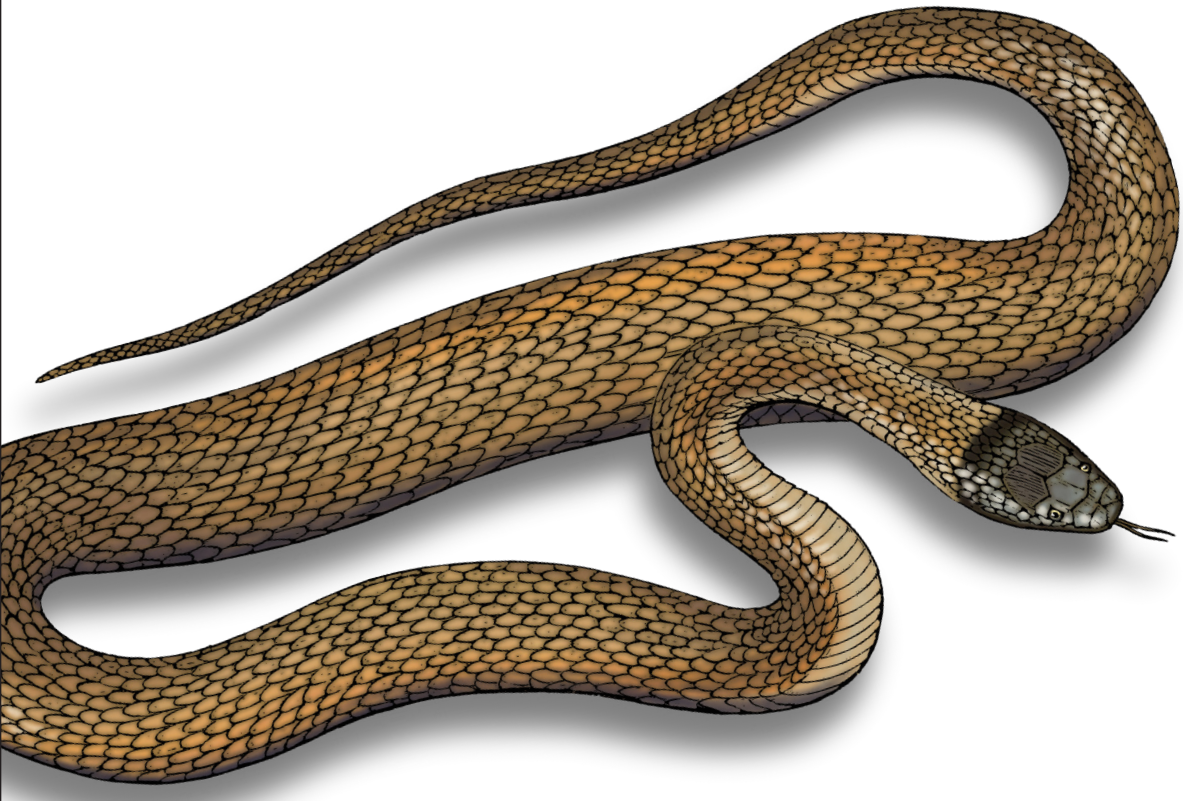
O'REILLY®

Financial Theory with Python

A Gentle Introduction



Yves Hilpisch

O'REILLY®

Financial Theory with Python

Nowadays, finance, mathematics, and programming are intrinsically linked. This book provides the relevant foundations of each discipline to give you the major tools you need to get started in the world of computational finance.

Using an approach where mathematical concepts provide the common background against which financial ideas and programming techniques are learned, this practical guide teaches you the basics of financial economics. Written by the best-selling author of *Python for Finance,* Yves Hilpisch, *Financial Theory with Python* explains financial, mathematical, and Python programming concepts in an integrative manner so that the interdisciplinary concepts reinforce each other.

- Draw upon mathematics to learn the foundations of financial theory and Python programming
- Learn about financial theory, financial data modeling, and the use of Python for computational finance
- Leverage simple economic models to better understand basic notions of finance and Python programming concepts
- Utilize both static and dynamic financial modeling to address fundamental problems in finance, such as pricing, decision-making, equilibrium, and asset allocation
- Learn the basics of Python packages useful for financial modeling, such as NumPy, SciPy, Matplotlib, and SymPy

"The way Yves makes the link between the financial theory and mathematics and then to coding is brilliant."

—Tomer Regev, PhD

Dr. Yves J. Hilpisch is the founder and CEO of The Python Quants, a group focused on the use of open source technologies for financial data science, artificial intelligence, algorithmic trading, and computational finance. He's also the founder and CEO of The AI Machine, a company focused on AI-powered algorithmic trading based on a proprietary strategy execution platform.

PYTHON / FINANCE

US $49.99 CAN $65.99

ISBN: 978-1-098-10435-1

Twitter: @oreillymedia
facebook.com/oreilly

Financial Theory with Python

A Gentle Introduction

Yves Hilpisch

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

Financial Theory with Python
by Yves Hilpisch

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://oreilly.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Acquisitions Editor: Michelle Smith
Development Editor: Michele Cronin
Production Editor: Daniel Elfanbaum
Copyeditor: Piper Editorial Conulting, LLC
Proofreader: Kim Cofer

Indexer: nSight, Inc.
Interior Designer: David Futato
Cover Designer: Karen Montgomery
Illustrator: Kate Dullea

October 2021: First Edition

Revision History for the First Edition
2021-09-23: First Release

See *http://oreilly.com/catalog/errata.csp?isbn=9781098104351* for release details.

978-1-098-10435-1

[LSI]

Table of Contents

Preface

> Python was quickly becoming the de-facto language for data science, machine learning and natural language processing; it would unlock new sources of innovation. Python would allow us to engage with its sizeable open source community, bringing state-of-the-art technology in-house quickly, while allowing for customization.[1]
>
> —Kindman and Taylor (2021)

Why This Book?

Technological trends like online trading platforms, open source software, and open financial data have significantly lowered or even completely removed the barriers of entry to the global financial markets. Individuals with only limited amounts of cash at their free disposal can get started, for example, with algorithmic trading within hours. Students and academics in financial disciplines with a little bit of background knowledge in programming can easily apply cutting-edge innovations in machine and deep learning to financial data—on the notebooks they bring to their finance classes. On the hardware side, cloud providers offer professional compute and data processing capabilities starting at 5 USD per month, billed by the hour and with almost unlimited scalability. So far, academic and professional finance education has only partly reacted to these trends.

This book teaches both finance and the Python programming language from the ground up. Nowadays, finance and programming in general are closely intertwined disciplines, with Python being one of the most widely used programming languages in the financial industry. The book presents relevant foundations—from mathematics, finance, and programming—in an integrated but not-too-technical fashion. Traditionally, theoretical finance and computational finance have been more or less separate disciplines. The fact that programming classes (for example, in Python but

1 Kindman, Andrew and Tom Taylor, "Why We Rewrote Our USD30 Billion Asset Management Platform in Python." (March 29, 2021), *https://oreil.ly/GghS6*.

also in C++) have become an integral part of Master of Financial Engineering and similar university programs shows how important programming skills have become in the field.

However, *mathematical foundations*, *theoretical finance*, and *basic programming techniques* are still quite often taught independently from one another and only later in combination with *computational finance*. This book takes a different approach in that the mathematical concepts—for example, from linear algebra and probability theory—provide the common background against which financial ideas and programming techniques alike are introduced. Abstract mathematical concepts are thereby motivated from two different angles: finance and programming. In addition, this approach allows for a new learning experience since both mathematical and financial concepts can directly be translated into executable code that can then be explored interactively.

Several readers of one of my other books, *Python for Finance* (2nd ed., 2018, O'Reilly), pointed out that it teaches neither finance nor Python from the ground up. Indeed, the reader of that book is expected to have at least some experience in both finance *and* (Python) programming. *Financial Theory with Python* closes this gap in that it focuses on more fundamental concepts from both finance and Python programming. In that sense, readers who finish this book can naturally progress to *Python for Finance* to further build and improve their Python skills as applied to finance. More guidance is provided in the final chapter.

Target Audience

I have written a number of books about Python applied to finance. My company, The Python Quants, offers a number of live and online training classes in Python for finance. For all of my previous books and the training classes, the book readers and training participants are expected to already have some background knowledge in both finance and Python programming or a similar language.

This book starts completely from scratch, with just the expectation that the reader has some basic knowledge in mathematics, in particular from calculus, linear algebra, and probability theory. Although the book material is almost self-contained with regard to the mathematical concepts introduced, an introductory mathematics book like the one by Pemberton and Rau (2016)[2] is recommended for further details if needed.

Given this approach, this book targets students, academics, and professionals alike who want to learn about financial theory, financial data modeling, and the use of Python for computational finance. It is a systematic introduction to the field on which to build through more advanced books or training programs. Readers with a

2 Find the full reference for this title in Chapter 7.

formal financial background will find the mathematical and financial elements of the book rather simple and straightforward. On the other hand, readers with a stronger programming background will find the Python elements rather simple and easy to understand.

Even if the reader does not intend to move on to more advanced topics in computational finance, algorithmic trading, or asset management, the Python and finance skills acquired through this book can be applied beneficially to standard problems in finance, such as the composition of investment portfolios according to modern portfolio theory (MPT). This book also teaches, for example, how to value options and other derivatives by standard methods such as replication portfolios or risk-neutral pricing.

This book is also suitable for executives in the financial industry who want to learn about the Python programming language as applied to finance. On the other hand, it can also be read by those already proficient in Python or another programming language who want to learn more about the application of Python in finance.

Overview of the Book

The book consists of the following chapters:

Chapter 1
: The first chapter sets the stage for the rest of the book. It provides a concise history of finance, explains the book's approach to using Python for finance, and shows how to set up a basic Python infrastructure suited to work with the code provided and the Jupyter Notebooks that accompany the book.

Chapter 2
: This chapter covers the most simple model economy, in which the analysis of finance under uncertainty is possible: there are only two relevant dates and two uncertain future states possible. One sometimes speaks of a *static two-state economy*. Despite its simplicity, the framework allows the introduction of such basic notions of finance as net present value, expected return, volatility, contingent claims, option replication, arbitrage pricing, martingale measures, market completeness, risk-neutral pricing, and mean-variance portfolios.

Chapter 3
: This chapter introduces a third uncertain future state to the model, analyzing a *static three-state economy*. This allows us to analyze such notions as market incompleteness, indeterminacy of martingale measures, super-replication of contingent claims, and approximate replication of contingent claims. It also introduces the Capital Asset Pricing Model as an equilibrium pricing approach for financial assets.

Chapter 4
: In this chapter, agents with their individual decision problems are introduced. The analysis in this chapter mainly rests on the dominating paradigm in finance for decision making under uncertainty: *expected utility maximization*. Based on a so-called representative agent, equilibrium notions are introduced, and the connection between optimality and equilibrium on the one hand and martingale measures and risk-neutral pricing on the other hand are illustrated. The representative agent is also one way of overcoming the difficulties that arise in economies with incomplete markets.

Chapter 5
: This chapter generalizes the previous notions and results in a setting with a finite, but possibly large, number of uncertain future states. It requires a bit more mathematical formalism to analyze this *general static economy*.

Chapter 6
: Building on the analysis of the general static economy, this chapter introduces dynamics to the financial modeling arsenal—to analyze two special cases of a dynamic economy in discrete time. The basic insight is that uncertainty about future states of an economy in general resolves gradually over time. This can be modeled by the use of stochastic processes, an example of which is the binomial process that can be represented visually by a binomial tree.

Chapter 7
: The final chapter provides a wealth of additional resources to explore in the fields of mathematics, financial theory, and Python programming. It also provides guidance on how to proceed after the reader has finished this book.

Conventions Used in This Book

The following typographical conventions are used in this book:

Italic
: Indicates new terms, URLs, email addresses, filenames, and file extensions.

`Constant width`
: Used for program listings, as well as within paragraphs to refer to program elements such as variable or function names, databases, data types, environment variables, statements, and keywords.

`Constant width bold`
: Shows commands or other text that should be typed literally by the user.

`Constant width italic`
: Shows text that should be replaced with user-supplied values or with values determined by context.

This element signifies a general note.

This element indicates a warning or caution.

This element indicates important information.

Using Code Examples

Supplemental material (code examples, exercises, etc.) is available for download at *https://finpy.pqp.io*.

If you have a technical question or a problem using the code examples, please send email to *bookquestions@oreilly.com*.

This book is here to help you get your job done. In general, if example code is offered with this book, you may use it in your programs and documentation. You do not need to contact us for permission unless you're reproducing a significant portion of the code. For example, writing a program that uses several chunks of code from this book does not require permission. Selling or distributing examples from O'Reilly books does require permission. Answering a question by citing this book and quoting example code does not require permission. Incorporating a significant amount of example code from this book into your product's documentation does require permission.

We appreciate, but generally do not require, attribution. An attribution usually includes the title, author, publisher, and ISBN. For example, this book would be attributed as "*Financial Theory with Python* by Yves Hilpisch (O'Reilly). Copyright 2022 Yves Hilpisch, 978-1-098-10435-1."

If you feel your use of code examples falls outside fair use or the permission given above, feel free to contact us at *permissions@oreilly.com*.

O'Reilly Online Learning

For more than 40 years, *O'Reilly Media* has provided technology and business training, knowledge, and insight to help companies succeed.

Our unique network of experts and innovators share their knowledge and expertise through books, articles, and our online learning platform. O'Reilly's online learning platform gives you on-demand access to live training courses, in-depth learning paths, interactive coding environments, and a vast collection of text and video from O'Reilly and 200+ other publishers. For more information, visit *http://oreilly.com*.

How to Contact Us

Please address comments and questions concerning this book to the publisher:

O'Reilly Media, Inc.
1005 Gravenstein Highway North
Sebastopol, CA 95472
800-998-9938 (in the United States or Canada)
707-829-0515 (international or local)
707-829-0104 (fax)

We have a web page for this book, where we list errata, examples, and any additional information. You can access this page at *https://oreil.ly/fin-theory-with-python*.

Email *bookquestions@oreilly.com* to comment or ask technical questions about this book.

For news and information about our books and courses, visit *http://oreilly.com*.

Find us on Facebook: *http://facebook.com/oreilly*.

Follow us on Twitter: *http://twitter.com/oreillymedia*.

Watch us on YouTube: *http://www.youtube.com/oreillymedia*.

Acknowledgments

This book has benefited from valuable feedback by delegates of our Certificate Programs in Python for Finance. They have pointed out numerous improvements over time.

I am thankful for several helpful comments that I have received from the technical reviewers.

I am also grateful for the help and support that I have experienced from the whole O'Reilly team.

I dedicate this book to my wife Sandra. You are the love of my life.

CHAPTER 1
Finance and Python

> The history of finance theory is an interesting example of the interaction between abstract theorizing and practical application.
>
> —Frank Milne (1995)

> Hedge funds have sucked in tens of billions of dollars in investments in recent years, assisted increasingly by technology. The same tech is also benefiting those people who make the financial decisions at these organisations.
>
> —Laurence Fletcher (2020)

This chapter gives a concise overview of topics relevant for the book. It is intended to provide both the financial and technological framework for the chapters to follow. "A Brief History of Finance" on page 2 starts by giving a brief overview of the history and current state of finance. "Major Trends in Finance" on page 3 discusses the major trends that have been driving the evolution of finance over time: mathematics, technology, data, and artificial intelligence. Against this background, "A Four-Languages World" on page 4 argues that finance today is a discipline of four closely interconnected types of languages: English, finance, mathematics, and programming. The overall approach of the book is explained in "The Approach of This Book" on page 5. "Getting Started with Python" on page 8 illustrates how an appropriate Python environment can be installed on the reader's computer. However, all the code can be used and executed via a regular web browser on the Quant Platform so that a local Python installation can be set up later.

A Brief History of Finance

To better understand the current state of finance and the financial industry, it is helpful to have a look at how they have developed over time. The history of finance as a scientific field can be divided roughly into three periods according to Rubinstein (2006):

The ancient period (pre-1950)
: A period mainly characterized by informal reasoning, rules of thumb, and the experience of market practitioners.

The classical period (1950–1980)
: A period characterized by the introduction of formal reasoning and mathematics to the field. Specialized models (for example, Black and Scholes's (1973) option pricing model) as well as general frameworks (for example, Harrison and Kreps's (1979) risk-neutral pricing approach) were developed during this period.

The modern period (1980–2000)
: This period generated many advances in specific subfields of finance (for example, computational finance) and tackled, among others, important empirical phenomena in the financial markets, such as stochastic interest rates (for example, Cox, Ingersoll, and Ross (1985)) or stochastic volatility (for example, Heston (1993)).

Fifteen years after the publication of the Rubinstein (2006) book, we can add fourth and fifth periods today. These two periods are responsible for the rise and the current omnipresence of Python in finance:

The computational period (2000–2020)
: This period saw a shift from a theoretical focus in finance to a computational one, driven by advances in both hardware and software used in finance. The paper by Longstaff and Schwartz (2001)—providing an efficient numerical algorithm to value American options by Monte Carlo simulation—illustrates this paradigm shift quite well. Their algorithm is computationally demanding in that hundreds of thousands of simulations and multiple ordinary least-squares regressions are required in general to value only a single option (see Hilpisch (2018)).

The artificial intelligence period (post-2020)
: Advances in artificial intelligence (AI) and related success stories have spurred interest to make use of the capabilities of AI in the financial domain. While there are already successful applications of AI in finance (see Hilpisch (2020)), it can be assumed that from 2020 onward there will be a systematic paradigm shift toward *AI-first finance*. AI-first finance describes the shift from simple, in general linear, models in finance to the use of advanced models and algorithms from AI

—such as deep neural networks or reinforcement learning—to capture, describe, and explain financial phenomena.

Major Trends in Finance

Like many other subjects and industries, finance has become a more formalized scientific discipline over time, driven by the increasing use of formal mathematics, advanced technology, increasing data availability, and improved algorithms, such as those from AI. Taken together, the evolution of finance over time can therefore be characterized by four major trends:

Mathematics
: Starting in the 1950s with the classical period, finance has become a more and more formalized discipline, making systematic use of different fields in mathematics, like linear algebra or stochastic calculus. The mean-variance portfolio (MVP) theory by Markowitz (1952) can be considered a major breakthrough in quantitative finance if not its starting point itself—leaving the ancient period characterized mainly by informal reasoning behind.

Technology
: The widespread availability and use of personal computers, workstations, and servers, starting mainly in the late 1980s and early 1990s, brought more and more technology to the field. While compute power and capacity in the beginning were rather limited, they have reached levels as of today that allow us to attack even the most complex problems in finance by sheer brute force, often rendering the search for rather specialized, efficient models and methods—that characterized the classical and modern periods—obsolete. The credo has become "Scale your hardware and use modern software in combination with appropriate numerical methods." On the other hand, the modern hardware found in most dorm and living rooms is already powerful enough that even high-performance approaches, like parallel processing, can generally be used on such commodity hardware—lowering the barriers of entry to computational and AI-first finance tremendously.

Data
: While researchers and practitioners alike mainly relied on printed financial information and data in the ancient and classical periods (think of the *Wall Street Journal* or the *Financial Times*), electronic financial data sets have become more widely available starting in the modern period. However, the computational period has seen an explosion in the availability of financial data. High-frequency intraday data sets have become the norm and have replaced end-of-day closing prices as the major basis for empirical research. A single stock might generate intraday data sets with well over 100,000 data points every trading day—this number is roughly the equivalent of 400 years' worth of end-of-day closing prices

for the same stock (250 trading days per year times 400 years). Even more recently, a proliferation in open or free data sets has been observed, which also significantly lowers the barriers of entry to computational finance, algorithmic trading, or financial econometrics.

Artificial intelligence
: The availability of ever more financial data ("big financial data") makes the application of AI algorithms—such as those from machine learning, deep learning, or reinforcement learning (see Hilpisch (2020))—not only possible but also in many cases these days necessary. Traditional statistical methods from financial econometrics are often not suited anymore to cope with today's complexities in financial markets. Faced with nonlinear, multidimensional, ever-changing financial environments, AI-based algorithms might often be the only option to discover relevant relationships and patterns, generate valuable insights, and benefit from improved prediction capabilities.

By reading this book, the reader lays the foundations in the areas of financial mathematics and modern technology used to implement formal financial models. The reader also acquires skills to work with typical financial data sets encountered in finance. Taken together, this prepares the reader to later on also explore more easily advanced topics in computational finance or AI as applied to finance.

Python and Finance

More and more, finance has become a field driven by computationally demanding algorithms, ever-increasing data availability, and AI. Python has proven to be the right programming language and technology platform to address the requirements and challenges that arise from the major trends observed in the field.

A Four-Languages World

Against this background, finance has become a world of four languages:

Natural language
: Today, the *English* language is the only relevant language in the field when it comes to published research, books, articles, or news.

Financial language
: Like every other field, *finance* has technical terms, notions, and expressions that describe certain phenomena or ideas that are usually not relevant in other domains.

Mathematical language
: *Mathematics* is the tool and language of choice when it comes to formalizing the notions and concepts of finance.

Programming language
: As the quote at the beginning of the preface points out, *Python* as a programming language has become the language of choice in many corners of the financial industry.

The mastery of finance therefore requires both the academic and practitioner to be fluent in all four languages: English, finance, mathematics, and Python. This is *not* to say that, for instance, English and Python are the *only* relevant natural or programming languages. It is rather the case that if you have only a limited amount of time to learn a programming language, you should most probably focus on Python—alongside mathematical finance—on your way to mastery of the field.

The Approach of This Book

How does this book approach the four languages needed in finance? The English language is a no-brainer—you are reading it already. Yet, three remain.

For example, this book cannot introduce every single piece of mathematics in detail that is needed in finance. Nor can it introduce every single concept in (Python) programming in detail that is needed in computational finance. However, it tries to introduce related concepts from finance, mathematics, and programming alongside one another whenever possible and sensible.

From Chapter 2 onward, the book introduces a financial notion or concept and then illustrates it on the basis of both a mathematical representation and the implementation in Python. As an example, have a look at the following table from Chapter 3. The table lists the financial topic, the major mathematical elements, and the major Python data structure used to implement the financial mathematics:

Finance	Mathematics	Python
Uncertainty	Probability space	`ndarray`
Financial assets	Vectors, matrices	`ndarray`
Attainable contingent claims	Span of vectors, basis of vector space	`ndarray`

The following is a walkthrough of one specific example, details of which are provided in later chapters. The example is only for illustration of the general approach of the book at this point.

As an example, take the central concept of *uncertainty* in finance from the preceding table. Uncertainty embodies the notion that future states of a model economy are not

known in advance. Which future state of the economy unfolds might be important, for example, to determine the payoff of a European call option. In a discrete case, one deals with a finite number of such states, like two, three, or more. In the most simple case of two future states only, the payoff of a European call option is represented mathematically as a *random variable*, which in turn can be represented formally as a *vector v* that is itself an element of the *vector space* $\mathbb{R}^2$. A vector space is a collection of objects—called vectors—for which addition and scalar multiplication are defined. Formally, one writes for such a vector v, for example:

$$v = \begin{pmatrix} v^u \\ v^d \end{pmatrix} \in \mathbb{R}^2_{\geq 0}$$

Here, both elements of the vector are assumed to be non-negative real numbers $v^u, v^d \in \mathbb{R}_{\geq 0}$. More concretely, if the uncertain, state-dependent price of the stock on which the European call option is written is given in this context by

$$S = \begin{pmatrix} 20 \\ 5 \end{pmatrix} \in \mathbb{R}^2_{\geq 0}$$

and the strike price of the option is $K = 15$, the payoff C of the European call option is given by

$$C = \max(S - K, 0) = \begin{pmatrix} \max(20 - 15, 0) \\ \max(5 - 15, 0) \end{pmatrix} = \begin{pmatrix} 5 \\ 0 \end{pmatrix} \in \mathbb{R}^2_{\geq 0}$$

This illustrates how the notions of the *uncertain price of a stock* and the *state-dependent payoff of a European option* can be modeled mathematically as a vector. The discipline dealing with vectors and vector spaces in mathematics is called *linear algebra*.

How can all this be translated into Python programming? First, *real numbers* are represented as *floating point numbers* or `float` objects in Python:

```
In [1]: vu = 1.5  ❶

In [2]: vd = 3.75  ❷

In [3]: type(vu)  ❸
Out[3]: float

In [4]: vu + vd  ❹
Out[4]: 5.25
```

❶ Defines a variable with the name `vu` and the value 1.5.

❷ Defines a variable with the name `vd` and the value 3.75.

❸ Looks up the type of the `vu` object—it is a `float` object.

❹ Adds up the values of `vu` and `vd`.

Second, one usually calls collections of objects of the same type in programming *arrays*. In Python, the package `NumPy` provides support for such data structures. The major data structure provided by this package is called `ndarray`, which is an abbreviation for *n*-dimensional array. Real-valued vectors are straightforward to model with `NumPy`:

```
In [5]: import numpy as np  ❶

In [6]: v = np.array((vu, vd))  ❷

In [7]: v  ❸
Out[7]: array([1.5 , 3.75])

In [8]: v.dtype  ❹
Out[8]: dtype('float64')

In [9]: v.shape  ❺
Out[9]: (2,)

In [10]: v + v  ❻
Out[10]: array([3. , 7.5])

In [11]: 3 * v  ❼
Out[11]: array([ 4.5 , 11.25])
```

❶ Imports the `NumPy` package.

❷ Instantiates an `ndarray` object.

❸ Prints out the data stored in the object.

❹ Looks up the data type for all elements.

❺ Looks up the shape of the object.

❻ Vector addition illustrated.

❼ Scalar multiplication illustrated.

This shows how the mathematical concepts surrounding vectors are represented and applied in Python. It is then only one step further to apply those insights to finance:

```
In [12]: S = np.array((20, 5))  ❶

In [13]: K = 15  ❷

In [14]: C = np.maximum(S - K, 0)  ❸

In [15]: C  ❹
Out[15]: array([5, 0])
```

❶ Defines the uncertain price of the stock as an `ndarray` object.

❷ Defines the strike price as a Python variable with an integer value (`int` object).

❸ Calculates the maximum expression element-wise.

❹ Shows the resulting data now stored in the `ndarray` object `C`.

This illustrates the style and approach of this book:

1. Notions and concepts in finance are introduced.
2. A mathematical representation and model is provided.
3. The mathematical model is translated into executable Python code.

In that sense, finance motivates the use of mathematics, which in turn motivates the use of Python programming techniques.

Getting Started with Python

One of the benefits of Python is that it is an open source language, which holds true for the absolute majority of important packages as well. This allows for easy installation of the language and required packages on all major operating systems, such as macOS, Windows, and Linux. There are only a few major packages that are required for the code of this book and finance in general in addition to a basic Python interpreter:

`NumPy`
: This package allows the efficient handling of large, *n*-dimensional numerical data sets.

`pandas`
: This package is primarily for the efficient handling of tabular data sets, such as financial time series data. Although not required for the purposes of this book, `pandas` has become one of the most popular Python packages in finance.

`SciPy`
: This package is a collection of scientific functions that are required, for example, to solve typical optimization problems.

`SymPy`
: This package allows for symbolic mathematics with Python, which sometimes comes in handy when dealing with financial models and algorithms.

`matplotlib`
: This package is the standard package in Python for visualization. It allows you to generate and customize different types of plots, such as line plots, bar charts, and histograms.

Similarly, there are only two tools that are required to get started with interactive Python coding:

IPython
: This is the most popular environment in which to do interactive Python coding on the command line (terminal, shell).

JupyterLab
: This is the interactive development environment in which to do interactive Python coding and development in the browser.

The technical prerequisites to follow along with regard to Python programming are minimal. There are basically two options for making use of the Python code in this book:

Quant Platform
: On the Quant Platform, for which you can sign up for free, you find a full-fledged environment for interactive financial analytics with Python. This allows you to make use of the Python code provided in this book via the browser, making a local installation unnecessary. After signing up for free, you have automatic access to all code and all Jupyter Notebooks that accompany the book, and you can execute the code right away in the browser.

Local Python environment
: It is also straightforward nowadays to install a local Python environment that allows you to dive into financial analytics and the book code on your own computer. This section describes how to do this.

About the Author

Dr. Yves J. Hilpisch is founder and CEO of The Python Quants, a group focusing on the use of open source technologies for financial data science, artificial intelligence, algorithmic trading, computational finance, and asset management. He is also founder and CEO of The AI Machine, a company focused on AI-powered algorithmic trading via a proprietary strategy-execution platform.

In addition to this book, he is the author of the following books:

- *Artificial Intelligence in Finance* (O'Reilly, 2020)
- *Python for Algorithmic Trading* (O'Reilly, 2020)
- *Python for Finance* (2nd ed., O'Reilly, 2018)
- *Listed Volatility and Variance Derivatives* (Wiley, 2017)
- *Derivatives Analytics with Python* (Wiley, 2015)

Yves is an adjunct professor of Computational Finance and lectures on Algorithmic Trading at the CQF Program. He is also the director of the first online training programs leading to University Certificates in Python for Algorithmic Trading, Python for Computational Finance, and Python for Asset Management, respectively.

Yves wrote the financial analytics library *DX Analytics* and organizes meetups, conferences, and bootcamps about Python for quantitative finance and algorithmic trading in London, Frankfurt, Berlin, Paris, and New York. He has given keynote speeches at technology conferences in the United States, Europe, and Asia.

Colophon

The animal on the cover of *Financial Theory with Python* is a crowned moon snake (*Furina ornata*). More commonly known as the orange-naped snake, this small venemous snake is native to northern and northwestern Australia. It is commonly identified by the red blotch on its nape that is not completely enclosed by the black bands above and below it.

The crowned moon snake's conservation status is "Least Concern." Many of the animals on O'Reilly covers are endangered; all of them are important to the world.

The cover illustration is by Karen Montgomery, based on a black and white engraving from Lydekker's *Royal Natural History*. The cover fonts are Gilroy Semibold and Guardian Sans. The text font is Adobe Minion Pro; the heading font is Adobe Myriad Condensed; and the code font is Dalton Maag's Ubuntu Mono.