Finance with Python

The Python Quants GmbH <training@tpq.io>

Table of Contents

Copyright

Preface

> *Like modern commodity analysts, the trade finance bankers of the future are probably going to be python coders.*[1]
>
> — *Etienne Amic (2020)*

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 (http://python.org) programming language from ground up. It presents all 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. This has changed somewhat recently in that programming classes (e.g. in C++) have become an integral part of Master of Financial Engineering and similar university programs.

However, *mathematical foundations*, *theoretical finance* and *basic programming techniques* are still quite often taught independent from each other and only later on combined to *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.

Target Audience

I have written a number of book about Python applied to finance. My company, The Python Quants offers a number of live and online training classes in Python for finance. All my books and the training classes expect the readers and participants to have already some decent background

knowledge in both finance and Python programming or a similar language.

This book starts completely from scratch, just expecting 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, it is recommended to use an introductory mathematics book like the one by Pemberton and Rau (2016) for references if needed.

Given this approach, the book targets students, academics, and professionals alike who want to learn (more) about financial theory, data analysis, and the use of Python for computational finance. It is a perfect introduction to the field on which to build through more advanced books or training programs offered by The Python Quants and others.

Overview of the Book

The book consists of the following chapters:

Finance and Python

The first chapter sets the stage for the rest of the book. It provides a concise history of finance, explains the approach of the book take towards using Python for finance, and shows how to set up a basic Python infrastructure suited to work with the code provided in the book and the Jupyter notebooks that accompany the book. The first chapter also provides a comprehensive overview of the literature referenced in the book or useful for a more detailed study of the different topics covered in the book.

Two State Economy

The 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 to introduce such basic notions of finance as net present value, expected return, volatility, contingent claims, option replication, arbitrage pricing, martingale measure, market completeness, risk-neutral pricing and mean-variance portfolios.

Three State Economy

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

Optimality and Equilibrium

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.

Static Economy

This chapter generalizes the previous notions and results to 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*.

Dynamic Economy

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.

1. Finance and Python

> *Python is now wide-spread across investment banking and hedge funds. Banks use Python for pricing, risk management and trade management platforms. More recently, they've been reprogramming their trading systems to run off Python rather than other, clunkier languages.*
>
> — *efinancialcareers (2016)*

1.1. Introduction

This chapter gives a concise overview of topics relevant for the course Finance with Python. It is intended to provide both the financial and technological framework for the chapters to follow.

1.2. A Brief History of Finance

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 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 (1973) option pricing model) as well as general frameworks (for example, Harrison and Kreps (1979) risk-neutral pricing approach) have been developed during this period.
- **The modern period (1980-2000)**: This period has generated many advances in specific sub-fields of finance (for example, computational finance) and has 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)).

One might add fourth and fifth ones today:

- **The computational period (2000-2020)**: This period saw a shift from a theoretical focus in finance towards 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 100,000s of simulations and multiple ordinary least-squares regressions are required in general to value a single option only.

- **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 onwards there will be a systematic paradigm shift towards *AI-first finance*.

The evolution of finance over time is 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 wide-spread availability and use of personal computers, work stations 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 beginnings were rather limited, they have reached levels as of today that allow to attack even the most complex problems in finance by sheer brute force, rendering the search for rather specialized, efficient models and methods — that characterized the classical and modern periods — often obsolete. The credo has become: "Scale your hardware and use modern software in combination with appropriate numerical methods." On the other hand, modern hardware found in most dorm and living rooms is already that powerful 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 (252 trading days per year times 40 years). Aven 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 non-linear, multi-dimensional, changing financial environments, AI-based algorithms might often be the only option to discover relevant relationships and patterns, to generate valuable insights and benefit from improved prediction capabilities.

1.3. A Four Languages World

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

- **Natural language**: The *English* language is today 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 probably not seen in many other ares
- **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 this chapter points out, *Python* (http://python.org) 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 student and practitioner to be fluent in all four languages: English, finance, mathematics and Python. This is *not* so say that, for instance, English and Python are the *only* relevant natural or programming languages. It is rather the case that if you only have 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.

1.4. The Approach of this Book

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

For example, this course 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 needed in computational finance. However, it tries to introduce related concepts from finance, mathematics and programming alongside each other whenever possible and sensible.

For example, take the central concept of *uncertainty* in finance. It 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. One writes for such a vector for example

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

Here, both elements of the vector are positive 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

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

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

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

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.

PYTHON

```
In [1]: vu = 1.5   1

In [2]: vd = 3.75   2

In [3]: type(vu)   3
Out[3]: float

In [4]: vu + vd   4
Out[4]: 5.25
```

1. Defines a variable with name `vu` and value 1.5.
2. Defines a variable with name `vd` and value 3.75.
3. Looks up the type of the `vu` object — it is a `float` object.
4. Adds up the values of `vu` and `vd`.

Second, one calls collections of objects of the same type in programming usually *arrays*. In Python, the package `NumPy` (http://numpy.prg) 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` .

PYTHON

```
In [5]: import numpy as np   1

In [6]: v = np.array((vu, vd))   2

In [7]: v   3
Out[7]: array([1.5 , 3.75])

In [8]: v.dtype   4
Out[8]: dtype('float64')

In [9]: v.shape   5
Out[9]: (2,)

In [10]: v + v   6
Out[10]: array([3. , 7.5])

In [11]: 3 * v   7
Out[11]: array([ 4.5 , 11.25])
```

1. Imports the `NumPy` package.
2. Instantiates a `ndarray` object.
3. Prints out the data stored in the object.
4. Looks up the data type for all elements.
5. Looks up the shape of the object.
6. Vector addition illustrated.
7. 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.

PYTHON

```
In [12]: S = np.array((20, 5))   1

In [13]: K = 15   2

In [14]: C = np.maximum(S - K, 0)   3

In [15]: C   4
Out[15]: array([5, 0])
```

1 Defines the uncertain price of the stock as a `ndarray` object.
2 Defines the strike price as a Python variable with an integer value (`int` object).
3 Calculates the maximum expression element-wise.
4 Shows the resulting data now stored in the `ndarray` object `C`.

This illustrates the style and approach of this course:

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.

1.5. Getting Started with Python

The technical prerequisites to follow along with regard to Python programming are minimal. There are basically two options of how to make use of the Python codes:

- **Quant Platform**: On the Quant Platform (http://pyesfi.pqp.io) 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. When you have signed up for free on this platform, you have been given access automatically to all code and all Jupyter Notebooks that accompany the book.
- **Local Python environment**: It is also straightforward to install a local Python environment that allows you to dive into financial analytics and the book code on your own computer. How to do this is what this section describes.

An easy and modern way of installing Python is by the use of the `conda` (http://conda.io) package and environment manager (see `conda` Web page).

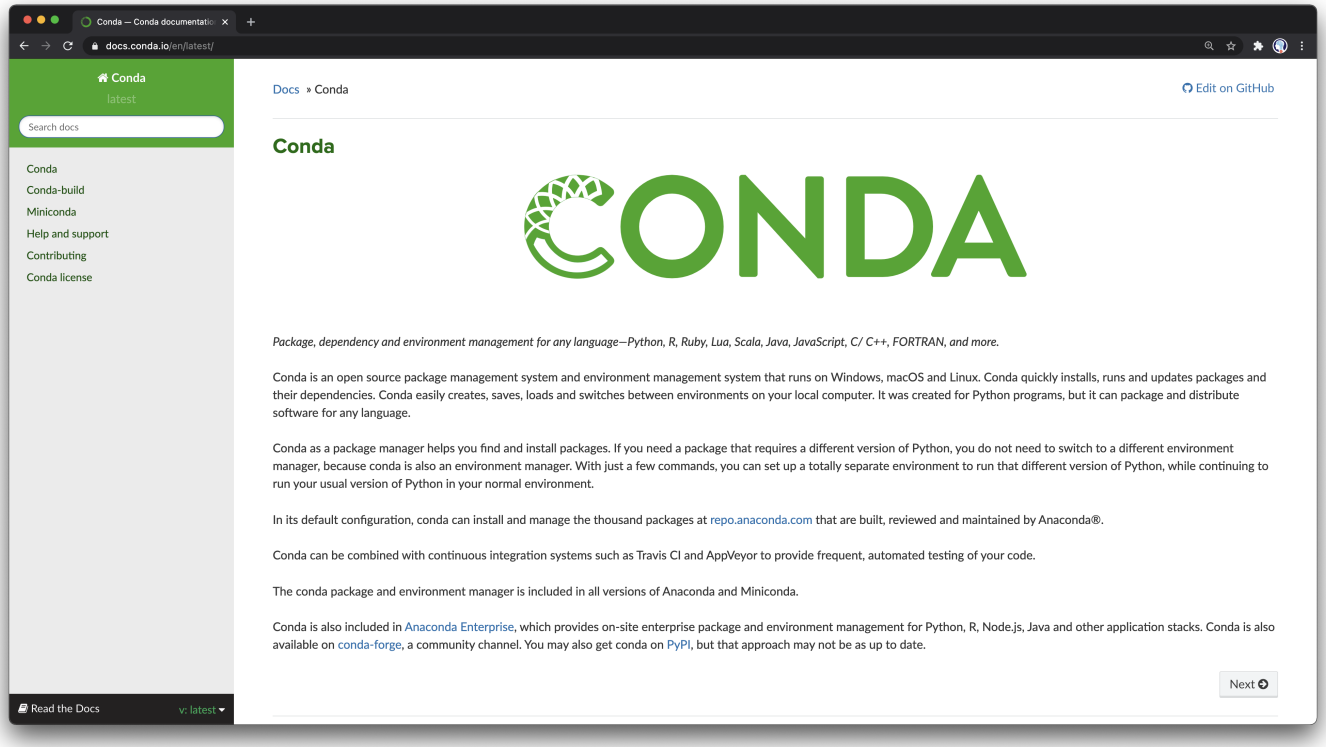

Figure 1. `conda` *Web page*

The most efficient way to install `conda` and a basic Python interpreter is via the Miniconda (https://conda.io/miniconda.html) distribution. On the `Miniconda` download page https://conda.io/miniconda.html, installer packages for the most important operating systems and Python versions are provided (see `Miniconda` download page).

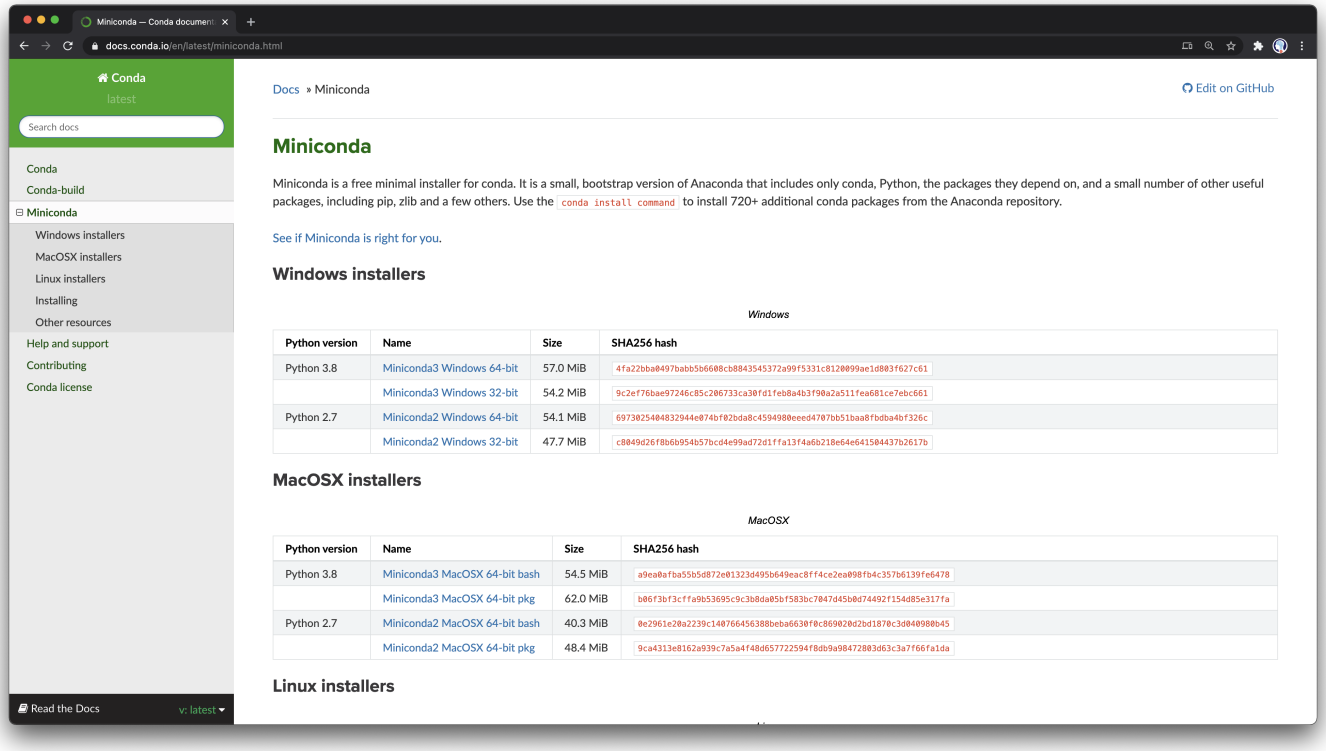

Figure 2. `Miniconda` *download page*

After having installed `Miniconda` according to the guidelines provided for your operating system, you should open a shell or command prompt and check whether `conda` is available. You should get an output similar to this:

```
pro:pyesfi yves$ conda --version
conda 4.9.2
pro:pyesfi yves$
```

The next step is to create a new *Python environment* as follows (and to answer "y" when prompted):

```
pro:pyesfi yves$ conda create --name pyesfi python=3.9
...
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate pyesfi
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

After the successful completion, activate the environment as follows:

```
pro:pyesfi yves$ conda activate pyesfi
(pyesfi) pro:pyesfi yves$
```

Notice how the prompt changes. Next, install the required tools IPython and Jupyter Lab as follows (and answer “y” when prompted):

```
(pyesfi) pro:pyesfi yves$ conda install ipython jupyterlab
...
```

After that, you should install the major Python packages used for financial data science as follows (the flag `-y` avoids the confirmation prompt):

```
(pyesfi) pro:pyesfi yves$ conda install -y numpy pandas matplotlib scipy sympy
...
```

This provides the most important Python packages for data analysis in general and financial analytics in particular. You might check whether everything has been installed as follows:

```
(pyesfi) pro:pyesfi yves$ conda list
# packages in environment at /Users/yves/miniconda3/envs/pyesfi:
#
# Name                    Version                   Build  Channel
anyio                     2.0.2           py39h6e9494a_2    conda-forge
appnope                   0.1.2           py39h6e9494a_0    conda-forge
argon2-cffi               20.1.0          py39h5a22ff9_2    conda-forge
async_generator           1.10                       py_0    conda-forge
attrs                     20.3.0           pyhd3deb0d_0    conda-forge
babel                     2.9.0            pyhd3deb0d_0    conda-forge
backcall                  0.2.0            pyh9f0ad1d_0    conda-forge
backports                 1.0                        py_2    conda-forge
...
webencodings              0.5.1                      py_1    conda-forge
wheel                     0.36.2           pyhd3deb0d_0    conda-forge
xz                        5.2.5              haf1e3a3_1    conda-forge
zeromq                    4.3.3              h74dc148_3    conda-forge
zipp                      3.4.0                      py_0    conda-forge
zlib                      1.2.11           h7795811_1010    conda-forge
zstd                      1.4.5              h289c70a_2    conda-forge
(pyesfi) pro:pyesfi yves$
```

An interactive Python session is then started by simply typing `python`.

```
(pyesfi) pro:pyesfi yves$ python
Python 3.9.1 | packaged by conda-forge | (default, Dec 21 2020, 22:06:14)
[Clang 11.0.0 ] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hello Finance World.')
Hello Finance World.
>>> exit()
(pyesfi) pro:pyesfi yves$
```

A better interactive shell is provided by IPython which is started via `ipython` on the shell.

```
(pyesfi) pro:pyesfi yves$ ipython
Python 3.9.1 | packaged by conda-forge | (default, Dec 21 2020, 22:06:14)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.19.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: import numpy as np

In [2]: np.random.random(10)
Out[2]:
array([0.29544518, 0.42983479, 0.04673849, 0.59572647, 0.38915588,
       0.62393223, 0.34299427, 0.79903732, 0.1940799 , 0.95673132])

In [3]: exit
(pyesfi) pro:pyesfi yves$
```

However, it is recommended — especially for the Python beginner — to work with Jupyter Lab in the browser. To this end, type `jupyter lab` on the shell which should give an output similar to the following:

```
pro:pyesfi yves$ jupyter lab
[I 2020-12-30 13:48:23.281 ServerApp] jupyterlab | extension was successfully linked.
...
[I 2020-12-30 13:48:23.600 ServerApp] Jupyter Server 1.1.3 is running at:
[I 2020-12-30 13:48:23.600 ServerApp] http://localhost:8888/lab?token=107dbf...
[I 2020-12-30 13:48:23.600 ServerApp]  or http://127.0.0.1:8888/lab?token=107db...
[I 2020-12-30 13:48:23.600 ServerApp] Use Control-C to stop this server and ...
```

In general, a new browser tab is opened automatically which then shows you the starting page of Jupyter similar to Jupyter Lab start page.

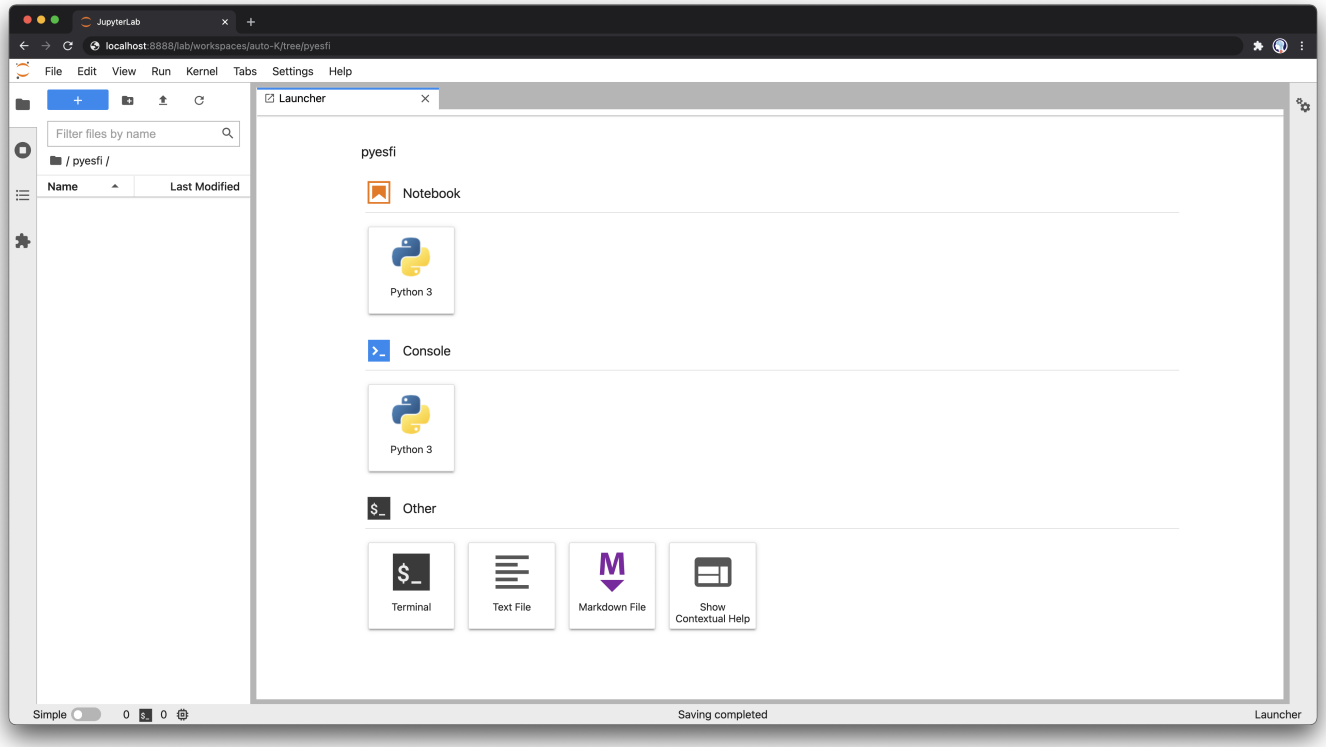

Figure 3. Jupyter Lab start page

You can then open a new Jupyter Notebook and start with interactive Python coding as shown in New Jupyter Notebook. To write code in a cell click on the cell. To execute the code, use `shift+return`, `ctrl+return` or `alt+return` (you will notice the difference).

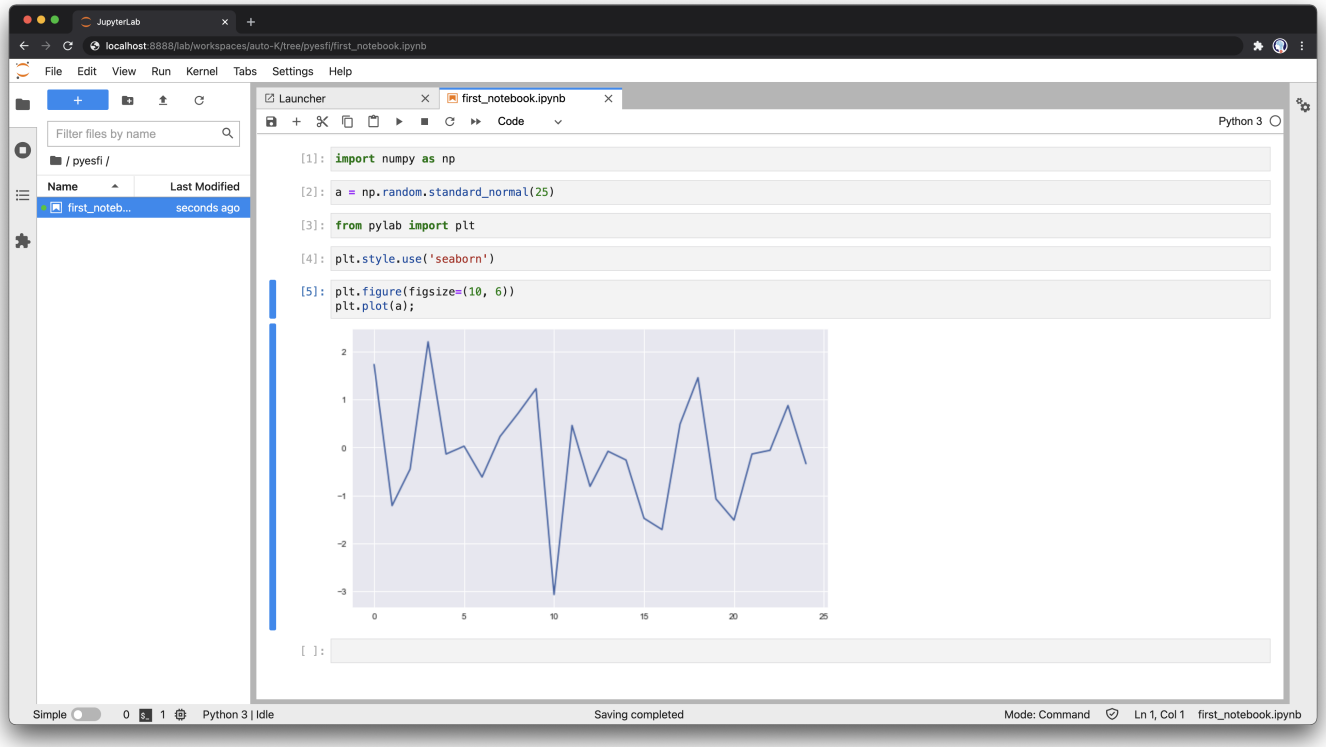

Figure 4. New Jupyter Notebook

You can also open one of the Jupyter Notebook files as provided with this book (see Jupyter Notebook accompanying the book).

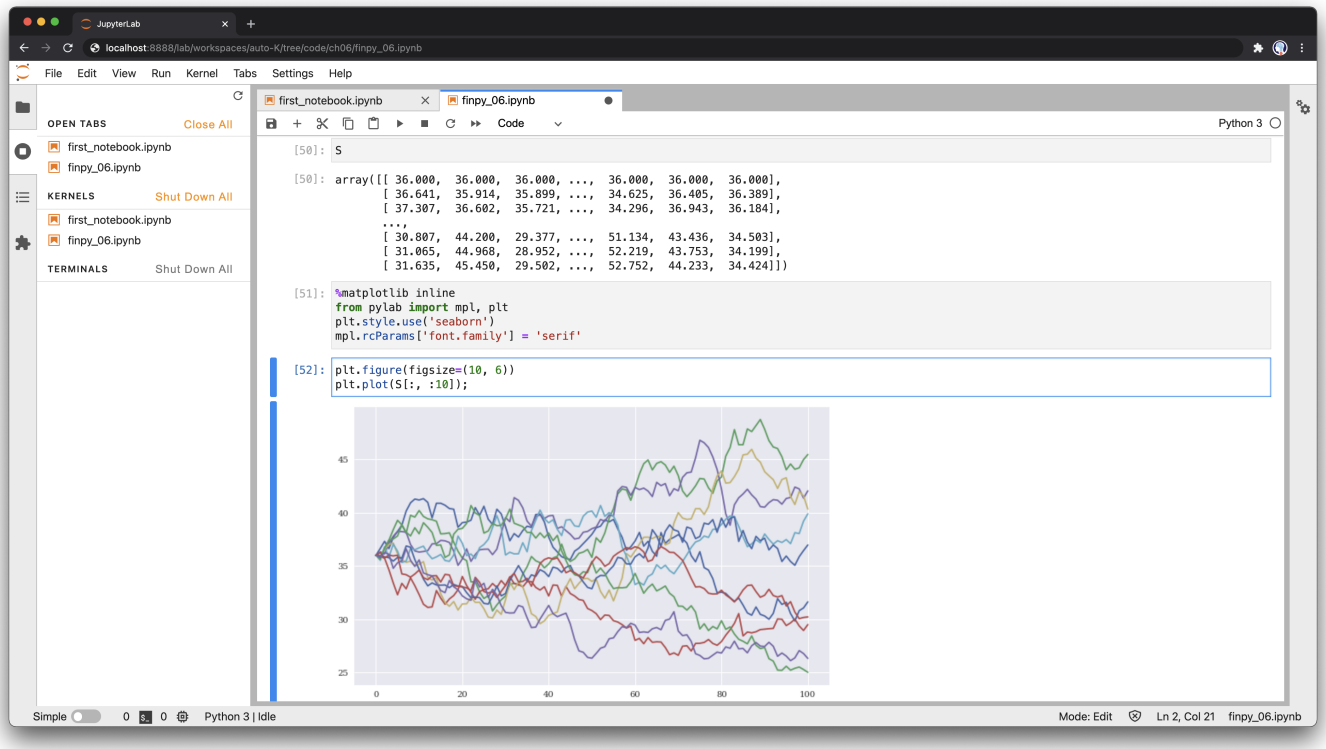

Figure 5. Jupyter Notebook accompanying the book

This section just provides the very basics to get started with Python and related tools such as IPython and Jupyter Lab. For more details — for example, about how work with IPython — refer to the book VanderPlas (2016).

1.6. Conclusions

Finance can look back on a long history. The period from 1950 to 1980 is characterized by the introduction of rigorous mathematical analysis to the field. From the 1980s onwards and in particular since 2000, the role of computers and computational finance has gained tremendously in importance. This trend will be further reinforced by the increasing role AI plays in the field, with is computationally demanding algorithms from machine learning (ML) and deep learning (DL).

The finance field makes use of four different types of language: *natural language* (English in general), *financial language* (notions and expressions special to the field), *mathematical language* (like linear algebra or probability theory) as well as *programming language* (like Python for the purposes of this book).

The approach of this book is to introduce related concepts from finance, mathematics, and Python programming alongside each other. The necessary prerequisites on the Python side are minimal, with the `conda` package and environment manager often as the tool of choice nowadays to manage Python environments.

1.7. Bibliography

The following provides an overview of published works used for and referenced in this book. The overview is comprehensive but for sure no complete, given the breadth of topics covered in the book. The single chapters have their own reference sections as well.

Market Risk Analysis book collection:

- Alexander, Carol (2008): *Market Risk Analysis I — Quantitative Methods in Finance*. John Wiley & Sons, Chicester.
- Alexander, Carol (2008): *Market Risk Analysis II — Practical Financial Econometrics*. John Wiley & Sons, Chicester.
- Alexander, Carol (2008): *Market Risk Analysis III — Pricing, Hedging and Trading Financial Instruments*. John Wiley & Sons, Chicester.
- Alexander, Carol (2008): *Market Risk Analysis IV — Value-at-Risk Models*. John Wiley & Sons, Chicester.

Mathematical books:

- Aleskerov, Fuad, Hasan Ersel and Dmitri Piontkovski (2011): *Linear Algebra for Economists*. Springer, Heidelberg et al.
- Bhattacharya, Rabi and Edward Waymire (2007): *A Basic Course in Probability Theory*. Springer Verlag, New York.
- Jacod, Jean and Philip Protter (2004): *Probability Essentials*. Springer, Berlin and Heidelberg.
- Pemberton, Malcolm and Nicholas Rau (2016): *Mathematics for Economists — An Introductory Textbook*. 4th ed., Manchester University Press, Manchester and New York.
- Rudin,Walter (1987): *Real and Complex Analysis*. 3rd ed., McGraw-Hill, London.
- Schneider, Hans and George Barker (1973): *Matrices and Linear Algebra*. Reprint 1989, Dover Publications, New York.
- Sundaram, Rangarajan (1996): *A First Course in Optimization Theory.* Cambridge University Press, Cambridge.
- Williams, David (1991): *Probability with Martingales*. Reprint 2001, Cambridge University Press, Cambridge.

Basic Python books:

- McKinney, Wes (2017): *Python for Data Analysis*. 2nd ed., O'Reilly, Sebastopol et al.
- Ramalho, Luciano (2016): *Fluent Python*. O'Reilly, Sebastopol et al.
- Ravenscroft, Anna, Steve Holden, Alex Martelli (2017): *Python in a Nutshell*. 3rd ed., O'Reilly, Sebastopol et al.
- VanderPlas, Jake (2016): *Python Data Science Handbook*. O'Reilly, Sebastopol et al.

Python for finance books:

- Hilpisch, Yves (2020): *Artificial Intelligence in Finance*. O'Reilly, Sebastopol et al.
- Hilpisch, Yves (2018): *Python for Finance*. 2nd ed., O'Reilly, Sebastopol et al.
- Hilpisch, Yves (2015): *Derivatives Analytics with Python*. Wiley Finance.

Finance papers:

- Black, Fischer and Myron Scholes (1973): "The Pricing of Options and Corporate Liabilities." *Journal of Political Economy*, Vol. 81, No. 3, 638–659.

- Boyle, Phelim (1977): “Options: A Monte Carlo Approach.” *Journal of Financial Economics*, Vol. 4, No. 4, 322–338.
- Cox, John and Stephen Ross (1976): “The Valuation of Options for Alternative Stochastic Processes.” *Journal of Financial Economics*, Vol. 3, 145-166.
- Cox, John, Jonathan Ingersoll and Stephen Ross (1985): “A Theory of the Term Structure of Interest Rates.” *Econometrica*, Vol. 53, No. 2, 385–407.
- Cox, John, Stephen Ross and Mark Rubinstein (1979): “Option Pricing: A Simplified Approach.” *Journal of Financial Economics*, Vol. 7, No. 3, 229–263.
- Harrison, Michael and David Kreps (1979): “Martingales and Arbitrage in Multiperiod Securities Markets.” *Journal of Economic Theory*, Vol. 20, 381–408.
- Harrison, Michael and Stanley Pliska (1981): “Martingales and Stochastic Integrals in the Theory of Continuous Trading.” *Stochastic Processes and their Applications*, Vol. 11, 215–260.
- Heston, Steven (1993): “A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options.” *The Review of Financial Studies*, Vol. 6, No. 2, 327–343.
- Longstaff, Francis and Eduardo Schwartz (2001): “Valuing American Options by Simulation: A Simple Least Squares Approach.” *Review of Financial Studies*, Vol. 14, No. 1, 113–147.
- Markowitz, Harry (1952): “Portfolio Selection.” *Journal of Finance*, Vol. 7, No. 1, 77-91.
- Merton, Robert (1976): “Option Pricing when the Underlying Stock Returns are Discontinuous.” *Journal of Financial Economics*, No. 3, Vol. 3, 125-144.
- Perold, André (2004): “The Capital Asset Pricing Model.” *Journal of Economic Perspectives*, Vol. 18, No. 3, 3-24
- Protter, Philip (2001): “A Partial Introduction to Financial Asset Pricing Theory.” *Stochastic Processes and their Applications*, Vol. 91, 169–203.
- Sharpe, William (1964): “Capital Asset Prices: A Theory of Market Equilibrium under Conditions of Risk.” *The Journal of Finance*, Vol. 19, No. 3, 425-442.

Basic finance and economics books:

- Copeland, Thomas, Fred Weston and Kuldepp Shastri (2005): *Financial Theory and Corporate Policy*. 4th ed., Addison Wesley, Boston et al.
- Eichberger, Jürgen and Ian Harper (1997): *Financial Economics*. Oxford University Press, New York.
- Milne, Frank (1995): *Finance Theory and Asset Pricing*. Oxford University Press, New York.

- Markowitz, Harry (1959): *Portfolio Selection — Efficient Diversification of Investments*. John Wiley & Sons, New York et al.
- Pliska, Stanley (1997): *Introduction to Mathematical Finance.* Blackwell Publishers, Malden and Oxford.
- Rubinstein, Mark (2006): *A History of the Theory of Investments*. Wiley Finance, Hoboken.
- Varian, Hal (1992): *Microeconomic Analysis*. 3rd ed., W.W. Norton & Company, New York and London.

Advanced finance books:

- Baxter, Martin and Andrew Rennie (1996): *Financial Calculus — An Introduction to Derivative Pricing*. Cambridge University Press, Cambridge.
- Björk, Tomas (2004): *Arbitrage Theory in Continuous Time*. 2nd ed., Oxford University Press, Oxford.
- Delbaen, Freddy and Walter Schachermayer (2004): *The Mathematics of Arbitrage*. Springer Verlag, Berlin.
- Duffie, Darrell (1988): *Security Markets — Stochastic Models*. Academic Press, San Diego et al.
- Elliot, Robert and Ekkehard Kopp (2005): *Mathematics of Financial Markets.* 2nd ed., Springer Verlag, New York.
- Glasserman, Paul (2004): *Monte Carlo Methods in Financial Engineering*. Springer Verlag, New York.

Additional articles cited in this chapter:

- efinancialcareers (2016): "The Six Hottest Programming Languages to Know in Banking Technology." Online Article (http://news.efinancialcareers.com/uk-en/137065/the-six-hottest-programming-languages-to-know-in-banking-technology) .

Author Biography

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

He is also the author of the following books (http://books.tpq.io):

- Artificial Intelligence in Finance (http://aiif.tpq.io) (O'Reilly, 2020)
- Python for Algorithmic Trading (http://py4at.tpq.io) (O'Reilly, 2020)
- Python for Finance (http://py4fi.tpq.io) (2nd ed., O'Reilly, 2018),
- Derivatives Analytics with Python (http://dawp.tpq.io) (Wiley, 2015) and
- Listed Volatility and Variance Derivatives (http://lvvd.tpq.io) (Wiley, 2017).

Yves is Adjunct Professor for Computational Finance and lectures on Algorithmic Trading at the CQF Program (http://cqf.com). He is also the director of the first online training programs leading to University Certificates in Python for Algorithmic Trading (http://certificate.tpq.io) and Python for Computational Finance (http://compfinance.tpq.io).

Yves wrote the financial analytics library DX Analytics (http://dx-analytics.com) 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.

---

1. Amic Etienne (2020): "Commodity traders Need to Embrace a Digital Future." *Financial Times*, 28. May 2020.
2. Note that this calculation for the volatility is only valid since an equal probability is assumed for the three states.
3. Utility is only to be understood in *ordinal terms*, that is in terms of bringing different plans into a certain order. A comparison of this numerical value with the optimal one from before does not make any sense because the utility functions are different.
4. The notation is changed here from i to r to emphasize that the *short rate* is meant from now on.
5. The parameters assumed in this chapter are from Longstaff-Schwartz (2001, table 1).
6. Note that only the numbers on and above the diagonal are relevant. Numbers below the diagonal can be ignored. They result from the specific vectorized operations implemented on the `ndarray` object.
7. Here, as also often seen in practice, there is a large number of cases for which the option expires worthless, that is, with a payoff of 0.

Last updated 2020-12-29 21:13:39 +0100