Computational Finance – Why Python Is Taking Over

Technology plays an important role in the financial industry. However, some areas have nevertheless been dominated by human beings for quite a long time. Among these areas is trading, where real people in general have at least the final word. This might change soon, as Robin Wigglesworth on the front page of the *Financial Times* from November 23, 2015 points out: "Traders used to be first-class citizens of the financial world, but that's not true anymore. Technologists are the priority now …" In this regard, banks, hedge funds, and asset managers are competing more and more with the big Silicon Valley employers, like Google and Facebook, for the brightest IT talent available.

In this context, the open-source programming language Python is becoming an increasingly important technology – used nowadays extensively in Silicon Valley as well as in finance. There are a few good reasons why this is the case.

The Python language

Python's syntax and idioms make it particularly attractive for science in general and finance in particular. Consider the "Hello World!" example of quantitative finance, the geometric Brownian motion model of Black–Scholes–Merton (1973) describing, for instance, the evolution of a stock index. In its risk-neutral form, the stochastic differential equation (SDE) reads as follows (with the parameters and symbols defined as usual and in particular Z_t being a Brownian motion):

$$dS_t = rS_t dt + \sigma S_t dZ_t$$

An Euler discretization scheme is then given by (with z_T being a standard normally distributed random variable)

$$S_T = S_0 \exp\left(\left(r - \frac{1}{2}\sigma^2\right) T + \sigma\sqrt{T} z_T\right)$$

In LaTeX you would describe this difference equation, for example, by

```
S_T = S_0 \exp (( r - 0.5 \sigma^2 ) T + \sigma
      \sqrt{T} z_T)
```

Something the majority of quants have done many, many times when documenting their work. The Python code for the difference equation is pretty close to its LaTeX representation:

```
In [1]: from pylab import *
S_0 = 100.; r = 0.01; T = 0.5; sigma = 0.2
    # parameters
z_T = standard_normal(10000)
    # pseudo-random numbers
S_T = S_0 * exp (( r - 0.5 * sigma ** 2 )
    * T + sigma * sqrt(T) * z_T)
```

With four lines of Python code, we import the necessary functionality (mainly from the NumPy library), draw 10,000 pseudo-random numbers, and simulate 10,000 end-of-period values. This vectorized approach not only leads to concise code (without loops) but is also quite fast, since the library used is mainly implemented in C.

The Python ecosystem

There are some other languages that have a similar syntax and also use approaches like vectorization. But there is one feature that distinguishes Python from other (general-purpose) programming languages: its comprehensive ecosystem of open-source scientific libraries. To mention just a few:

- **NumPy** (fast, vectorized array operations)
- **SciPy** (collection of scientific classes/functions)
- **Cython** (static compiler for hybrid Python and C code)
- **pandas** (times series and tabular data management)
- **ibis** (Pythonic interaction with, e.g., relational databases)
- **PyTables** (hardware-bound IO operations)
- **TsTables** (high-performance tick data storage/retrieval)
- **scikit-learn** (machine learning algorithms)
- **statsmodels** (statistical classes/functions)
- **xlwings** (Python–Excel integration)

In addition, there are a wealth of libraries for efficient and nice visualizations, like matplotlib, seaborn, or plotly.

Financial libraries

When it comes to dedicated financial libraries, the number of options is also steadily growing. A few examples are:

- **DX Analytics** (global valuation of multi-risk derivatives and portfolios)
- **pyfolio** (portfolio management, performance analysis)
- **PyThalesians** (data, backtesting, trading)
- **zipline** (backtesting of trading algorithms and strategies)

However, there is no single, unified financial library available at the beginning of 2016 that has accomplished what pandas has done on the data analytics side.

Functionality-wise there is also nothing comparable, for instance, to the open-source library QuantLib (although a Python wrapper exits).

DX Analytics

DX Analytics (dx-analytics.com) is an open-source Python library for advanced financial and derivatives analytics written and maintained by The Python Quants (tpq.io). It is particularly suited to model multi-risk derivatives and to do a consistent valuation of complex derivatives portfolios. It mainly uses Monte Carlo simulation, since that is the only numerical method capable of valuing and risk-managing complex, multi-risk derivatives books.

In addition to plain vanilla instruments, derivatives that can be modeled with DX Analytics include "maximum call" options on multiple assets with European exercise, "minimum put" options on multiple assets with American exercise, and many, many more. It is not possible to give a comprehensive overview of the instruments that can be modeled with the library since it allows for a very flexible, modular modeling approach.

The Python Quants plan to build this library out to a fully fledged analytics suite in 2016/2017. Current development efforts include, among others:

- the addition of a SABR/LMM model with calibration capabilities;
- more sophisticated portfolio optimization techniques and classes;
- standardized data API support for Thomson Reuters Eikon;
- more sophisticated parallelization techniques (for clusters).

High-performance Python

As an interpreted language, Python is generally not fast when it comes to the execution of (nested) loops. This is a problem in finance since many financial algorithms are based, when implemented technically, on such loops. We have already seen a solution to this: vectorization of code with NumPy. For certain algorithms the memory burden of this approach might, however, be prohibitively heavy.

There are performance libraries available, like Numba, which make use of the LLVM infrastructure to dynamically compile Python code to machine code – without the need to adjust the original Python code. This is the same approach used, for instance, by the high-performance language Julia. In many cases, speed-ups of 50 times and more can be reached.

When it comes to input–output (IO) operations, Python has in general high performance already built in. Storing, for example, numerical data in binary format on disks usually works at the speed of the hardware available. Using a typical notebook with SSD drive, Python might allow you to write and read data at 500 MB/s while newer SSD technologies allow write and read speeds of well beyond 1 GB/s.

Quant Platform

Deployment of open-source technologies like Python can be a risky, costly, and time-consuming endeavor – especially in larger organizations. The Python Quants offer the Quant Platform (quant-platform.com) as an easy way to deploy and use multiple open-source technologies in an efficient and scalable manner. Among others, the platform offers:

- **Jupyter Notebook** (interactive data and financial analytics in the browser)
- **Python** (full stack)
- **R and Julia** (reduced stacks)
- **File Manager** (easy file management in the browser)
- **Web Editor** (editing of all text and code files)
- **System Shell** (full access to Linux-based infrastructure)
- **User Administration** (Linux-based user management)

There are free trial accounts for single users available. The platform can also be deployed individually in the Cloud or on-premise using Docker containers, making it compliant with typical IT policies of financial institutions.

Education and training

In addition to the technical ecosystem, Python also offers a quickly growing non-technical ecosystem. This consists of schools and universities using Python to teach programming and finance, conferences and books about Python and finance as well as professional education programs. The following is a list of resources especially for the intersection of Python and quant finance:

- **books** – *Python for Finance* (O'Reilly, pff.tpq.io); *Derivatives Analytics with Python* (Wiley Finance, dawp.tpq.io); *Listed Volatility and Variance Derivatives* (Wiley Finance, lvvd.tpq.io)
- **conferences** – For Python Quants (fpq.io); Open Source in Quantitative Finance (osqf.tpq.io); PyData (pydata.org)
- **meetups** – Python for Quant Finance (meetup.com/Python-for-Quant-Finance, meetup.com/Python-for-Quant-Finance-NYC)
- **trainings** – For Python Quants workshops (fpq.io); Python for Finance Certification (pfc.tpq.io)

Conclusions

Python is becoming more and more important in finance and in particular in computational finance. The reasons for this are, among others, its syntax, the ecosystem of scientific libraries, the financial libraries available in Python, its performance for financial algorithms and IO operations, as well as the non-technical ecosystem of books, conferences, meetup groups, and training offerings. Against this background, Python is likely to stay in finance for quite a while.

Disclaimer: Do not use Python at your own risk.

About the Author

Yves J. Hilpisch is founder and managing partner of The Python Quants (http://tpq.io), a group focusing on the use of open source technologies for computational finance and financial data science. He is the author of the books *Python for Finance* (O'Reilly, 2014), *Derivatives Analytics with Python* (Wiley, 2015) and *Listed Volatility and Variance Derivatives* (Wiley, forthcoming). Yves lectures on computational finance at the CQF Program (http://cqf.com) and has written the financial analytics library DX Analytics (http://dx-analytics.com). He is also organizer of meetups and conferences about Python for quantitative finance in Frankfurt, London, and New York.