MILLIMAN WHITE PAPER

Using Python as an actuarial platform

A series of short papers on efficient actuarial modeling with Python

Karol Maciejewski
Mehdi Echchelh

Milliman

The use of Python is on the rise. While this is not news, in recent years this programming language has steadily gained a foothold in various actuarial applications, and the ripples that creates in the actuarial world seem to be widening. In this short article, the first in a new series focusing on Python in actuarial modeling, we will look into the question of whether Python can go a step further and take its well-deserved place as an actuarial modeling platform.

Introduction

By most measures Python is currently the most-used general programming language[1], providing an implementation platform for a variety of applications, from serving as the backend for some of the most popular websites and advanced scientific simulations to being the backbone of artificial intelligence (AI) large language models (LLM). With its wide ecosystem of libraries and packages, Python can be easily used for virtually any purpose, and it is probably the most popular environment for data science, machine learning, and AI applications. Its accessibility and simplicity make it an attractive choice for university curricula, so it should not come as a surprise that we have been observing an increased interest in and popularity of Python also in the actuarial world for some time.

Using Python in actuarial modeling is by no means new. Python has been in the actuarial toolbox for several years, mostly in non-life applications and as helper scripts. Many articles about using Python in the actuarial domain can be found online. Milliman has also produced briefing notes and white papers on the topic, including a general introduction[2] and a more specialized application to life insurance,[3] respectively. Based on our recent experiences, it seems that in the insurance and risk domains, Python is gaining increasingly more traction. New generations of actuaries come equipped with Python skills and are more open to using it as a platform for a broadening spectrum of applications. The fact that Python is the main language used nowadays for AI research and applications certainly adds to its appeal and popularity. The complexity and breadth of solutions considered in Python have also been increasing, as have the expectations toward the model quality. It is often no longer sufficient (nor reasonable) to have a model in the form of a Python script or notebook. A proper production-worthy software application is preferred.

Therefore, we would like to revisit the topic of Python in actuarial applications, and over a series of short publications address some of the important questions, decisions, and issues that one might encounter when working with this language. While the fundamentals of Python are easy to learn and nowadays often taught at universities, being able to design and develop sophisticated, production-grade, and highly efficient models require significantly more expertise, especially from actuaries who are rarely also software engineers.

In this first paper, we will look at some truths and myths about using Python as a modeling platform for actuarial applications and help the reader answer a fundamental question: Is Python a good choice for an actuarial modeling platform?

Actuarial modeling platform

Many factors should be considered when deciding whether a software platform or programming environment is a good basis for a given purpose. For actuarial modeling, a reasonable starting point might be:

- Model performance
- Training, documentation, and support
- Reliability
- Widespread use
- Multipurpose and interoperability

1. See rankings referenced in section titled “Ecosystem and community.”
2. McGinley D. (23 May 2021). The Rise of Python: Powerful, robust calculation software. Milliman Briefing Note. Retrieved 24 March 2025, from https://www.milliman.com/en/insight/The-Rise-of-Python-Powerful-robust-calculation-software.
3. Maciejewski K., Echchelh M., Sznajder D. (13 March 2023). Building a high-performance in-house life projection and ALM model: Architecture and implementation considerations in Python. Milliman White Paper. Retrieved 24 March 2025, from https://www.milliman.com/en/insight/building-in-house-projection-alm-model-python.

Performance is an obvious point, as everybody wants their models to run as fast as possible (or even faster). Although many people underestimate the importance of choosing the right algorithms, data structures, and code optimization, which are all key ingredients of a fast runtime, the choice of programming language or software platform also has a significant effect on performance.

Learning ease is a very important factor for actuarial modeling purposes. Most actuaries, even among the modeling ones, are not software engineers or computer programmers. They typically prefer something accessible, well documented, and with a good level of training and support to help them out when they encounter issues.

Another important aspect is reliability. This can mean several different things. In the narrowest understanding, it provides a stable computational environment with reproducible results. But looking at it in a broader context, some other important elements include the ability to follow modern development operations (DevOps) techniques[4] for model development, testing, and deployment and high confidence that the platform will be maintained and supported for a long time.

The following aspect might seem somewhat less important, but widespread use of the programming language or software means that there is a large pool of people who know it and can be hired in case of employee turnover or team growth. This is especially important in the case of actuarial modeling, where resources are generally scarce in most markets. Locking oneself into a niche software or programming language will certainly become a problem in the longer term.

Last, but not least, it is helpful if the selected language or software platform comes equipped with a plethora of libraries that can be used in a wide variety of applications and domains. This is typically linked with the openness of the system, broad and active community of users, and the possibility of seamless interfacing with other systems and processes.

Of course, these points are only a subjective selection of many possible criteria. Each person and organization might have their own additional preferences and priorities when choosing the modeling platform, such as independence of third-party vendors, consistency with the tooling already used in the company or group, cost, and more.

Python performance

One of the most-asked questions and biggest concerns when considering Python as an actuarial modeling platform is if it is fast enough. One common criticism of Python is that it is slow and therefore might not be suitable for actuarial modeling. There are also benchmarks circulating around[5], showing that Python fares poorly in comparison with majority of other programming languages.

On the one hand, this is true. Python has not been designed with speed of execution as a priority. It is an interpreted language, meaning the source code is analyzed and converted into an executable at the moment of invoking the script, instead of preparing it in advance (as is done in compiled languages). This obviously adds up to the waiting time for the results of an execution.

However, when used the right way, employing high-performance calculation libraries, Python enables users to achieve very-fast runtimes. There are several things to consider here.

While pure Python is not particularly fast, there are plenty of publicly available Python libraries focused on performance. They are written in fast languages like C or C++ and can be used directly from within Python, combining the best of both worlds. Using these libraries is common practice in the scientific and data science communities, with millions of users every day. Several libraries allow pre-compilation of the critical, computationally intense Python code blocks into efficient machine code, giving the user a combination of simple Python syntax and performance on par with the fastest languages available.

Secondly, many benchmarks comparing different programming languages focus on purely algorithmic problems that might not be representative of actuarial models' computational complexity. Therefore, their results might be misleading without understanding the full context and the specificities of actuarial use cases. Actuarial models usually focus on statistical computations, medium-sized data processing, or a high number of repetitive calculations (simulations or individual policy modeling) that can be vectorized. As mentioned above, there are highly optimized libraries for each of these purposes, which minimize the negative performance impact of pure Python.

4. DevOps refers to software development operations; widely accepted guidelines; and standards of automated, collaborative, agile, reliable, and secure iterative software improvements that quickly deliver a better product. See more details in the following sections.

5. Pereira R. et al., (4 January 2021). Ranking programming languages by energy efficiency. Retrieved 24 March 2025, from https://haslab.github.io/SAFER/scp21.pdf

In practice, one wouldn't use Python for actual actuarial use cases the way it is used in these general benchmarks. Benchmarks focused on actuarial applications show very different results.[6]

Thirdly, some benchmarks compare execution of compiled languages with a full execution of scripted languages. While for extremely-fast benchmarks executing in milliseconds, this indeed might create a significant difference, will these additional few seconds at the start of the Python script matter when running actuarial models that take minutes (or in extreme cases, even hours)?

Finally, Python comes with many libraries that allow programmers to easily take advantage of parallel or distributed computing. When using machines equipped with graphics processing units (GPUs), several libraries enable the transfer of compute-intensive tasks to GPUs or even write custom GPU kernels to tackle specific problems. This offers unparalleled performance for problems that parallelize well—and all this can be done directly from Python, without knowledge of Compute Unified Device Architecture (CUDA) C++, or OpenGL.

An additional thing to consider here is that when thinking about a platform for actuarial models, actuaries will rarely consider coding directly in the high-performance languages like C or C++ (or nowadays Rust). Plus, any third-party vendor solution written in any of these languages will also usually come with a boilerplate that will make models in that platform perform slower than the raw benchmarks in the relevant language. This can also be seen in our simple projection model benchmark presented in our 2023 white paper[7].

When it comes to performance in Python, it is impossible to overstate the importance of having an expert with an understanding of code complexity and efficiency of algorithms and data structures and at least fundamental knowledge of software development involved in setting up the architecture and development of the models. While Python will probably never be the fastest of all possible options (if one considers building their models in C, C++, or Rust), when designed by a programming-savvy actuary, it can be made very fast, usually substantially faster than the current solution. And whatever it might lack on the performance side, it makes up for in the other aspects of a good actuarial modeling platform.

And a final question for a thoughtful reader: If its performance was truly so bad, would Python be the language of choice for the cutting-edge tech companies dealing with extremely compute-intensive generative AI?

Ecosystem and community

Three of the proposed characteristics of a good actuarial modeling platform revolve around Python's popularity, extremely-big ecosystem, and community. The bigger the community of users, the more resources are available with respect to training and support. This is true for any software, but even more so for active open-source projects. By definition, a big community will also mean widespread adoption, a large pool of potential resources skilled in it, and, most likely, a broad spectrum of uses.

All this is true for Python. It is ranked by various groups as the most popular programming language in the world. The TIOBE language index places Python at an undisputed first place,[8] as does the Popularity of Programming Language (PYPL) index.[9] A 2024 Octoverse report by GitHub, the biggest software code repository with more than 500 million projects hosted, shows continued growth of Python's popularity and also places it in the first place for the first time.[10] A Stack Overflow[11] survey places Python at third place, but the first two places are taken by HTML and JavaScript, which are the backbones of the World Wide Web and not general programming languages. Stack Overflow, probably the biggest community forum for programming questions and problems with 22 million active questions and 36 million answers,[12] has over 2.2 million questions (and answers) about Python,[13] the second-biggest group, again just after JavaScript. No other programming language scores anywhere close or anywhere so consistently across various sources. With millions of active users and a contribution-oriented mindset, it's no surprise that the main Python package repository (Python Package Index – PyPI) consists of over 600,000 projects and almost 1 million people contributing to their development.[14] Each of those projects is a code package that anybody can download and use in their models.

6. See simple examples as cited in Maciejewski K, Echchelh M, & Sznajder D., 2023.
7. See benchmarks in Maciejewski K, Echchelh M, & Sznajder D., 2023.
8. Jansen, P. (March 2025). TIOBE Index for March 2025. Retrieved 24 March 2025, from https://www.tiobe.com/tiobe-index/.
9. PYPL Index (March 2025). PYPL Popularity of Programming Language Index. Retrieved 24 March 2025, from https://pypl.github.io/PYPL.html.
10. GitHub. (22 November 2024). Octoverse: AI leads Python to top language as the number of global developers surges. Retrieved 24 March 2025, from https://github.blog/news-insights/octoverse/octoverse-2024/#the-most-popular-programming-languages.
11. Stack Overflow. (May 2024). 2024 Developer Survey. Retrieved 24 March 2025, from https://survey.stackoverflow.co/2024/technology#most-popular-technologies-language.
12. StackExchange (15 March 2025). StackExchange Data Explorer. Retrieved 24 March 2025, from https://data.stackexchange.com/.
13. StackExchange (n.d.). Tags. Retrieved 24 March 2025, from https://stackoverflow.com/tags.
14. Python Package Index (PyPI). (2025). PyPI.org. Retrieved 24 March 2025, from https://pypi.org/.

Besides these publicly available and free resources, there are countless courses and tutorials, both free and paid. Most serious Python libraries come with very-high-quality documentation as well, which is an invaluable source of information. If one encounters issues with any of the libraries, there is usually a direct way to communicate with the developers themselves. This is a starkly different experience from many vendor solutions, where one often needs days to go through several iterations with customer service before getting to someone who can answer the question.

The final point, related to the popularity of Python, is that in recent years it has achieved at least the same status as R used to have in terms of academic curricula. It is now being taught in most programs that produce future actuaries—mathematics, physics, econometrics, or data science. Most of the graduates coming to work in actuarial and risk teams will likely have at least a basic knowledge of Python. One must keep in mind, of course, as indicated earlier, that being able to write a Python script to pass an assessment of a university course does not mean that person can develop a production-grade, high-performance model, but it is a good starting point for further education.

Reliability and interoperability

That leaves us with two points on the proposed actuarial modeling platform "wish list"—reliability and interoperability—and unsurprisingly Python also checks these boxes.

As shown in the previous section, Python has a massive community, hundreds of thousands of contributors to the popular library projects and millions of users, including institutional and corporate. This is a pretty good assurance for reliability of this platform. Neither Python nor these popular libraries will disappear in any time frame that would not allow their users to work out an alternative plan (at least comparable with a time frame a vendor might stop its support for its software). Additionally, most of these packages are open source, meaning anyone, even an insurance company, could take the source over and maintain it for the time needed to migrate to another solution. And finally, when we see some libraries fade away, it is usually because there is a new replacement that provides better functionalities or better performance, as the Python ecosystem is constantly evolving and improving.

Besides these conceptual arguments for the reliability of Python and its libraries, there is more practical proof. A large part of the connected world as we know it is built using Python in its tech stack. Applications of companies such as Netflix, Spotify, Uber, YouTube, Google, Instagram and many others are mainly, or in a significant part, coded in Python.[15] In fact, developers from such companies are also part of the contributor community making Python's ecosystem so active and evolving. This distributed development and maintenance model, incorporating a large degree of diversification, is exactly what assures longevity and robustness of Python and its ecosystem and makes the risk of software discontinuation much more pronounced for any single vendor. In the case of some of these applications, besides the usual machine learning and orchestration uses, Python is also handling the operational workload—millions of client requests per second. This is a testament to another dimension of reliability of the Python platform.[16]

One other aspect of platform reliability mentioned earlier was the ability to use modern DevOps techniques during model development and later maintenance. Python can be fully integrated with all key DevOps elements, like source code versioning, automated testing during developments, continuous integration and automated test pipelines when committing and releasing the code, automatic building of simple test cases from code documentation, automatic checking tools for code correctness and style, and many others. Many actuarial models are built on platforms which do not support the use of these techniques, and they can truly improve code quality and reliability, as well as shorten the development and testing time required for new model releases.

Because of its widespread use and broad set of libraries, Python also comes with countless libraries allowing interfacing and connection with a plethora of other systems. It can run on most operating systems; it can communicate over many kinds of networking and communication protocols; it can read, write, and manipulate data using common consumer file formats (e.g., CSV, Excel), high-performance data stores (Parquet, Feather, HDFS), and many types of databases (SQL, NoSQL); it can consume and serve application programming interfaces (APIs), allowing easy client-server communication across web interfaces; it can serve interactive web-based dashboards and many, many others. With a few lines of code (using readily available libraries), one can integrate data flow and processing steps in different applications on different machines into a single automated pipeline.

15. Goyal, A. (25 July 2022). Best Websites Written on Python. Retrieved 24 March 2025, from https://www.octalsoftware.com/blog/best-websites-written-on-python.

16. Woodruff, B. (15 August 2019). Static Analysis at Scale: An Instagram Story. Retrieved 24 March 2025, from https://instagram-engineering.com/static-analysis-at-scale-an-instagram-story-8f498ab71a0c.

Actuarial models and processes

Having reviewed the proposed key characteristics of what makes a good actuarial modeling platform and how Python performs in each of those categories, one might wonder about the scope of potential use of Python in the actuarial world.

We are accustomed to having different dedicated tools for different actuarial purposes: one tool for extracting data from an admin system and cleaning, processing, and storing it in a data lake or warehouse; another tool for life cashflow projections and asset-liability management (ALM); another for non-life pricing; another for non-life reserving; another for Solvency II reporting; yet another one for generating IFRS 17 figures; and so on.

Python could be used as an efficient substitute for pretty much all of them. It would allow unification of actuarial modeling skills, facilitate automation of processes and elimination of many manual steps, and make it possible to reuse calculation modules between (now) different models instead of maintaining them several times in different platforms and struggling to reconcile the numbers in different processes every time. At the end of each process, there could be a modern, clean visualization dashboard allowing interactive exploration of the results. All of that could be developed using a single platform.

Of course, there is a long road before this could happen, and it would involve significant investments, change management, and recalibration of the skill set of actuarial modeling teams. But because of Python's versatility, popularity, simplicity, and performance, Python has the potential to be a fully-fledged actuarial modeling platform, to the extent like no other platform so far. And the first step is to try using Python in your next model or process automation.

Solutions for a world at risk™

Milliman leverages deep expertise, actuarial rigor, and advanced technology to develop solutions for a world at risk. We help clients in the public and private sectors navigate urgent, complex challenges—from extreme weather and market volatility to financial insecurity and rising health costs—so they can meet their business, financial, and social objectives. Our solutions encompass insurance, financial services, healthcare, life sciences, and employee benefits. Founded in 1947, Milliman is an independent firm with offices in major cities around the globe.

milliman.com

CONTACT

Karol Maciejewski
karol.maciejewski@milliman.com

Mehdi Echchelh
mehdi.echchelh@milliman.com

Milliman