From Spreadsheets to Python

Transforming Excel Workflows into Production-Ready Applications

August 2025

DL Lim

CONTENTS

1 EXECUTIVE SUMMARY

Spreadsheets have been the backbone of financial modelling and analytics for decades, providing flexibility and accessibility. They enabled analysts, actuaries, and finance professionals to build complex valuation models, run scenario analyses, and conduct regulatory calculations with relative speed. However, as organisations scale and regulatory requirements become more stringent, **the limitations of spreadsheets become increasingly apparent**. Version control, user access management, auditability, and scalable computation are not optional features; they are essential for sustainable, compliant operations.

Python and modern code-based systems offer a pathway beyond these constraints. By enabling reproducible, auditable, and scalable workflows, production-ready systems unlock efficiency and transparency that spreadsheets alone cannot deliver. They allow teams to centralise complex logic, integrate with enterprise data sources, and automate repetitive calculations, while still preserving flexibility for exploration and prototyping. Organisations that embrace this transition can achieve faster, more reliable decision-making, improve operational resilience, and gain a competitive edge in a market where precision and speed are paramount.

Building such systems requires **more than just writing code. Success demands the right combination** of expertise, team structure, processes, and strategic planning. Thoughtful design, reusable frameworks, and robust integration practices ensure that migration from spreadsheets to production systems is efficient, low-risk, and future-proof. Organisations that invest in these capabilities position themselves to meet regulatory and operational demands while leveraging their data and models as strategic assets, enhancing competitiveness and driving growth.

In this context, expert guidance and experience can significantly accelerate the journey. **Leveraging proven practices and purpose-built solutions** helps organisations prioritise high-impact projects, avoid common pitfalls, and realise the full potential of Python-based systems for finance and insurance operations.

2 INTRODUCTION

For decades, Microsoft Office’s Excel has been the **natural evolution** from pen, paper, and calculators. It replaced manual ledgers and human “computers” with a flexible, accessible tool that empowered professionals to model scenarios, perform calculations, and share results at speed. In finance and insurance especially, **spreadsheets became indispensable**, offering a way to prototype, test, and deliver insights without needing specialist software development skills. Entire business processes, even mission-critical ones, have been built around Excel.

But just as Excel once supplanted the manual methods before it, code is now steadily replacing Excel. Languages like Python are becoming the next logical step, enabling organisations to move from individual workbooks towards systematic, production-grade solutions. Python not only mirrors Excel’s accessibility but expands on it, offering scalability, reproducibility, and integration into modern enterprise systems. Where Excel formulas remain bound to a grid, Python modules can be version-controlled, tested, deployed, and scaled across servers or cloud environments.

This shift ought to matter most in finance and insurance, where regulatory compliance, auditability, and performance at scale are non-negotiable. Regulators expect firms to reproduce calculations years later, competitors demand faster turnaround, and customers reward accuracy and transparency. Yet, the industry is still slow on the uptake compared to other tech-based companies. Excel can provide quick answers, but Python can provide sustained advantage. The migration is not about abandoning a useful tool, but about realising a new competitive edge recognising that the future of reliable, scalable, and compliant analytics lies in code.

3 THE SPREADSHEET CEILING

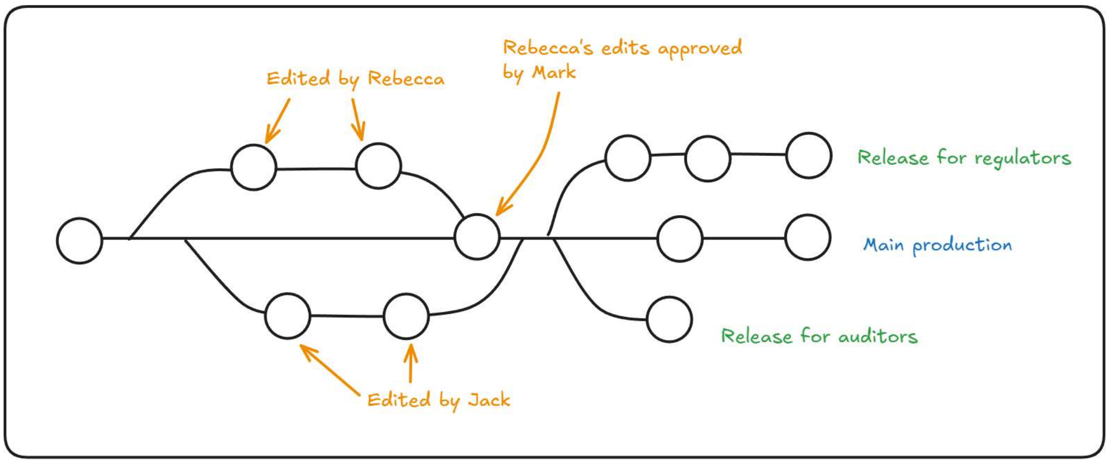
Excel has carried organisations a long way. It has enabled actuaries, analysts, and financial professionals to design and run valuation models, perform model validation tests, calculate solvency ratios, price insurance products, analyse reserve movements, conduct experience studies, measure embedded value, and even develop credit risk models. These capabilities made Excel a cornerstone of financial and insurance workflows, democratising access to advanced calculations without the need for formal programming.

However, as financial products, regulations, and organisational demands have become more complex, the cracks in the spreadsheet-run organisational model are impossible to ignore. What was once sufficient for personal productivity or team-level collaboration no longer scales to the demands of modern, regulated enterprises. This is the spreadsheet ceiling: the point at which Excel is no longer enough to support the accuracy, auditability, and scale required by the business.

3.1 VERSION CONTROL

Every analyst has faced the problem of "MotorPricing_v2_final_FINAL_CLEAN.xlsx". Spreadsheets are inherently file-based, leading to duplication, confusion, and errors when multiple people are working on the same logic. Unlike code repositories, there is no native mechanism to track who changed what, when, and why. This makes it nearly impossible to enforce consistent processes across teams, or to return confidently to a trusted prior version.

In high-stakes environments like solvency reporting or risk modelling, the absence of robust version control creates material risk. Regulators may ask firms to reproduce a calculation from years ago, but if workbooks have evolved in an uncontrolled way, that reproducibility cannot be guaranteed.

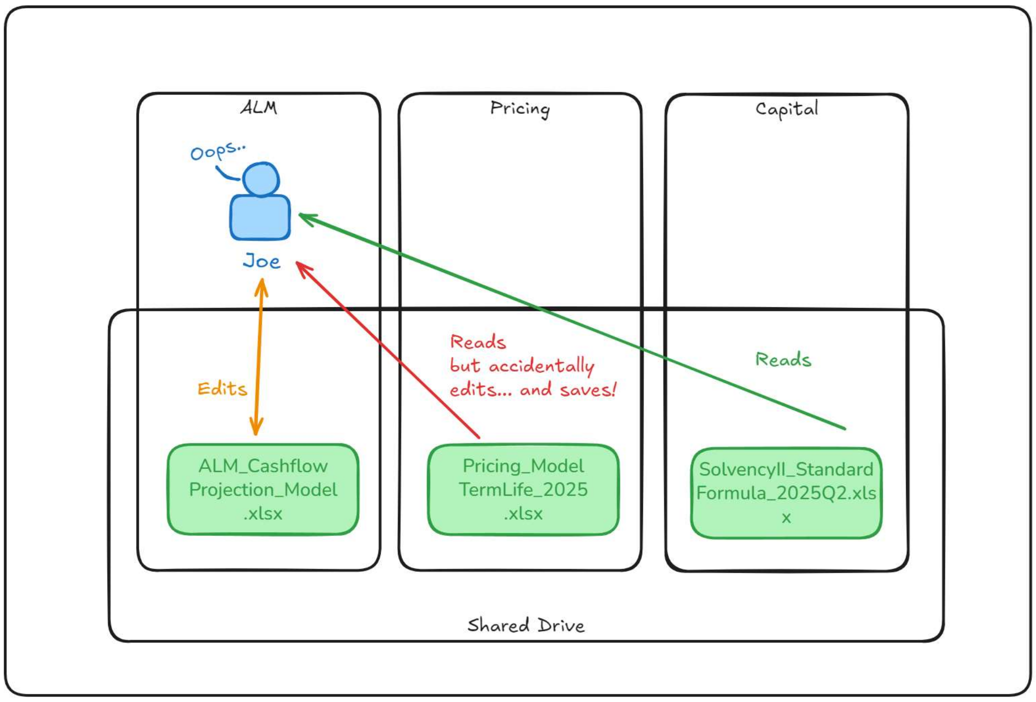


3.2 USER ACCESS CONTROL

Excel is highly permissive. **Anyone with access to a file can usually see and change everything**. While it's possible to password-protect an Excel file, this method is both easily bypassed and ineffective for managing granular permission levels securely.

Although the level of flexibility and openness with Excel files supports collaboration, it comes at the cost of weakened governance. Sensitive assumptions or formulas can be altered (by mistake) without oversight, and entire sheets of data can be copied and shared and errors compounded across the organisation.

This lack of granular access control poses compliance challenges in finance and insurance, where regulators expect evidence of proper handling of sensitive data. Without robust permissions, audit teams struggle to confirm that only the right people had access to the right parts of a model.



3.3 SCALABLE COMPUTE

Spreadsheets were never designed to handle enterprise-scale compute.

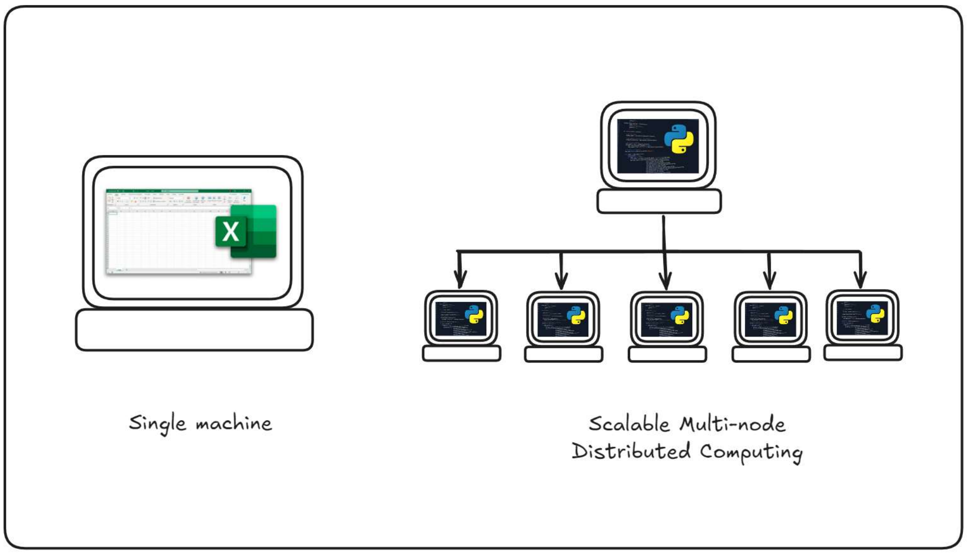
Earlier 32-bit versions of Excel (which is still broadly in use today!) often crashes with files over 100MB, maxing out their 2GB of RAM. **Formula-heavy spreadsheets seldom perform well once they're over 40MB**. More modern 64-bit versions of Excel can handle much larger files up to the limits of the system's RAM, but even so, they can't scale multi-node, as it is a desktop-interface.

Such kinds of large datasets quickly push the boundaries of what Excel can manage, often leading to crashes, corrupted files, or unacceptably long calculation times. Complex actuarial or risk models that may need to run tens of thousands of simulations simply cannot be executed efficiently in Excel.

This lack of scalability forces analysts to cut corners, simplify assumptions, or run fewer simulations than ideal. The result is not only slower decision-making but also reduced accuracy in areas where precision is critical.

Where Excel is bound to the computational and memory limitations of the single machine on which it is installed, code unlocks the ability to run in a distributed fashion over a number of nodes. For instance, **frameworks like Apache Spark allow workloads to be broken into smaller tasks, which are distributed across a cluster of worker nodes**. A central manager, often called the driver, coordinates these workers, ensuring that tasks are executed in parallel and results are gathered efficiently.

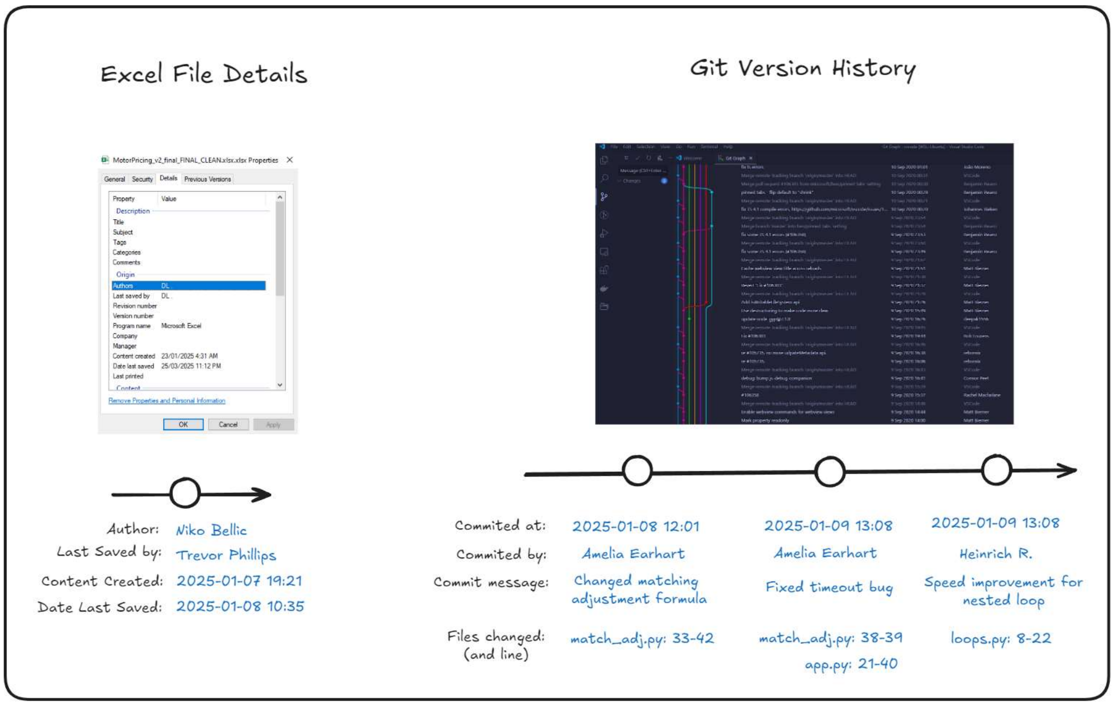
This model allows organisations to scale calculations well beyond the limits of a single laptop or desktop. Instead of waiting hours (or days) for a spreadsheet to grind through complex simulations, distributed systems can process millions of records and run thousands of scenarios in a fraction of the time. For data-heavy industries such as finance and insurance, this parallelism is a game-changer: regulatory stress tests, reserve calculations, or pricing models that once strained Excel can now be computed at enterprise scale, with consistent accuracy and far greater speed.



3.4 AUDIT LOGGING

One of Excel's greatest weaknesses in regulated environments is the absence of proper audit logging. While it is possible to track some changes with features like "Track Changes," these are often clunky, incomplete, and not robust enough for enterprise needs. Critical questions such as "Who changed this formula?" or "When was this assumption last updated?" cannot be answered with confidence.

In insurance and finance, where every number must stand up to scrutiny, the inability to produce a defensible audit trail exposes organisations to compliance risk. Without an audit log, even well-constructed spreadsheets remain fragile and difficult to trust.



MotorPricing_v2_final_FINAL_CLEAN.xlsx.xlsx Properties

Property	Value
Description	
Title	
Subject	
Tags	
Categories	
Comments	
Origin	
Authors	DL
Last saved by	DL
Revision number	
Version number	
Program name	Microsoft Excel
Company	
Manager	
Content created	23/01/2025 4:31 AM
Date last saved	25/03/2025 11:12 PM
Last printed	
Content	

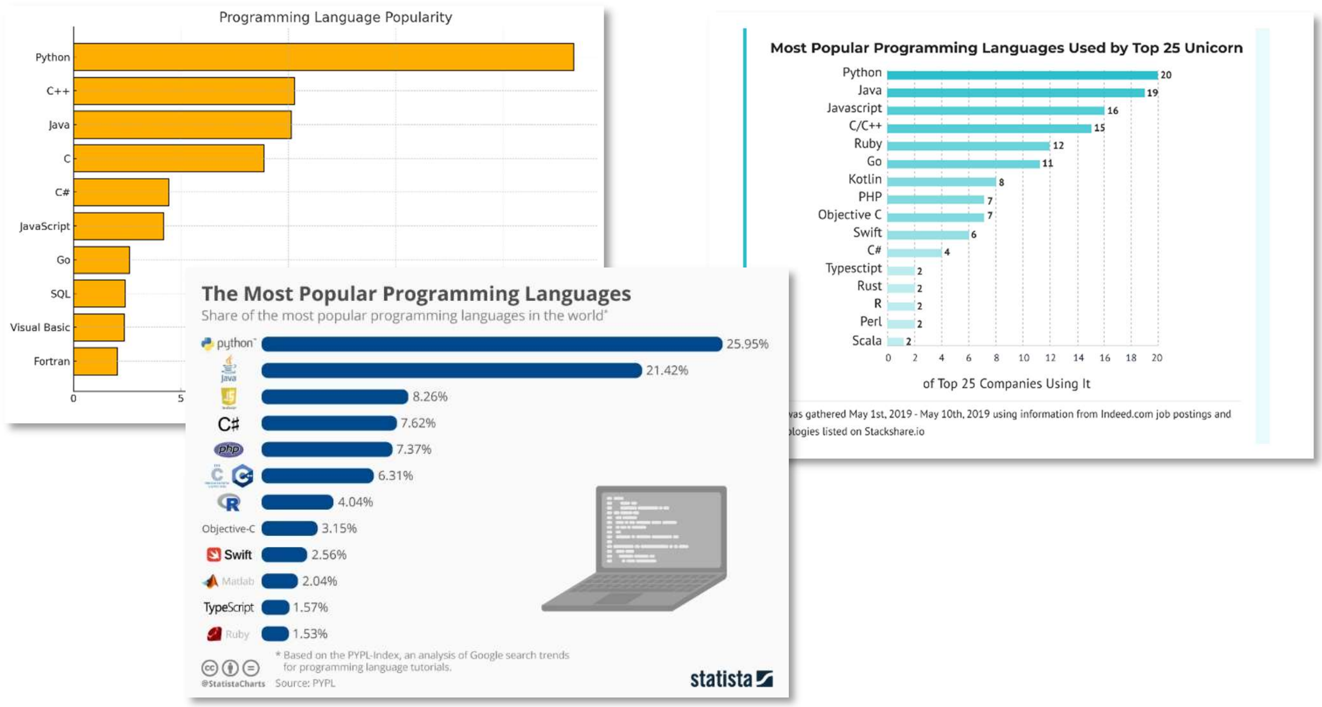
Remove Properties and Personal Information

Author:	Niko Bellic
Last Saved by:	Trevor Phillips
Content Created:	2025-01-07 19:21
Date Last Saved:	2025-01-08 10:35

Commited at:	2025-01-08 12:01	2025-01-09 13:08	2025-01-09 13:08
Commited by:	Amelia Earhart	Amelia Earhart	Heinrich R.
Commit message:	Changed matching adjustment formula	Fixed timeout bug	Speed improvement for nested loop
Files changed: (and line)	match_adj.py: 33-42	match_adj.py: 38-39 app.py: 21-40	loops.py: 8-22

4 WHY PYTHON

While any code language with good software engineering practices should be able to accomplish what Python could, Python’s dominance comes from its vast ecosystem, low barrier to entry, the sheer size of its global community, and popularity amongst developers.



As such Python emerges as the natural successor to Excel in modern financial and insurance workflows. It is not only the dominant language in data and machine learning but also an approachable, versatile tool that allows teams to scale beyond the spreadsheet ceiling.

4.1 THE LANGUAGE OF DATA AND MACHINE LEARNING

Python is the undisputed leader in the fields of data science, machine learning, and artificial intelligence. Virtually all modern frameworks from numpy and PyTorch to pandas and scikit-learn are all written in or support Python. This means that actuarial teams, quants, and analysts who already rely on data-intensive methods can tap into a massive ecosystem of tools that Excel simply cannot match.

4.2 COMMONLY UNDERSTOOD ACROSS DISCIPLINES

Unlike niche or legacy programming languages, Python enjoys near-universal recognition. Its simple syntax makes it accessible to beginners, while its depth allows experts to build sophisticated systems. This broad adoption means analysts, developers, and even business stakeholders are more likely to understand and engage with Python-based solutions. It reduces silos and fosters a shared language across teams.

4.3 OVERCOMING THE SPREADSHEET CEILING

Where Excel reaches its limits, Python excels. It addresses the pain points of spreadsheets by:

- Enabling **Git and version control** for transparent history and collaboration
- Enforcing **user access and permissions** within applications
- Scaling calculations across clusters or cloud infrastructure
- Capturing a full **audit trail** of inputs, changes, and outputs

In short, Python turns fragile spreadsheet models into robust, production-grade systems.

4.4 ECOSYSTEM AND EXTENSIBILITY

Python is not just a language, it is a platform. Libraries exist for virtually every need: from numerical computation with NumPy, to actuarial modelling with specialised packages, to web deployment with FastAPI or Django. [Read our paper on this]. This ecosystem allows organisations to evolve incrementally, starting with replacing an Excel workbook and eventually scaling up to enterprise-wide data systems, all without switching tools.

4.5 REGULATORY CONFIDENCE AND TRANSPARENCY

For industries like finance and insurance, regulators demand reproducibility, transparency, and evidence of control. Python, combined with the right frameworks and solid DevOps (e.g. MLFlow, git, CI/CD pipelines), enables models to be documented, versioned, and validated systematically.

Tagged releases can be used to re-run models and specific points in time.

This produces not only accurate results but also confidence in auditability and compliance; something spreadsheets struggle to provide.

4.6 FUTURE-PROOF SKILLSET

Finally, Python equips teams with skills that are forward-looking. While Excel proficiency remains valuable, organisations that embrace Python prepare their teams for the growing demands of automation, artificial intelligence, and scalable systems. This ensures that investments made today continue to deliver value tomorrow.

5 CHALLENGES

Transitioning from spreadsheets to code is not a matter of simply swapping one tool for another. It requires a **cultural shift as much as a technical one**, and organisations need to **be mindful of the human** and operational **factors** involved.

5.1 EXCEL STILL HAS ITS PLACE

Excel should not be seen as an enemy to be eliminated. Its flexibility and ease of use make it the perfect environment for quick experimentation, exploratory analysis, or proof-of-concept work. When an analyst has an idea for a new valuation approach, a fresh way to look at reserve movements, or a novel pricing adjustment, Excel allows them to model it rapidly without waiting for a full development cycle. For tasks where speed and accessibility are more important than scale or auditability, Excel continues to shine.

The point, then, is **not to eliminate Excel but to understand its scope**. It should serve as the **launchpad for ideas, not the final destination for enterprise-critical systems**. Once a model proves its value and becomes integral to business operations or regulatory reporting, that is the time to migrate it into code. In this way, **Excel and Python can complement one another**: Excel providing the agility for discovery, and Python delivering the structure, scalability, and controls required for production.

5.2 CULTURAL SHIFT AND FEAR OF CHANGE

Replacing spreadsheets with code can feel daunting to many professionals. Excel has been the familiar companion of analysts, actuaries, and finance teams for decades, and there is comfort in its grid of rows and columns. Moving to Python, with its scripts and functions, can feel like stepping into unfamiliar territory. Some may even fear that learning a new tool signals the end of their current role, or that automation will make their jobs obsolete.

In reality, this transition represents an opportunity for growth rather than loss. By learning Python, spreadsheet specialists expand their toolkit and become more versatile. They are not being replaced but empowered to take on bigger, more complex problems that Excel cannot handle. Upskilling in this way allows professionals to remain central to the organisation's future, while also ensuring their careers remain relevant in a rapidly changing industry. The shift is less about giving up what you know and more about building on it with new, powerful capabilities.

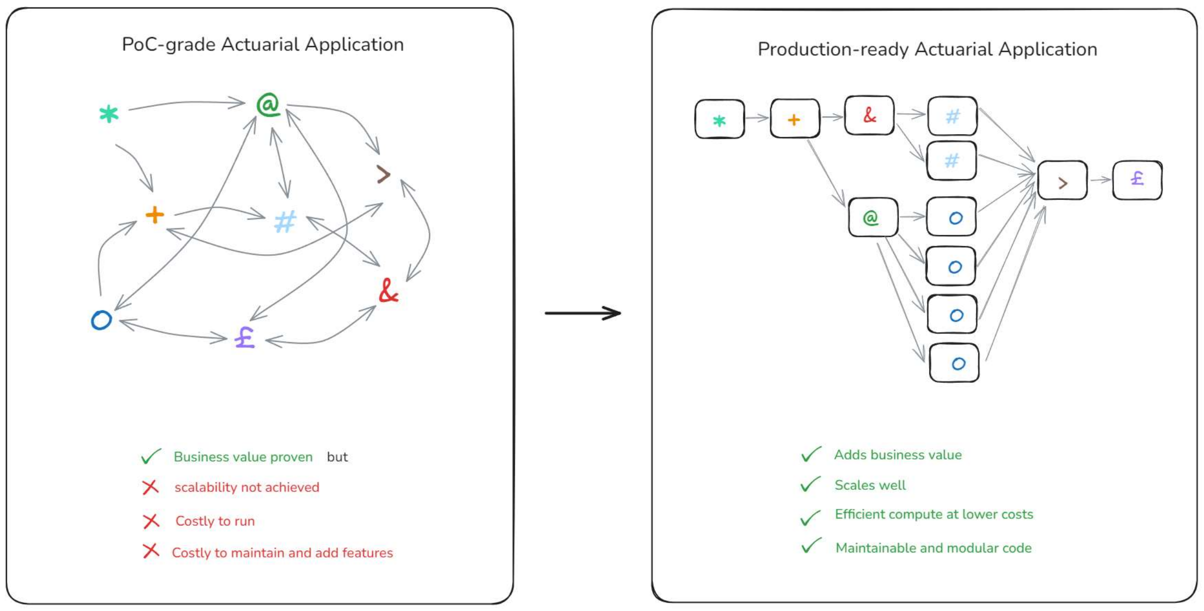
5.3 FROM PROTOTYPE TO PRODUCTION

Depending on what an individual's level of comfort is with Python, they may use LLM's and AI to accelerate their output. **It is often easy to build a quick prototype in Python.** With a few lines of code and access to the right libraries, analysts can replicate models that once required sprawling spreadsheets.

This ease of prototyping is one of Python's great strengths, but it is only the beginning. **Moving from a proof-of-concept to a production-ready system introduces a whole new set of challenges.**

Productionising a model means ensuring it integrates with other systems, runs reliably on cloud infrastructure, and adheres to security and compliance requirements. It means adding monitoring, logging, and automated testing. It also means designing for scalability so the model works not only for one analyst but across entire teams or business units.

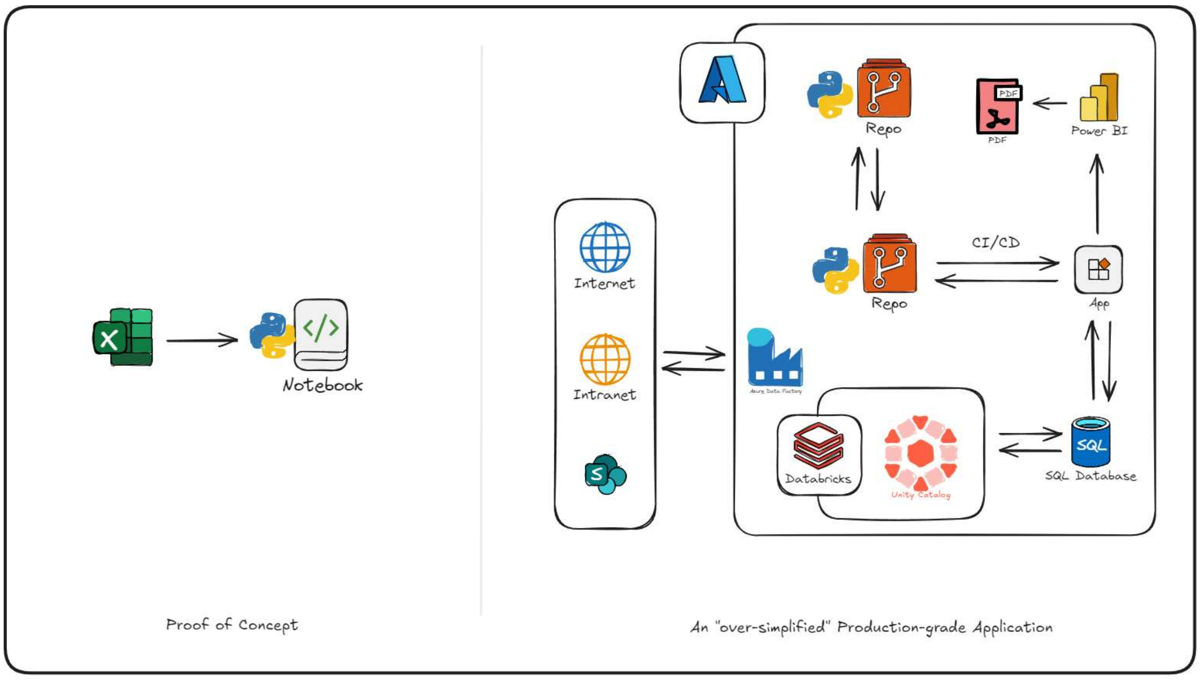
These are skills rooted in software engineering and systems design, and they need to be learned and embedded alongside modelling expertise. **Organisations that understand this difference set themselves up for long-term success, while those that underestimate it risk building fragile systems that fail under real-world pressures**.


PoC-grade Actuarial Application
✓ Business value proven but
✗ scalability not achieved
✗ Costly to run
✗ Costly to maintain and add features
Production-ready Actuarial Application
✓ Adds business value
✓ Scales well
✓ Efficient compute at lower costs
✓ Maintainable and modular code

5.4 BALANCING AGILITY AND RIGOUR

One of Excel's enduring strengths is agility. It lets analysts test an idea in minutes, make adjustments on the fly, and share results almost instantly. By contrast, **production systems** in Python **require discipline**: documenting assumptions, writing tests, enforcing version control, and following structured deployment practices. For teams used to the instant feedback of spreadsheets, this discipline can feel like a slowdown.

The challenge lies in finding the balance. Organisations must preserve the creative spark and rapid experimentation that spreadsheets encourage, while also embracing the rigour of production systems. **This is not a choice between speed and structure but an integration of both.** Prototyping can remain fast and flexible in Excel or ad-hoc Python, while production systems are designed with robustness in mind. With the right practices, organisations can move smoothly from exploration to enterprise-grade deployment, keeping the best of both worlds.



6 PATHWAYS TO MIGRATION

Moving beyond Excel does not mean abandoning it altogether or attempting a wholesale switch overnight. **Successful transitions happen in stages**, guided by clear criteria and supported by reusable foundations. By **treating migration as a journey rather** than a one-off project, organisations can de-risk the process and maximise the value of their investments.

6.1 IDENTIFY CANDIDATES FOR MIGRATION

Not every spreadsheet needs to be converted into a Python system. Attempting to migrate all spreadsheets at once is inefficient and risky. The first step is to **categorise spreadsheets based on business impact, complexity, and lifecycle**. This ensures resources are focused on models where migration delivers the greatest value.

Criteria for selecting spreadsheets to migrate:

- **Competitive Advantage**: Does building this in a systematic way help the organisation gain an edge in the market?
- **Business Criticality**: Does the spreadsheet underpin regulatory reporting, financial statements, or key business decisions?
- **Complexity of Logic**: Does it include advanced formulas, iterative calculations, or embedded macros that are difficult to maintain in Excel?
- **Frequency of Use**: Is it updated regularly, reused across teams, or shared across multiple stakeholders?
- **Collaboration Needs**: Do multiple people need access to modify, validate, or review the spreadsheet simultaneously?
- **Data Volume and Scalability Requirements**: Does it handle large datasets that exceed Excel's memory or computational limits?
- **Audit and Compliance Requirements**: Is it subject to internal audits or external regulatory scrutiny? Does having a system help mitigate some of those risks?
- **Repetitive Tasks**: Does it require frequent manual updates, copy-pasting, or repetitive calculations?

Examples of spreadsheets that are strong candidates for migration:

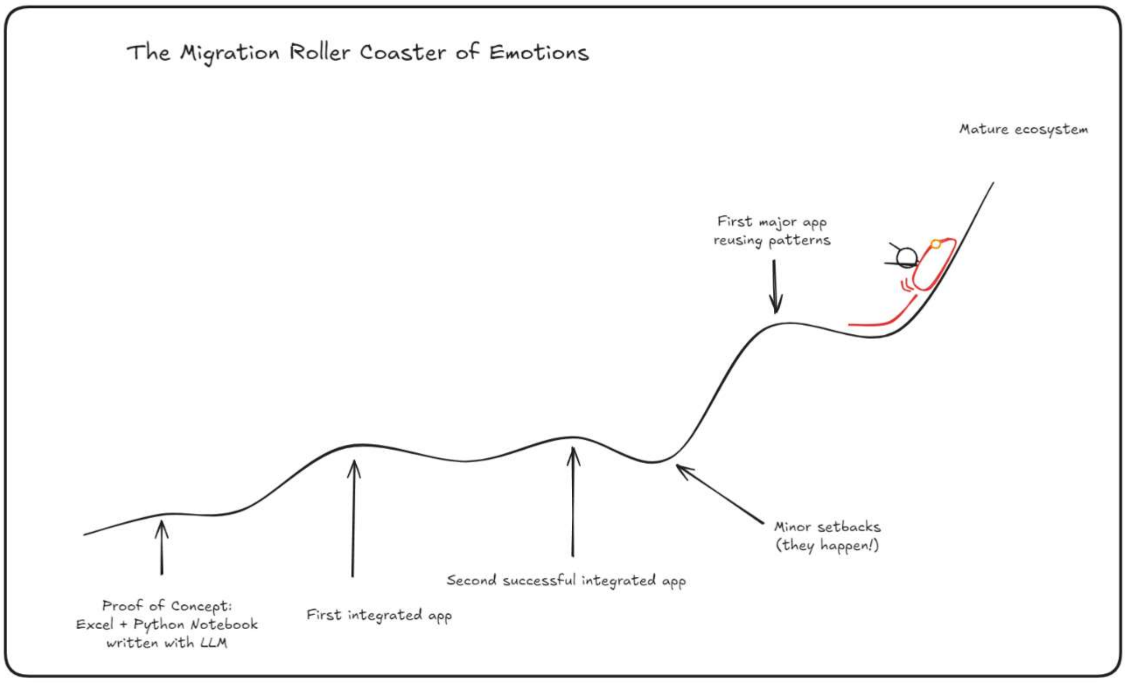
- **Valuation Models**: Complex calculations for reserves, policy values, or embedded value analysis.
- **Solvency and Risk Calculations**: Stress tests, credit risk models, or capital adequacy assessments.
- **Pricing Models**: Insurance premium calculators, product pricing simulators, or scenario analysis templates.
- **Reserve Movement Analyses**: Spreadsheets tracking historical reserve changes, claims development, or loss triangles.
- **Experience Studies**: Mortality, lapse, or claim experience studies that feed into assumptions for actuarial models.
- **Regulatory Submissions**: Files that are used to compile reports required by regulators.

Selecting candidates with these criteria ensures that the migration effort is **strategic, risk-managed, and high-impact**. Less critical spreadsheets, or those used for short-term analysis or proof-of-concepts, can remain in Excel, allowing teams to maintain agility while focusing development efforts where it matters most.

6.2 START SMALL, SCALE GRADUALLY

The temptation may be to tackle the largest, most business-critical spreadsheet first, but this is often a recipe for frustration. Instead, a more effective approach is to **start small and select a moderately complex model** that highlights the benefits of Python without overwhelming the team. Early wins build confidence, demonstrate tangible value, and provide learning experiences that can be applied to larger projects later.

Scaling gradually also helps to manage organisational change. Stakeholders can see that Python-based systems deliver the same outputs with greater control and transparency, while the teams involved gain practical skills and confidence. By the time the organisation is ready to migrate its most critical spreadsheets, the people, processes, and infrastructure are already in place to support success.


The Migration Roller Coaster of Emotions
Mature ecosystem
First major app reusing patterns
Minor setbacks (they happen!)
Second successful integrated app
First integrated app
Proof of Concept: Excel + Python Notebook written with LLM

6.3 BUILD REUSABLE FOUNDATIONS

Migration is faster and more consistent when teams **avoid reinventing the wheel** for every model. Establishing a set of reusable foundations is therefore essential. This could include shared Python libraries for common actuarial or financial functions, standardised approaches for input validation and reporting, and frameworks for automated testing.

These foundations serve as a springboard, enabling analysts to focus on the logic that differentiates their models rather than the plumbing required to make them work. Over time, a **library of reusable components** reduces duplication, improves quality, and ensures consistency across teams. It also accelerates future migrations, as each new project builds on the lessons and tools developed before it.

6.4 INTEGRATION WITH ENTERPRISE SYSTEMS

A standalone Python script is already an improvement over a fragile spreadsheet, but **the real power comes when models are integrated into the wider enterprise ecosystem**. This might mean connecting to a central data warehouse, exposing results through an API, or deploying models onto cloud infrastructure that scales on demand.

Integration transforms models from isolated tools into shared organisational assets. Rather than emailing spreadsheets back and forth, results can be consumed directly by downstream systems or visualised in dashboards. This reduces duplication, ensures consistency, and enables more agile decision-making. Integration also ensures that Python-based models meet the same standards of governance, security, and reliability as other enterprise systems.

6.5 UPSKILLING AND CHANGE MANAGEMENT

Last but not least, perhaps, **the most important pathway to migration is people**. Analysts and actuaries who have relied on Excel for years need support to learn Python, and organisations must create a culture that encourages this shift. Formal training, mentoring, and hands-on projects all play a role, but equally important is fostering a mindset of curiosity and continuous improvement.

Change management is critical here. Migration should be framed not as a loss of familiar tools but as **an opportunity to gain new skills and increase impact**. Upskilled professionals remain domain experts while also becoming more technically versatile, bridging the gap between business and technology. This dual expertise ensures that the organisation does not just replace spreadsheets with code but elevates its overall modelling capability.

7 CONCLUSION

Excel has served as the workhorse of financial modelling, but its limitations become acute as organisations scale and regulatory scrutiny intensifies. Version control, user access, scalability, and audit logging are not optional extras in this environment; they are fundamental requirements if an organisation is to have forward-looking success.

While it is excellent for prototyping and exploration, but once models underpin regulatory filings, business-critical decisions, or enterprise-scale workflows, **the spreadsheet ceiling becomes unavoidable,** and the **need to move into something more robust** in code and in systems becomes apparent.

Nevertheless, **building production-ready systems is inevitably more complex** than working in spreadsheets. It requires careful design, robust infrastructure, and a blend of technical and domain expertise. Organisations that treat this process seriously — investing in the right skills, teams, and resources — are the ones that consistently achieve reliable, scalable, and auditable models.

Such organisations **gain a tangible competitive edge against other players** in the market by moving more quickly. With streamlined automation, faster calculations, and centralised access to **trusted data**, actuaries and analysts can make better-informed decisions more readily. This not only reduces costs and operational friction but also enables teams to respond nimbly to business opportunities and regulatory demands, ultimately supporting stronger client outcomes and growth.

If you are currently exploring moving your critical models from spreadsheets into robust systems, having guidance from experienced practitioners can make a meaningful difference. Proper planning and expert insight ensure the transition is smooth, efficient, and aligned with your organisation's strategic goals.

Let us know which low-hanging fruit projects you're currently looking at productionising into code. Perhaps there are purpose-built solutions that make the transition both painless and immediate.

Get In Touch ›

https://lunoxtech.com/get-in-touch

8 REFERENCES

Statistics Graphs:

https://www.statista.com/chart/16567/popular-programming-languages/

https://www.devoriales.com/post/374/python-tops-the-tiobe-index-the-most-popular-programming-languages-january-2025

https://coding-bootcamp-2021.acmbpdc.org/02-overview-of-programming-languages/

https://www.anaconda.com/blog/python-in-excel-for-finance

https://www.researchgate.net/publication/392758009_Python_for_Finance_A_Modern_Guide_to_Building_and_Interpreting_Financial_Models