Four Python Myths Debunked: Lessons Learned Building a Python Email API

We built our Python email API to handle billions of request per week, and dispelled four myths along the way.

Tasia Potasinski | July 8, 2019

In less than two years, we built a Python email API that's trusted by tens of thousands of developers across 22 countries. Our API is implemented in applications from companies you're sure to recognize - brand names like Hyundai, Ellie Mae, and Newscorp.

Despite Python's surging popularity (solidly ranked 3rd among the top 10 most used languages) as a mainstay programming language, it's often dismissed as "a scripting language," "too slow," and "not as good for enterprise-scale production."

What belies these statements are the flourishing Python ecosystem and its many libraries — the multitude of open-source Python packages enabling almost every use case and optimization.

Myth #1: Python is Old and Outdated

Okay, so Python *may* be an old language, but it's certainly not outdated — it's grown and evolved with the times.

Python is a high-level, general-purpose programming language that can be applied to many different classes of problems. Programming instructors have come to favor Python's versatility and accessibility for introductory coding workshops as of late, but the language has a long history that even predates Java.

First released in 1991, Python has come a long way with the support of its creator Guido van

Rossum, essential core developers, and an ever-growing community of users and fans.

Core upgrades, especially Python 2 and the relatively recent major release of Python 3, have provided tools not just for scripting, but for building complex software applications that are performant and scalable as well.

Myth #2: Python Doesn't Scale

We don't need to look any further than companies like YouTube, Dropbox, or Instagram, who run massive scale operations with a backend running on Python, to disprove this myth. In the rest of this post, we'll discuss some of the reasons why this language makes a good choice for enterprise web services and why Python powers the Nylas sync engine.

Myth #3: Python Is Unreliable

Running an API software service at scale means coordinating distributed systems and making the most of limited memory resources to handle an unceasing deluge of data. It isn't long before a single server is overpowered and becomes a bottleneck for operational growth. Imagine the specifications for a server that would be powerful enough to run YouTube on its own! The answer lies in the cloud: networked machines cooperating with each other to efficiently distribute the load.

> *"Python is a programming language that lets you work quickly and integrate systems more effectively."*
>
> — Python.org

Myth #4: Python Is Slow

One topic that comes up with regard to speed is the advantage held by lower-level languages. Programming languages that offer lower-level access to underlying computation processes, such as C, are naturally able to operate faster thanks to their fine-grained control. But that fine-grained control comes with many usability trade-offs which much be weighed into the pros and cons of the language choice. Higher-level languages, such as Python, are more usable and easier to write, but not as fast as lower-level languages. To get around this limitation, Python has bindings to languages like C, and these bindings, in essence, enable a developer to write code in Python and have it run as fast as if it were written directly in C.

Why We Built the Nylas Email API on Python

When Nylas first got its start, we had to consider how we'd build our sync engine (the code that powers our API) and what language to build it in. We decided that our best approach would be to

mirror the contents of mailboxes and serve as many requests as we could directly from our own servers. This was driven primarily by the need to deliver reliability to organizations that depend on very high uptime. We sharded databases and distributed our services across multiple machines, which ultimately positioned us to better tackle bigger challenges, like threading emails, syncing tags and folder management.

> *"Python is standard, reliable and (best of all for a startup like ours) boring."*
>
> — Christine Spang, Nylas CTO

We chose Python because of its simplicity, its diversity of libraries, and its ability to work well with servers. The language syntax is simple and expressive, there are tons of open source modules and frameworks available, and the community has values Nylas stands behind like diversity, openness, and transparency.

To build out our architecture, we doubled down on Python because we wanted to make use of its many existing libraries. This allows us to leverage the work accomplished by other smart developers on the individual pieces of the problem, while we focus on the greater sync system we are building. For example, we use flanker, a well-developed parsing library, to help facilitate email deliverability. We want our Python stack to make use of multiple cores and threads, so we use Gevent, a coroutine library, to sync about 100 accounts on a single process.

We use many other well-known Python libraries like Flask, SQLAlchemy, and pytest in addition to other web stack tools like HAproxy, nginx, gunicorn, MySQL, ProxySQL, Ansible, Redis and more. We recently wrote about 5 Python Libraries We Love for building systems that scale.

The Nylas team architected a reliable tech stack that doesn't waste time trying to reinvent the wheel. It simply helps to sync and deliver email while meeting everyone's expectations on uptime, scale, and stability. Quite simply, Python offered the best set of tools for the job.

The result of our efforts has been the Nylas cloud-core engine that helps process billions of email, calendar, and contacts events every week. Our core code base has more than 275,672 lines of Python code, (some of which is open source). It powers a RESTful API that enables developers to integrate messaging into their applications. It includes tests and migrations and helps to wrangle an ecosystem of protocols, protocol offshoots, parsing, encoding and much more.

Python Prevails

Python is reliable and battle-tested as an enterprise-level web services programming language. Python offers a variety of functions that aid in building a reliable product efficiently. With its robust libraries, your Python code will be readable, maintainable, and compatible with other major platforms and systems as you scale. Using Python will also simplify complex development issues

and make it easy to implement vital testing measures. Python may be the latest rising star, but its fame will surely be no passing fad!