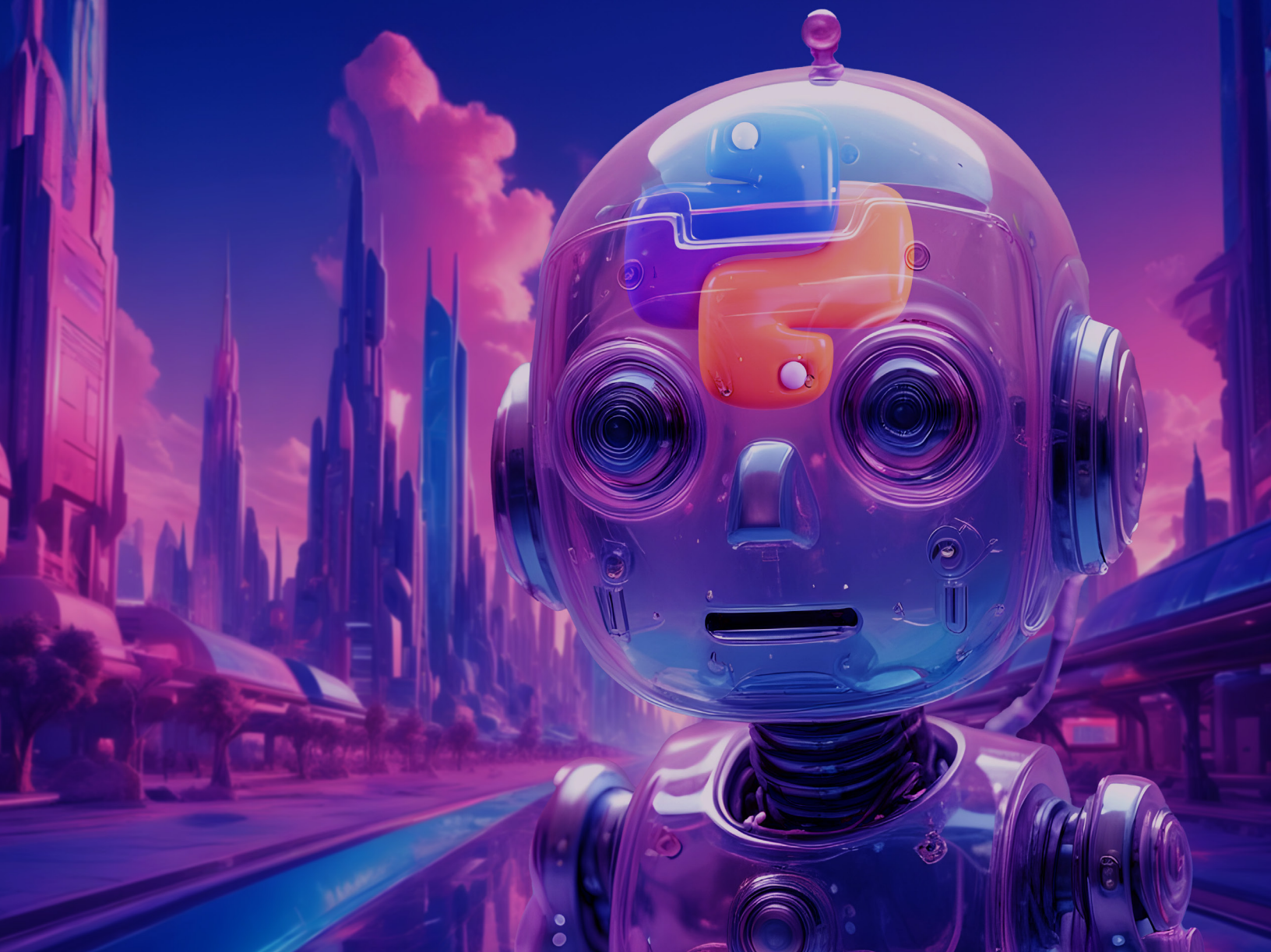
ActiveState

ACTIVESTATE VS ANACONDA: **PYTHON FOR DATA SCIENTISTS**



Executive Summary

With the rise of the citizen developer, the need for IT to source easy-to-use tooling has become paramount since citizen developers like data scientists are rarely coding experts. Because Python is the primary language for data scientists, easy-to-use Python tooling is key.

But data scientists and the code they create will need to be productized by enterprise developers, who will require an enterprise-grade version of Python to work with. To keep tech stacks manageable, IT requires a single hand to shake: one Python provider that can fulfill both their citizen and enterprise developer needs. Unfortunately, that kind of "simple but powerful" quality is typically more prevalent in the marketing of a Python distribution than the actual use of it.

Issues arise whenever a Python distribution attempts to mask complexity, such as:

- **Proprietary Package Management -** conflicts between a proprietary package manager (like conda) and Python's native package manager (pip) can corrupt environments.
- **Package Installation -** Anaconda's Python catalog is limited, forcing users to source (or build) the packages they need elsewhere.
- **Dependency Management -** clarity around changes to environments when installing (or removing) packages is critical to prevent dependency conflicts.

Issues like these have driven Anaconda users to search for alternative Python providers, such as ActiveState Python that fulfills the requirements of both citizen and enterprise developers:

- Replaceable drop-in for Anaconda Python environments that can even be run alongside existing Anaconda deployments.
- 100% compatible with open source Python.
- A Web GUI that dramatically simplifies creating and working with Python environments for non-professional developers.
- A powerful command line interface (State Tool) that supports the way enterprise developers prefer to work.

Introduction

Citizen developers are an idea whose time has come. With languages like Python and coding tools such as the AI-enabled Copilot making programming easier than ever, citizen developers are finally empowered to translate their expertise into digital assets.

Citizen developers are an attempt by enterprises to empower their Subject Matter Experts (SMEs) while also promoting innovation. The trend is being driven by two concerns:

- The availability of skilled developer resources, which are always at a premium in any organization.
- The length of time it takes coders to bridge the gap between subject matter expertise and software requirements, which too often ends up imperfect anyway.

The thinking is that if SMEs can encode their expertise in applications by becoming citizen developers, the result should benefit the organization, despite the additional burden it often places on IT.

One of the most common types of citizen developer is that of the data scientist. While they work primarily with the Python programming language, few data scientists would characterize themselves as Python programmers. Rather, Python is simply a tool that allows them to get their data analysis, scientific computing, AI or Machine Learning work done.

But just like citizen developers, data scientists must work with runtime environments approved and sanctioned by IT, who are generally looking for a single hand to shake: one provider that can fulfill both their citizen and enterprise developer needs. Unfortunately, that kind of “simple but powerful” quality is typically more prevalent in the marketing of a solution than the actual use of it.

Trying to find a “one size fits all” solution can be difficult. You’ll want to include a representative from each internal team to test drive the tool in order to ensure that it can fulfill a wide range of use cases from the novice to the power user. But what’s effective in a sandboxed data science environment rarely scales to production use. Developing a data analysis routine is one thing (a mortgage risk calculator, for example), but it’s quite a different task to implement that routine as a software solution (e.g., an online mortgage qualifier).

The result is too often a recommendation to implement multiple tools – one for each user type -- or else compromise on a substandard solution, which can mean:

- Multiple versions of Python for IT to support.
- Extra work to abstract away complexities or shortcomings.
- Workarounds to integrate third party tools, and so on.

Data Scientists & Python

While there are numerous programming languages (R, Julia, etc), as well as ecosystems like MatLab that are specifically designed for scientific computing, Python is where the greatest innovation has been happening for the better part of a decade, especially in the fields of AI and Machine Learning.

As a result, the majority of data scientists have established a working relationship with Python in general, and the Anaconda Python distribution, in particular. Founded in 2012, Anaconda is widely seen as the commercial Python market leader:

- Anaconda Python ships with Microsoft Visual Studio, making it the default version of Python on Windows developer desktops.
- Anaconda Python is frequently cited as the distribution of choice for “learn Python / learn Data Science” web sites since it ships with 250 of the most popular data science packages.

But Anaconda has moved on from centering their business around commercial support for their Python distribution, and is now focused on their data science platform instead. Data scientists that embrace Anaconda’s platform can gain an enterprise-strength solution for their needs. But those that don’t may struggle to adapt the Anaconda distribution to fulfill use cases it was never designed to support.

For example, one of the key selling points of Anaconda for citizen developers is its ease of use. By bundling a number of the most popular data analysis and data visualization packages in a single distribution, data scientists have ready access to whatever they want to install in their current project, either via their command line tool (conda) or (even better for newbies) via their GUI (conda Navigator). Conda is a fully capable package manager, able to install not only Python packages but C, Fortran and other binary dependencies that are a staple of data science packages since they speed up numeric computation.

Unfortunately, such an approach raises a number of issues, such as:

- **Dependency Management -** Citizen developers are generally unaware of how package management works, especially when it comes to a package’s dependencies. As a result, they don’t understand why installing or updating a single package can result in a cascade of changes that Anaconda will typically take a very long time to resolve, or perhaps won’t be able to resolve, which may leave the user with an unworkable environment.

ActiveState takes a different approach. ActiveState’s command line tool (State Tool) is meant for power users, but the ActiveState Platform Web GUI will walk inexperienced users through the changes that occur when a package is installed, updated or removed. The UI is visually instructive, letting users know exactly what changes are going to happen (both at the package and dependency level) BEFORE committing to them, and resolving any conflicts that may occur so users don’t end up with a corrupted environment.

In contrast, to work around Anaconda's dependency management issues, IT usually encapsulates the Python environment in a Virtual Machine (VM) or a container so they can quickly spin up a new version. This adds more work for IT, as well as more complexity for citizen developers. Worse, unless these containers/VMs are periodically checked into a version control system, restoring the original environment loses any changes citizen developers may have made.

ActiveState takes a git-like snapshot of the Python environment every time a change is made. Restoring a previous version is as simple as clicking "Revert" within the Web GUI.

- **Package Availability -** While Anaconda ships with hundreds of the most popular data science packages, it can be confusing for citizen developers when a package they need is not available in their Anaconda distribution. While Anaconda (the company) will, for a fee, build the package in a few days/weeks, it's often cheaper, easier and quicker to check whether the community has already built it.

  If neither of these paths work, citizen developers will need to adopt Python's default package management tool (pip) and install the package they require from the Python Package Index (PyPI). Unfortunately, installing a package with pip can bring along dependencies that conflict with existing Anaconda installed dependencies, breaking their environment.

ActiveState effectively solves these issues by mirroring all of PyPI, including all of the most popular data science packages so users rarely experience a missing package issue. Daily ingestion of new packages as they are published on PyPI means ActiveState's catalog of Python packages is always up to date, compared to the Anaconda distribution, which is infrequently updated. Additionally, ActiveState maintains a deep catalog of previous package releases, ensuring you can always reproduce an older environment. This addresses an issue with Anaconda where confusion can arise when a package suddenly becomes deprecated.

- **Production Readiness -** When citizen developers have finished their work, they typically hand it off to enterprise developers to bring it to production. But enterprise developers are unlikely to work with Anaconda Python given its large footprint that results in excessive RAM consumption, making it extremely slow when it comes to loading, implementing changes, or even switching between multiple environments. Miniconda is preferred since it starts with very few pre-installed packages, but enterprise developers may find it overly opinionated, and will undoubtedly also run into limitations when it comes to package availability.

ActiveState makes it a point of pride to support both citizen developers (typically via our Web GUI) and enterprise developers (typically via our CLI, State Tool). The GUI's wizard-driven approach to creating a Python environment simplifies getting started, as well as making modifications to the environment at any time. And since ActiveState Python defaults to installing into a virtual environment, dependency conflicts are avoided.

But for ActiveState, being able to go from sandbox to production is as simple as supporting a single command to check out and start working in an environment (state activate) for citizen developers, while providing enterprise developers with more sophisticated capabilities, such as the ability to checkout multiple environments (state checkout), and switch between them at will (state use). And like conda, State Tool supports not only Python, but also the C, Fortran, Rust and other binary libraries common to data science tools.

Conclusions

Given that citizen developers are business experts rather than technology experts, professional developers will need to become active participants in the development process at some point in order to ensure quality, security and functionality.

The responsibility for sourcing and supporting solutions for both citizen and professional developers typically devolves to IT. While modern tech stacks include multiple open source languages, and each language will require multiple tools, IT departments would prefer not to support non-standard tooling, especially if interoperability with an ecosystem's standard tooling has been shown to be problematic, as with Anaconda.

ActiveState provides an alternative to Anaconda that can fulfill the requirements of both citizen and enterprise developers:

- Replaceable drop-in for Anaconda Python environments that can even be run alongside existing Anaconda deployments.
- 100% compatible with open source Python.
- A Web GUI that dramatically simplifies creating and working with Python environments for non-professional developers.
- A powerful command line interface (State Tool) that supports the way enterprise developers prefer to work.

But ActiveState also comes with a number of other benefits, including:

- **Environment Reproducibility -** ensure projects are always deployed in a consistent, reproducible manner.
- **Supply Chain Security -** take advantage of a continually updated catalog of vetted packages automatically built securely from source code.
- **Vulnerability Management -** centrally track vulnerabilities per project, identify fixes, and then automatically rebuild vulnerable environments in minutes.

In other words, ActiveState Python can not only replace Anaconda Python, but is the only solution you'll need to take your project from sandbox to production without compromise.

ActiveState

Secure open source integration

www.activestate.com
Toll-free in NA: 1-866.631.4581
solutions@activestate.com