INFOWORLD TECH WATCH
By Serdar Yegulalp, Senior Writer, InfoWorld
JUN 27, 2016

About
Informed news analysis every weekday

4 languages poised to out-Python Python

Swift, Go, Julia, and R are all potential contenders for Python's crown of convenience and versatility. Here's how each could win out -- and how Python could prevail

Nothing lasts forever -- including programming languages. What seems like the future of computing today may be tomorrow's footnote, whether deserved or undeserved.

Python, currently riding high on the list of languages to know, seems like a candidate for near-immortality at this point. But other languages are showing that they share Python's strengths: convenient to program in, decked out with powerful ways to perform math and science work, arrayed with a huge number of convenient third-party libraries.

[The art of programming is changing rapidly. We help you navigate what's hot in programming and what's going cold and give insights into the technologies that are changing how developers work. | Keep up with hot topics in programming with InfoWorld's Application Development newsletter.]

Here's how four potential challengers to Python shape up against it, and how Python can still keep its edge.

Swift

What it is: Apple's language, originally for iOS development, but now open source and shaping up to be of interest for server-side development as well.

How it's a challenge: Writing code in Swift is a frictionless experience, more akin to a scripting language (like, say, Python!) than a compiled language like Swift's indirect predecessor, Objective-C. Where Swift has a decided advantage is execution speed -- it's compiled to machine code by way of the LLVM compiler framework, so it supports true multithreading, which Python is still struggling with.

If developer speed is more important than execution speed, another major Python selling point, Swift also has an interpreted "Playground" mode via the Xcode IDE.

How Python still has its lead: For one, Swift's still a new language compared to Python, and so Python has all of the advantages inherent to any incumbent language -- a big captive userbase, plenty of libraries, broad and well-tested platform support. Swift doesn't even yet run on Windows (barring third-party efforts), although that's planned for sometime in the near future. Swift was also originally created to directly complement Apple's toolchain (e.g., Xcode), while Python has fewer dependencies.

Go

What it is: Google's "expressive, concise, clean, and efficient" language, now powering everything from Docker and its associated projects to the InfluxDB database, the Ethereum blockchain system, and Canonical's Snappy package manager.

How it's a challenge: Like Swift, Go compiles to platform-native binaries, so it not only runs far faster than Python for many tasks, it can be deployed cross-platform without needing a Python runtime at the target. Go programs also compile so quickly that it hews closer to an interpreted language rather than a compiled one in terms of its development speed.

How Python still has its lead: While Go isn't as new as Swift -- it debuted to the public in 2009 -- Python still has the larger user base and library assortment. Also, Go's syntax and approach to error handling are alienating to current Python users. Consequently, it's unlikely existing Pythonistas will switch projects to Go, although none of that will stop newcomers from picking up on the language. And as far as runtimes go, utilities like Pyinstaller have made it far easier to bundle Python apps -- not to mention that on most any Linux system, a Python runtime is a standard-issue item.

Julia

What it is: Unveiled in 2012, Julia is dedicated to technical applications, such as data analysis and linear algebra.

How it's a challenge: One of Python's major use cases is for math and science applications, thanks to libraries like Numpy and the interactive IPython notebook format. Julia is aimed at much the same user base, and like Go and Swift, it's faster at its core than Python. It also features a growing list of packages, covering not just math and science applications, but also other functionalities associated with Python, like connectivity to data sources on cloud providers.

How Python still has its lead: Julia has a relatively package index compared to Python. But beyond that, the existing community of development around Python for math and science work isn't sitting on its laurels -- it's advancing both the core language and the environment around it, nonstop. It's also not as if Python can't run as fast as Julia (or many of Python's other competitors), as long as you use the right libraries for the right job.

There's also skepticism about the way Julia has been put together. Random example: Julia's arrays are 1-indexed rather than zero-indexed -- in stark contrast not just to Python, but almost every other language out there. (It's likely this was meant to complement packages like Mathematica that also use 1-indexing, as a way to bring in users of that system, but it's still jarring.)

R

What it is: A long-standing project -- both a language and a development environment -- for statistical computing.

How it's a challenge: R has many of the benefits Python likes to claim for itself, such as a rich ecosystem of third-party packages. R is also designed with statistical computing in mind and remains focused on that. Python does math and stats among other things, but math and stats are what R is about from top to bottom.

R's also drawn the attention of some big names. Microsoft acquired the makers of one of the standard implementations of the language to complement its own cloud-based data services. Hewlett-Packard has developed a Distributed R product that can run across many nodes at once. With their involvement, future versions of R could push Python off the map when it comes to statistical work.

How Python still has its lead: Sometimes, though, being a general-purpose language has its advantages. R is rather limited in what it can deal with -- there's little in the way of creating interactivity with running R applications, for instance. It's also generally easier to get onboard with Python as a language than with R -- or to use a package like RPy2 to connect Python to R and get the best of both worlds.

Finally, if the involvement of Microsoft seems like a slam-dunk benefit for R, bear in mind Microsoft's also providing Python with a few helping hands so it will run well in Azure.

Serdar Yegulalp is a senior writer at InfoWorld, focused on machine learning, containerization, devops, the Python ecosystem, and periodic reviews.

Follow

YOU MIGHT LIKE