Python vs. Other Programming Languages

Find out how Python compares to other popular languages and decide if it's the best choice for your software project.

STX NEXT
python powerhouse

Table of Contents

Introduction	1
Why Python?	2
Python vs. Golang	6
Python vs. JavaScript	9
Python vs. Node.js	12
Python vs. Java	16
Python vs. Ruby	19
Python vs. PHP	21
Python vs. R	24
Python vs. C++	27
Python vs. C#	30
Final Thoughts	32

Introduction

Building software is like building a house. In both cases, what you need the most is a strong foundation.

Use a weak foundation, and you will struggle with expansion, suffer costly repairs, or possibly be forced to rebuild the whole thing from scratch down the line. But use a strong foundation, and you will scale up smoothly, the upkeep will be a breeze, and your project will be built to last.

If the house is your software, then the foundation is your programming language.

We're going to show you why Python is one of the best programming languages out there and explain the many reasons you should consider choosing it for your software project. However, robust as it may be, Python isn't the only programming language worth its salt. There are many other Python alternatives to choose from, including:

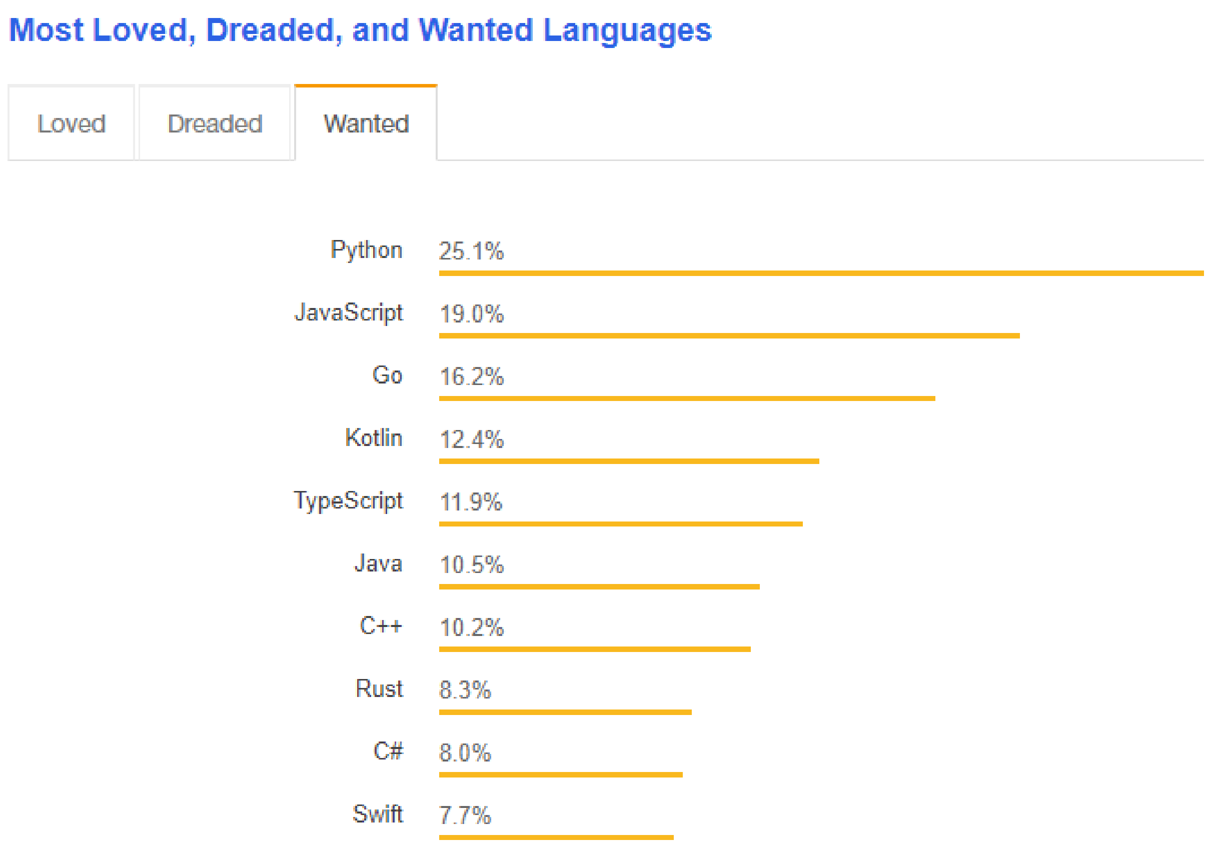
- Golang,
- JavaScript,
- Node.js,
- Java,
- Ruby,
- PHP,
- R,
- C++.
- C#.

We're also going to show you how Python compares to these programming languages, as impartially and informatively as we can. We'll let you decide which of them is the best choice for your software. Let's start with Python, then move on to the others.

Why Python?

Since 2012, Python has been consistently growing in popularity, and the trend is likely to continue, if not increase, in the future.

Per the Stack Overflow Developer Survey 2018:



Python	25.1%
JavaScript	19.0%
Go	16.2%
Kotlin	12.4%
TypeScript	11.9%
Java	10.5%
C++	10.2%
Rust	8.3%
C#	8.0%
Swift	7.7%

The demand and support for Python are also on the rise, and if projections are to be believed, Python will overtake Java in the coming years and claim the top spot.

Is Python's popularity a good thing?

Yes, very much so. While generally what is popular isn't always the best, in the case of programming languages the popularity pays off.

Thanks to Python's popularity, you're likely to find a ready-made solution to any problem you may be experiencing. The community of Python enthusiasts is strong and they are working tirelessly on improving the language every day.

Python also has a number of corporate sponsors, pushing to popularize the language further still. Among them are tech giants such as Google, which itself is using Python.

Fast development speed in Python

Python is designed to be accessible. This makes writing Python code very easy and developing software in Python very fast.

What does that mean for your development team? Less time wasted struggling with the language and more time spent building your product.

Python's numerous libraries and frameworks

A huge advantage of Python is the wide selection of libraries and frameworks it offers. Your time-to-market will improve if you leverage them, since you won't be coding features manually.

There's a Python library for everything:

- data visualization,
- machine learning,
- data science,
- natural language processing,
- complex data analysis.

From NumPy to TensorFlow—you name it, Python has it.

The same is true for frameworks, which help get your project off the ground and save you time and effort.

There's a variety of frameworks to choose from, depending on your needs, such as:

- Django,
- Flask,
- Pyramid,
- Twisted,
- Falcon.

Performance of Python

One of the biggest criticisms of Python is the runtime, relatively slow when compared to other languages. However, there's a workaround to this specific challenge.

When performance takes priority, Python gives you the ability to integrate other, higher-performing languages into your code. Cython is a good example of such a solution. It optimizes your speed without forcing you to rewrite your entire code base from scratch.

Besides, the priciest resource isn't CPU time; it's your developers' time. Reducing your time-to-market should therefore always take precedence over fast runtime execution.

Easy maintenance in Python

Python is intuitive to read, because it resembles actual English. This makes the language effortless to decipher and maintain.

Additionally, Python has a clear syntax and doesn't require as many lines of code as Java or C to give you comparable results.

What are the benefits of Python's high readability?

Python's simplicity is particularly helpful in reading code—yours or someone else's. Because Python code has fewer lines and mimics English, reviewing it takes a lot less time. This is a major benefit.

Reducing the time you need to spend on code review is invaluable, since the productivity of your developers should be your top priority.

Reliable scalability of Python

Scalability is unpredictable. You never know when your user numbers surge and you find yourself prioritizing the ability to scale over anything else.

That’s why Python is such an optimal choice, with its reliability and scalability. Some of the biggest players on the web, like YouTube, have bet on Python for that very reason.

Takeaways: Should you use Python?

Why Python? Because:

- it’s popular, fast, readable, intuitive, simple, clear, and scalable;
- it offers a ton of useful frameworks and libraries;
- it enjoys a vast and ever-growing community of supporters and enthusiasts.

Head over here for an in-depth look into why Python is the right choice for your tech stack.

Python vs. Golang

Effortless development in Python and Golang

Python is so flexible and readable that it can be understood without any prior knowledge of the language—the same is true for Golang. Neither of the two requires so much as reading one tutorial to follow their code.

Go in particular is easy to find your way around. Within the first 24 hours of being introduced to Go, you're able to start making changes to software written in it.

Similarities between Golang and Python

The main similarity between Python and Golang lies in high-level types.

Go's *slices* and *maps* resemble Python's *lists* and *dicts*, only statically typed.

Also, *enumerate* in Python functions as *range* in Golang.

And... this is where the similarities end.

Differences between Python and Go

There are many more differences than similarities between Python and Go, some of them likely to shock Python developers.

For instance, Golang doesn't have *try-except*, instead allowing functions to return an error type along with a result. Therefore, you need to check whether an error was returned before you use a function.

The greatest difference between the two languages, however, lies in typing. Python is dynamically typed, while Go is statically typed. Python is also an interpreted language, as opposed to Golang, which is a compiled language.

Go has a number of other surprises in store for Python developers to learn, including:

- *channels* (sending messages between *goroutines*),
- *defer* (replacing *try-finally*),
- *structs* (compound types).

High readability of both Go and Python

The reason Python developers are able to understand Golang without much trouble is because the design of Python and the design of Go are based on similar principles.

If we compare the Zen of Python with the guiding principles of Golang, we notice that both languages prioritize simplicity and minimizing clutter:

1) Python values readability, while the clean syntax of Go leads precisely to high readability.
2) "Simple is better than complex" for Python, and the same is achieved thanks to the orthogonality of Golang.
3) The static typing of Go aligns with the rule of "explicit is better than implicit" in Python.

Which language is better—Python or Golang?

Python is an excellent choice for data science and the web. Meanwhile, because Golang is compiled and statically typed, its performance is much faster than that of an interpreted and dynamically typed language like Python.

So should you choose one over the other? We don't think so.

The most optimal approach is to use Python and Go together.

Microservices or serverless are likely the best ways to go about it. When code performance is your top priority, consider writing the code in Golang and using Python for everything else.

Takeaways: Should you use Golang or Python?

The design similarities between Python and Golang make transitioning from one to the other seamless and enjoyable.

Hopefully, we will see more and more projects combining the two languages in the near future.

Read the full story of a lifelong Pythonista's venture into Golang here.

Python vs. JavaScript

Comparing Python and JavaScript is not easy. They were designed to deliver different outcomes, so it's difficult to point out which one is "the best."

As artists use different brushes to create a magnificent masterpiece, software developers might want to work in several programming languages to achieve an optimal result.

Many pieces of software you use every day are based on multiple programming languages.

But before looking at examples of JavaScript and Python combined, let's look at the top qualities of them both.

Advantages of Python: simple, clean, and fast to write

Python is a great general-purpose language. Due to easy syntax, it gained popularity not only among software engineers but also with data scientists and academic researchers.

Its simplicity is best for overcoming complex problems and make it the most popular choice for machine learning and data processing.

In terms of web development, Python is typically applied to build the backend and for server-side scripting.

Advantages of JavaScript: great for web apps and frontend development

The things that everyone associates with JavaScript are web applications and frontend development. Together with HTML and CSS, it allows developers to create interactive elements on websites. JavaScript is well known for its frameworks that established it as a go-to in terms of web development.

In the 2020 StackOverflow developers survey, JavaScript was used by the majority of respondents (67.7%)—the second most popular language was Python, used by 44.1%.

Why is comparing JavaScript and Python difficult?

As written above, JavaScript and Python serve different purposes. That itself makes them hard to compare. Yet, there are more reasons why this task is a challenge. For instance, if we want to indicate which of them is better in terms of performance, it depends on various details.

In general, JavaScript works better in I/O with big amounts of data to be processed in real-time (like on social media platforms) while Python runs better in CPU-intensive situations (like building a machine learning model that needs to solve a specific problem).

In terms of web development, environments like Node.js makes JavaScript a more efficient language at runtime. Python can process requests only in a single flow, not multithreading, which makes it slower.

However, with the help of optimized C code (using tools like NumPy or Cython), Python software developers can take advantage of C modules that speed up the execution of code.

Why not both? Are Python and JavaScript a match made in heaven?

Seeing how different Python and JavaScript are, the question is if comparing them even makes sense. Another approach would be to use them in one tech stack. It's not without reason that world-class applications like Instagram run on a combination of Python and JavaScript.

Expectations for Instagram are high—the software has to be able to process massive amounts of data, from traveling chickens to miserable geography. Simplicity and practicality play a crucial role here. These might be the reasons why the entire Instagram servers run on Python—software engineers can quickly understand code and debug if necessary.

On the other hand, the Instagram interface is built with React Native, a JavaScript framework. It is perfect for mobile apps as it provides high developer velocity and allows engineers to write the code once and ship features to both iOS and Android. Maintaining the app in two ecosystems at once allows companies to optimize costs and development time.

The same stack of Python and JavaScript can be found in the architecture of PayPal.

PayPal uses JavaScript (via the Node.js framework) to build their application due to its fast performance on the web. While for the mission-critical components, it runs on Python (along with a few other languages).

The combination of JavaScript and Python provides the best of two worlds: fast time-to-market and satisfying speed. Instead of competing, the two languages can yield great results working together.

The essential differences between Python and JavaScript

- JavaScript works better to build the frontend, whereas Python is better for backend and server-side scripting work.
- JavaScript runs faster compared to Python, but it takes less time to produce Python code.
- Python is better for data analytics, machine learning, or artificial intelligence as it's easier to understand and maintain than JavaScript.

Takeaways: Should you use JavaScript or Python?

Summing up, JavaScript and Python are so different that comparing them can be unproductive. A more helpful approach is to understand the strong suits of each and use them to build better software.

Check out the entire article why comparing JavaScript and Python isn't fair.

However, there is one situation where JavaScript and Python can go head to head: that's deciding between writing the app's backend in Python vs. using Node.js.

Python vs. Node.js

Comparing Node.js and Python

It’s important to remember that Node.js is a runtime environment for JavaScript, not a programming language like Python. Because of that, writing in Node.js means you’re using the same language on the frontend and the backend.

Advantages of Python over Node.js

1) More beginner-friendly

At a more advanced level, JavaScript can be difficult to understand for developers with less Node.js experience. They may make some fairly common mistakes, slowing down development in the process.

This is not the case with Python, since it’s easier to use for less experienced developers. The mistakes made by them will have less of a negative impact on development.

2) Lower entry point

Frameworks such as Django are mature, increase the quality of your code, and speed up the process of writing it—all without the need to lean on highly skilled developers.

3) More applications

Node.js is mostly used for the web, while the applications of Python are far greater.

The universality and versatility of Python are among the top reasons why the language is a great fit for trending technologies such as data science.

4) Better implementation

JavaScript runtime environments and frameworks all implement the language differently; Node.js is no exception. Frankly, the ecosystem of JavaScript is a bit of a mess—though nowhere near as bad as it used to be.

Python doesn't have that problem, which is why it's simpler and easier to use. It also makes the language faster to write in, although Node.js is anything but slow.

Advantages of Node.js over Python

1) More flexible developers

It's necessary to know JavaScript if you wish to use Node.js, since you're dealing with the same language on the frontend and the backend.

This requires more flexibility and higher understanding of the project from your developers.

2) Less opinionated ecosystem

Packages for Node.js are often simple and designed for one purpose only. This pushes developers to think more carefully about what they want to use and when they want to use it.

Because of this, Node.js requires your developers to be more advanced. Writing Python code in Django isn't anywhere near as demanding.

3) Coding everything in JavaScript

The number of Python or JavaScript developers your software product is going to require is often impossible to predict as you assemble your development team.

Using JavaScript on both for the frontend and the backend with the help of Node.js solves that problem altogether. You rely on fewer technologies and save time in the process.

4) Fast growth and large community

Since 2012, Python has been consistently praised for its great community and support—and rightly so. But the days of its huge frameworks and libraries advantage are over now.

These days, JavaScript is just as well supported. It keeps growing with no signs of stopping and remains very much in the lead of the most dynamically expanding languages in the business.

Development history of Python and JavaScript

JavaScript has seen its fair share of growing pains. The code was all over the place back when it was first being developed, and its old versions are still causing compatibility problems to this day.

Consequently, the growing popularity of JavaScript in recent years has brought with it an endless swarm of updates and significant documentation issues. This hurts the quality of Node.js.

Once again, Python has the upper hand here. The documentation and coverage of Python are both superior to Node.js. When it comes to reliability, Python has always been ahead of JavaScript.

Trending technologies with Node.js and Python

The chaotic ecosystem of JavaScript also makes Node.js too unstable and unpredictable to rely on for trending technologies.

Because of the significant fluidity of JavaScript trends, JavaScript technologies become outdated much more quickly. This is why Node.js is a risky choice for emerging technological trends, even though it allows you to use them earlier.

Python, on the other hand, doesn't pose that risk, since it introduces substantial changes very slowly. The language is a perfect fit for trending technologies such as machine learning or data science, with its top-notch experts and library support.

Performance and speed of Python and Node.js

Node.js may struggle with executing a lot of tasks at once. If the code isn't written very well, your product will perform poorly and work slowly.

This may also happen with Python, but Python frameworks such as Django come with ready-made solutions to help your software withstand high load.

It's yet another example of Python making life easier for your developers.

Team composition in Node.js and Python

Your team composition is everything—the number one factor to consider when deciding on the programming language for your software product.

Python works better for some projects and Node.js for others. Your choice should depend entirely on whether you have good Python or JavaScript developers on your team.

Granted, this argument is invalid if you happen to have full-stack developers with both languages; however, those are hard to come by, so you usually have to keep this in mind.

Takeaways: Should you use Node.js or Python?

All things considered, the scale is tipped in Python's favor in one regard: it is much friendlier for junior or inexperienced developers. Furthermore, you generally shouldn't choose Node.js if you don't have an advanced team on hand.

But the real difference lies in your development team, not the language. They are what decides your project's success or failure, so you should go with whichever option works better for them.

Click here to learn more about how Python compares to Node.js.

Python vs. Java

Interpreted and dynamically typed vs. compiled and statically typed

Python is an interpreted and dynamically typed language, whereas Java is a compiled and statically typed language.

Python code doesn't need to be compiled before being run. Java code, on the other hand, needs to be compiled from code readable by humans to code readable by the machine.

Simply put, this generally means that Python has faster launch time and slower run time, while Java has slower launch time and faster run time.

Entry point in Java and Python

For Python, the entry point is famously low, which is why it's perfect for newbies and junior developers. The language is extremely user-friendly.

Conversely, Java has a high entry point with a clear learning curve. Learning how to write in Java—not to mention mastering it—is a significant time investment.

In a nutshell, getting started on Python takes weeks, while getting started on Java takes months.

Stability of Python and Java

There is a preconception that Java is the enterprise solution for software development. Corporations consider Java to be a strong, robust language because of its large code volume. They believe it makes the language more stable and secure than, for instance, Python.

However, the notion isn't entirely correct. Python also has what it takes to handle software products for big businesses—fintech, in particular.

To call Python unstable would be unfair and false. So why the prejudice in Java's favor?

It's not as much code volume as it is enterprise-friendly library support. These libraries are the actual reason why Java really is a little more stable than Python for corporate purposes.

Speed of Java and Python

Building an MVP with Java can take months because of its high code complexity and volume. Consequently, projects written in Java often go on for years and demand more developers on the team.

Python doesn't have any of these problems, thanks to its lightning-fast development speed. You can build an MVP with Python in mere weeks, finish the whole project in a matter of months, and use only a handful of developers for the job.

Beating deadlines is Python's specialty. If time is your number one concern—especially if you're a startup—look no further.

Resources with Python and Java

Development in Java is a bigger investment all around; it requires more time and money. If you have a lot of those on your hands, you should be perfectly satisfied with Java.

Python is less expensive, which is why for most projects it's the preferred choice. Remember, just because something costs more doesn't automatically make it better.

Trending technologies for Java and Python

No programming language is better suited for trending technologies than Python.

Whether it's artificial intelligence or machine learning, Python's design and features give it an advantage over all other languages for these relatively new purposes.

The main reason why Python's been adopted as the go-to language for trending technologies is its extensive AI/ML library support.

Furthermore, there's every indication that this trend will continue in the future.

Takeaways: Should you use Java or Python?

Python is clear to read, easy to write, and simple to modify. So if it's development speed you care about the most, go with Python.

Java, on the other hand, is perfectly suited to handle really complicated tasks. Therefore, if you value software stability above anything else, you might be better off with Java.

This isn't everything we have to say about Python and Java. Read the full comparison here.

Python vs. Ruby

Simplicity vs. creative flair

Both Python and Ruby allow developers to reach similar results when building web apps. While web development is what Ruby is primarily used for, Python is capable of much more.

Other than their use cases, the two languages also differ in their philosophies and approaches to solving problems. The use of Ruby has been declining over the past decade, whereas Python's popularity has skyrocketed.

Similarities between Python and Ruby

Both languages are:

- server-side, high-level, and scripting;
- dynamically typed, cross-platform, and general-purpose;
- well-established and mature, used by tech giants from many different fields (for example, Airbnb or Twitter use Ruby, while Facebook or Spotify use Python).

Differences between Ruby and Python

1) Popularity

In short, Python enjoys much higher adoption rates among developers than Ruby.

GitHub's Octoverse has found that Ruby's popularity has been declining by the year. It went from ranking 5th in 2014 to being 10th four years later.

Python, on the other hand, has been growing exponentially. Stack Overflow has referred to it as the "fastest-growing major programming language."

2) Flexibility

Ruby's flexible syntax allows developers to come up with highly creative solutions. This has led some to describe the language as "magical." Conversely, Python focuses on clear and simple solutions.

3) Philosophy

The approach to solving problems is the greatest difference between Python and Ruby. While the former features simple, singular solutions, the latter usually offers more than one way to get something done.

Although this may be seen as an advantage of Ruby, it could in fact compromise readability and simplicity as well as make errors more difficult to debug.

Takeaways: Should you use Ruby or Python?

Unless the project you're working on requires you to use Ruby, going with Python is the smarter choice.

Anything you can do with Ruby, you can do with Python. However, the rule doesn't apply the other way around. There are plenty of areas—such as academia, science, machine learning, or data analysis—where Python has a clear advantage over Ruby.

Despite the fact that the popularity of Ruby has been declining, the language still has plenty to offer when it comes to web development. There have been voices that the language is going obsolete, though this doesn't seem to be the case for now.

However, given the plethora of Python's use cases, choosing between Python and Ruby is a no-brainer. Python's dynamic growth, application in many different industries, and ease of use clearly make it the better pick.

For a more detailed comparison of Ruby and Python, head over to our article.

Python vs. PHP

What is PHP used for?

PHP is a widely used open-source language that's become the default web server technology. Often called a "scripting language for the web," PHP powers about 80% of website servers.

Although the language is often used for "traditional" web projects that don't need plenty of calculations or latest features, it has been applied across the board, from user authentication through database support to real-time applications.

PHP is extremely easy for a beginner to learn, yet it offers a lot of advanced features that lead to top-notch results.

What is Python used for?

Even though Python is currently used predominantly for web development, this area was not intended to be its chief focus at the start.

The range of Python use cases is wide and keeps on growing. It can be applied successfully in machine learning, data analytics, statistics, science, academia, and the Internet of Things.

In comparison with other programming languages, Python is very easy to learn, clear to read, and simple to write in.

Advantages of Python over PHP

1) Versatility

From data analytics through machine learning models to powerful web apps, there is little that Python can't do. In terms of versatility, it beats PHP hands down.

2) Structure

Since there have been fewer releases of Python than PHP, it tends to be more organized, secure, and easier to maintain.

3) Popularity

Given its application in areas such as artificial intelligence, machine learning, and the Internet of Things, as well as a vast array of uses that remain beyond the scope of PHP, Python has enjoyed a huge popularity spike in recent years.

Even though PHP has traditionally been more popular than Python, it's been gradually losing traction of late.

Advantages of PHP over Python

1) Features

PHP comes with more out-of-the-box features than Python. The latter does, however, make use of plenty of libraries to make up for that slight inconvenience.

2) Ease of installation

PHP is easier to install on any platform than Python, unless you stick exclusively to Linux.

Takeaways: Should you use PHP or Python?

Even though both Python and PHP are mature, well-established languages that enjoy widespread use and support, they couldn't be more different from each other when it comes to their syntax and philosophy.

However, when faced with a choice between the two, the decision should always come down to the individual requirements of your project.

If you're after a website, a blog, or a simple web service, the end result will most likely be the same whether you go with Python, PHP, or any other leading web development technology. You shouldn't be able to notice any difference in terms of performance, speed, or user design.

However, if the scope of your project is more varied and includes, for instance, machine learning, data analytics, or the Internet of Things, you should pick Python.

All in all, you can't go wrong with Python. Since it's great for web development and has plenty of other uses, you won't need to worry about not being able to expand the scope of your project in the future.

Plus, with its popularity skyrocketing in the last few years, Python is bound to stick around for a long time—unlike PHP.

Eager to learn more? Here's our full-length comparison of Python and PHP.

Python vs. R

These days, data science is an integral part of a growing number of people's jobs. The increased availability of data, the importance of analytics-driven decisions, and powerful computing in business make data science a huge deal in the tech world.

When it comes to tools for data science work, Python and R are two of the most popular tech stack choices.

Both are flexible, open-source programming languages with new libraries and tools added to their catalogs frequently that are oriented toward data science and enjoy large support communities. That makes it a challenge to pick one out of the two for your data analytics.

However, Python takes a more general approach to data science, whereas R is primarily used for statistical analysis. Therefore, anyone interested in leveraging these languages for their data science project should know the key differences and benefits of using one language over the other.

Let's take a look at the main features and advantages of Python and R.

Advantages of R: 12,000 packages and exceptional reporting

R was developed by academics and statisticians. Because of this, the programming language offers one of the richest ecosystems for performing data analysis.

With R, you'll be able to find a library for almost any analysis you wish to perform by leveraging the 12,000 packages available in CRAN, an open-source repository. In fact, the wide variety of libraries is what makes R the top choice for statistical analysis, particularly for specialized analytical work.

Another factor that makes R a cutting-edge language is the output it generates. R features impressive tools that are efficient in communicating the results. Rstudio includes the library knitr that was written by Xie Yihui, which makes communicating your findings in the form of a presentation or document seamless and almost trivial.

Advantages of Python: great fit for machine learning and artificial intelligence

Python can do almost all the tasks as R—engineering, data wrangling, feature selection, web scraping, app development, and so much more. Basically, Python code is easier to maintain and stronger than R. Hence, Python is often used to deploy and implement machine learning at a large scale.

A few years back, Python didn't have a lot of data analysis and machine learning libraries, but the language is catching up fast. Currently, it features some truly cutting-edge APIs for machine learning and artificial intelligence. You can often do most data science jobs using five Python libraries: NumPy, SciPy, Seaborn, Pandas, and scikit-learn.

Python really stands out when it comes to making replicability and accessibility easier than R. In fact, Python is your best bet if you're looking to use the results of your analysis in an application or website.

Takeaways: Should you use R or Python?

Since R was developed by academics and scientists, it's designed to solve statistical problems and also works well for machine learning and data science. Hence, it's fair to say the language is the right tool for data science, thanks to its powerful communication libraries. Additionally, R also contains many packages that can be used to perform time series analysis, data mining, and panel data.

Conversely, if you're a beginner in the field of data science and wish to learn how the algorithm works and deploys the model, you should ideally start learning Python.

Python comes with influential libraries for math, statistics, and artificial intelligence, making it a key player in machine learning. The programming language also features fantastic libraries that help greatly with manipulating matrices or coding algorithms.

Python is fairly easy to work with if you're looking to build a model from scratch, allowing you to later switch to the functions from the machine learning libraries. Similarly, you can also use R and Python together when going into data analysis.

R is excellent when you're focusing on statistical methods. Still, Python is the better choice if you want to do more than statistics—deployment and reproducibility, for instance.

To put it briefly, R and Python are basically on par these days when applied to data science. You can even use a combination of both languages if you need. However, Python can be used in many more areas, such as web development, network programming, and software prototyping. Ultimately, this makes Python a much more versatile choice.

Looking to further explore the true differences between Python and R? Head over here.

Python vs. C++

Python and C++ are two different programming languages with entirely different features and behavior. However, they do have one thing in common—strong support for object-oriented programming.

Below, we take a look at the key differences between Python and C++.

What is Python?

Python is a high-level, object-oriented, and interpreted programming language. It features extensive library support, making implementing various programs and algorithms easier. Its design aims to help programmers write clear and logical code for various projects.

What is C++?

C++ is a high-level, general-purpose programming language that was created as an extension of C. The language has improved significantly over time—the modern C++ is generic, object-oriented, and equipped with functional features in addition to facilities for low-level memory manipulation.

Differences and similarities between Python and C++

1) Ease of learning with C++ and Python

Ease of learning is one of the major factors for beginners. A programming language with a high entry point will be a nightmare for programmers to learn. Python's syntax is similar to English, which makes it much easier to learn. C++, though, is based on object-oriented concepts that deal with memory allocation. Writing the wrong program in C++ can destroy the entire system.

2) Speed of Python and C++

Compared to Python, C++ is faster. Memory management is hard in Python, while in C++, memory can be allocated to the variables and further deallocated when the variables are no longer used in the code.

3) Memory management in C++ and Python

As previously stated, memory in C++ should be allocated to new variables and deallocated when a variable is no longer used. Not doing this accurately can result in a memory leak. Therefore, it's safe to say that C++ doesn't offer built-in garbage collection and dynamic memory management.

Meanwhile, Python features built-in garbage collection and dynamic memory management mechanisms. It automatically allocates and deallocates memory.

4) Compilation with Python and C++

Python demands an interpreter at the time of compilation—for that reason, it's called an interpreted programming language. However, C++ is a pre-compiled programming language, meaning it doesn't need an interpreter at the time of compilation.

5) Readability of C++ and Python

C++ has a complex syntax that is difficult to read and write. It demands the user to follow specific programming rules like using curly brackets and semicolons at the end of a statement. In contrast, Python doesn't feature such programming rules.

On the other hand, Python uses indentation rules, which are similar to the English language. This makes it easier for programmers to understand Python code.

6) Variable declaration in Python and C++

C++ is a statically typed programming language—variables should be declared by mentioning the type and name before using them. Conversely, Python is a dynamically typed programming language—no declaring variables before using them is necessary.

Takeaways: Should you use C++ or Python?

All sorts of comparisons between Python and C++ lead to several conclusions. Python is ideal for beginners—whether that's because of the programming language's easy-to-read code or simple syntax. Python is also a better option for backend web development, whereas C++ isn't too popular in terms of web development of any kind.

Furthermore, Python is a leading language for data analysis and machine learning. Though C++ can be used for machine learning purposes, it doesn't fare as well as Python.

If simplicity is at the top of your list of features you're after in a programming language, Python is the ideal option for you. It's much easier to use and has a great support system when it comes to AI and ML frameworks.

Read our full comparison of the two languages if you wish to learn more about C++ vs. Python.

Python vs. C#

Simply put, Python and C# are juggernauts when it comes to programming languages.

Both are incredibly popular and are some of the best tools you could choose for any of your software development needs.

Each language is a versatile tool in its own right, but that doesn't mean they're equals. Picking the one that is right for your project will depend on what suits your business and technical requirements the most.

Advantages of Python: open, flexible, malleable

What makes Python unique is its incredible versatility. It's an all-purpose, high-quality, open-source language that allows you to do a lot with very little.

The structure of Python is extremely simple and clear, making it the ideal choice for situations where your team is still learning the ropes or is working on a piece of software that's meant to be compatible with multiple platforms.

Advantages of C#: familiar territory

What C# lacks in terms of simplicity—after all, the language is structured more traditionally, with various instances of punctuation making it less readable than Python—it gains by being native to Microsoft.

C# is overall the optimal choice for you if you're certain you'll be building a product for a Microsoft-based platform, since a team that is already familiar with that framework can achieve much more specific things way faster.

When to use Python or C#?

Python will work great in most instances. With its incredibly large community constantly providing you with free plugins to choose from, as well as its general user-friendliness and smooth learning curve, you can accomplish very much without expending a lot of resources.

Still, if you know you're writing for Microsoft-based platforms and you have a team that's familiar with any C-related language, you can be far more efficient if you give C# a shot.

Takeaways: Should you use C# or Python?

To sum up, Python and C# are among the top languages available on the market nowadays—for good reason. Both are multipurpose and have tremendous community support.

While Python wins out just slightly thanks to the sheer volume of community support the language boasts, it's hard to find a better option than C# if you're developing an app solely for a Microsoft-based platform.

This is far from everything there is to say about Python and C#. Read our full comparison here.

Final Thoughts

Thank you for reading our comparisons of Python to other programming languages. We hope our 15+ years of experience writing software in Python have helped answer all the questions you may have had in the matter.

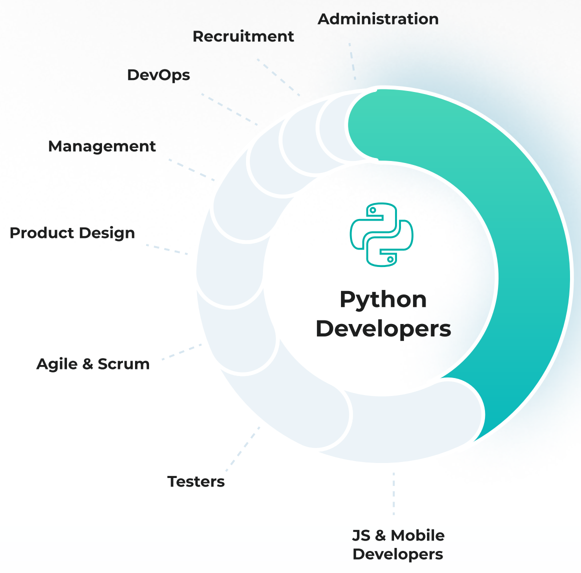
And if you do decide that Python is the right choice for your software project's tech stack, maybe we could also interest you in outsourcing your Python software development?

We'll be updating the "Python vs." page on our website several times in the near future, since there are many more technologies worth looking into with regard to how they compare to Python. Stay tuned.

Hire an exclusive Python development team

Accelerate your software project with Europe's largest Python software house.
For companies with big projects and fast deadlines.

Team Extension	End-to-End Development	Consulting & Expertise
Additional developers or experts supporting your development efforts within 14 days	Full development team taking your project all the way from discovery to deployment	Solving your problems or improving your product with the help of subject matter experts



Over 400 developers

Ready to empower any project with well-reviewed code and a results-driven Agile process

17+ years
market experience

750+
projects delivered

3.5+ years
average partnership

300+
clients served

550+
professionals on board

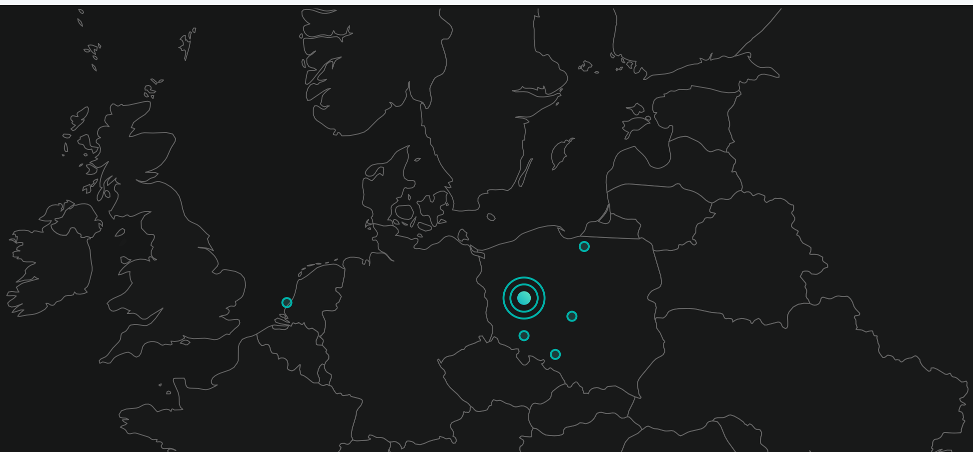
6.5+ years
average experience of our developers

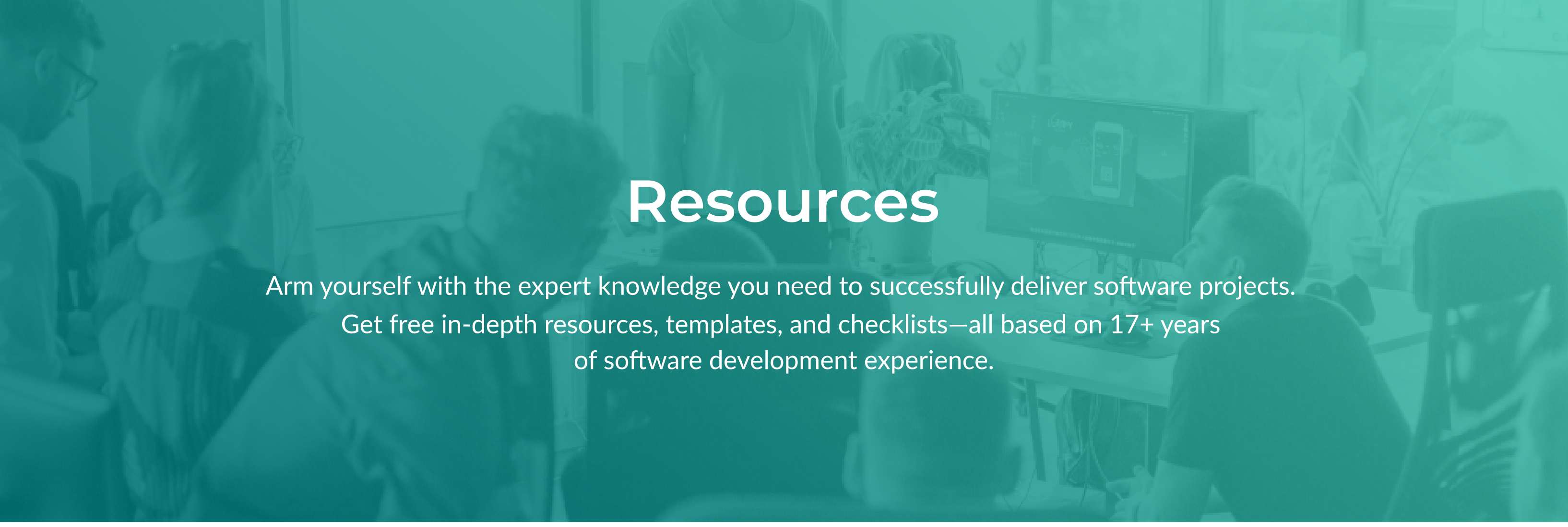
Locations

Poznań (HQ)

Mostowa 38
61-854 Poznań, Poland
+48 61 610 01 92

Wrocław
Olsztyn
Katowice
Łódź
Hague (Netherlands)





Resources

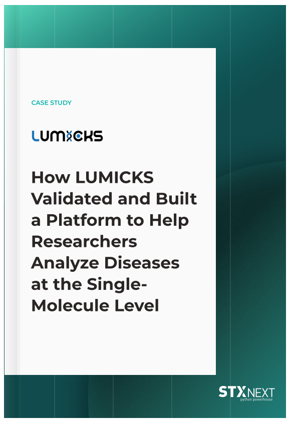
Arm yourself with the expert knowledge you need to successfully deliver software projects. Get free in-depth resources, templates, and checklists—all based on 17+ years of software development experience.

Reports and case studies

TheGlobalCTOSurvey × STXNEXT

SURVEY RESULTS & INSIGHTS

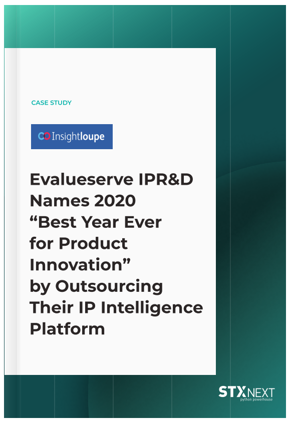
The Global CTO Survey 2021 Report


CASE STUDY

LUMICKS

How LUMICKS Validated and Built a Platform to Help Researchers Analyze Diseases at the Single-Molecule Level

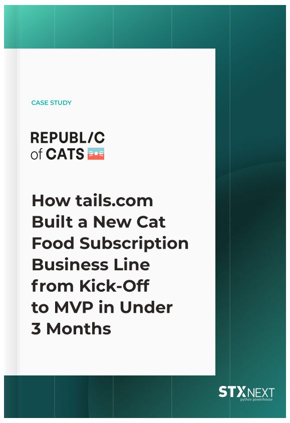
STXNEXT


CASE STUDY

Insightloupe

Evalueserve IPR&D Names 2020 "Best Year Ever for Product Innovation" by Outsourcing Their IP Intelligence Platform

STXNEXT


CASE STUDY

REPUBL/C of CATS

How tails.com Built a New Cat Food Subscription Business Line from Kick-Off to MVP in Under 3 Months

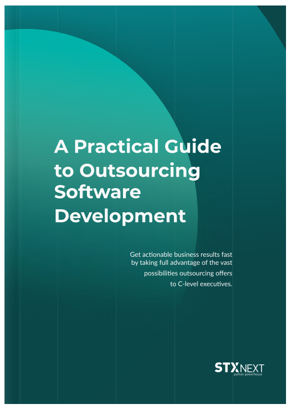
STXNEXT

Guides

What Is Python Used for?

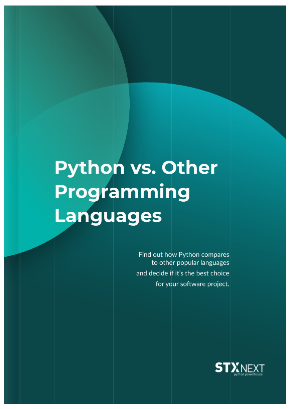
Discover why Python is such a great fit for so many industries and what areas of technology or business it benefits the most.

STXNEXT python powerhouse


A Practical Guide to Outsourcing Software Development

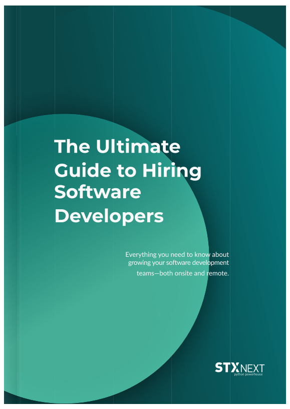
Get actionable business results fast by taking full advantage of the vast possibilities outsourcing offers to C-level executives.

STXNEXT python powerhouse


Python vs. Other Programming Languages

Find out how Python compares to other popular languages and decide if it's the best choice for your software project.

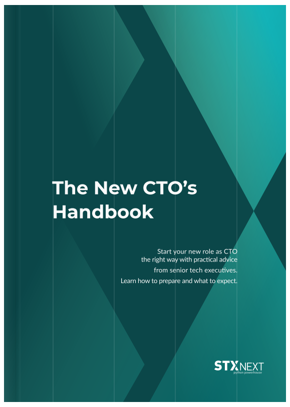
STXNEXT python powerhouse


The Ultimate Guide to Hiring Software Developers

Everything you need to know about growing your software development teams—both onsite and remote.

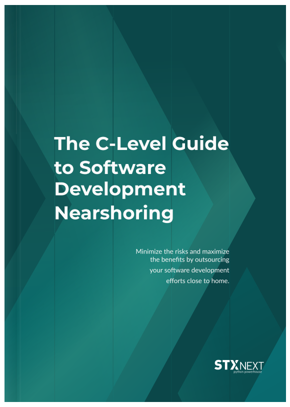
STXNEXT python powerhouse

Ebooks


The New CTO's Handbook

Start your new role as CTO the right way with practical advice from senior tech executives. Learn how to prepare and what to expect.

STXNEXT python powerhouse


The C-Level Guide to Software Development Nearshoring

Minimize the risks and maximize the benefits by outsourcing your software development efforts close to home.

STXNEXT python powerhouse

TECH LEADERS HUB by STX Next

Tech Leaders Hub: Management & Growth

Become a great leader by leveraging the huge experience of tech leadership experts who have spent years managing and growing teams.

STXNEXT python powerhouse

The True Cost of Hiring In-House Developers

From training through benefits to paid leave, the cost of adding new members to your team is never just the salary.

STXNEXT python powerhouse

DISCOVER MORE →

Services

Your project is all that matters. We'll build it like it was our own. Whether it's team extension, end-to-end product development, or expert consulting you're after, we'll do everything in our power to meet your needs.

- Python Development
- JavaScript Development
- .NET Development
- Web Development
- Software Testing & QA
- Mobile Development
- Django Development
- Node.js Development
- React Native Development
- Fintech Development
- Machine Learning
- Data Engineering
- DevOps
- Product Design
- Discovery Workshops

REVIEWED ON Clutch ★★★★★ 82 REVIEWS

python SOFTWARE FOUNDATION

Tell us about your project

Speed up work on your software projects and outpace the competition.

HIRE US →

Marta Błażejewska
DIRECTOR OF SALES
marta@stxnext.com
+48 506 154 343

Sebastian Resz
HEAD OF SALES
sebastian@stxnext.com
+48 690 433 578

Follow us