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.

STX NEXT
python powerhouse

Table of Contents

Introduction 1

Python for Web Development 2

Python for the Internet of Things 8

Python for Machine Learning 14

Python for Startups 20

Python for Fintech 24

Python for Data Science 29

Python for Finance 32

Python for Data Engineering 34

Python for Natural Language Processing 37

Final Thoughts 39

Resource List 41

Introduction

Google. YouTube. Instagram. Spotify. Reddit.

Aside from being some of the most popular software services in the world, what else do they have in common?

That's right: **they all use Python.**

Python is everywhere. You may not even realize how widespread it is.

The prominence of Guido van Rossum's creation can be attributed to a number of factors.

Most of all, Python is easy to learn, clear to read, and simple to write in. This speeds up development without sacrificing reliability or scalability.

Thanks to the high demand for Python, it's also very well supported in the community and keeps on growing in popularity.

But what exactly is Python used for? What areas of technology or business does Python benefit the most?

Read on if you're looking to break into any of the following fields and are considering whether to choose Python for your tech stack.

Python For Web Development

In the current market, a business without a website might as well not exist. Moreover, the trends are pushing for more and more impressive web apps that, among others, include:

- flawless mobile and desktop versions,
- asymmetrical layouts,
- Progressive Web Apps,
- integrated animations,
- ML-powered chatbots.

Nowadays, more than ever, it's important to choose the right tools when the time comes to build (or, quite likely, rebuild) your own website or web app.

Advantages of Python for web development

There are many advantages of Python that help you get results fast within the field of web development:

1) **Python has a large selection of pre-built libraries for just about anything.**

Scientific computing, image processing, data processing, machine learning, deep learning—you name it, Python has it.

2) **Python code takes less time to write due to its simple and clean syntax.**

Because of this, code written in Python lends itself very well to creating quick prototypes.

3) Python accelerates the ROI of commercial projects.

The reason behind this is similar to the previous point: you can write and ship your code faster. This is especially important for startups.

4) Python has a built-in framework for unit tests.

This helps you ship bug-free code.

In addition to Python's standard features, one of its major strengths in web development is the variety of web frameworks it offers.

With a large selection of well-supported frameworks, you can find the right starting point for any project. Python gives you the tools to get the job done reliably, no matter whether you're looking to focus on:

- **fast-to-implement and out-of-the-box solutions,**
- **solutions that require many specialized microservices working together,**
- **an app where performance is crucial.**

Top Python web frameworks

Click here to view the infographic in high resolution

1) Django

The most widely used Python web framework—at least until recently.

Django's trademark is its completeness, as it aims to offer all the tools you need to build a web app in a single package.

It's the perfect choice if your app is fairly standard, since it allows you to skip most of the initial steps and get a working solution faster.

2) Flask

Compared to Django, Flask is much more geared towards microservices, which may be the reason why it's the new #1 in popularity according to JetBrains.

Contrary to the all-in-one-package philosophy of Django, Flask works more like the glue that allows you to combine libraries with each other.

Flask lends itself well to an iterative approach of adding new features and services "one drop at a time."

3) Bottle

Bottle is another framework that would rather stay out of the way than overwhelm the user with every single thing they might need.

The framework is lightweight and has no external dependencies other than the Python standard library (stdlib).

It works great for prototyping, as a learning tool, or for building and running simple personal web apps.

4) Pyramid

The maturity of Pyramid stems from the legacy of two previous frameworks: Pylons and repoze.bfg. Now merged into Pyramid, Pylons used to be one of the top Python frameworks.

Pyramid's prime advantage over Django is that it's very easy to customize, whereas Django is more "opinionated." This makes Pyramid a great choice for non-standard projects that could be more complex.

We've barely scratched the framework surface here, so to speak. Head over to our article for more exotic examples of Python web frameworks.

Takeaways on Python for web development

Python's simplicity and wide selection of pre-existing libraries and frameworks make it a great fit for both garden-variety web projects and highly complex web apps.

Python For The Internet Of Things

The Internet of Things can be variously understood, depending on your perspective.

For the sake of this explanation, let's assume we're talking about physical objects fitted into an embedded system that connects them to the Internet.

These "things" now have their own IP addresses and can interact with other "things," remote or local, using the network.

IoT often plays a role in projects involving wireless sensor networks, data analytics, cyber-physical systems, big data, and machine learning. Additionally, IoT projects often involve real-time analytics and processes.

Ideally, your programming language for an IoT project should already be a strong choice for the aforementioned fields, while also being lightweight and scalable.

Python fits these criteria very well. Here's how.

Advantages of Python for IoT

1) Python's popularity is a considerable asset.

The language is supported by **a large, helpful community,** which has led to the creation of an extensive set of pre-written libraries, making it easier to implement and deploy working solutions.

2) **Python is portable, expandable, and embeddable.**

This makes Python not system-dependent and allows it to support many of the single-board computers currently available on the market, **regardless of the architecture or operating system.**

3) **Python works great for managing and organizing complex data.**

For IoT systems that are particularly data-heavy, this is especially useful.

4) **Python is easy to learn without forcing you to get acquainted with many formatting standards and compiling options.**

The most immediate consequence of this are faster results.

5) **Python code is easily readable and compact thanks to its clean syntax.**

This is helpful on small devices with limited memory and computational power. Additionally, the syntax is partly responsible for Python's growing popularity, further strengthening its community.

6) **Python's close relation to scientific computing has allowed it to gain ground in IoT development.**

If a social scientist or biologist wants to create a program for their smart device in the lab, they'll happily use their favorite language.

In the majority of cases, that will be Python, since it's the go-to technology for scientific computing.

7) **Python is the language of choice for the Raspberry Pi.**

It matters a great deal, since the Raspberry Pi is one of the most popular microcontrollers on the market.

8) **Python offers tools that streamline the IoT development process, such as webrepl.**

This gives you the option to use your browser to run Python code for IoT. Additionally, the mqtt messaging protocol lets you update your code/config.

9) **Since Python is an interpreted language, you can easily test your solution without compiling the code or flashing the device.**

With a C program, you would have to compile the code on your PC, then upload it to your "thing."

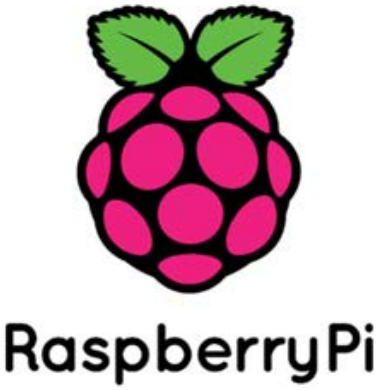
Python allows you to log into the interpreter directly on your "thing," making it easier to test various solutions.

10) **AWS offers a Python SDK for AWS IoT.**

Think of it as the cherry on top of an already delicious cake.

What tools are available for Python in IoT?

1) Raspberry Pi



Have you ever seen an interesting IoT project around the web?

If so, there's a good chance the Raspberry Pi was involved.

The Raspberry Pi:

- is small (85 mm × 56 mm for the Raspberry Pi 3);
- consumes very little energy;
- comes equipped with USB ports, an HDMI port, an Ethernet port, and Micro SD support.

Most importantly, it has a Linux distro on board, which means it also uses Python, making coding for the Raspberry Pi simple and straightforward.

The Raspberry Pi is an incredibly versatile device you can use to build anything:

- a media center,
- a retro gaming machine,
- a time-lapse camera,
- a robot controller,
- an FM radio station,
- a web server,
- a motion-capture security system,
- a Twitter bot,
- a mini-desktop PC.

It's also one of the most popular tools for teaching programming.

2) MicroPython

When it comes to Python solutions for IoT, it doesn't get much smaller than **MicroPython: a small microcontroller optimized to run Python on a board that's only a few square inches in size.**

The kit includes a software package, so if you're just starting out in IoT with Python, you don't need to look a lot further.

One feature of MicroPython that's especially enticing is WebREPL (read-evaluate-print loop), which is similar to a command line and accessible through a webpage. With WebREPL, you can use a simple terminal in your browser to run Python code on your IoT device without the need for a serial connection.

To make the deal even sweeter, you don't need to connect the board to WiFi, since it can create its own network.

3) Zerynth

Zerynth hails itself as “the middleware for IoT and Industry 4.0.”

It provides developers with a full ecosystem of tools, including an IDE, a toolchain for development, a multithreaded RTOS (real-time operating system), a device manager, and a convenient mobile app to monitor and control Zerynth-powered devices.

Zerynth speeds up IoT development by allowing you to write in Python or a hybrid of C and Python.

You can use Zerynth to program the most popular 32-bit microcontrollers, connect them to Cloud infrastructures, and keep your devices running the latest version of your software with Firmware Over-the-Air updates. It’s also very compact, requiring just 60–80 kB of Flash and 3–5 kB of RAM.

4) Home Assistant

Home Assistant is an open-source Python project for intelligent home automation. You can install it on a PC or a Raspberry Pi.

Home Assistant drives automation; for example, it can control the lights in your house and measure the temperature in each room.

On top of that, Home Assistant is compatible with a variety of drivers and sensors.

Desktop programming vs. IoT programming

To Python's great advantage, IoT programming is inching closer to desktop programming every day. As the capabilities of "smart things" continue to grow, their similarity to desktop computers is sure to follow.

We've already mentioned that a device like the Raspberry Pi could even serve as a miniaturized desktop PC. This trend works in favor of using Python for IoT, because with greater memory and computational power comes greater freedom of choice in terms of picking the right programming language.

Therefore, when developers and project managers are looking to choose a language that brings results fast and makes life easier, they tend to go with Python.

Takeaways on Python for IoT

Writing in Python is as quick, easy, and painless in IoT as in other fields.

In the current IoT environment, you choose your programming language just like you would choose it for any other project. Ease of writing matters more than the choice of language, and Python has that in droves.

Python For Machine Learning

Machine learning is the latest craze in the software development world. It's been steadily rising in popularity due to its seemingly limitless possibilities—and rightly so.

The very idea that computers can actively learn instead of operating in strict accordance with codified rules is simply exhilarating. It offers a whole new approach to problem solving.

At the forefront of machine learning is Python. Multiple studies unequivocally hail Python as the most popular language for machine learning and data science.

But why is that? What is Python's secret?

Advantages of Python for ML

There are several reasons why Python is a perfect fit for machine learning:

1) **Python's syntax is efficient and precise;**
2) **Python has a low entry point;**
3) **Python integrates well with other programming languages.**

But there is one more argument to be made for Python here, which in the case of machine learning is greater than all the others combined: **extensive open-source library support.**

Top Python libraries for machine learning

Python is famous for rich selection of libraries, especially for data science. It's the root cause why Python is considered the go-to solution for machine learning.

Here are the most popular Python libraries for machine learning.

1) scikit-learn

Scikit-learn is the best known and arguably most popular Python library for machine learning. Built on SciPy and NumPy—and designed to interoperate with them—scikit-learn is open source, accessible to all, and reusable in a number of contexts.

The library features a wide variety of algorithms for: classification, regression, clustering, dimensionality reduction, model selection, and preprocessing. These algorithms include: support-vector machines, random forests, gradient boosting, k-means, and DBSCAN.

Yet despite the barrage of options scikit-learn provides, the data mining and data analysis tools it offers are both simple and efficient.

2) TensorFlow

TensorFlow was originally developed by engineers and researchers at Google to meet their needs for a system that can build and train neural networks to find and decipher correlations and patterns. The process was designed as analogous to the ways humans reason and learn.

The flexible, high-performance architecture of the open-source library makes it easy to deploy numerical computation across multiple platforms, as well as from desktops to server clusters to mobile devices.

TensorFlow is used by companies such as Uber, Dropbox, eBay, Snapchat, or Coca Cola—not to mention Google, of course. This pretty much speaks for itself.

3) nilearn

Nilearn is a high-level Python library for easy and fast statistical learning on neuroimaging data. The library leverages scikit-learn for a plethora of advanced machine learning techniques, such as pattern recognition or multivariate statistics. The applications of this include predictive modelling and connectivity analysis, among others.

Constructing domain-specific feature engineering is the highest value nilearn holds for machine-learning experts. This means shaping neuroimaging data into a matrix of features perfect for statistical learning, or the other way around.

4) mlpy

Mlpy is a high-performance Python library for predictive modeling, built on top of SciPy, NumPy, and the GNU Scientific Libraries. Multiplatform and open source, mlpy aims to provide solutions for supervised and unsupervised problems, offering an extensive range of cutting-edge methods.

Finding a sensible compromise between efficiency, modularity, reproducibility, maintainability, and usability is the prime goal of mlpy.

What machine learning challenges can Python libraries help you solve?

1) Python for supervised learning

Supervised machine learning is one of the most widely-applied uses of AI. In supervised learning, an algorithm learns from a labeled dataset with the output already being known. Two of the main techniques within this category are classification and regression.

Classification is used to categorize data into desired, distinct classes and predict a discrete value. It can serve to assess creditworthiness or help with medical diagnostics.

Regression is used in problems that involve continuous numbers, including demand and financial forecasting, as well as property price estimation. The predicted outcome here is an estimation of a numeric value.

Both classification and regression problems can be solved thanks to a large number of Python libraries, including:

- scikit-learn (support-vector machines, linear and quadratic discriminant analysis, nearest neighbor algorithms, naïve Bayes classifiers, decision trees, ensemble methods, and more);
- TensorFlow;
- Keras;
- PyTorch;
- Caffe2 (deep learning);
- XGBoost;
- CatBoost;
- LightGBM (gradient boosting).

2) Python for unsupervised learning

In unsupervised machine learning, the algorithm relies on its own ability to solve problems after it's been handed an unlabeled dataset without training instructions and a known outcome.

Clustering and matrix factorization are two of the most common unsupervised machine learning methods. Frequently applied in user segmentation and recommender systems, both methods are used to group elements based on the similarity between object properties.

Some of the most popular libraries used in clustering and recommendation system engines are:

- Surprise (neighborhood-based methods, SVD, PMF, SVD++, NMF);
- LightFM (hybrid latent representation recommender with matrix factorization);
- Spotlight (uses PyTorch to build recommender models).

3) Python for reinforcement learning

Reinforcement learning algorithms learn to modify their behavior to make the right decisions after receiving feedback. They have been **tested in self-learning solutions,** including video games and traffic light control systems.

Problems posed by reinforcement learning are often highly specific and finding solutions to them may prove quite challenging. These Python libraries can help:

- Keras-RL (deep reinforcement learning for Keras);
- TensorForce (TensorFlow library for applied reinforcement learning);
- Coach (NAF, DQN, DFP, and more).

Takeaways on Python for ML

Python is the top choice for machine learning because its myriad of pre-prepared, tried-and-true libraries does most of the heavy lifting during the development process.

The tools are all there, simple and ready to use, complemented by extensive documentation and a vibrant community to go with it.

If you're looking to get into machine learning in Python yourself, we have a comprehensive tutorial covering all the bases, written by our very own machine learning experts.

And if you're not new to machine learning and wish to expand your Python skills in ML, maybe you'll be interested in a more advanced resource: an extensive overview of tree-based ensemble models in Python.

Python For Startups

Startups are very special cases. Building them from the ground up is a substantial undertaking, as they require a different approach than an established company. Making the process easier in any way you can is essential.

That's why when you choose the programming language for your project, you need one that will let you get started quickly and deliver the results you expect efficiently. At the same time, you can't compromise on high quality.

Python can give you all of that.

Why is Python the best programming language for your startup?

There are plenty of reasons why Python is a dream come true for any startup. Just to name a few, it's because the language is:

- **intuitive,**
- **reliable,**
- **scalable,**
- **ubiquitous,**
- **cutting edge.**

Notable startups with Python in their tech stack

We understand that theory is one thing and practice is another.

Here are just a few startups that have rightly bet on Python as their programming language of choice.

1) 21 Buttons

21 Buttons is the leading European fashion social network, with a user base of over 6 million bloggers and influencers. The startup provides a platform to share clothing designs and outfits with other like-minded fashion enthusiasts.

21 Buttons built their product using Python and a variety of Python frameworks, like Django or Flask.

2) Deuce Tennis

Deuce Tennis is a startup connecting tennis players with courts and clubs near them. Their goal is to grow the popularity of the sport and make playing tennis easy, enjoyable, and affordable for everyone.

By leveraging trending technologies such as React Native together with Python, the previously tedious process of booking and managing tennis sessions and courts becomes effortless with Deuce.

Deuce Tennis is available both as a mobile and web application. They have already made a splash on the worldwide tennis scene after they received backing from tennis legend Andy Murray.

3) TravelPerk

TravelPerk made headlines a while back as one of the hottest Barcelonian startups—and for good reason. They offer an all-in-one solution for business travel booking and management.

The startup hopes to change the way businesspeople travel for good. And the best part? It's totally free. The first and so far only one of its kind to be this bold.

TravelPerk is built in Python and React, putting code quality and user experience over anything else, with the backing of top tech investors like Spark Capital.

4) Zappi

Like the previous entries, Zappi also enables individuals to get in contact with others. However, they stand out from the startup crowd, since their focus is putting substitute teachers in contact with schools.

The startup's mission is to revolutionize the painstaking and stressful process of finding supply staff. Aware of the advantages Python offers, Zappi was adamant about using the language for their software product.

Their platform consists of a mobile solution for teachers, a web application for schools, and an administration platform for the Zappi hub that contains a centralized database of both the substitute teachers and the schools.

The Zappi app is available on Google Play and the App Store.

Takeaways on Python for startups

Startups are both exciting and terrifying endeavors.

They need to launch and grow fast, it takes a while before they turn a profit, and their financial foundations are often shaky.

There's a lot at stake and the risk of failure is high. That's why research and choice of programming language are crucial before you set out to launch a startup of your own.

Thankfully, the development speed, high code quality, and efficient scalability of Python make the language seem almost custom-made to help your startup overcome every challenge in its way.

Python For Fintech

While Python may not be a new technology, the rise in popularity it enjoys among the hedge fund and investment banking industries is a fairly recent development. But the fact that Python is the fastest-growing language in finance should come as no surprise.

If your company is about to venture into the world of fintech, you need a programming language that is high performing, easy to scale, and mature. The tech stack you choose should also have a wealth of ready-made solutions and libraries to fall back on.

This makes Python and fintech a match made in heaven.

Advantages of Python for fintech

Hedge fund and investment banking industries have long decided that Python is an ideal choice for fintech because the language addresses many of their highly specific needs:

- **creating platforms for risk and trade management;**
- **quantitative rate problem solving;**
- **simplifying data regulation, compliance, and analytics by leveraging the abundance of Python libraries.**

Why should you choose Python for your fintech software product?

Fintech belongs with Python for a number of reasons:

1) Clean syntax

Python code is easy to understand, because it resembles actual English. This allows developers to learn it quickly and to become fairly competent in it within a short time.

2) Fast time to market

Python is a dynamically typed language, making development in it quicker than in statically typed languages such as Java.

When writing in Python, you need less code, which in turn allows for faster deployment.

3) Helpful libraries

Python boasts an extensive array of libraries for a plethora of purposes; a lot of those are excellent for fintech and finance.

Do you need an algorithmic trading library? Try **pyalgotrade.** Something for scientific and technical computing? There is **SciPy.** How about quantitative economics? Check out **quantecon.py.**

Whatever your question, Python has the answer.

Top fintechs with Python in their tech stack

Many fintech companies have chosen to use the many benefits Python offers to their advantage.

Here are some examples.

1) Venmo

Venmo started out as a fintech startup. These days, it's more of a social media network. Either way, Venmo is an intuitive tool for people to split bills or pay one another.

Like PayPal, Venmo is an easy way to move money around. Linking a credit card, debit card, or checking account is required to receive or send payments.

Originally owned by Braintree, Venmo was acquired by PayPal in 2013 for $800 million. In Q2 of 2017 alone, Venmo processed a staggering $8 billion in payments.

Backend software developers at Venmo are expected to have experience working in Python and Django.

2) Newable Business Finance

Newable Business Finance is a platform that can be used to apply for business loans. They offer financial services to clients unwilling or unable to receive loans from traditional banking institutions.

The platform does away with tedious reliance on paperwork and bureaucracy using printed documentation, replacing it with a seamless and efficient online alternative. Business owners get their money quickly and reliably.

Newable Business Finance was built under tremendous time constraints. Within 2 months, the development team delivered the complete product, having started from nothing. Since day one, the platform has been running smoothly and pitfall-free.

How do we know that? We had a hand in building the platform.

3) Zopa

Zopa was founded on a bold principle: cut out the middleman and let borrowers and lenders deal directly with each other. The clients' investments are distributed across multiple loans, meaning not one of them holds more than 1% of a given investment.

The experiment paid off. In 2018, Zopa surpassed £3 billion in lending, the first consumer peer-to-peer lending platform to do so. As regulators put increasing pressure on traditional peer-to-peer lending, Zopa set out to launch its digital challenger bank.

Zopa uses a tech stack that is combination of Python, Java, and C#. Relying on a variety of frameworks, from Django to Pandas, they consider Python to be the key weapon in their arsenal.

4) Vyze

Vyze is a leading fintech for manufacturers and retailers. They provide a comprehensive lending service, combining support, technology, and supply to elevate businesses to brand new financing heights.

Vyze offers extraordinary financing to clients of all shapes and sizes. Their solutions are smart, adaptable, and most of all simple in an effort to give you a more satisfying purchasing experience, wherever you shop.

STX Next supported Vyze in building their cutting-edge platform. Some of the solutions Vyze needed to build were nowhere to be found in other fintech products. This meant that the developers had to use every trick in Python's proverbial book to create the solutions from scratch.

Takeaways on Python for fintech

The languages and frameworks you choose for your fintech will to a large extent decide the longevity and ultimate fate of your product. They determine what you can build, who you can hire to build it with, and how long the market validation will take.

This is what you'll get if you go with Python for your fintech:

- **optimal time to market;**
- **fewer errors and less bug fixing;**
- **open-source tools and libraries offering pre-prepared solutions.**

We think the choice should be clear.

Python For Data Science

Since Python is the go-to programming language for domains such as artificial intelligence, machine learning and deep learning, it's no surprise that it's also a fundamental tool for any data scientist.

With data often described as the new oil, the success of any business that works with it depends on the ability to extract insights from large databases and make strategic decisions based on them.

Python allows organizations to analyze and visualize data in meaningful ways to identify patterns, track down relationships, and help solve complex business problems.

What makes Python the best choice for data science?

Ease of use

Since it was designed with ease of use in mind, Python is a great tool for every data scientist, from a beginner without tech background to a pro with years of professional experience. Its flexibility and high code readability are some of the reasons 87% of data scientists reported using it daily.

Library support

Python's popularity for data science is, in a large part, due to the power of its multiple libraries.

Data scientists typically manage complex problems that can be solved with four general steps: data collection and cleaning, data exploration, data modeling, and data visualization. Python provides libraries that help with each stage.

They also make the language compatible with high-performance algorithms that allow it to combine with technologies such as artificial intelligence, machine learning, and predictive analysis.

A lot of data science tasks, such as web scraping, can be done using dedicated Python libraries, for instance Scrapy, Beautiful Soup, or Requests.

On top of that, Python gives you access to many deep learning frameworks, including Caffe, TensorFlow, PyTorch, and Keras.

Versatility

The great thing about Python is that there are plenty of ways it can be applied without compromising on the quality of the final result and without the need to make use of any other languages.

It can be used to predict outcomes, automate tasks, streamline processes, and offer business intelligence insights. Besides, Python integrates well with most cloud and PaaS providers, and supports numerous file export and sharing options.

Flexibility

By now, you surely know that Python is incredibly flexible. In the context of data processing, it means it allows you to build data models, systematize data sets and create ML-powered algorithms with ease.

What are the best Python libraries for data science?

Even though it's feasible to use Python on its own to work with data, the job becomes much easier with some of its useful libraries. Here are just some of the most common use cases and popular libraries.

General purpose

- NumPy
- Pandas

Web scraping

- Scrapy
- Beautiful Soup

Distributed processing

- PySpark
- Dask

Visualization

- Matplotlib
- Bokeh
- Plotly
- Seaborn
- pydot

Machine learning

- scikit-learn
- XGBoost

Deep learning

- TensorFlow
- PyTorch
- Keras

Python For Finance

The accelerating pace of technological change is one of the most creative—and destructive—forces shaping the financial services industry.

The disruption caused by fast-moving, up-and-coming start-ups, has forced the traditionally conservative sector to innovate, or risk falling behind.

Coupled with rising customer expectations, security and data protection concerns, as well as evolving regulations, financial services companies need robust technologies to rely on.

With its flexibility, high performance and access to a large ecosystem of scientific libraries, Python is the most reached-for programming language to help companies in this sector not just adapt to the changes but also spearhead them.

Python is the language of financial services

Knowledge of Python is one of the most popular skills that banks require when they hire developers, which points to the sector's reliance on it.

Given its ease of use, it's not uncommon to see Python skills outside of software development roles in finance. It's a great skill to have by professionals such as asset managers, financial analysts, business analysts and risk managers to help them better understand the technology underpinning the operations. It's often more effective than Excel for collecting, presenting and analyzing data.

Python can be used to create solutions in virtually any area of financial services, including:

- portfolio construction and management,
- financial analysis,
- risk analytics,

- algorithmic trading,
- credit scoring,
- consumer data management,
- fraud detection,
- cybersecurity.

Why is Python the best choice for financial services companies?

1) Great match for big data

Financial operations, such as risk analytics, creditworthiness assessment, or fraud detection, involve processing huge amounts of data. Using Python, a leading language in machine learning and data science, financial service professionals can analyze and visualize data faster than ever to recognize patterns, spot opportunities, and reduce risk.

2) Rapid development

Due to tight competition and strict regulations, financial institutions need technologies that enable them to rapidly release new products or updates. Python offers access to a number of frameworks and scientific libraries, such as NumPy, Pandas, SciPy or Matplotlib. They allow organizations to build sophisticated tools without the need to create every solution from scratch.

3) Ease of use

Python's syntax is the closest to the mathematical syntax that is used to describe scientific problems or financial algorithms. It can integrate seamlessly into trading and other financial services thanks to its ability to handle complex applications and real-time analysis while maintaining its ease of use. This makes the prototyping and development of financial services solutions time- and cost-effective.

4) Safety

Financial institutions need tools that easily adapt to the evolving industry regulations. Since Python is one of the fastest-growing programming languages in the world, it can be used to build solutions that are flexible and scalable. It reduces the potential error rate and helps ensure that the software won't become obsolete.

Python For Data Engineering

Python doesn't just make it easier for data engineers to do their work; in many cases, it makes it possible in the first place.

As the go-to language for data scientists, Python is also a great alternative for specialized languages such as R for machine learning. Often branded the language of data, it's indispensable in data engineering.

What are some of the most common use cases of Python in data engineering? Read on to find out.

Python in the cloud

Python is one of the few programming languages supported by the serverless computing services of all three platforms (AWS Lambda Functions, GCP Cloud Functions, and Azure Functions).

Over the years, Python has managed to incentivize cloud platform providers to use it for implementing and controlling their services.

The largest cloud providers, including Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure, keep Python users in mind when designing solutions to a variety of problems.

Data ingestion with Python

Business data may originate from various sources, such as databases (both SQL and noSQL), flat files (e.g. CSVs), other files used by companies (such as spreadsheets), external systems, APIs, and web documents.

Thanks to how popular Python is, you can pick from a large number of libraries and modules, including those used to access data, such as SQLAlchemy for some SQL databases, Scrapy, Beautiful Soup, and Requests.

One of the libraries that we find particularly interesting is Pandas. It allows you to read data into "DataFrames" from a variety of different formats, such as CSVs, TSVs, JSON, XML, HTML, LaTeX, SQL, Microsoft, open spreadsheets, and several other binary formats.

Pandas is based on other scientific and calculationally optimized packages, and it offers a rich programming interface with a large number of functions that are needed to process and transform data.

To facilitate DataFrame operations, AWS Labs maintains the aws-data-wrangler library called "Pandas on AWS." It can be used as a Layer for Lambda Functions, which helps make the deployment of serverless functions much easier.

Parallel computing with PySpark

Apache Spark is an open-source engine used to process large amounts of data. It leverages the parallel computing principle in a very efficient and fault-tolerant way.

Even though it was originally implemented in Scala and natively supported the language, it now has a popular and commonly used interface in Python, named PySpark.

What makes developing ETL jobs really straightforward for adepts of Pandas is the fact that PySpark supports most of Spark's features, including Spark SQL, DataFrame, Streaming, MLlib (Machine Learning), and Spark Core.

PySpark is also supported by the largest cloud providers, namely Amazon Web Services (AWS), Google Cloud Platform (GCP), and Microsoft Azure.

As a result, PySpark is a powerful tool to help you transform and aggregate huge volumes of data. It also makes it ready for consumption by end users, such as business analysts, or by further components, for instance by involving machine learning.

Job scheduling with Apache Airflow

The fact that there are very popular Python-based tools for on-premise systems has prompted platform cloud providers to commercialize them as "managed" services that, as a result, are easier to set up and work with.

This applies to Amazon's Managed Workflows for Apache Airflow, among others. Launched in 2020, it facilitates the use of Airflow in certain AWS zones.

Written in Python, Apache Airflow is an open-source workflow management platform that allows you to programmatically author and schedule workflow processing sequences, and then monitor them via a built-in Airflow user interface.

The alternatives to Airflow include the Python-based workflow orchestrators Perfect and Dagster. The tools go a certain way toward addressing the problems in Airflow, and workflows in both of them can be managed with Python.

Takeaways on Python for DE

If you're a data engineer, Python's versatility can help you maintain high velocity across all the tasks you need to perform.

The language has become one of the most popular tools for performing ETL tasks thanks to its access to countless libraries and ease of use. In fact, many data engineers use Python instead of ETL tools, as it proves to be more flexible and effective for these tasks.

Finally, since most of the latest and relevant technologies can be implemented and controlled with Python, the language has a plethora of uses in data engineering.

Python For Natural Language Processing

These days, we're coexisting with machines on a daily basis. This is remarkable in its own right, but also pretty mind-boggling, considering how much we differ from them. The language barrier alone is quite steep between us, though we still managed to come up with a solution to that particular issue.

Natural Language Processing, or NLP, is the bridge that can help us close that gap. And as usual, Python comes in as our main tool to that end, offering a number of libraries that can help you make the most of NLP.

Rule-based NLP and statistics-based NLP

Natural language processing comes in two main types: rule-based and statistical. The names are pretty self-explanatory—rule-based NLP operates mostly on a set of established rules, while statistical NLP uses large amounts of data to make judgments.

One is more intuitive, while the other gives you more control. That is, if you have the right skills. However, both are just alternatives to helping you accomplish the same thing.

What are the main challenges of NLP?

First of all, humans and machines operate on completely different wavelengths. The latter is very orderly, while the former is comparatively chaotic and unpredictable—and that's just one gap we're still struggling to bridge.

Human communication isn't always easy to parse for machines, since we often speak using terms that aren't even remotely literal. Not to mention the fact that all our languages are constantly evolving!

What are NLP libraries?

Natural language processing libraries are essentially databases that make it much easier to get an NLP to work as needed. Python comes with a fair share of them, so we'd like to showcase some of the more impressive ones for you now.

Top Python NLP libraries

- Natural Language Toolkit (NLTK)
- spaCy
- TextBlob
- Gensim
- CoreNLP
- Pattern
- polyglot
- PyNLPl
- scikit-learn
- PyTorch

Takeaways on Python for NLP

When it comes to teaching machines how to understand human communication, NLP is the way forward—there's no doubt about it.

We're hoping that, with the help of Python, taking the plunge and reaching out for all the opportunities that come with the technology will be a bit less daunting for you.

Final Thoughts

The applications of Python are numerous.

The language is perfect for web development, the Internet of Things, machine learning, startups, and fintech—among many others.

We've discussed at length why Python is such a good fit for all of these purposes.

However, just to give you a quick recap:

1) **Python lets you optimize your development resources by writing code faster, thanks to its high readability and ease of use;**
2) **Python has a clear and simple syntax, allowing you to review your code effortlessly and efficiently;**
3) **Python gives you a multitude of ready-made, tried-and-tested frameworks and libraries to lean on, instead of building your entire product from scratch;**
4) **Python offers extensive support from a variety of tutorials and guides, as well as a robust and thriving community of enthusiasts;**
5) **Python is used by tech giants like Google, YouTube, or Reddit, so if they put their trust in the language, there's no reason why you shouldn't do the same.**

This page gives you an overview of what Python is most useful for, but it doesn't nearly exhaust the subject, so expect more updates to come in the future.

Meanwhile, maybe you're looking for answers to different questions. Like, say, whether to choose Python or a different programming language for your software project.

If that's the case, we have another resource that will be perfect for you: a detailed look on how Python compares to other programming languages.

We also have other free resources you may benefit from:

- a manual on hiring software engineers,
- an introduction to nearshoring,
- a detailed breakdown of how much it really costs to hire an in-house developer,
- a general primer on Python from the perspective of a tech manager.

Resource List

On Python for web development

- Top 9 Web Application Trends in 2019 – Oxidian.ch
- Why Python & Django Are Your Top Choice for Web Development – The Startup on Medium
- 20 Advantages of Doing Web Development with Python and Django – Worthwhile
- How to Use the Bottle Micro Framework to Develop Python Web Apps – DigitalOcean
- Bottle – Full Stack Python
- A Beginner's Introduction to Python Web Frameworks – STX Next

On Python for the Internet of Things

- The Top 6 Programming Languages for IoT Projects – TechBeacon
- The Six Best-Paid IoT Programming Languages – IoT World Today
- MicroPython Basics: ESP8266 WebREPL Access – Adafruit
- IoT with Python: Essential Packages – element14 Community
- "What Is the Use of Python in IoT?" by Ashish Jhanwar – Quora

On Python for machine learning

- The Hitchhiker's Guide to Machine Learning in Python – freeCodeCamp on Medium
- Machine Learning Is Fun! The World's Easiest Introduction to Machine Learning – Medium
- The Most Popular Language for Machine Learning Is… – IBM Community
- Most Popular Programming Languages for Machine Learning and Data Science – Fossbytes

- The Most Popular Python Scientific Libraries – STX Next
- Tutorial: Getting Started with Machine Learning in Python – STX Next
- Machine Learning from the Woods: Exploring Tree-Based Ensemble Models in Python – STX Next

On Python for startups

- Why Python Is the Best Programming Language for Startups and Why Our Developers Love It – Net Solutions
- "Python vs. Java for Backend Web Development, What Are the Pros and Cons?" by Sebastian Buczyński – Quora
- "Why Python Is Perfect for Startups" by Vadim Nareyko – LinkedIn
- The 7 Most In-Demand Programming Languages of 2018 – Coding Dojo
- Why Python Should Be the Programming Language for Your Startup – STX Next
- Zappi: UK Education Project Using Python Django and React Native – STX Next
- Deuce Tennis: UK Tennis App Built with Python Django and React Native – STX Next

On Python for fintech

- Redrawing the Lines: FinTech's Growing Influence on Financial Services – PwC
- The Incredible Growth of Python – Stack Overflow
- Emerging Languages Overshadowed by Incumbents Java, Python in Coding Interviews – HackerRank
- Why Python Should Be the Technology Choice for Your Fintech – STX Next
- Top 17 Fintech Companies That Include Python in Their Tech Stack – STX Next
- 8 Fintech Companies with Python in Their Tech Stack: The Insurtech Edition – STX Next
- Newable Business Finance: UK Fintech Project Using Python Flask and Angular – STX Next
- Vyze: US Fintech Project Using Python Django and Angular – STX Next

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

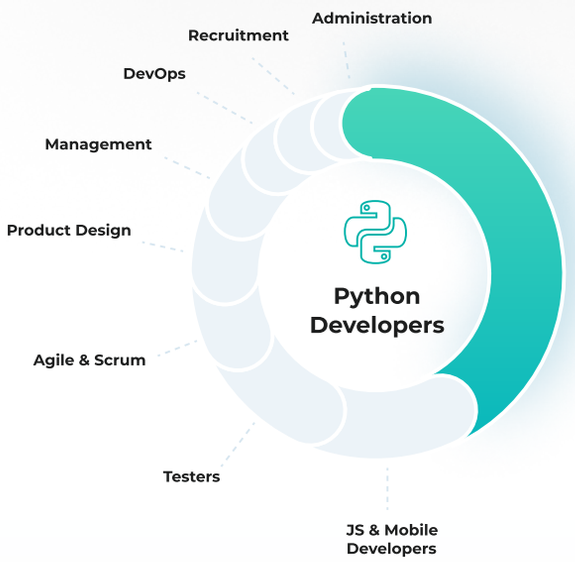
Additional developers or experts supporting your development efforts within 14 days

End-to-End Development

Full development team taking your project all the way from discovery to deployment

Consulting & Expertise

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

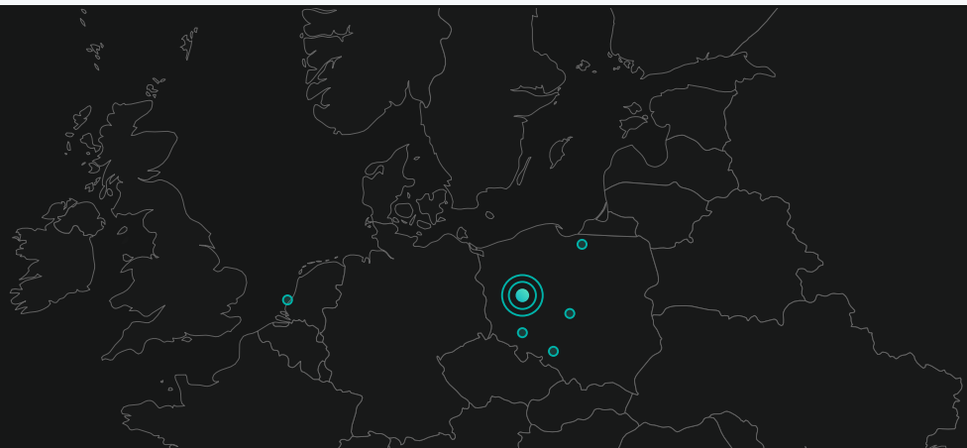
6.5+ years
average experience of our developers

Locations

Poznań (HQ)

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

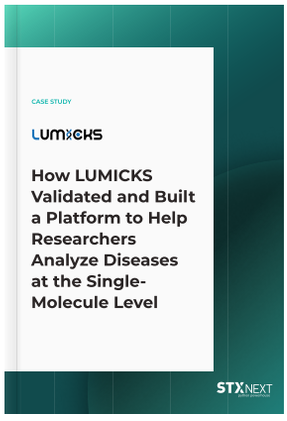
- Wrocław
- Olsztyn
- Katowice
- Łódź
- Hague (Netherlands)

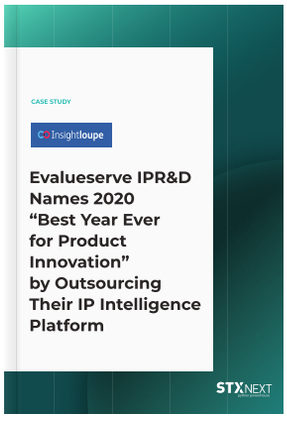


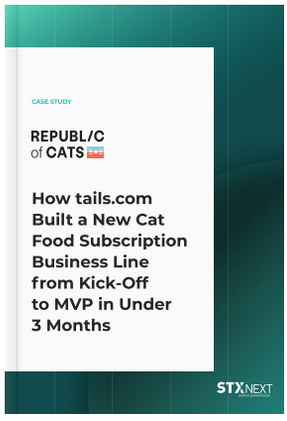
Resources

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.

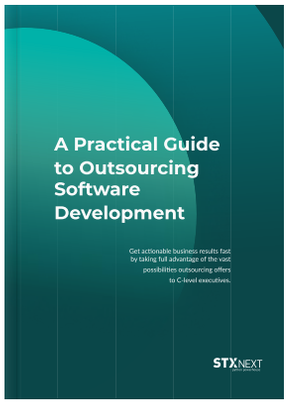
Reports and case studies

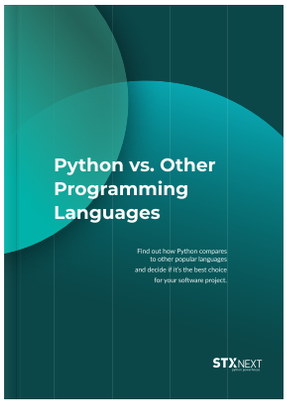




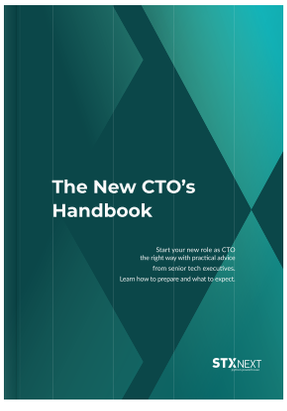


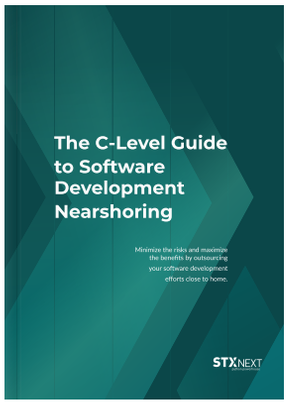
Guides





Ebooks





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