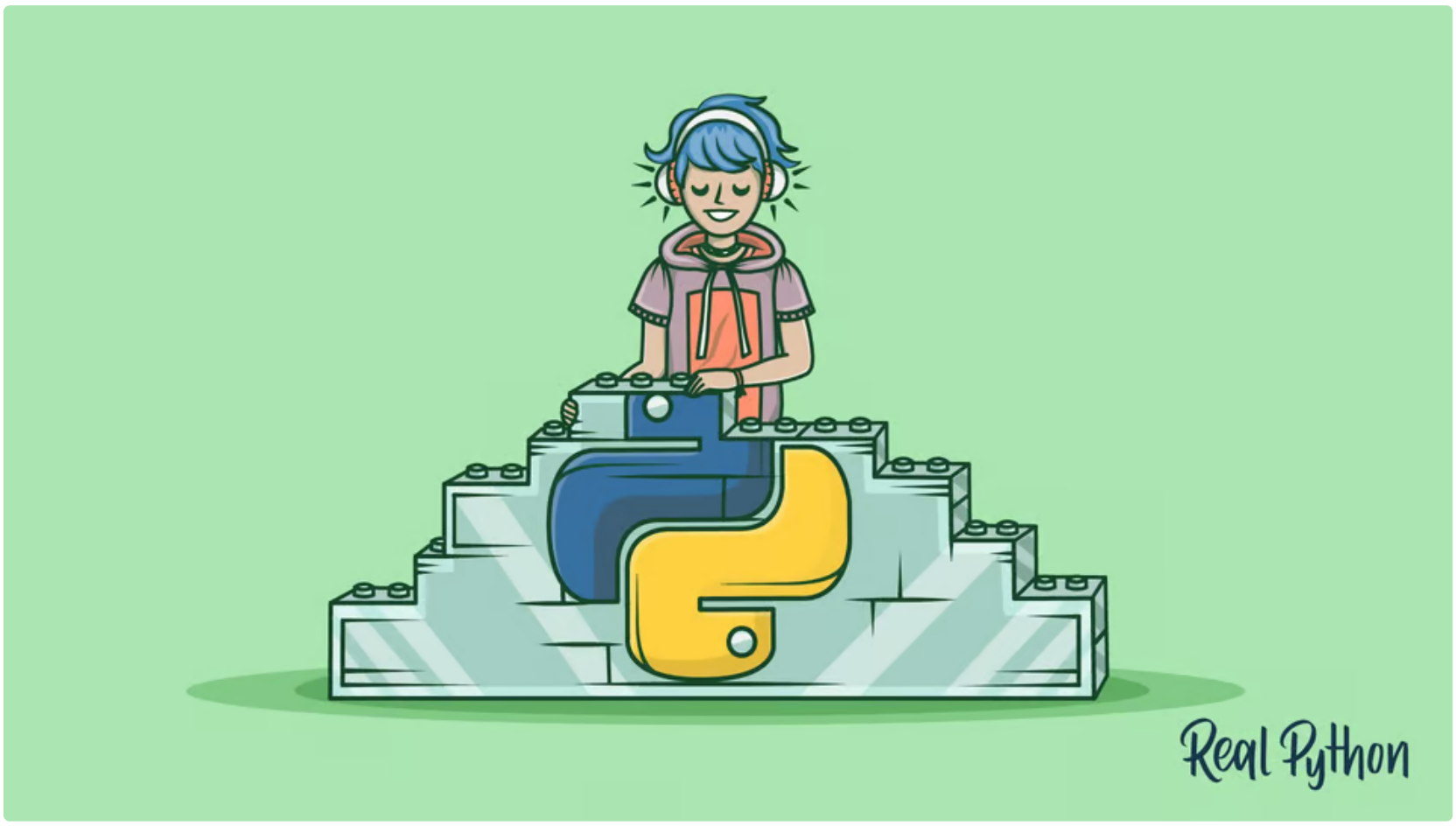


What Can I Do With Python?

by Leodanis Pozo Ramos 23 Comments

basics career

Mark as Completed

Tweet Share Email

Table of Contents

You've finished a course or finally made it to the end of a book that teaches you the basics of programming with Python. You've learned about variables, lists, tuples, dictionaries, `for` and `while` loops, conditional statements, object-oriented concepts, and more. So, what's next? What can you do with Python nowadays?

Python is a versatile programming language with many use cases in a variety of different fields. If you've grasped the basics of Python and are itching to build something with the language, then it's time to figure out what your next step should be.

In this article, you'll see how you can use Python for:

- Doing general **software development**
- Diving into **data science and math**
- Speeding up and automating your **workflow**
- Building **embedded systems** and **robots**

You'll also find ideas for practical projects, resources, and tutorials that you can use to start building things with Python right away.

Python in the Real World

Python is a high-level and general-purpose programming language. As this definition implies, you can use Python for several purposes, from web development to data science, machine learning, and robotics. Python's real-world use cases are limitless.

You're probably wondering what people are successfully building with Python. If you take a quick look at companies using the language, then you'll find world-class companies, such as Google, YouTube, Facebook, Instagram, Spotify, Netflix, and more.

Google has used Python from the start, and it's gained a place as one of the tech giant's main server-side languages. Guido van Rossum, Python's creator, worked there for several years, overseeing the language's development.

Instagram likes Python for its simplicity. The service is known for running "the world's largest deployment of the Django web framework, which is written entirely in Python."

Spotify uses the language for data analysis and back-end services. According to its team, Python's ease of use leads to a lightning-fast development pipeline. Spotify performs a ton of analysis to give recommendations to its users, so it needs a productive tool that works well. Python to the rescue!

You'll also find that Python has been vital for science and space exploration, with a lot of exciting use cases in robotics and hardware control.

In this article, you'll see how you can use your Python skills in a wide range of areas.

Develop Cool Software

Python's ecosystem provides a rich set of frameworks, tools, and libraries that allow you to write almost any kind of application. You can use Python to build applications for the Web as well as desktop and mobile platforms. You can even use Python to create video games.

Web Development

Developing web applications with Python is one of the most in-demand skills, with a lot of opportunities for you out there. In this field, you'll find several useful Python frameworks, libraries, and tools for developing cool web applications, APIs, and more. Here are some of the most popular Python web frameworks:

Framework	Description
Django	Django is a high-level framework that encourages rapid web application development with a clean and pragmatic design. It allows you to focus on writing your applications without having to reinvent the wheel.
FastAPI	FastAPI is a fast and performant web framework for building web APIs. It's built on top of modern Python type hint features and enables asynchronous programming.
Flask	Flask is a lightweight framework for creating WSGI web applications. It allows you to get started quickly and to scale up to complex applications if needed.
Tornado	Tornado is a web framework and asynchronous networking library. It uses non-blocking network I/O, so you can write applications that can scale to tens of thousands of open connections.

To get started with web development, check out:

- Python Web Development Tutorials
- Django Tutorials
- Flask Tutorials
- FastAPI Tutorial

If you want some practical project ideas for applying your web development skills right away, then you can build a portfolio web application with Django. With so many jobs and career opportunities out there, it's a great idea to have a personal portfolio these days, so go ahead and give it a try. You don't need to know anything about Django to get started with this step-by-step tutorial. It's perfect if you're itching to get your hands dirty with web development in Python.

CLI Development

Another field in which Python shines is command-line interface (CLI) application development. CLI applications are everywhere and allow you to automate repetitive and boring tasks in your day-to-day work by creating small and large tools for your command line.

In Python, you have an impressive set of CLI libraries and frameworks that can make your life more pleasant and help you build command-line tools quickly:

Library	Description
`argparse`	`argprse` is a standard library module that allows you to write user-friendly command-line interfaces. You can define the arguments you want to take at the command line and parse them nicely. It automatically generates help and usage messages and issues errors when your users provide invalid input.
Click	Click is a Python package for creating beautiful command-line interfaces with as little code as needed. It's highly configurable and comes with sensible defaults out of the box. Its goals include making the process of writing command-line tools quick and fun.
Typer	Typer is a library for building CLI applications that users will love using and developers will love creating. It provides automatic help messages and automatic completion for all shells. It minimizes code duplication and facilitates debugging.

To get started with CLI development, check out:

- How to Build Command Line Interfaces in Python With argparse
- Comparing Python Command-Line Parsing Libraries – Argparse, Docopt, and Click

Additionally, if you want to jump into building a CLI application project, then you can start by creating a directory tree generator tool for your command line. In this step-by-step project, you'll build a command-line tool for generating ASCII diagrams that display the contents of a directory or folder in your file system.

Creating applications with a user-friendly and intuitive command-line interface is a valuable skill for any Python developer.

GUI Development

Creating traditional graphical user interface (GUI) applications for desktop environments is also an attractive option in Python. If you're interested in building this kind of application, then Python has you covered with a wide range of GUI libraries, frameworks, and toolkits to choose from:

Library	Description
Kivy	Kivy is a library for rapid development of applications with innovative user interfaces, such as multi-touch applications. It runs on Linux, Windows, macOS, Android, iOS, and Raspberry Pi.
PyQt	PyQt is a set of Python bindings for the Qt application framework. It includes classes for building GUI applications. It also provides classes for networking, threads, SQL databases, and more. It supports the Windows, Linux, and macOS platforms.
PySimpleGUI	PySimpleGUI is a library that aims to transform the tkinter, Qt, wxPython, and Remi GUI frameworks into a simpler interface. It uses Python core data types to define windows and simplify event handling.
Qt for Python (`PySide6`)	Qt for Python is a project that provides the official set of Python bindings (`PySide6`) for the Qt framework.
tkinter	tkinter is a standard Python interface to the Tk GUI toolkit. It allows you to build GUI applications without the need for third-party dependencies. It's available on most Unix platforms as well as on Windows systems.
wxPython	wxPython is a Python binding for the wxWidgets C++ library. It allows you to create applications for Windows, macOS, and Linux with a single code base. It gives applications a native look and feel because it uses the platform's native API.

A quick way to start building your GUI applications is to use `tkinter`. This module comes in the Python standard library. Practice using `tkinter` and watch your vision materialize on the screen. Once you've got your feet wet, you can branch out and start working with other Python GUI toolkits.

To get started with GUI programming, check out:

- Python GUI Programming Tutorials
- Python GUI Programming Learning Path
- GUI Programming With PyQt Learning Path

Building back-end services is an essential part of development. However, you also need a front end. Creating applications that users can effectively interact with is paramount.

If you want to start creating real-world GUI applications, then you can build a calculator using PyQt. Completing this calculator project will help you grasp the fundamentals of this full-featured GUI framework, so you can start building nice things for your desktop immediately.

You can also find some other practical projects to help you out with your GUI programming journey. Take a look at the following resources:

- Build a Bulk File Rename Tool With Python and PyQt

- Build a Contact Book With Python, PyQt, and SQLite

These projects will guide you through the process of building GUI applications with PyQt and Python. They will also help you integrate a wide variety of skills to create fully functional real-world applications.

Game Development

Creating computer games is a great way to learn how to program not only in Python but also in any other language. To develop games, you'll need to use variables, loops, conditional statements, functions, object-oriented programming, and more. Game development is an excellent option to integrate multiple skills.

Computer games have played an important role in programming. Many people get into programming because they love games and want to re-create their favorite games or build their own. Developing computer games can be a fun and rewarding adventure, in which you can live the great experience of playing the game you just created.

You'll find several tools, libraries, and frameworks for creating games quickly in the Python ecosystem. Here's a small sample of them:

Library	Description
Arcade	Arcade is a Python library for creating 2D video games. It's ideal for people learning to program because they don't need to learn a complex game framework to start creating their own games.
PyGame	PyGame is a set of Python modules designed for writing video games. It adds functionality on top of the SDL library. It allows you to create full-featured games and multimedia programs. The library is highly portable and runs on several platforms and operating systems.
pyglet	pyglet is a powerful Python library for creating games and other visually rich applications on Windows, macOS, and Linux. It supports windowing, user interface event handling, OpenGL graphics, loading images, and playing videos and music.

To get started with game programming, check out:

- Python Game Development Tutorials
- PyGame: A Primer on Game Programming in Python

You can use Python to create arcade games, adventure games, and puzzle games that you can deploy within a few hours. You can also code classic games, such as hangman, tic-tac-toe, rock paper scissors, and more with your newly acquired programming skills.

If you want to dive into building your first game, then you can start by building an Asteroids game with Python and PyGame. If you want to go a step further and build your first platform game, then check out Build a Platform Game in Python With Arcade.

Dive Into Data Science and Math

Data science is a field that involves cleaning, preparing, and analyzing data to extract knowledge from it. Data science combines statistics, mathematics, programming, and problem-solving skills to extract useful information from data.

Python plays a fundamental role in the fields of data science and math. The language has become popular among scientists because of its readability, productivity, flexibility, and portability. The Python ecosystem around science has grown immensely. You'll find mature Python solutions in almost every major field in math and science.

Python includes tools for machine learning (ML), artificial intelligence (AI), scientific computing, data analysis, and data visualization. The language also provides efficient tools for collecting, mining, and manipulating data.

Machine Learning

Machine learning can be the first step for someone interested in artificial intelligence. Machine learning studies algorithms that learn through experience. These algorithms build models based on samples of training data to make predictions and decisions.

Machine learning can be an intimidating field to get started with because the space is fast and ever-changing. Here's a summary of some of the most popular tools for doing machine learning with Python:

Library	Description
Keras	Keras is an industrial-strength deep learning framework with an API designed for human beings. It allows you to run new experiments and try more ideas quickly. It follows best practices for reducing cognitive load.
NLTK	NLTK is a platform for building Python programs to work with human language data. It provides libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning.
PyTorch	PyTorch is an open source machine learning framework that accelerates the path from research prototyping to production deployment.
scikit-learn	scikit-learn is an open source machine learning library that supports supervised and unsupervised learning. It's an efficient tool for predictive data analysis that's accessible to everybody and reusable in various contexts.
TensorFlow	TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that will help you build and deploy ML-powered applications.

To get started with machine learning, check out:

- Python Machine Learning Tutorials
- Machine Learning With Python Learning Path

Scientific Computing

Another field in which Python plays a significant role is scientific computing. Scientists use advanced computing capabilities available through supercomputers, clusters of computers, and even desktop and laptop computers to understand and solve complex problems.

Here are some of the libraries and tools you can use for scientific computing in Python these days:

Library	Description
NumPy	NumPy is a fundamental package for scientific computing with Python. It offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more. It provides a high-level syntax that makes it accessible and productive.
SciPy	SciPy is a Python-based collection of open source software for mathematics, science, and engineering.
SimPy	SimPy is a process-based discrete-event simulation framework based on Python. It can help you simulate real-world systems, such as airports, customer services, highways, and more.

To get started with scientific computing, check out:

- Math for Data Science Learning Path
- NumPy, SciPy, and Pandas: Correlation With Python
- SimPy: Simulating Real-World Processes With Python

The libraries and tools in this section are fundamental pieces in the data science space in Python. Some of them are core components of higher-level libraries for machine learning, data analysis, and more.

Data Analysis and Visualization

Data analysis is a process of collecting, inspecting, cleansing, transforming, and modeling data to discover useful information, make predictions, arrive at conclusions, support decision-making processes, and more. Data analysis is closely related to data visualization, which deals with the graphical representation of data.

In Python, you'll also find mature and well-established libraries for data analysis and data visualization. Here are some of them:

Library	Description
Bokeh	Bokeh is an interactive data visualization library for web browsers. It provides tools for constructing elegant and versatile graphics. It can help you quickly make interactive plots, dashboards, and data applications.
Dash	Dash is a Python framework for building web analytic applications quickly. It's ideal for building data visualization applications with custom user interfaces that render in the browser.
Matplotlib	Matplotlib is a library for creating static, animated, and interactive data visualizations in Python.
pandas	pandas is a powerful and flexible open source tool for analyzing and manipulating data. It provides fast, flexible, and expressive data structures to work with relational or labeled data.
Seaborn	Seaborn is a Python data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics that allow you to explore and understand your data. It integrates closely with pandas data structures.

To get started with data analysis and visualization, check out:

- Data Collection & Storage Learning Path
- Data Visualization With Python Learning Path
- Data Science With Python Core Skills Learning Path
- Pandas for Data Science Learning Path Learning Path
- Develop Data Visualization Interfaces in Python With Dash

If you want to level up your data analysis skills by building a practical project, then you can create a gradebook with Python and pandas. This step-by-step project guides you through the process of creating a Python script that loads the grade data and calculates letter grades for a group of students. The project involves loading the data from a comma-separated values (CSV) file, exploring the data, and calculating and plotting the grades using pandas.

Web Scraping

One of the most significant sources of information for doing data science is the Web. The process of collecting and parsing raw data from the Web with an automated tool (crawler) is known as web scraping.

Python has a great set of tools and libraries for scraping data from the Web. Here are some of them:

Library	Description
Beautiful Soup	Beautiful Soup is a Python library for pulling data out of HTML and XML files into parse trees. The library provides methods and Pythonic idioms to navigate, search, modify, and extract information from parse trees.
`requests`	`requests` is an elegant and powerful HTTP library for Python. It provides an intuitive and concise API designed for human beings.
Scrapy	Scrapy is a fast, high-level web crawling and web scraping framework. It allows you to crawl websites and extract structured data from their pages.
`urllib.request`	`urllib.request` is a standard library module that defines functions and classes to help you open URLs. It also allows you to work with basic and digest authentication, redirections, cookies, and more.

To scrape data from the web, check out:

- Python Web Scraping Tutorials
- Python Web Scraping Learning Path

Once you know the basics of web scraping, you can dive into a practical project and build your own web scraper with Python and Beautiful Soup. After finishing this practical project, you'll be able to apply the same process and tools to any other static websites out there. These skills allow you to extract relevant information and use it in your applications. Go ahead and give it a try!

> **Note:** Before using your Python skills for web scraping, you should check the use policy of your target website to make sure that scraping it with automated tools isn't a violation of its terms of use.

A second project you can build right away is a Bitcoin price notification service. Since topping out at a price of just over $40,000 in January 2021, the cryptocurrency has been on the minds of millions. Its price continues to fluctuate, but many people out there would consider it a worthwhile investment.

If you're looking to cash in on the virtual gold rush and just need to know when to make your move, then you'll need to stay on top of Bitcoin's prices. The foundation of this project is the creation of IFTTT (If This Then That) applets. You'll learn how to use `requests` to send HTTP requests and how to use a webhook to connect your application to external services.

This Bitcoin price notification service is the perfect starter project for a beginner Pythonista with interest in crypto. Then you can extend the service you'll build in this tutorial to monitor other currencies as well.

Thanks to the Internet—and, increasingly, the Internet of Things—you now have access to hordes of data that weren't available years ago.

Analytics is a huge part of any field that works with data. What are people talking about? What patterns can you see in their behavior? Twitter is a great place to get answers to some of these questions. If you're interested in data analysis, then a Twitter sentiment analysis project is a great way to use your Python skills to answer questions about the world around you.

In this project, you'll learn how to mine Twitter data and analyze user sentiment with a Docker environment. You'll learn how to register an application with Twitter, which you'll need to do in order to access their streaming API. You'll see how to use Tweepy to filter which tweets you want to pull, TextBlob to calculate the sentiment of those tweets, Elasticsearch to analyze their content, and Kibana to visualize the results.

Speed Up and Automate Your Workflow

Computers are extremely good at performing repetitive and boring tasks. They can be doing the same thing for a long time without making mistakes. This is a valuable feature that can help you make your day-to-day work more pleasant and productive.

With Python, you can automate a lot of tasks in your workflow. You can automate and manage your DevOps operations, build an effective Python development environment, handle the packaging and deployment process in your development cycle, test your software, manage your database systems, and more.

DevOps

DevOps comprises software development and general IT operations. DevOps allows you to handle the entire life cycle of your applications and software products. It includes development, testing, packaging and deployment, and other related operations.

Python is one of the primary technologies people use for DevOps. Its flexibility and accessibility make Python an excellent fit for this job, enabling development teams to improve their workflow and to be more efficient and productive.

In the Python ecosystem, you'll find that some popular DevOps tools are written in Python. You'll also find that you can use Python to control most of those tools. Here are a few of them:

Library	Description
Ansible	Ansible is a tool for software provisioning, configuration management, and application deployment. It enables infrastructure as code.
Docker Compose	Docker Compose is a tool for defining and running multicontainer Docker applications. You can configure your application's services with a YAML file. Then, with a single command, you can create and start all the services from your configuration file. It works on production, staging, development, testing, and more.

To get started with DevOps, check out:

- Python DevOps Tutorials
- DevOps With Python Learning Path

With these resources, you'll build various skills and learn to use tools and technologies that any DevOps engineer working with Python should know.

Development Environment

Constructing a productive and effective environment for you and your teammates is a fundamental part of software development. To this end, Python has a great set of tools that allows you to isolate your packages, libraries, and Python version in per-project virtual environments.

Here are some of the most popular tools:

Tool	Description
`conda`	`conda` is an open source package and environment management system. It allows you to quickly install, run, and update packages and their dependencies. It helps you find and install packages.
`pip`	`pip` is a package management tool for Python. It allows you to install packages from PyPI and other indexes.

Tool	Description
Pipenv	Pipenv is a tool that aims to bring the best of all packaging worlds to the Python world. It allows you to create and manage virtual environments for your projects. It provides a way to use `pip` and `virtualenv` together through a unified interface.
pipx	pipx is a tool that helps you install and run end-user applications written in Python in isolated environments. It creates an isolated environment for each application and its associated packages. It makes the applications available in your command line or shell.
pyenv	pyenv is a tool for installing and managing multiple Python versions. It lets you switch between them quickly. It also allows you to define per-project Python versions.

To build an effective development environment, check out:

- Python Development Tools Tutorials
- Perfect Your Python Development Setup Learning Path
- An Effective Python Environment: Making Yourself at Home

Learning how to build an effective Python environment for your development adventure will push your productivity to the next level, so it's important to take the time to polish this skill.

Software Packaging and Deployment

Another critical part of your software development cycle is to package, distribute, and deploy your products to your end users or clients. In Python, a quick and popular way to deploy applications and libraries is to publish them to PyPI.

Here are some of the tools you can use for this purpose:

Tool	Description
Flit	Flit is a tool that provides a quick way to put your Python packages and modules on PyPI. It helps you set up the information about your package, so you can publish it to PyPI with minimal effort.
Poetry	Poetry is a tool for creating, building, installing, and packaging Python projects. It also allows you to publish your projects to PyPI. It tracks and resolves your project's dependencies. It uses your current virtual environments or creates new ones to isolate your packages from your system-wide Python installation.
PyInstaller	PyInstaller is a tool that freezes Python applications into stand-alone executables that work under Windows, GNU/Linux, macOS, and others.
setuptools	setuptools is a collection of enhancements to the Python distutils that allows you to build and distribute Python distributions, especially those that depend on other packages.
Twine	Twine is a utility for publishing Python packages on PyPI. It allows you to upload source and binary distributions of your projects.

To get started, check out:

- How to Publish an Open Source Python Package to PyPI
- Using PyInstaller to Easily Distribute Python Applications

With these resources, you can get started with packaging and deploying your Python applications, libraries, and packages to your end users, clients, and colleges. Also, the Python Packaging Authority provides a lot of useful information and tutorials to help you distribute Python packages with modern tools.

Database Systems

Most of the applications you'll build in your career as a developer will interact with data in some way. This interaction commonly happens through a database management system (DBMS) that allows you to define, create, maintain, and control access to your database or databases.

To connect and manipulate your databases with Python, you have several options that include both standard library packages and third-party packages and libraries. You also have options for SQL and NoSQL databases in Python.

Object-relational mapping tools (ORMs) are another important type of tool you'll probably use to work with databases in Python. These tools allow you to use object-oriented programming to create and manipulate your databases.

Here are some Python libraries you can use for connecting and operating databases:

Library	Database	Description
MongoEngine	MongoDB	MongoEngine is a document-object mapper for working with MongoDB using object-oriented programming in Python.
MySQL Connector/Python	MySQL	MySQL Connector is a self-contained Python driver for communicating with MySQL servers.
Psycopg	PostgreSQL	Psycopg is a PostgreSQL database adapter for the Python programming language.
PyMongo	MongoDB	PyMongo is a Python distribution containing tools for working with MongoDB databases. It provides a native Python driver for this type of database system.
SQLAlchemy	SQL	SQLAlchemy is a Python SQL toolkit and object-relational mapper for SQL databases.
`sqlite3`	SQLite	`sqlite3` is a lightweight disk-based database that doesn't require a separate server process. It allows you to access databases using a nonstandard variant of SQL. It's freely available and comes in the Python standard library.

To get started with databases, check out:

- Python Database Tutorials
- Data Collection & Storage Learning Path

Creating and working with databases is a powerful way to manage data in your Python applications. Databases add significant functionality and versatility to your programs and allow you to provide exciting features to your users and client. Managing databases is a fundamental skill in your developer education.

Software Testing

When you're beginning with Python or with programming, you probably start by creating small programs and scripts that you can run and test manually to make sure they work as you expect. However, when your programs grow and get more complex, testing them by hand is near to impossible. This is when automated testing comes into the scene.

Unfortunately, developers make mistakes, and no code is perfect. So, you'll need a testing process that helps you identify bugs and avoid getting them into production. Testing can also drive your code's design and help you check non-functional features, such as performance, security, usability, regulatory compliance, and more. Testing,

therefore, is an important component of software development.

Python has some of the best tools when it comes to testing. You can use these tools to write consistent tests and to run them automatically. Here's a small sample of these tools:

Tool	Description
doctest	doctest is a standard module that searches your docstrings for pieces of text that look like interactive Python sessions and executes them to verify that they work correctly.
pytest	pytest is a robust and mature testing framework that allows you to write and automate tests. It can scale from small unit tests to complex functional tests for your applications and libraries.
tox	tox is a generic virtualenv management and test command-line tool. It allows you to check if your packages install correctly within different Python versions and interpreters. It can run your tests in each of the configured environments.
`unittest`	`unittest` is a unit testing framework available in the Python standard library. It supports test automation, setup and teardown of tests, aggregation of tests into collections, and more.

To get started with testing, check out:

- Python Testing Tutorials
- Test Your Python Apps Learning Path

As a developer, you need to produce reliable code that works correctly. This means that you need to test your code every time you change it or add new features. Automated tests are the way to go in these situations.

Develop Embedded Systems and Robots

Writing your own applications for the Web or desktop is cool, but writing code that controls how hardware systems and robots work can be even cooler! Fields like Internet of Things, home automation, self-driving cars, and robotics have become more and more popular with advances in science and technology.

Python has gradually jumped into the world of sensors, electrical motors, circuits, microcontrollers, and robots. Today, you can find several Python projects that move in that direction. Here are some of them:

Library	Description
BBC micro:bit	BBC micro:bit is a pocket-sized computer that introduces you to how software and hardware work together. It is programmable with Python.
CircuitPython	CircuitPython is a programming language designed to simplify experimenting and learning to code on low-cost microcontroller boards.
MicroPython	MicroPython is a lean and efficient implementation of Python. It includes a small subset of the Python standard library. It's optimized to run on microcontrollers and in constrained environments.
PythonRobotics	PythonRobotics is a compilation of various robotics algorithms with visualizations. It's focused on autonomous navigation. Its goal is to allow you to understand the basic ideas behind each robotic algorithm it provides.
Raspberry Pi	Raspberry Pi is a general-purpose, Linux-based computer. It has a complete operating system with a GUI interface that is capable of running many different programs at the same time. Python comes built in on the Raspberry Pi.
rospy	rospy is a client library for ROS (Robot Operating System). Its API enables Python programmers to quickly interface with ROS to create complex and reliable robot behaviors.

To get started with embedded Python, check out:

- MicroPython: An Intro to Programming Hardware in Python
- Episode 5: Exploring CircuitPython
- Episode 161: Resources and Advice for Building CircuitPython Projects
- Embedded Python: Build a Game on the BBC micro:bit

If you want to start creating a hardware-related project with Python, then look at how to build physical projects with Python on the Raspberry Pi. In this project, you'll learn how to set up a Raspberry Pi, run Python code on it, read input from its sensors, send signals to its electronic components, and more.

What You Probably Shouldn't Do With Python

Python is a highly versatile language, and there's a lot you can do with it. However, you can't do everything. There are things that Python isn't very well suited for at all.

As an interpreted language, Python has trouble interacting with low-level devices, like device drivers. You'd have a problem if you wanted to write an operating system with Python. You're better off sticking with C or C++ for low-level applications.

However, even that might not be true for long. As a testament to Python's flexibility, there are people out there who are working on projects that extend Python's usability to low-level interactions. MicroPython and CircuitPython are just some of these projects designing low-level capability for Python.

What Else Can I Do With Python?

The list of ideas in this tutorial isn't exhaustive. There are countless other fields you can work on with Python. If you're looking for practical projects that Python is well suited for, then check out 13 Project Ideas for Intermediate Python Developers to find inspiration.

You can also do your own research to find projects that pique your interest. If you're not sure where to begin, then follow *Real Python* on Twitter. You'll find cool and interesting Python projects from the community there. Maybe you'll find something you can't wait to contribute to!

Conclusion

Having a basic understanding of what you can do with Python is key for you to keep leveling up your Python skills. You can use Python in a variety of different fields ranging from application development to robotics!

In this article, you saw that you can use Python for:

- General **software development**
- **Data science** and **math**
- **Workflow** speedup and automation
- **Embedded systems** and **robotics**

You also saw ideas for several practical projects that you can build to take your Python skills to the next level.

Next Steps

So there you have it! An extensive list of topics and practical projects to start working your way from Python beginner to savvy Pythonista.

No matter where you choose to begin, you'll be opening up countless avenues for growing your programming skills. Pick something and get started! Do you have an idea for a practical project that isn't here? Leave a comment down below! You could suggest the perfect project for a fellow programmer.

If you get stuck and need a nudge in the right direction, then check out 11 Beginner Tips for Learning Python Programming to help get yourself back on track.

Another great way to get unstuck is to talk it out. Coding doesn't have to be a solitary activity. If you need a way to ask questions and get answers quickly from knowledgeable Python developers, then consider joining The *Real Python* Member's Slack community. Everyone is welcome, no matter how much experience you have. You can always help others and also get help from others.

About **Leodanis Pozo Ramos**

Leodanis is an industrial engineer who loves Python and software development. He's a self-taugh Python developer with 6+ years of experience. He's an avid technical writer with a growing numb of articles published on Real Python and other sites.

» More about Leodanis

Each tutorial at Real Python is created by a team of developers so that it meets our high quality standards. The team members who worked on this tutorial are:

Adriana

Bartosz

Dan

Geir Arne

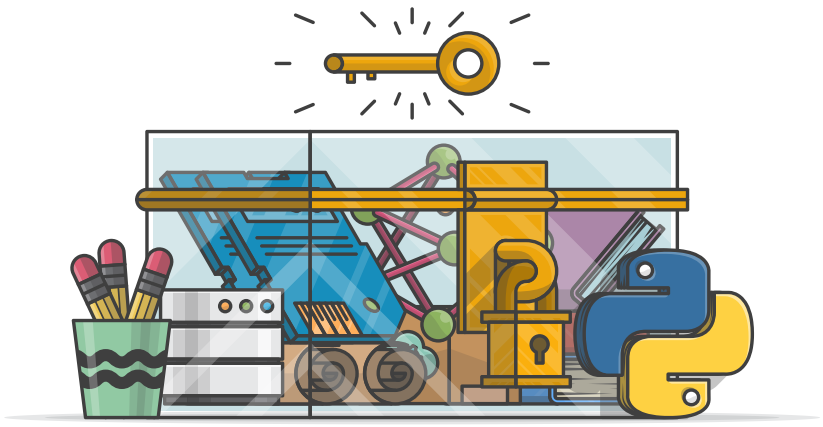
Jaya

Joanna

Jacob

Martin

Master Real-World Python Skills With Unlimited Access to Real Python



Join us and get access to thousands of tutorials, hands-on video courses, and a community of expert Pythonistas:

Level Up Your Python Skills »

What Do You Think?

Rate this article:

Tweet Share Share Email

What's your #1 takeaway or favorite thing you learned? How are you going to put your newfound skills to use? Leave a comment below and let us know.

Commenting Tips: The most useful comments are those written with the goal of learning from or helping out other students. Get tips for asking good questions and get answers to common questions in our support portal.

Looking for a real-time conversation? Visit the Real Python Community Chat or join the next "Office Hours" Live Q&A Session. Happy Pythoning!

Keep Learning

Related Tutorial Categories: basics career