Engineer

Python

the language most of this site's tooling is in

What it is

Python is a general-purpose programming language whose source is run by an interpreter rather than compiled ahead of time into a separate binary. Its own documentation calls it interpreted, interactive and object-oriented, with modules, exceptions, dynamic typing and high-level data types built in. The trait most people notice first is that indentation is syntax: a block is whatever is indented under the line that opens it, so the layout a reader sees is the structure the interpreter runs. Guido van Rossum began it over the 1989 Christmas holidays, drawing on his work on the ABC language at CWI in Amsterdam, posted it in February 1991, and named it after the scripts of Monty Python's Flying Circus he was reading at the time.

Typing is dynamic. A name is bound to an object while the program runs, and the type belongs to the object, not to the name, so the same variable can hold a number on one line and a string on the next. Optional annotations, standardised by PEP 484, let a function declare what it expects and returns, but that PEP is explicit that no type checking happens at runtime and that Python is going to stay a dynamically typed language. The annotations are read by separate tools, type checkers and editors, while the interpreter stores them without enforcing them.

The language ships with a large standard library, which the tutorial describes as batteries included: files, JSON, SQLite, regular expressions, subprocesses, argument parsing and more, all without installing anything. Everything else comes as third-party packages, usually installed with pip from the Python Package Index. Because one interpreter can only hold one version of each package, the standard library also includes `venv`, which creates a self-contained environment per project, so that one application can use version 1.0 of a library while another uses 2.0.

New versions arrive on a calendar. Since Python 3.9, PEP 602 has put a feature release out every October after about seventeen months of development, and each one is maintained for five years; from 3.13 on, that means two years of regular bugfix releases, roughly every other month, then three years of security fixes only. Python 3.15.0 came out on 9 October 2026. Python 3.10 reached its end of life on 1 October 2026, and Python 2.7, the last of the second series, stopped receiving support on 1 January 2020. The reference interpreter, CPython, has long had a global interpreter lock that lets only one thread execute Python code at a time; PEP 703 added a build without it, and PEP 779 named 3.14 as the time frame for the phase in which that free-threaded build is officially supported but still optional.

Use the concept in a decision

Put “What is Python?” to work in a real decision. We can help relate the concept to the system you are building or buying. The part users never see: where the data lives, what the rules are, and how other systems are allowed to talk to it.

Explore our services Get in touch →

Test the assumptions behind the term

Use the explanation and linked work to frame questions for your project. We can examine the assumptions with you before a tool is chosen. Designed and ran 6 ETL/ELT pipelines — 6 ETL pipelines across three database engines

Designed and ran 6 ETL/ELT pipelines Get in touch →

Why it matters

Python is often the shortest path from a question to a working program. A script that reads a file, queries a database and writes a report can be written with nothing but the standard library, run immediately, and changed without a build step. That is why it so often ends up as the glue in data pipelines, build tooling and test harnesses.

Readability is a deliberate design goal rather than a side effect. The Zen of Python, PEP 20, says that readability counts, and PEP 8, the style guide for the standard library, settles the questions a team would otherwise argue about, from naming to the 79-character line.

The flexibility has a cost, and it arrives late. A misspelt attribute, a `None` where a string was expected or a wrong argument type is an error only when that line runs, which may be in production. Teams that rely on Python for anything lasting therefore add the checks the language leaves out: a linter, annotations with a type checker in the gate, and tests that actually execute the code paths.

The calendar matters as much as the code. Every interpreter version has a fixed end date, so an installation nobody touches drifts out of security support on a published schedule. A project that names its minimum version and moves forward within the two bugfix years trades an emergency for one planned upgrade a year.

What it looks like in practice

In the portfolio, Python powered a database-driven CV and portfolio generator at one documented stage: 41 modules and 10,580 lines that rendered six output formats, from HTML and PDF to Markdown and JSON, out of one SQLite database of 23 tables. A pipeline of 19 steps built that database, each step safe to repeat. Runtime dependencies were held to exactly two, a template engine and a PDF renderer, on the argument that a document generator that cannot be installed in five years has preserved nothing. In return, every document then depended on that Python code, with no other way left to produce one. These counts describe the implementation at that documented stage.

Python also won a direct comparison in an hourly electricity consumption pipeline, which was built three ways, in Python, in JQ and in SQL on PostgreSQL, to turn irregular readings into one row per hour and zone, with net consumption as production plus imports minus exports. The version written as pure Python functions was the fastest, handling a day of data in 34 ms and a 30-day dataset in 180 ms.

This site is built by Hugo. In the tooling snapshot recorded in October 2026, its 22 helper scripts, 8,408 lines in all, were written in Python. Every one of them used the standard library alone except the script that imported drawings, which needed Pillow to trim them. The `.tool-versions` file named Python 3.14.7 with the policy that the interpreter could be newer but never older, and `task doctor` asserted that against the installed binary rather than trusting the file. Ruff was the linter, and basedpyright checked types at its `standard` level, because its default level reported 358 findings that were almost all about missing annotations in code that was plain on purpose. On its first run the type checker found a value that could be `None` being used as a dictionary key, a defect Ruff had passed clean.

Where it goes wrong

Most Python trouble comes from code that the interpreter accepts without complaint and that behaves differently from how it reads.

A default argument remembers earlier calls. A function returns a list that still holds items from previous calls, so results depend on call order. A default value is evaluated once, when the `def` statement runs, so a mutable default such as a list or dictionary is one shared object for every call that omits the argument. Use `None` as the default and create the list inside the body; Ruff's B006 rule reports the pattern.

```
def add_tag(tag, tags=None):
    if tags is None:
        tags = []  # a new list on every call
    tags.append(tag)
    return tags
```

Projects break each other's dependencies. Upgrading a library for one project breaks another, or `pip install` stops with an error about an externally managed environment. Both are packages installed straight into a shared interpreter: the projects compete for one copy of each library, and under PEP 668 an operating system distributor can mark its own Python as off limits to pip. Give each project its own environment with `python -m venv .venv`, install into that, and record the dependencies in a file so the environment can be rebuilt.

Type hints are trusted as if they were checks. A function annotated to take an `int` receives a string from a JSON file, and the failure surfaces three calls later. Annotations describe intent; the interpreter does not enforce them. Run a type checker as part of the gate so mismatches between annotated code fail the build, and validate data where it enters the program, since no annotation can vouch for input.

An interpreter ages out of support. A server keeps the version it was installed with until security releases for it stop and packages drop it from their supported range. Each feature version gets five years, and the five years of 3.10 ended on 1 October 2026. Name the minimum version in a file that a check reads, move up within the two bugfix years of each release, and run the test suite on the new version before switching.

Further reading

- Python documentation — the tutorial, the library reference and the language reference for the current release.
- The Python Tutorial — the language from first principles, including the warning about mutable default arguments.
- PEP 8 – Style Guide for Python Code — the layout and naming conventions most Python code follows.
- PEP 602 – Annual Release Cycle for Python — why a feature release lands every October and how long each is supported.
- PEP 484 – Type Hints — the annotation syntax and why the interpreter leaves checking to other tools.
- Virtual Environments and Packages — creating one environment per project with `venv` and installing into it with pip.

Glossary

Choose for your actual constraints

Describe the workflow, the existing interfaces and the users who need the change. We can define a testable delivery and a handover that lets your team maintain it.

See what it costs Get in touch →

Make the terminology practical

Ask how “What is Python?” applies to your organisation. A concrete example from your current system is enough to start the discussion.

Explore our services

welcome@engineer.company +45 50 10 61 84 Copenhagen, Denmark

Take this page with you

Contact card

Carefully Crafted With Love by D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Copyright © 2025 – present · Engineer ApS · All rights reserved.

Printed from https://engineer.company/glossary/python/