L2: Introduction to Python

Lesson Introduction

Python is currently the number-one language in the world. But what makes it so popular? In this lesson, you will find the answer to this question. First, you will take a quick look at the history of Python and the philosophy behind the language. Then, you will explore the main characteristics of Python, its pros and cons, and how it is different from other languages. In the final section, you will find out in which industries Python is commonly used.

Duration: **20 minutes**

The Founding of Python

Python originated within the walls of a research institute in Amsterdam called CWI.

Guido van Rossumjoined CWI in late 1982 as a programmer in an ABC group. ABC was conceived as a programming language and development environment for improving and eventually replacing BASIC. A few years later, Guido started working on another project—Amoeba, a kernel-based distributed system.

The idea for the Python language came to Guido during his time on the Amoeba project. Every application on this project was either a shell script or a C program. Guido could see the downsides of both. He was looking for a third language that would unite the strengths of both C and shell but that would be interpreted, easier to use, and more concise.

Work on Python was started **in late December 1989**, and the first version was released early in 1990. The new language was named Python in honor of Guido's favorite comedy show, Monty Python's Flying Circus.

On **February 20, 1991**, it was released as an open-source project. It immediately received a lot of feedback from the community, which soon began to grow. These people spread the word about Python and started contributing.

The community has inspired and motivated developers to constantly update and refine the language. To this day, Python remains a dynamic language that strives to address the developing needs of the industry.

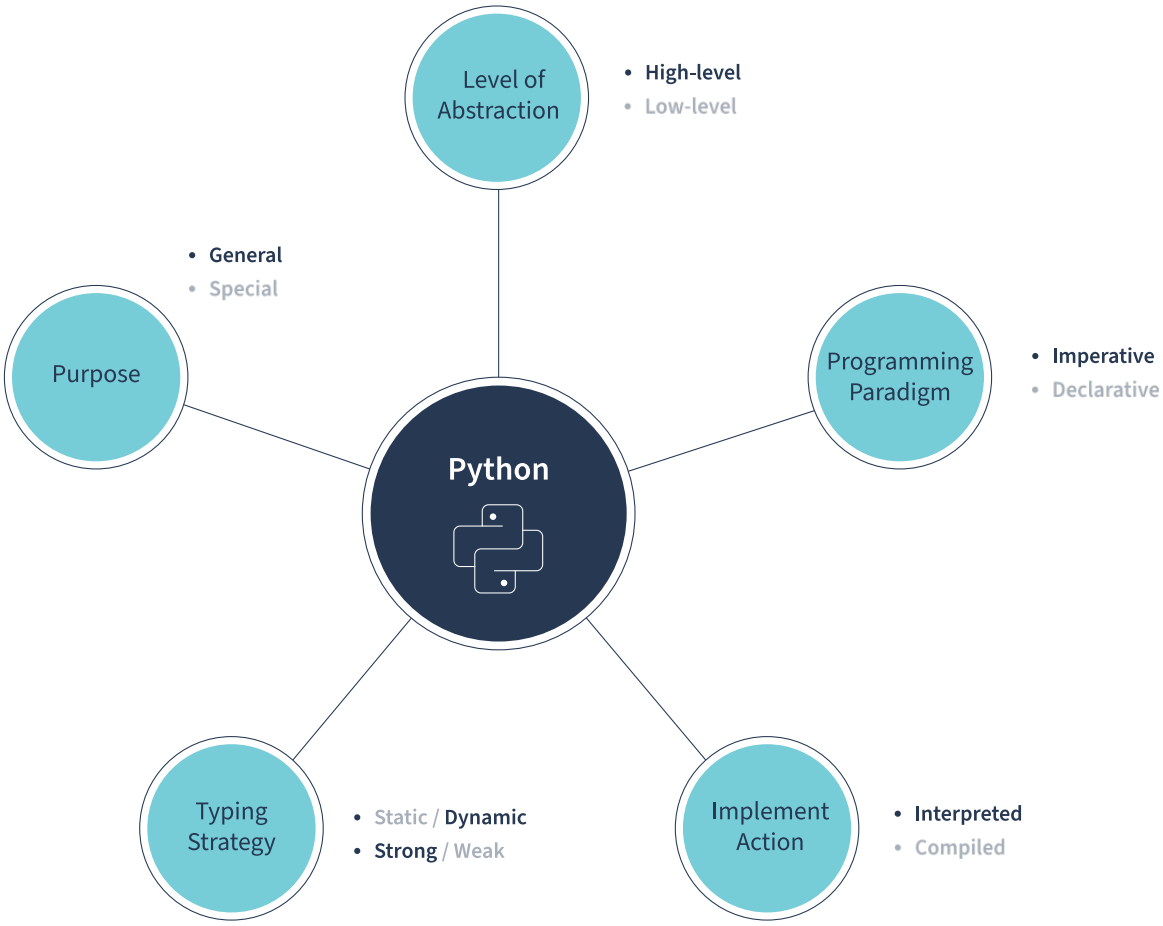
The following versions of Python have been released over the years:

Versions

Python 1.0 Jan 1994	Python 2.0 16 Oct 2000	Python 3.0 3 Dec 2008
Python 1.5 - 31 Dec 1997	Python 2.1 - 17 Apr 2001	Python 3.1 - 27 Jun 2009
Python 1.6 - 05 Sep 2000	Python 2.2 - 21 Dec 2001	Python 3.2 - 20 Feb 2011
	Python 2.3 - 29 Jul 2003	Python 3.3 - 29 Sep 2012
	Python 2.4 - 30 Nov 2004	Python 3.4 - 16 Mar 2014
	Python 2.5 - 19 Sep 2006	Python 3.5 - 13 Sep 2015
	Python 2.6 - 01 Oct 2008	Python 3.6 - 23 Dec 2016
	Python 2.7 - 03 Jul 2010	Python 3.7 - 27 Jul 2018
		Python 3.8 - 14 Oct 2019
		Python 3.9 - 05 Oct 2020
		Python 3.10 - 04 Oct 2021

What is Python?

Python is an easy-to-learn, powerful programming language that is perfect for scripting and rapid development. Below you can see how Python fits into the family of programming languages:



You're probably already familiar with these classifications, but it would be good to explore them in more detail.

General-purpose

Python is an industry-independent language, so it is used in many different domains, including:

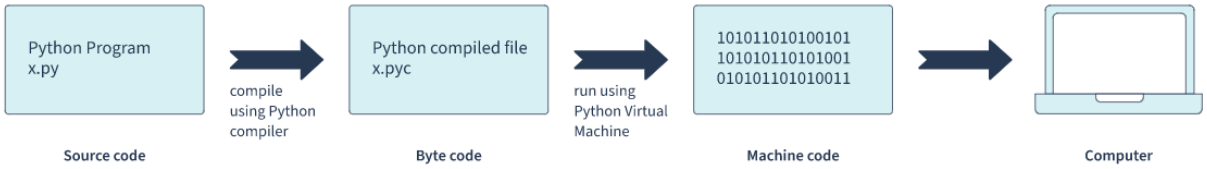
- Scripting
- Web development
- Data analysis
- Machine learning and AI
- DevOps and system administration
- Tests automation
- Prototyping

High-level

Python is a high-level programming language. It has a strong abstraction from the computer characteristics and resembles natural human language.

Interpreted

Even though Python is generally considered an interpreted language, this is not completely true. The source code of the program is first compiled into an intermediate format called bytecode. Then, this low-level instruction runs on a Python virtual machine (PVM). The PVM is software for converting bytecode line by line into machine code so that the computer can execute the instructions and display the final output.



Object-oriented

Python is a member of the family of an object-oriented languages (OOLs).An OOL is a high-level computer programming language that devises objects with their associated methods to create software programs. Therefore, Python supports OOP. It employs objects with clearly defined inter-connections and interactions. However, though it is an OOP, Python can also be used for functional programming.

Dynamic typing

Python performs type checking at runtime. That means developers don't have to declare a variable's type explicitly. It will be automatically recognized at runtime based on the value assigned to the variable. For example:

In the example below, **x** has an integer value and **y** – string.

```
>>> x = 1
>>> y = "Hello!"
```

The same variable can change its type as many times as necessary during program execution.

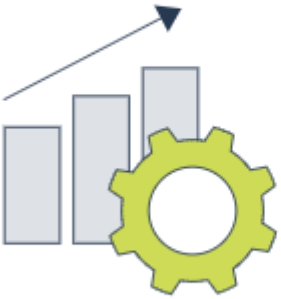
Just like with any language, you should consider Python's advantages and disadvantages before deciding to use it.

Pros

Python's benefits have made it the most widely used language in the industry. Here are just a few of them:

Easy to learn

Python is well known as one of the most beginner-friendly languages to learn. Due to its simplicity and human-language-like syntax, novice programmers can concentrate on fundamentals and good coding practices rather than language structure.

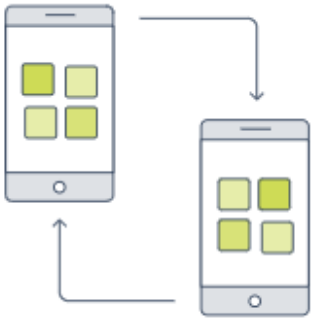


Fast development

Python's clear, concise syntax simplifies both learning and the development process: There are several standard libraries and extensions in PyPI (the library of Python packages). Additionally, Python has frameworks that provide pre-coded components(e.g., Flask for web development). Instead of creating everything from scratch, developers can take advantage of ready-to-use building blocks.

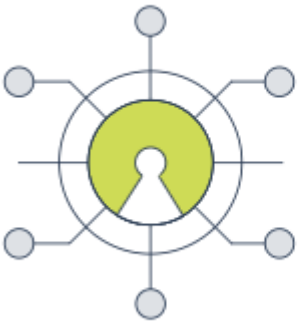
Dynamic typing

Python doesn't know the type of variable until we run the code. It automatically assigns the data type during execution. Programmers don't need to worry about declaring variables and their data types.



Portability and extensibility to other languages

Python is a platform-independent language, which means you can run the same code across different operating systems like Windows, Linux, or macOS. Portability is achieved thanks to bytecode and a PVM that serve as an intermediary between the programmer and the CPU executing the program. Python can also team up with other languages using extensions like `Cython` for C, `Gython` for Go, `Jython` for Java, and `IronPython` for .Net, which allow developers to mix languages and cover some places in the application.



Free and open source

Python is covered under an open-source license, making it free to use and distribute. You can download the source code, modify it, and even distribute your own version of Python. This is useful for organizations that want to adjust specific behaviors and develop using their own version.

Cons

Speed Limitations

Though Python was designed to make programming faster, it can't boast the same results in execution speed as other popular compiled languages such as C ++ or Java (see here for more information). A Python program is interpreted at runtime, line by line, instead of being compiled to the machine code in one piece. It brings advantages in terms of debugging, but this leads to performance loss during execution.

Also worth mentioning is that dynamic semantics contribute to the slowing down of program execution as well.

No multithreading

Python uses a mechanism called a global interpreter lock (GIL), which allows only one thread to be executed at a time. The GIL creates a bottleneck for multithreaded programs that run multiple workflows simultaneously. This prevents you from taking full advantage of modern multicore processors, which can execute several tasks concurrently.

High memory consumption

Python is often criticized for its high resource usage. Even though Python has a garbage collector to manage memory, it doesn't return resources to the system immediately. Moreover, if any of the objects in your code save any references on this outdated object, it can't be deleted. Due to these factors, Python programs tend to run out of memory easily.

Ineffective in mobile computing

Python is commonly used in server-side programming. However, it is hardly suitable for client-side computing in mobile applications. The main reason is Python's low memory efficiency and processing power compared to other languages.