Engineer

Python

lenguaje de la mayoría de las herramientas de este sitio

Qué es

Python es un lenguaje de programación de propósito general cuyo código fuente ejecuta un intérprete, en lugar de compilarse de antemano en un binario independiente. Su propia documentación lo describe como interpretado, interactivo y orientado a objetos, con módulos, excepciones, tipado dinámico y tipos de datos de alto nivel integrados. El rasgo que la mayoría de las personas nota primero es que la sangría forma parte de la sintaxis: un bloque es lo que aparece sangrado bajo la línea que lo abre, de modo que la disposición que ve quien lee es la estructura que ejecuta el intérprete. Guido van Rossum empezó a crearlo durante las vacaciones de Navidad de 1989, basándose en su trabajo con el lenguaje ABC en el CWI de Ámsterdam, lo publicó en febrero de 1991 y lo llamó así por los guiones de Monty Python's Flying Circus que leía entonces.

El tipado es dinámico. Un nombre se vincula a un objeto mientras se ejecuta el programa, y el tipo pertenece al objeto, no al nombre, de modo que una misma variable puede contener un número en una línea y una cadena en la siguiente. Las anotaciones opcionales, normalizadas por PEP 484, permiten que una función declare qué espera y qué devuelve, pero ese PEP especifica que no se comprueban tipos en tiempo de ejecución y que Python seguirá siendo un lenguaje de tipado dinámico. Las anotaciones las leen herramientas independientes, comprobadores de tipos y editores, mientras que el intérprete las almacena sin exigir que se cumplan.

El lenguaje incluye una biblioteca estándar amplia, que el tutorial describe como «pilas incluidas»: archivos, JSON, SQLite, expresiones regulares, subprocesos, análisis de argumentos y más, sin instalar nada. El resto llega en paquetes de terceros, normalmente instalados con pip desde el Python Package Index. Como un intérprete solo puede contener una versión de cada paquete, la biblioteca

estándar también incluye `venv`, que crea un entorno autónomo por proyecto para que una aplicación pueda usar la versión 1.0 de una biblioteca mientras otra utiliza la 2.0.

Las versiones nuevas llegan conforme a un calendario. Desde Python 3.9, PEP 602 establece una versión con nuevas funciones cada octubre, tras unos diecisiete meses de desarrollo, y cada una recibe mantenimiento durante cinco años; desde la 3.13, eso significa dos años de versiones regulares de corrección de errores, aproximadamente cada dos meses, y después tres años solo de correcciones de seguridad. Python 3.15.0 se publicó el 9 de octubre de 2026. Python 3.10 llegó al final de su vida útil el 1 de octubre de 2026, y Python 2.7, el último de la segunda serie, dejó de recibir soporte el 1 de enero de 2020. El intérprete de referencia, CPython, ha tenido durante mucho tiempo un bloqueo global del intérprete que solo permite a un hilo ejecutar código Python a la vez; PEP 703 añadió una variante del intérprete sin él, y PEP 779 señaló la 3.14 como marco temporal de la fase en la que esa variante sin bloqueo global recibe soporte oficial, aunque sigue siendo opcional.

Utilice el concepto en una decisión

Aplique «¿Qué es Python?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. La parte que el usuario no ve: dónde se guardan los datos, qué reglas se aplican y cómo pueden comunicarse otros sistemas con ella.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. Diseño y operación de 6 flujos ETL/ELT — 6 flujos ETL en tres motores de bases de datos

Diseño y operación de 6 flujos ETL/ELT Póngase en contacto →

Por qué importa

Python suele ser el camino más corto entre una pregunta y un programa que funciona. Un script que lee un archivo, consulta una base de datos y escribe un informe puede hacerse solo con la biblioteca estándar, ejecutarse de inmediato y modificarse sin una fase previa de construcción. Por eso termina tan a menudo uniendo las piezas de los flujos de procesamiento de datos, las herramientas de construcción y los sistemas de ejecución de pruebas.

La legibilidad es un objetivo de diseño deliberado, no un efecto secundario. El Zen de Python, PEP 20, dice que la legibilidad cuenta, y PEP 8, la guía de estilo de la biblioteca estándar, resuelve las cuestiones que un equipo discutiría de otro modo, desde los nombres hasta la línea de 79 caracteres.

La flexibilidad tiene un coste, y llega tarde. Un atributo mal escrito, un `None` donde se esperaba una cadena o un argumento de tipo incorrecto solo es un error cuando se ejecuta esa línea, lo que puede ocurrir en producción. Por eso, los equipos que confían en Python para algo duradero añaden las comprobaciones que el lenguaje omite: un linter, anotaciones con un comprobador de tipos en el control de calidad obligatorio y pruebas que ejecuten realmente las rutas del código.

El calendario importa tanto como el código. Cada versión del intérprete tiene una fecha final fija, de modo que una instalación que nadie toca pierde el soporte de seguridad conforme a un calendario publicado. Un proyecto que indica su versión mínima y avanza dentro de los dos años de corrección de errores cambia una emergencia por una actualización planificada al año.

Cómo se aplica en la práctica

En el portafolio, Python sustentaba un generador de CV y portafolios basado en una base de datos en una etapa documentada: 41 módulos y 10.580 líneas que generaban seis formatos de salida, desde HTML y PDF hasta Markdown y JSON, a partir de una única base de datos SQLite de 23 tablas. Un flujo de procesamiento de 19 pasos construía esa base de datos, y cada paso podía repetirse con seguridad. Las dependencias de ejecución se limitaron exactamente a dos, un motor de plantillas y un motor de renderizado de PDF, con el argumento de que un generador de documentos que no pueda instalarse dentro de cinco años no ha conservado nada. A cambio, cada documento dependía entonces de ese código Python, sin ninguna otra forma de producirlo. Estos recuentos describen la implementación en esa etapa documentada.

Python también ganó una comparación directa en un flujo de procesamiento del consumo eléctrico por horas, construido de tres formas, en Python, en JQ y en SQL sobre PostgreSQL, para convertir lecturas irregulares en una fila por hora y zona, con el consumo neto calculado como producción más importaciones menos exportaciones. La versión escrita como funciones de Python puro fue la más rápida: procesó un día de datos en 34 ms y un conjunto de datos de 30 días en 180 ms.

Este sitio se construye con Hugo. En la instantánea de sus herramientas registrada en octubre de 2026, sus 22 scripts auxiliares, con 8.408 líneas en total, estaban escritos en Python. Todos utilizaban únicamente la biblioteca estándar, salvo el script que importaba dibujos, que necesitaba Pillow para recortarlos. El

archivo `.tool-versions` indicaba Python 3.14.7, con la política de que el intérprete podía ser más reciente, pero nunca anterior, y `task doctor` lo comprobaba contra el binario instalado en lugar de confiar en el archivo. Ruff era el linter, y basedpyright comprobaba tipos en su nivel `standard`, porque su nivel predeterminado había señalado 358 problemas, casi todos sobre anotaciones ausentes en código deliberadamente sencillo. En su primera ejecución, el comprobador de tipos encontró un valor que podía ser `None` utilizado como clave de diccionario, un defecto que Ruff no había detectado.

Dónde surgen los problemas

La mayoría de los problemas con Python proviene de código que el intérprete acepta sin quejarse y que se comporta de forma distinta a lo que su lectura sugiere.

Un argumento predeterminado recuerda llamadas anteriores. Una función devuelve una lista que todavía contiene elementos de llamadas anteriores, de modo que los resultados dependen del orden de las llamadas. Un valor predeterminado se evalúa una vez, cuando se ejecuta la sentencia `def`, por lo que un valor mutable predeterminado, como una lista o un diccionario, es un único objeto compartido por todas las llamadas que omiten el argumento. Use `None` como valor predeterminado y cree la lista dentro del cuerpo; la regla B006 de Ruff detecta este patrón.

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

Los proyectos rompen las dependencias de otros. Actualizar una biblioteca para un proyecto rompe otro, o `pip install` se detiene con un error sobre un entorno gestionado externamente. En ambos casos son paquetes instalados directamente en un intérprete compartido: los proyectos compiten por una copia de cada biblioteca y, según PEP 668, un distribuidor de un sistema operativo puede marcar su propio Python como fuera del alcance de pip. Dé a cada proyecto su propio entorno con `python -m venv .venv`, instale los paquetes en él y registre las dependencias en un archivo para poder reconstruir el entorno.

Se confía en las anotaciones de tipo como si fueran comprobaciones. Una función anotada para recibir un `int` recibe una cadena de un archivo JSON, y el fallo aparece tres llamadas después. Las anotaciones describen la intención; el intérprete no las hace cumplir. Ejecute un comprobador de tipos como parte del control de calidad obligatorio para que las incompatibilidades entre código anotado hagan fallar la construcción, y valide los datos donde entran en el programa, porque ninguna anotación puede garantizar la entrada.

Un intérprete se queda sin soporte. Un servidor conserva la versión con la que se instaló hasta que dejan de publicarse actualizaciones de seguridad para ella y los paquetes la excluyen de las versiones que admiten. Cada versión con nuevas funciones recibe cinco años, y los cinco años de la 3.10 terminaron el 1 de octubre de 2026. Indique la versión mínima en un archivo que lea una comprobación, actualice dentro de los dos años de corrección de errores de cada versión y ejecute el conjunto de pruebas con la nueva versión antes de cambiar.

Lecturas adicionales

- Documentación de Python — El tutorial, la referencia de la biblioteca y la referencia del lenguaje de la versión actual.
- El tutorial de Python — El lenguaje desde sus fundamentos, incluida la advertencia sobre los argumentos predeterminados mutables.
- PEP 8: guía de estilo del código Python — Las convenciones de disposición y nombres que sigue la mayor parte del código Python.
- PEP 602: ciclo anual de versiones de Python — Por qué cada octubre llega una versión con nuevas funciones y cuánto dura su soporte.
- PEP 484: anotaciones de tipo — La sintaxis de las anotaciones y por qué el intérprete deja la comprobación a otras herramientas.
- Entornos virtuales y paquetes — Cómo crear un entorno por proyecto con `venv` e instalar en él con pip.

Glosario

Elija según sus restricciones reales

Describa el flujo de trabajo, las interfaces existentes y los usuarios que necesitan el cambio. Podemos definir una entrega comprobable y un traspaso que permita a su equipo mantener lo entregado.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es Python?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

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

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/python/