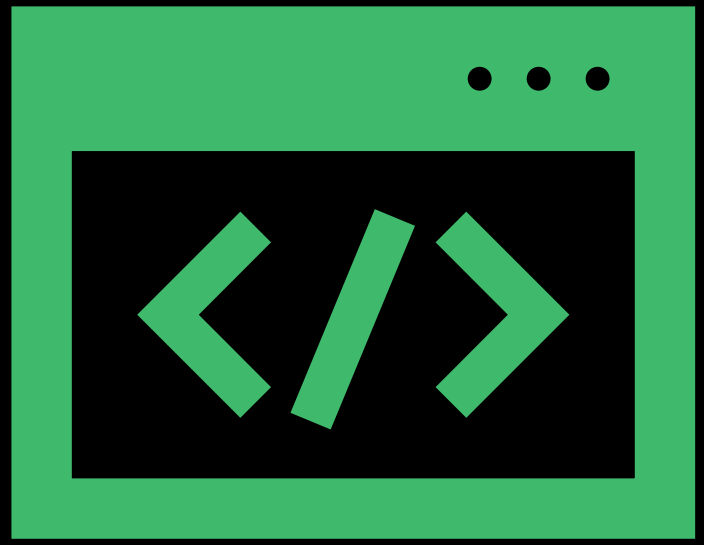
INTRODUCCION A PYTHON



¿Qué es Python?

- Python es un lenguaje de programación de alto nivel, interpretado, orientado a objetos y de uso generalizado con semántica dinámica, que se utiliza para la programación de propósito general.

¿Quién creó Python?

- Una de las características sorprendentes de Python es el hecho de que en realidad es el trabajo de una persona. Por lo general, los grandes lenguajes de programación son desarrollados y publicados por grandes compañías que emplean a muchos profesionales, y debido a las normas de derechos de autor, es muy difícil nombrar a cualquiera de las personas involucradas en el proyecto. Python es una excepción.

Guido van Rossum

NO EXISTEN MUCHOS LENGUAJES DE PROGRAMACIÓN CUYOS AUTORES SEAN CONOCIDOS POR SU NOMBRE. PYTHON FUE CREADO POR **GUIDO VAN ROSSUM**, NACIDO EN 1956 EN HAARLEM, PAÍSES BAJOS. POR SUPUESTO, GUIDO VAN ROSSUM NO DESARROLLÓ Y EVOLUCIONÓ TODOS LOS COMPONENTES DE PYTHON.

LA VELOCIDAD CON LA QUE PYTHON SE HA EXTENDIDO POR TODO EL MUNDO ES EL RESULTADO DEL TRABAJO CONTINUO DE MILES DE (MUY A MENUDO ANÓNIMOS) PROGRAMADORES, TESTERS, USUARIOS (MUCHOS DE ELLOS NO SON ESPECIALISTAS EN TI) Y ENTUSIASTAS, PERO HAY QUE DECIR QUE LA PRIMERA IDEA (LA SEMILLA DE LA QUE BROTÓ PYTHON) LLEGÓ A UNA CABEZA: LA DE GUIDO.

Los objetivos de Python

UN LENGUAJE **FÁCIL E INTUITIVO** TAN PODEROSO COMO LOS DE LOS PRINCIPALES COMPETIDORES.

DE **CÓDIGO ABIERTO**, PARA QUE CUALQUIERA PUEDA CONTRIBUIR A SU DESARROLLO.

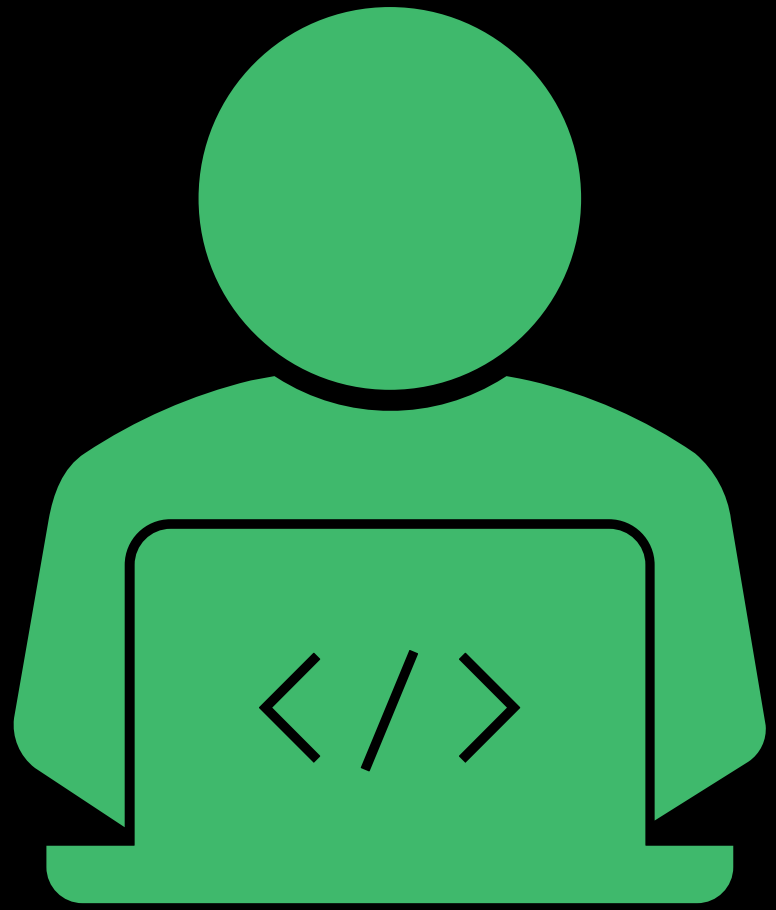
EL CÓDIGO QUE ES TAN **COMPRENSIBLE** COMO EL INGLÉS SIMPLE.

ADECUADO PARA TAREAS COTIDIANAS, PERMITIENDO TIEMPOS DE DESARROLLO CORTOS.

¿Qué hace que Python sea tan especial?

¿Por qué Python?

- ¿Por qué los programadores, jóvenes y viejos, experimentados y novatos, quieren usarlo? ¿Cómo fue que las grandes empresas adoptaron Python e implementaron sus productos al usarlo?
- Existen muchas razones. Ya hemos enumerado algunas de ellas, pero vamos a enumerarlas de una manera más práctica:



- es **fácil de aprender** - el tiempo necesario para aprender Python es más corto que en muchos otros lenguajes; esto significa que es posible comenzar la programación real más rápido.

- es **fácil de enseñar** - la carga de trabajo de enseñanza es menor que la que necesitan otros lenguajes; esto significa que el profesor puede poner más énfasis en las técnicas de programación generales (independientes del lenguaje), no gastando energía en trucos exóticos, extrañas excepciones y reglas incomprensibles.

- es **fácil de utilizar** para escribir software nuevo - a menudo es posible escribir código más rápido cuando se emplea Python.

- es **fácil de entender** - a menudo, también es más fácil entender el código de otra persona más rápido si está escrito en Python.

- es **fácil de obtener, instalar y desplegar** - Python es gratuito, abierto y multiplataforma; no todos los lenguajes pueden presumir de eso.

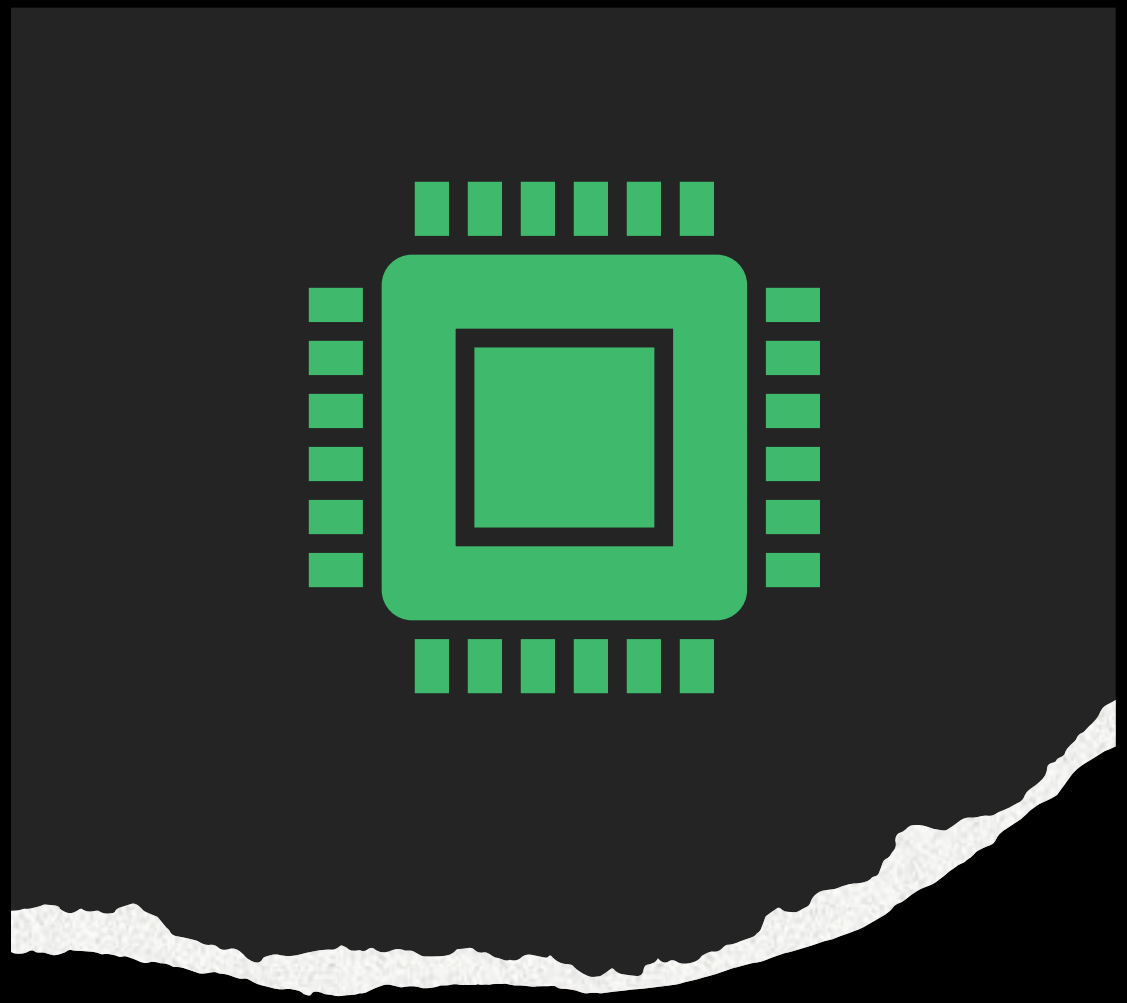
¿Dónde podemos ver a Python en acción?

- Lo vemos todos los días y en casi todas partes. Se utiliza ampliamente para implementar complejos **Servicios de Internet** como motores de búsqueda, almacenamiento en la nube y herramientas, redes sociales, etc. Cuando utilizas cualquiera de estos servicios, en realidad estás muy cerca de Python.
- Muchas **herramientas de desarrollo** se implementan en Python. Cada vez se escriben más **aplicaciones de uso diario** en Python. Muchos **científicos** han abandonado las costosas herramientas patentadas y se han cambiado a Python. Muchos **testers** de proyectos de TI han comenzado a usar Python para llevar a cabo procedimientos de prueba repetibles. La lista es larga.

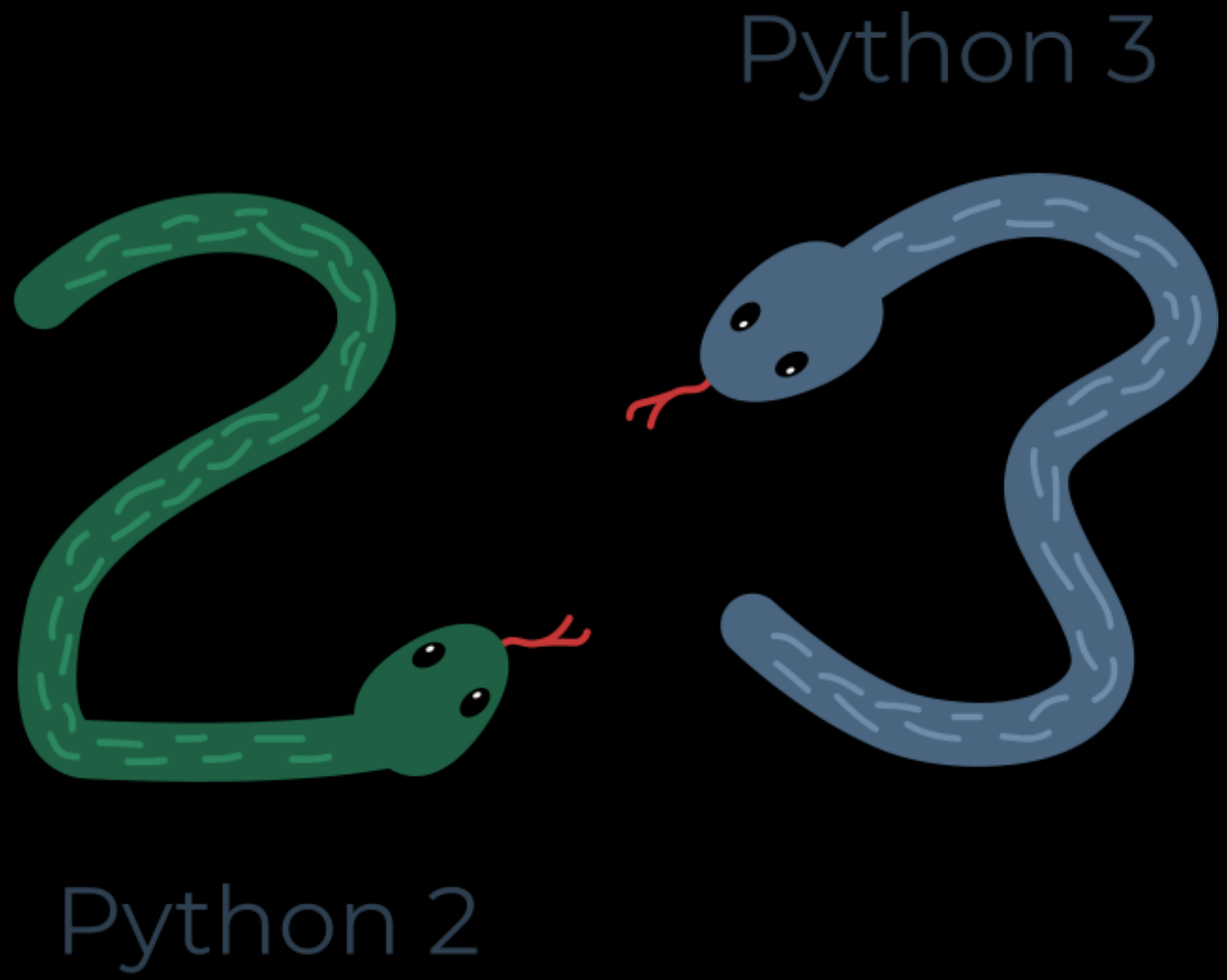
Existe más de un Python

Python 2 vs. Python 3

- Existen dos tipos principales de Python, llamados Python 2 y Python 3.
- Python 2 es una versión anterior del Python original. Su desarrollo se ha estancado intencionalmente, aunque eso no significa que no haya actualizaciones. Por el contrario, las actualizaciones se emiten de forma regular, pero no pretenden modificar el idioma de manera significativa. Prefieren arreglar cualquier error recién descubierto y agujeros de seguridad. La ruta de desarrollo de Python 2 ya ha llegado a un callejón sin salida, pero Python 2 en sí todavía está muy vivo.
- **Python 3 es la versión más nueva (para ser precisos, la actual) del lenguaje. Está atravesando su propio camino de evolución, creando sus propios estándares y hábitos.**



- Estas dos versiones de Python no son compatibles entre sí. Las secuencias de comandos de Python 2 no se ejecutarán en un entorno de Python 3 y viceversa, por lo que si deseas que un intérprete de Python 3 ejecute el código Python 2 anterior, la única solución posible es volver a escribirlo, no desde cero, por supuesto. Grandes partes del código pueden permanecer intactas, pero tienes que revisar todo el código para encontrar todas las incompatibilidades posibles. Desafortunadamente, este proceso no puede ser completamente automatizado.



Implementaciones de Python

Además de Python 2 y Python 3, hay más de una versión de cada uno.

Siguiendo la Página wiki de Python, una *implementación* de Python se refiere a "un programa o entorno que brinda soporte para la ejecución de programas escritos en el lenguaje Python

Cython es una de las posibles soluciones al rasgo de Python más doloroso – la falta de eficiencia. Los cálculos matemáticos grandes y complejos pueden ser fácilmente codificados en Python (mucho más fácil que en "C" o en cualquier otro lenguaje tradicional), pero la ejecución del código resultante puede requerir mucho tiempo.

¿Cómo se reconcilian estas dos contradicciones?

Una solución es escribir tus ideas matemáticas usando Python, y cuando estés absolutamente seguro de que tu código es correcto y produce resultados válidos, puedes traducirlo a "C". Ciertamente, "C" se ejecutará mucho más rápido que Python puro.

Esto es lo que pretende hacer Cython: traducir automáticamente el código de Python (limpio y claro, pero no demasiado rápido) al código "C“ (complicado y hablador, pero ágil).

Otra versión de Python se llama Jython.

"J" es de "Java". Imagina un Python escrito en Java en lugar de C. Esto es útil, por ejemplo, si desarrollas sistemas grandes y complejos escritos completamente en Java y deseas agregarles cierta flexibilidad de Python.

El tradicional CPython puede ser difícil de integrar en un entorno de este tipo, ya que C y Java viven en mundos completamente diferentes y no comparten muchas ideas comunes. Jython puede comunicarse con la infraestructura Java existente de manera más efectiva. Es por esto que algunos proyectos lo encuentran útil y necesario.

Nota: la implementación actual de Jython sigue los estándares de Python 2. Hasta ahora, no hay Jython conforme a Python 3.

PyPy - un Python dentro de un Python. En otras palabras, representa un entorno de Python escrito en un lenguaje similar a Python llamado RPython (Restricted Python). En realidad es un subconjunto de Python.

El código fuente de PyPy no se ejecuta de manera interpretativa, sino que se traduce al lenguaje de programación C y Tluego se ejecuta por separado.

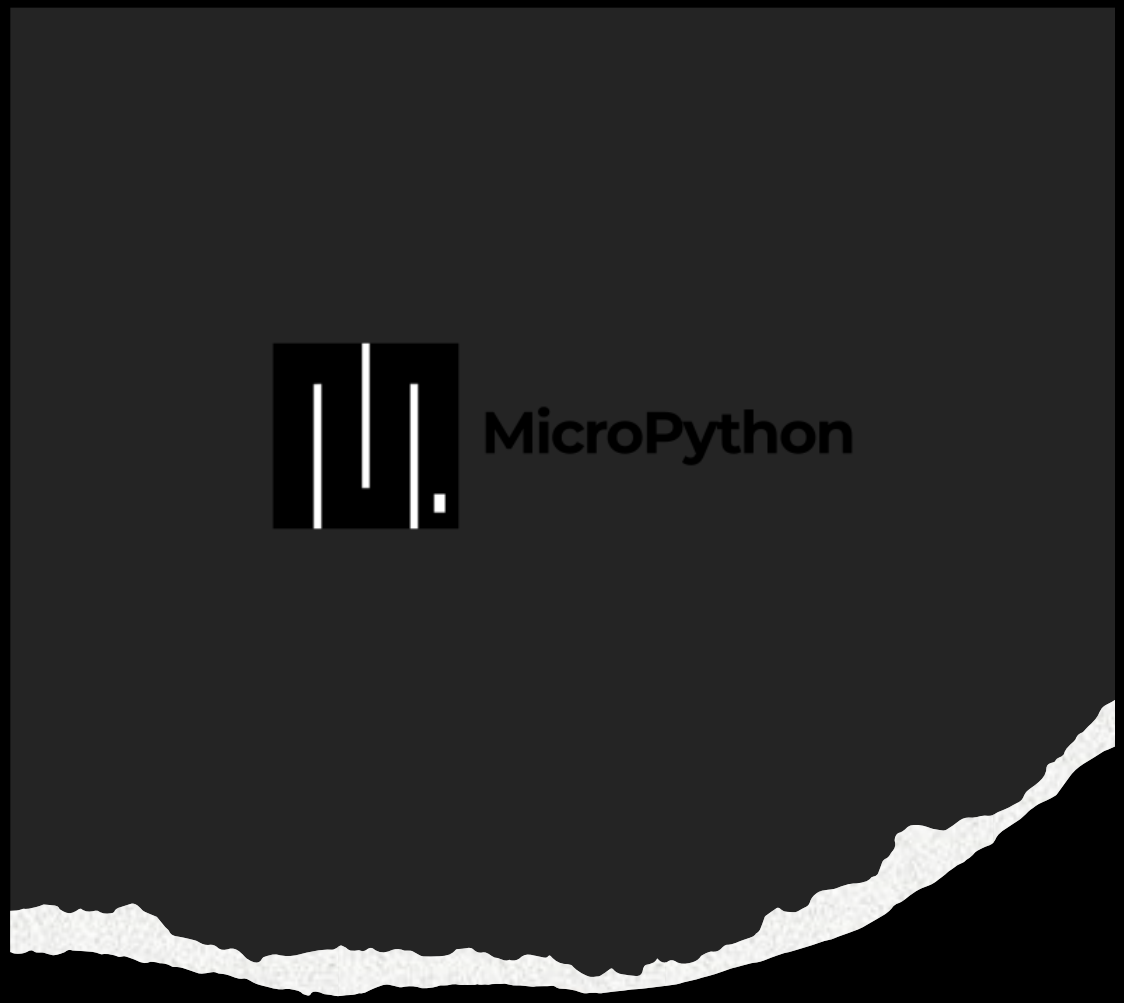
Esto es útil porque si deseas probar cualquier característica nueva que pueda ser o no introducida en la implementación de Python, es más fácil verificarla con PyPy que con CPython.

Esta es la razón por la que PyPy es más una herramienta para las personas que desarrollan Python que para el resto de los usuarios.

Esto no hace que PyPy sea menos importante o menos serio que CPython.

Además, PyPy es compatible con el lenguaje Python 3.

Hay muchos más Pythons diferentes en el mundo. Los encontrarás sí los buscas, pero este curso se centrará en CPython.



MicroPython es una implementación eficiente de software de código abierto de Python 3 que está optimizada para ejecutarse en microcontroladores.

Incluye un pequeño subconjunto de la biblioteca estándar de Python, pero está repleto de una gran cantidad de funciones, como mensajes interactivos o números enteros de precisión arbitraria, así como módulos que dan acceso al programador a hardware de bajo nivel.


Gracias por su atención