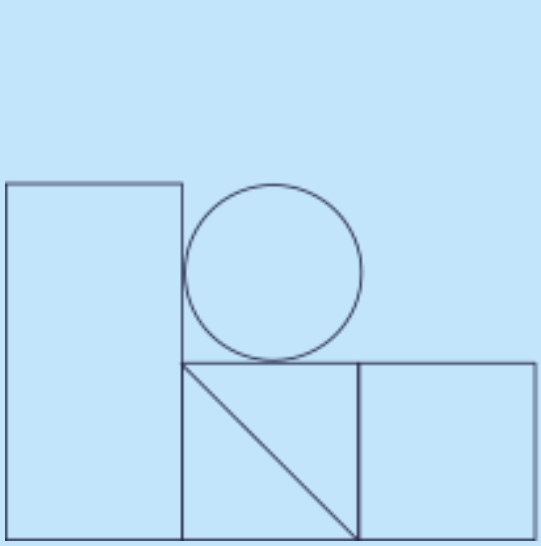
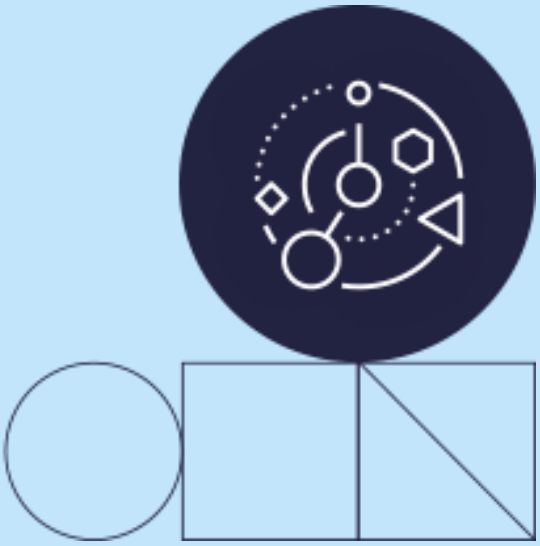
IBM SkillsBuild

IBM

Conceptos básicos y sintaxis de Python

Introducción al lenguaje Python





Índice

Introducción

Python suele ser el lenguaje con el que muchas personas comienzan en el mundo de la programación. Es un lenguaje multiparadigma en el que se puede trabajar con programación estructurada, con programación orientada a objetos o programación funcional y tiene una curva de aprendizaje significativamente menor que otros lenguajes de programación de alto nivel.

Python es un lenguaje de programación interpretado que se caracteriza por su legibilidad y por fomentar el software de calidad. Si nos quedamos con la descripción fría, Python es un lenguaje multiparadigma (permite programar de manera imperativa, con orientación a objetos y, en menor medida, de manera funcional), interpretado y de tipado dinámico.

Python fue creado en 1991 por Guido Van Rossum, quien, hasta hace poco, seguía dirigiendo su desarrollo bajo el título de Dictador Benevolente Vitalicio o BDFL (Benevolent Dictator For Life).

El desarrollo de **Python** está coordinado bajo el paraguas de la Python Software Foundation, quien provee recursos y un marco organizativo. **Python** es Software Libre, al estar licenciado bajo la licencia Python Software License.

¿Por qué utilizar Python?

Con la cantidad de lenguajes de programación que existen en la actualidad es fácil preguntarse ¿por qué aprender **Python**? ¿qué ventajas ofrece este lenguaje? A continuación, podéis ver un listado de las principales ventajas que han hecho de **Python** un lenguaje ampliamente utilizado.

Calidad de Software

Python tiene como foco la legibilidad, coherencia y calidad de software. Esto le ha permitido situarse en un lugar destacado comparado con otros lenguajes de programación. **Python** está diseñado para que los programas escritos en este lenguaje sean muy legibles y, por lo tanto, más mantenibles. La uniformidad del código **Python** permite que un programa en **Python** sea más fácil de entender, incluso por personas que no han escrito ese código. Tal es el impacto que ha supuesto este foco, que otros lenguajes de programación de reciente aparición también están apostando por estos aspectos.

Aumento de la productividad del programador

Los programas escritos en **Python** son, típicamente, entre un tercio y un quinto (¡a veces incluso más!) más cortos que sus equivalentes escritos en C, C++ o Java. Esto implica que los programas de **Python**, no sólo son más rápidos de escribir, sino que, por lo general, también son más robustos. Tener que escribir menos líneas de código implica que hay que menos código que “debuguear” y menos que mantener. A esto hay que añadir que, al ser **Python** un lenguaje interpretado, los programas se pueden ejecutar directamente, sin perder el tiempo en largos procesos de compilación y linkado requeridos en otros lenguajes compilados.

Portabilidad de código

La mayoría de software escrito en **Python** puede ser portado a los mayores Sistemas Operativos y plataformas con ningún o muy poco esfuerzo. La mayoría de software en **Python** puede ser portado de Linux a Windows, por ejemplo, simplemente copiando el código de una máquina a otra.

Librería estándar

Python incluye una gran colección de librerías, la librería estándar, que cubren una multitud de funcionalidades: interfaces con el sistema operativo, manipulación de textos, conexión con bases de datos, etc. Por ello se suele decir que **Python** viene con las baterías incluidas (batteries included). A la librería estándar se le une el enorme ecosistema de librerías de terceros disponible en la red. Desde frameworks de para robótica hasta librerías de computación numérica que rivalizan con (y en muchos casos superan) a Matlab, pasando por frameworks web, etc. Todos ellos completamente libres y gratuitos para el usuario.

Disfrute por la programación

Esta última es más subjetiva, aunque es una razón muy comentada por muchos programadores de Python. Debido a la facilidad de programación y al extenso ecosistema de librerías disponibles, muchos desarrolladores afirman que programar en **Python** se convierte más en un placer que en una tarea aburrida. Algunos hablan también de la belleza del código en **Python**, tanto por su estructura como por su coherencia. Es más, algunos desarrolladores que han tenido que volver a programar en otros lenguajes tras haber estado utilizando **Python** durante un tiempo, han expresado su frustración al tener que volver a lenguajes menos “bellos”.

Diferencias entre Python 2 y 3

Actualmente hay dos ramas principales en Python: 2.x, liberada a principios del año 2000, y 3.x, liberada por primera vez en 2008. A primera vista, Python 3 puede parecer similar a Python 2, pero hay numerosas diferencias profundas, incluyendo el soporte extenso de Unicode en Python 3, módulos renombrados, cambios en las importaciones, cambios en las divisiones y más. Antes de comenzar a aprender, debemos tener en cuenta algunas cosas muy importantes.

Para empezar, debemos hacernos a la idea de que Python 2.x y Python 3.x son lenguajes completamente diferentes. En realidad, son el mismo Python, pero las diferencias son tantas que es difícil imaginarlos como un mismo lenguaje. Un script compatible con Python 2.x podría no ejecutarse en Python 3.x y viceversa. Lo mejor es hacerse a la idea de que son lenguajes diferentes.

En primer lugar, por pura lógica entenderemos que Python 3 es más moderno y actualizado que Python 2, y dado que un programador debe estar siempre trabajando con las últimas versiones de los lenguajes, lo recomendable es aprender Python 3. Está más optimizado, más extendido y por supuesto, es el futuro del lenguaje.

Python 2 dejó de tener soporte a partir del año (2020), quiere decir que Python 2.7 fue la última versión de Python 2.x, nunca habrá un Python 2.8, si vamos a aprender Python, no tiene sentido invertir tiempo en Python 2.

La única razón aceptable para aprender Python 2, es hacerlo con la intención de trabajar en mantenimiento de aplicaciones ya creadas en Python 2. Lo lógico es pensar que las empresas que en su día apostaron por Python 2.x harían una migración, han tenido mucho tiempo. Pero la verdad, en el mundo real, las empresas siempre tienen código viejo que necesitan mantener y no siempre se actualiza.

Ahora bien, si una empresa piensa hacer un desarrollo nuevo con Python 2.x, la verdad es que, como programadores responsables, deberíamos advertir de todo lo que eso involucra. Las debilidades del software al no tener actualizaciones ni soporte por parte del lenguaje, puede convertirse en un gran problema a largo plazo.

Aun así, existen formas de poder migrar e incluso mantener ambas versiones en un mismo proyecto:

Python 3 propone implementar lanzamientos más rápidos, mejoras del rendimiento, nuevas funciones en el uso de las cadenas, nuevos operadores de unión y API internas más consistentes y estables. ¿Qué diferencias hay entre Python 2 y 3?.

Mejor soporte Unicode

En Python 2, las cadenas se almacenan como ASCII de forma predeterminada y debíamos agregar una u’ si queríamos almacenarla como cadenas Unicode. En Python 3 esto ya no es necesario ya que las cadenas de texto se almacenan de forma predeterminada en Unicode.

Esto es muy importante ya que Unicode es mucho más versátil que ASCII. Las cadenas Unicode pueden almacenar letras en distintos idiomas, números romanos, símbolos, emojis, etc., ofreciéndonos muchísimas más opciones.

Mejora la división entera

En Python 2, si escribimos un número sin ningún dígito después del punto decimal, se redondea su cálculo al número entero más cercano. Por ejemplo, si intentamos dividir 5 entre 2 (5/2) el resultado será 2 debido al redondeo.

Tendríamos que escribirlo como 5.0 / 2.0 para obtener la respuesta exacta de 2.5. Sin embargo, en Python 3, la expresión 5/2 devolverá el resultado

esperado de 2.5 sin tener que preocuparnos por agregar esos ceros adicionales.

Sin duda, este pequeño y simple ejemplo demuestra como la sintaxis de Python 3 termina siendo mucho más intuitiva, algo que le dará mucha más comodidad a los novatos que intentan aprender el lenguaje de programación Python.

Diferencias de Sintaxis en determinados comandos

Es probable que veamos esta diferencia un tanto trivial, y es que sin duda lo es ya que no afecta en nada la funcionalidad de Python, pero es una diferencia de sintaxis de declaración de print muy notoria que no debemos pasar por alto.

En Python 3, la declaración de print ha sido reemplazada por una función print(). Por ejemplo, en Python 2 se escribe print “hola” pero en Python 3 se escribe print(“hola”). Como principiantes que buscamos comenzar a aprender lenguaje de programación en Python, esta diferencia no debería preocuparnos ya que no va a afectarnos mucho, pero no deja de ser un punto a tener en cuenta.

Así mismo, la función raw_input() se renombra input().