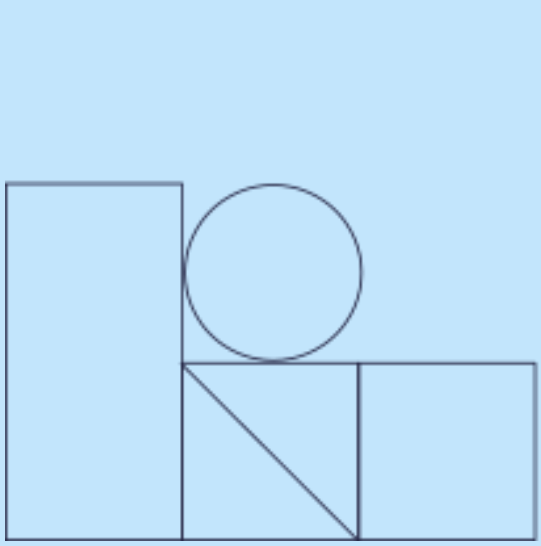
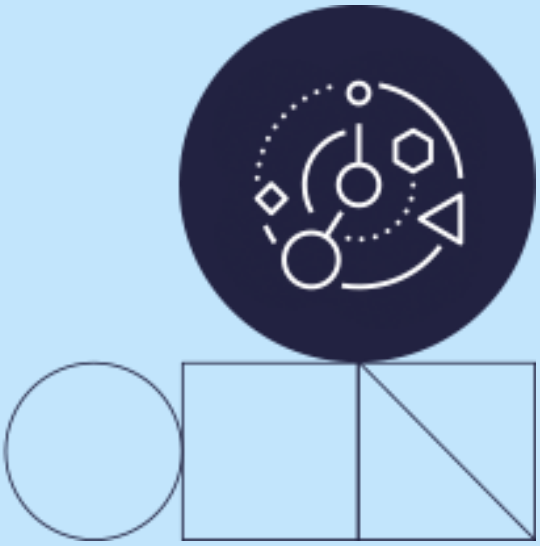
IBM SkillsBuild

IBM

Conceptos básicos y sintaxis de Python

Sintaxis Python





Índice

Introducción

La sintaxis de **Python**, viendo cómo podemos empezar a usar el lenguaje creando nuestras primeras variables y estructuras de control.

El termino sintaxis hace referencia al conjunto de reglas que definen como se tiene que escribir el código en un determinado lenguaje de programación. Es decir, hace referencia a la forma en la que debemos escribir las instrucciones para que el ordenador, o más bien lenguaje de programación, nos entienda.

En la mayoría de lenguajes existe una sintaxis común, como por ejemplo el uso de = para asignar un dato a una variable, o el uso de {} para designar bloques de código, pero **Python** tiene ciertas particularidades.

La sintaxis es a la programación lo que la gramática es a los idiomas. De la misma forma que la frase “Yo estamos aquí” no es correcta, el siguiente código en Python no sería correcto, ya que no respeta las normas del lenguaje.

```
if ($variable){
    x=9;
}
```

Lo veremos a continuación en detalle, pero **Python** no soporta el uso de $ ni hace falta terminar las líneas con ; como en otros lenguajes, y tampoco hay que usar {} en estructuras de control como en el if.

Por otro lado, de la misma forma que un idioma no se habla con simplemente saber todas sus palabras, en la programación no basta con saber la sintaxis de un lenguaje para programar correctamente en él. Es cierto que sabiendo la sintaxis podremos empezar a programar y a hacer lo que queramos, pero el uso de un lenguaje de programación va mucho más allá de la sintaxis.

Cadenas, operadores aritméticos y el uso del condicional if

Para empezar a perderle el miedo a la sintaxis de **Python**, vamos a ver un ejemplo donde vemos cadenas, operadores aritméticos y el uso del condicional if.

El siguiente código simplemente define tres valores a, b y c, realiza unas operaciones con ellos y muestra el resultado por pantalla.

```
# Definimos una variable x con una cadena
x = "El valor de (a+b)*c es"

# Podemos realizar múltiples asignaciones
a, b, c = 4, 3, 2

# Realizamos unas operaciones con a,b,c
d = (a + b) * c

# Definimos una variable booleana
imprimir = True

# Si imprimir, print()
if imprimir:
    print(x, d)

# Salida: El valor de (a+b)*c es 14
```

Como puedes observar, la sintaxis de **Python** es muy parecida al lenguaje natural o pseudocódigo, lo que hace que sea relativamente fácil de leer. Otra ventaja es que no necesitamos nada más, el código anterior puede ser ejecutado tal cual está. Si conoces otros lenguajes como **C** o **Java**, esto te resultará cómodo, ya que no es necesario crear la típica función main().

Comentarios

Los comentarios son bloques de texto usados para comentar el código. Es decir, para ofrecer a otros programadores o a nuestro yo futuro información relevante acerca del código que está escrito. A efectos prácticos, para **Python** es como si no existieran, ya que no son código propiamente dicho, solo anotaciones.

Los comentarios se inician con # y todo lo que vaya después en la misma línea será considerado un comentario.

```
# Esto es un comentario
```

Al igual que en otros lenguajes de programación, podemos también comentar varias líneas de código. Para ello es necesario hacer uso de triples comillas bien sean simples ''' o dobles """. Es necesario usarlas para abrir el bloque del comentario y para cerrarlo.

```
'''
Esto es un comentario
de varias líneas
de código
'''
```

Es interesante que, como programadores, nos acostumbremos a comentar todo lo posible nuestro código. La mayoría de los proyectos de cierta envergadura se hacen en equipo y dejar nuestro código comentado facilita la labor al compañero que venga después de nosotros. En el caso de que programemos en solitario, comentar el código también es una excelente idea, pues es probable que dentro de un tiempo (meses o años) tengamos que revisar o escalar nuestro código, y si lo dejamos bien comentado podremos retomar el trabajo donde lo dejamos con mucha mayor facilidad.

Identación y bloques de código

En la mayoría de los lenguajes de programación los elementos que contienen código (como procedimientos, funciones, condicionales, bucles...etc) hacen uso de paréntesis, corchetes y llaves para incluir dicho código. El **Python** no es así. En **Python** los bloques de código se representan con identación, es decir, poniendo el código que hay dentro de uno de los elementos anteriormente mencionados, varios espacios a la derecha, y aunque hay un poco de debate con respecto a usar tabulador o espacios, la norma general es usar cuatro espacios.

En el siguiente código tenemos un condicional if. Justo después tenemos un print() identado con cuatro espacios. Por lo tanto, todo lo que tenga esa identación pertenecerá al bloque del if.

En Python:

```
if True:
    print("True")
```

En otros lenguajes de programación la sintaxis sería algo como:

```
if (True)
{print("True")};
```

Esto es muy importante ya que el código anterior y el siguiente no son lo mismo. De hecho, el siguiente código daría un error ya que el if no contiene ningún bloque de código, y eso es algo que no se puede hacer en **Python**.

```
if True:
print("True")
```

Por otro lado, a diferencia de en otros lenguajes de programación, no es necesario utilizar ; para terminar cada línea.

```
# Otros lenguajes como C
# requieren de ; al final de cada línea
x = 10;
```

Sin embargo, en **Python** no es necesario, basta con un salto de línea.

```
x = 5
y = 10
```

Pero se puede usar el punto y coma ; para tener dos sentencias en la misma línea.

```
x = 5; y = 10
```

Múltiples líneas

En algunas situaciones se puede dar el caso de que queramos tener una sola instrucción en varias líneas de código. Uno de los motivos principales podría ser que fuera demasiado larga, y de hecho en la especificación **PEP8** se recomienda que las líneas no excedan los 79 caracteres.

Haciendo uso de \ se puede romper el código en varias líneas, lo que en determinados casos hace que el código sea mucho más legible.

```
x = 1 + 2 + 3 + 4 +\
    5 + 6 + 7 + 8
```

Si por lo contrario estamos dentro de un bloque rodeado con paréntesis (), bastaría con saltar a la siguiente línea.

```
x = (1 + 2 + 3 + 4 +
     5 + 6 + 7 + 8)
```

Se puede hacer lo mismo para llamadas a funciones:

```
def funcion(a, b, c):
    return a+b+c

d = funcion(10,
23,
3)
```