Entero o int

Los enteros en Python o también conocidos como int, son un tipo de datos que permite representar números enteros, es decir, positivos y negativos no decimales. Si vienes de otros lenguajes de programación, olvídate de los int8, uint16 y demás. Python nos da un sólo tipo que podemos usar sin preocuparnosdel tamaño del número almacenado por debajo.

Introducción a int

Los tipos enteros o int en Python permiten almacenar un valor numérico no decimal ya sea positivo o negativo de cualquier valor. La función type() nos devuelve el tipo de la variable, y podemos ver com efectivamente es de la clase int.

i = 12

print(i) #12

print(type(i)) #<class 'int'>

En otros lenguajes de programación, los int tenían un valor máximo que pueden representar. Dependiendo de la longitud de palabra o wordsize de la arquitectura del ordenador, existen unos números mínimos y máximos que era posible representar. Si por ejemplo se usan enteros de 32 bits el rango a representar es de -2^31 a 2^31–1, es decir, -2.147.483.648 a 2.147.483.647. Con 64 bits, el rango es de -9.223.372.036.854.775.808 hasta 9.223.372.036.854.775.807. Una gran ventaja de Python es que ya no nos tenemos que preocupar de esto, ya que por debajo se encarga de asignar más o menos memoria al número, y podemos representar prácticamente cualquier número.

x = 250\*\*250

print(x)

print(type(x))

#305493636349960468205197939321361769978940274057232666389361390928129162652472045770185723510801522825687515269359046715531785342780428396973513311420091788963072442053377285222203558881953188370081650866793017948791366338993705251636497892270212003524508209121908744820211960149463721109340307985507678283651836204093399373959982767701148986816406250000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

#<class 'int'>

Para saber más: Si vienes de otras versiones de Python, tal vez eches de menos el tipo long. Puedes leer la PEP237, donde se explica porqué se unificó los long y los int.

Ejemplos

En el primer ejemplo hemos asignado un número decimal a una variable, pero también es posible asignar valores en binario, hexadecimal y octal. El prefijo 0b indica que lo que viene a continuación será interpretado como un número binario. Para el caso hexadecimal es con 0x y octal con 0c. Al imprimir, el número se convierte en decimal para todos los casos.

a = 0b100

b = 0x17

c = 0o720

print(a, type(a)) #4 <class 'int'>

print(b, type(b)) #23 <class 'int'>

print(c, type(c)) #464 <class 'int'>

Con el siguiente ejemplo podemos ver como Python asigna diferente número de bits a las variables en función del número que quieren representar. La función getsizeof() devuelve el tamaño de una variable en memoria.

import sys

x = 5\*\*10000

y = 10

print(sys.getsizeof(x), type(x))

print(sys.getsizeof(y), type(y))

Aunque como hemos dicho, Python puede representar casi cualquier número, hay casos límite en los que nos podemos encontrar con una excepción como la que mostramos a continuación.

print(5e200\*\*2)

# OverflowError

Un caso curioso es que si intentamos representar un número aún mayor, nos encontraremos con lo siguiente en vez de con una excepción.

print(2e2000\*\*2)

# inf

Convertir a int

El posible convertir a int otro tipo. Como hemos explicado, el tipo int no puedo contener decimales, por lo que si intentamos convertir un número decimal, se truncará todo lo que tengamos a la derecha de la coma.

b = int(1.6)

print(b) #1