Float

El tipo numérico float permite representar un número positivo o negativo con decimales, es decir, números reales. Si vienes de otros lenguajes, tal vez conozcas el tipo doble, lo que significa que tiene el doble de precisión que un float. En Python las cosas son algo distintas, y los float son en realidad double.

Para saber más: Los valores float son almacenados de una forma muy particular, denominada representación en coma flotante. En el estándar IEEE 754 se explica con detalle.

Por lo tanto si declaramos una variable y le asignamos un valor decimal, por defecto la variable será de tipo float.

f = 0.10093

print(f) #0.10093

print(type(f)) #<class 'float'>

Conversión a float

También se puede declarar usando la notación científica con e y el exponente. El siguiente ejemplo sería lo mismo que decir 1.93 multiplicado por diez elevado a -3.

f = 1.93e-3

También podemos convertir otro tipo a float haciendo uso de float(). Podemos ver como True es en realidad tratado como 1, y al convertirlo a float, como 1.0.

a = float(True)

b = float(1)

print(a, type(a)) #1.0 <class 'float'>

print(b, type(b)) #1.0 <class 'float'>

Rango representable

Alguna curiosidad es que los float no tienen precisión infinita. Podemos ver en el siguiente ejemplo como en realidad f se almacena como si fuera 1, ya que no es posible representar tanta precisión decimal.

f = 0.99999999999999999

print(f) #1.0

print(1 == f) #True

Los float a diferencia de los int tienen unos valores mínimo y máximos que pueden representar. La mínima precisión es 2.2250738585072014e-308 y la máxima 1.7976931348623157e+308, pero si no nos crees, lo puedes verificar tu mismo.

import sys

print(sys.float\_info.min) #2.2250738585072014e-308

print(sys.float\_info.max) #1.7976931348623157e+308

De hecho si intentas asignar a una variable un valor mayor que el max, lo que ocurre es que esa variable toma el valor inf o infinito.

f = 1.7976931348623157e+310

print(f) #inf

Si quieres representar una variable que tenga un valor muy alto, también puedes crear directamente una variable que contenga ese valor inf.

f = float('inf')

print(f) #inf

Precisión del float

Desafortunadamente, los ordenadores no pueden representar cualquier número, y menos aún si este es uno irracional. Debido a esto, suceden cosas curiosas como las siguientes.

Dividir 1/3 debería resultar en 0.3 periódico, pero para Python es imposible representar tal número.

print("{:.20f}".format(1.0/3.0))

# 0.33333333333333331483

Por otro lado, la siguiente operación debería tener como resultado cero, pero como puedes ver esto no es así.

print(0.1 + 0.1 + 0.1 - 0.3)

# 5.551115123125783e-17

Afortunadamente, salvo aplicaciones muy concretas esto no resulta un problema. Al fin y al cabo, ¿a quién le importa lo que haya en la posición decimal 15?