Clase 01 - Números y cadenas de caracteres

# **Números en Python**

## **Tipos de número**

Los números de Python están relacionados con los números matemáticos, pero están sujetos a las limitaciones de la representación numérica en las computadoras.

Python distingue entre enteros, números de punto flotante y números complejos.

## **Enteros**

Los números enteros son aquellos que no tienen decimales, tanto positivos como negativos (además del cero). En Python se pueden representar mediante el tipo int (de integrer, entero) o el tipo long (largo). La única diferencia es que el tipo long permite almacenar números más grandes. Es aconsejable no utilizar el tipo long a menos que sea necesario, para no malgastar memoria.

**Ejemplo**:

**1, 2, 525, 0, -817**

## **Long**

Los números enteros largos o long en Python son iguales a los enteros, no tienen decimales, y pueden ser positivos, negativos o cero. Se tratan de números de cualquier tamaño. Se puede definir con una L al costado de nuestro número.

**Ejemplo:**

**9812893712387912379123L**

**897538475389475198237891249823L**

**12387349587373L**

## **Float / Decimal**

Los números reales, son los que tienen decimales, en Python se expresan mediante el tipo **float**. Desde Python 2.4 cuenta con un nuevo tipo Decimal, para el caso de que se necesite representar fracciones de forma más precisa.

**Ejemplo:**

**0,270**

**-12,1233**

**989,87439124387**

**-74,9349834**

## **Complejos**

Los números complejos son los que tienen parte imaginaria, es muy probable que no lo vayas a necesitar nunca. Este tipo se llama complex, se almacena usando reales, ya que es una extensión de dichos números.

**Ejemplo:**

**2,1j**

**-41,832i**

**88,23 254j**

# 

# **Operaciones numéricas**

## **Operaciones numéricas en Python**

En programación y en matemáticas, los operadores aritméticos son aquellos que manipulan los datos de tipo numérico, es decir, permiten la realización de operaciones matemáticas (sumas, restas, multiplicaciones, etc.). El resultado de una operación aritmética es un dato aritmético, es decir, si ambos valores son números enteros el resultado será de tipo entero; si alguno de ellos o ambos son números con decimales, el resultado también lo será.

## 

## 

## 



# **Procedencia**

## **Procedencia de los operadores**

Al igual que ocurre en matemáticas, en programación también tenemos una prioridad en los operadores. Esto significa que si una expresión matemática es precedida por un operador y seguido de otro, el operador más alto en la lista debe ser aplicado por primera vez. Las expresiones con paréntesis se evalúan de dentro a fuera, el paréntesis más interno se evalúa primero.

## **Procedencia**

1. Términos entre paréntesis.

2. Potenciación y raíces.

3. Multiplicación y división.

4. Suma y resta.

El orden normal de las operaciones es de izquierda a derecha, evaluando en orden los siguientes operadores:

## **Procedencia**

1. ( )

2. \*\*

3. X, /, %, //

4. +, -

En el lenguaje de programación de Python se representan los operadores con el siguiente orden:

# **Cadenas de texto**

## **Cadenas de texto en Python**

Las cadenas (o strings) son un tipo de datos compuestos por secuencias de caracteres que representan texto. Estas cadenas de texto son de tipo str y se delimitan mediante el uso de comillas simples o dobles.

**Ejemplo**:

**“Esto es una cadena de texto”**

**‘Esto también es una cadena de texto’**

En el caso de que queramos usar comillas (o un apóstrofe) dentro de una cadena tenemos distintas opciones. La más simple es encerrar nuestra cadena mediante un tipo de comillas (simples o dobles) y usar el otro tipo dentro de la cadena. Otra opción es usar en todo momento el mismo tipo de comillas, pero usando la barra invertida (\) como carácter de escape en las comillas del interior de la cadena para indicar que esos caracteres forman parte de la cadena.

**Ejemplos:**

**“Esto es un ‘texto’ entre comillas dobles”**

**‘Esto es otro “texto” entre comillas simples’**

**“Esto es otro \“texto\” todo en comillas dobles”**

**‘Esto otro \’texto\’ todo en comillas simples’**

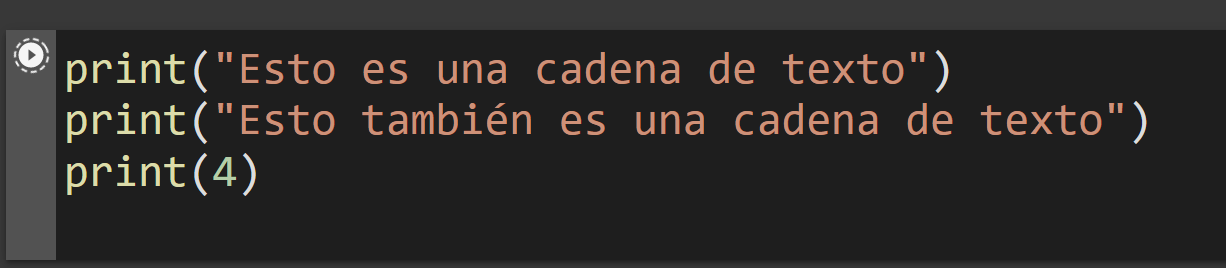
## 

## 

## **PRINT**

**¿Para qué sirve?**

La forma correcta de mostrar cadenas de texto (u otros objetos) por pantalla en Python es utilizando una función llamada print (imprimir). Se indica lo que se desea mostrar por pantalla entre paréntesis.



**Ventajas**

Usar print tiene sus ventajas. Por ejemplo, nos deja mostrar por pantalla caracteres especiales, como tabulación o saltos de línea.

Ejemplo:

**[in] print('Una cadena\tcon tabulación’)**

**[out]Una cadena con tabulación**

**[in] print('Otra cadena\ncon salto de línea’)**

**[out]Una cadena**

**con salto de línea**

Si por ejemplo quisiéramos imprimir el directorio de una carpeta, sería de la siguiente forma: **print(‘C:\nombre\directorio’).** Pero va a tomar el **\n** como carácter especial para salto de línea. Para poder ignorar estos caracteres especiales Python tiene una forma de “printear” cruda o raw. Lo indicamos con una **r** delante de lo que se va a imprimir, y Python automáticamente lo interpretará para no tomar en cuenta los caracteres especiales.

**print(r’C:\nombre\directorio’)**

Otra funcionalidad que tiene es permitir mostrar una cadena en distintas líneas, de forma que con un solo print se muestran varias líneas de cadenas.

Para lograrlo tenemos que pasarlo entre tres comillas dobles, o tres comillas simples.

Ejemplo:

**print("""una cadena**

**otra cadena**

**otra cadena más**

**""")**

# **Variables**

## **Variables en matemáticas**

Las cadenas (o strings) son un tipo de datos compuestos por secuencias de caracteres que representan texto. Estas cadenas de texto son de tipo str y se delimitan mediante el uso de comillas simples o dobles.

Ejemplo:

**“Esto es una cadena de texto”**

**‘Esto también es una cadena de texto’**

Dependiendo del contexto, las variables significan cosas distintas. En el caso del Álgebra, una variable representa una cantidad desconocida que se relaciona con otras. Consideremos por ejemplo la ecuación:

*x + 3 = 5*

## **Variables en programación**

En algunos lenguajes de programación, las variables se pueden entender como "cajas" en las que se guardan los datos, pero en Python las variables son "etiquetas" que permiten hacer referencia a los datos (que se guardan en unas "cajas" llamadas objetos).

Python es un lenguaje de programación orientado a objetos y su modelo de datos también está basado en objetos. Para cada dato que aparece en un programa, Python crea un objeto que lo contiene.

**Cada objeto tiene:**

* Un **identificador** único (un número entero, distinto para cada objeto). El identificador permite a Python referirse al objeto sin ambigüedades.
* Un **tipo de datos** (entero, decimal, cadena de caracteres, etc.). El tipo de datos permite saber a Python qué operaciones pueden hacerse con el dato.
* Un **valor**, el propio dato.

Las variables en Python no guardan los datos, sino que son simples nombres para poder hacer referencia a esos objetos.

## **Variables en programación**

En Python, si escribimos la instrucción: a = 2. Se crea el objeto "2". Ese objeto tendrá un identificador único que se asigna en el momento de la creación y se conserva a lo largo del programa. En este caso, el objeto creado será de tipo número entero y guardará el valor 2. Se asocia el nombre a al objeto número entero 2 creado.

Al describir la instrucción anterior no habría que decir 'la variable **a** almacena el número entero 2', sino que habría que decir 'podemos llamar a al objeto número entero 2'. La variable a es como una etiqueta que nos permite hacer referencia al objeto "2", más cómoda de recordar y utilizar que el identificador del objeto.

## **Definir una variable**

Siempre se escribe a la izquierda de la igualdad, de lo contrario, Python generará un mensaje de error:

**>>> 2 = mi\_variable**

**SyntaxError: can’t assign to literal**

Para mostrar el valor de la variable hay que escribir su nombre, o “printearlo”.

**>>> mi\_variable = 2**

**>>> mi\_variable**

**2**

**>>>print(mi\_variable)**

**2**

Si una variable no se ha definido previamente, al escribir su nombre o printear la variable generará un error:

**>>> x = -10**

**>>> y**

**Traceback (most recent call last):**

**File "<pyshell#1>", line 1, in <module>**

**y**

**NameError: name 'y' is not defined**

Una variable puede almacenar números, texto o estructuras más complicadas (que se verán más adelante). Si se va a almacenar texto, el texto debe escribirse entre comillas simples (') o dobles ("), que son equivalentes. A las variables que almacenan texto se les suele llamar cadenas (de texto).

**>>> nombre = "Pepito Conejo"**

**>>> nombre**

**'Pepito Conejo'**

**>>> print(nombre)**

**'Pepito Conejo'**

Si no se escriben comillas, Python supone que estamos haciendo referencia a otra variable (que, si no está definida, genera un mensaje de error):

**>>> nombre = Pepe**

**Traceback (most recent call last):**

**File "<pyshell#0>", line 1, in <module>**

**nombre = Pepe**

**NameError: name 'Pepe' is not defined**

**>>> nombre = Pepito Conejo**

**SyntaxError: invalid syntax**

Aunque no es obligatorio, se recomienda que el nombre de la variable esté relacionado con la información que se almacena en ella para que sea más fácil entender el programa.

Si el programa es trivial o mientras se está escribiendo un programa, esto no parece muy importante, pero si se consulta un programa escrito por otra persona o escrito por uno mismo hace tiempo, resultará mucho más fácil entender el programa si los nombres están bien elegidos.

## **Nombres de variables**

El nombre de una variable debe empezar por una letra o por un guión bajo (\_) y puede seguir con más letras, números o guiones bajos (esto en inglés se llama snake case).

**>>> fecha\_de\_nacimiento = "27 de octubre de 1997"**

**>>> fecha\_de\_nacimiento**

**'27 de octubre de 1997'**

Los nombres de variables no pueden incluir espacios en blanco.

**>>> fecha de nacimiento = "27 de octubre de 1997"**

**SyntaxError: invalid syntax**

Los nombres de las variables pueden contener mayúsculas, pero tenga en cuenta que Python distingue entre mayúsculas y minúsculas (en inglés se dice que Python es case-sensitive).

**>>> nombre = "Pepito Conejo"**

**>>> Nombre = "Numa Nigerio"**

**>>> nomBre = "Fulanito Mengánez"**

**>>> nombre**

**'Pepito Conejo'**

**>>> Nombre**

**'Numa Nigerio'**

**>>> nomBre**

**'Fulanito Mengánez'**

## **Input**

En Informática, la "entrada" o input de un programa son los datos que llegan al programa desde el exterior. Actualmente, el origen más habitual es el teclado. Python tiene una función llamada input() la cual permite obtener texto escrito por teclado. Al llegar a la función, el programa se detiene esperando que se escriba algo y se pulse la tecla Intro.

Ej.: **>>> nombre = input()**

Otra solución, más compacta, es aprovechar que a la función input() se le puede enviar un argumento que se escribe en la pantalla (sin añadir un salto de línea):

Ejemplo:

**>>> nombre = input(“¿Cómo te llamas?”)**

## **Conversión de tipos**

De forma predeterminada, la función input() convierte la entrada en una cadena, aunque escribamos un número. Si intentamos hacer operaciones, se producirá un error.

Si se quiere que Python intérprete la entrada como un número entero, se debe utilizar la función int() de la siguiente manera:

Ejemplo: **>>> nombre = int(input(“¿Que edad tienes?”))**

## **Indexación de strings**

Cada uno de los caracteres de una cadena (incluidos los espacios) tiene asignado un índice. Este índice nos permite seleccionar su carácter asociado haciendo referencia a él entre corchetes (**[]**) en el nombre de la variable que almacena la cadena.

Si consideremos el orden de izquierda a derecha, el índice empieza en 0 para el primer carácter, etc.

También se puede considerar el orden de derecha a izquierda, en cuyo caso al último carácter le corresponde el índice -1, al penúltimo -2 y así sucesivamente.

Este método es útil si por ejemplo queremos acceder a caracteres en las últimas posiciones de una cadena con muchos caracteres de la cual no conocemos su longitud.

**>>> cadena = 'Python’**

**>>> cadena[0] 🡪 ‘P’**

**>>> cadena[-1] 🡪‘n’**

## **Longitud de strings**

Python nos da una función llamada len. Esta función nos permite saber cuál es la longitud de un string, sin la necesidad de contar uno a uno los caracteres que tiene. También nos sirve en el caso de que no sepamos qué valor tiene una variable, pero tenemos que sacar determinados caracteres por índice.

Ejemplo de len:

**>>> palabra = 'Python'**

**>>> len(palabra)**

**6**

**>>> otra\_palabra = 'Hola, como están? Yo bien! '**

**>>> len(otra\_palabra)**

**26**

## **Rebanar string (slicing)**

Es el índice del último carácter no incluido de la porción de la cadena que queremos seleccionar. Indica cada cuantos caracteres seleccionamos entre las posiciones de inicio y fin.

**>>> cadena = 'Python'**

**>>> cadena[0:4:1] 🡪 ‘Pyth’**

**>>> cadena[2:6:2] 🡪 ‘to’**

## **¿Reasignar valor?**

En algún momento nos preguntaremos si es posible traer el valor de una cadena de un índice a otro, ¿eso significa que puedo cambiarle el valor de un índice a uno que yo quiera?

Observemos el siguiente ejemplo:

**>>> palabra = “Pithon”**

¡Cometimos un error! Debería decir Python, no Pithon

>>> palabra[1] = “y”

Traceback (most recent call last):

File "<pyshell#0>", line 1, in <module>

palabra[1] = “y”

TypeError: ‘str’ object does not support item assignment

Pero, ¿qué pasó acá? ¿Por qué se rompió?

**Para recordar**

En Python, las cadenas de texto o strings, son inmutables esto significa, que no se puede sustituir ninguno de sus caracteres individualmente. Pero esto no es un gran problema.

Python es flexible, ¡podemos modificar el string que deseemos con slicing!

**>>> palabra = 'Pithon’**

**>>> palabra = palabra[0:1] + 'y' + palabra[2:]**

De esta forma podremos mostrar Python y no Pithon.

## 

## **Promedio ponderado o pesado**

No todos los valores tienen el mismo “peso” o valor.

* Promedio entre 3 y 10 es: (1.3 + 1.10) / 2, este es el promedio tradicional donde todos los valores tienen un peso de 1.
* Promedio pesado entre 3 y 10 es: (13.3 + 2.10) / 15, acá vemos que el peso de 3 es 13, y el peso del 10 es 2, por lo que el 3 es más importante, se divide por la suma de los pesos.