Tipos de datos básicos

Tipos simples

- Jupyter notebook ofrece una herramienta de ejecución interactiva con la cual es posible dar órdenes directamente al intérprete y obtener una respuesta inmediata para cada una de ellas.
- Python es un **lenguaje interpretado**. El intérprete actúa como una simple calculadora. Es posible introducir una expresión y éste escribirá el resultado de evaluar la expresión.
- La sintaxis es sencilla: los operadores +, -, * y / funcionan como en la mayoría de los lenguajes; los paréntesis (()) pueden ser usados para agrupar.
- No es necesario escribir un programa completo para empezar a obtener resultados de ejecución.

```
2 + 5
```

```
7
```

```
6 + 8 - 9
```

```
5
```

La tabla siguiente tabla resume las características de los operadores Python: su aridad (número de operandos), asociatividad y precedencia.

Operación	Operador	Aridad	Asociatividad	Precedencia
Exponenciación	**	Binario	Por la derecha	1
Identidad	+	Unario	—	2
Cambio de signo	-	Unario	—	2
Multiplicación	*	Binario	Por la izquierda	3
División	/	Binario	Por la izquierda	3
Módulo (o resto)	%	Binario	Por la izquierda	3
Suma	+	Binario	Por la izquierda	4
Resta	-	Binario	Por la izquierda	4

Enteros

```
# potencia
2 ** 3      # 2 elevado a 3
```

```
8
```

```
# resto de la división entera (operación módulo)
5 % 2
```

```
1
```

```
2 * 4 - (7 - 1)
```

```
2
```

```
# división entera: El resultado es un número sin decimales
5 // 2
```

```
2
```

```
# división con decmales: el resultado es un número con decimales.
5 / 2
```

```
2.5
```

En ocasiones deseamos que el ordenador recuerde ciertos valores para usarlos más adelante. Necesitamos el concepto de **variable**. Por ejemplo, supongamos que deseamos efectuar el cálculo del perímetro de un círculo de radio 1.298373.

```
# perímetro del círculo :2*pi*r        esta línea es un comentario que no se ejecuta

r = 1.298373                    # en la varaible r almacenamos el valor del radio
pi = 3.14159265359              # en la variable pi almacenamos el valor de pi
perimetro = 2 * pi * r      #perímetro
```

Podemos preguntar por el valor de la variable perímetro:

```
print(perimetro)      # print es una función predefinida en python
```

```
8.157918156839218
```

En la celda anterior hemos utilizado una función predefinada de python. Se trata de la función `print`. Como ves, sirve para mostrar el contenido de una variable. La usaremos de aquí en adelante en

muchas ocasiones.

```
perimetro            # escribiendo solo la variable, jupyter muestra el contenido
```

```
8.157918156839218
```

Si queremos conocer el valor del perímetro de nuevo, tenemos que volver a a realizar el cálculo. Lo mejor es almacenar el valor del perímetro en una variable nueva:

Variables:

Una **variable** es un espacio para almacenar datos en la memoria de un ordenador. En Python, una variable se define con la sintaxis:

```
nombre_de_la_variable = valor_de_la_variable
```

- Cada variable, tiene un nombre y un valor.
- Así podemos almacenar un valor en una variable y posteriormente podemos utilizar dicha variable en expresiones. A esta acción de almacenar un valor a una variable se denomina **asignación**.

```
# usamos las variables pi y r definidas aneriormente para calcular el área
area = pi * ( r ** 2 )          # área
```

Los nombres de las variables en Python pueden contener caracteres alfanuméricos (empezando con una letra) a-z, A-Z, 0-9 y otros símbolos como la _.

Por cuestiones de estilo, las variables suelen empezar con minúscula. Podéis consultar la guia de estilo para código escrito en Python en el siguiente enlace https://www.python.org/dev/peps/pep-0008/

> Algunos nombres no pueden ser usados como nombres de variables (son palabras reservadas por python):
>
> and, as, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while, with, yield

Pero las órdenes que puede ejecutar el intérprete de Python no son únicamente operaciones aritméticas y asignaciones. También puede ejecutar **funciones**.

```
d = 6
print(d)
```

```
6
```

```
print('el valor total es: ', d)
```

```
el valor total es:  6
```

```
d = 88888
```

```
max(1,5,34,35,3,5)
```

```
35
```

```
print('Hola Mundo')
```

```
Hola Mundo
```

Python es un lenguaje de tipado dinámico: el tipo se determina en el momento de la asignación.

- Cada variable, tiene un nombre y un valor. El valor define también el tipo de datos de la variable.

```
a = 9.0                 # el tipo de a es float o real
a = 'hola'              # aquí el tipo de a es cadena o string
type(a)
```

```
str
```

En la celda anterior, usamos la función `type` para preguntar acerca del tipo de a variable `a`. La respuesta es `str` que se refiere al tipo cadena o string.
Para conocer el tipo de una variable podemos ejecutar la función `type` de Python.

```
type(d)
```

```
int
```

Las divisiones por cero lanzan un error:

```
1 // 0
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-20-a477bb877ddd> in <module>()
----> 1 1 // 0

ZeroDivisionError: integer division or modulo by zero
```

Los errores se capturan mediante el uso de excepciones (como en otros lenguajes de programación).
Las operaciones entre enteros son, generalmente, más rápidas que las operaciones con reales. Así pues, utilizaremos enteros a menos que de verdad necesitemos números con decimales.

Aritmética de coma flotante

Los valores de tipo real se representan mediante el punto, separando así la parte entera de la parte decimal. Estos valores decimos que son de tipo `float`.

```
precio = 3.     # El punto indica la coma decimal
precio
```

```
3.0
```

```
type(precio)
```

```
float
```

Es posible mezclar en una misma expresión datos de tipos distintos. Si alguno de los operandos en una expresión es real, el resultado es también real:

```
precio / 2
```

```
1.5
```

```
precio // 2
```

```
1.0
```

```
precio * 2   # multiplicación
```

```
6.0
```

A continuación mostramos un ejemplo sencillo con variables y operadores aritméticos:

```
sueldo_bruto = 35000
retenciones = 17.3
neto = sueldo_bruto - (sueldo_bruto * retenciones / 100)
neto
```

```
28945.0
```

Funciones y Casting de tipos

Python proporciona funciones que podemos utilizar en las expresiones. Estas funciones se dice que están **predefinidas** (como la función **print**):

```
# Valor absoluto
abs( 2 - 7 )
```

```
5
```

```
max(1 , 5 , 8 , 7)   # máximo de un conjuto de valores (separados por comas)
```

```
8
```

```
min( -1 , 1 , 0 )    # mínimo de un conjuto de valores
```

```
-1
```

```
round( 18.6 )       # redondea un float
```

```
19
```

Podemos **convertir expresiones** a `int, float, bool, str` mediante funciones:

```
int( 18.6 )
```

```
18
```

```
float( 1 )
```

```
1.0
```

```
float( 8 + 9.9 )
```

```
17.9
```

```
complex(2)
```

```
(2+0j)
```

```
str ( 256568 )    # convierte un entero a una cadena de caracteres
```

```
'256568'
```

Tipo de datos Boolean

Las constantes booleanas son `True` y `False`.

```
v = True  # cierto
f = False  # falso
```

Los operadores booleanos son los habituales, `and` para la conjunción, `or` para la disyunción y `not` para la negación.

Tabla de Verdad del Operador Lógico "AND"

Operando1	Operando 2	AND
V	V	V
V	F	F
F	V	F
F	F	F

Tabla de Verdad del Operador Lógico "OR"

Operando1	Operando 2	OR
V	V	V
V	F	V
F	V	V
F	F	F

```
a = True
a and False
```

```
False
```

```
a or False
```

```
True
```

```
not a
```

```
False
```

```
print( a )
```

```
True
```

Además dispone de los operadores habituales de comparación. Los operadores de comparación son:

- == igual a
- != distinto de
- < menor que
- <= menor o igual que

> Devolverán un valor booleano: `True` o `False`

```
f = 2
2.0  ==  f and f == True
```

```
False
```

```
a = True
b = False
a == b
  #  la comparación devuelve un valor lógico
```

```
False
```

```
5 * 4 <= 100  and 3 > 9
```

```
False
```

```
print( a , b )        # la función print con dos parámetros
                    # imprime el valor de a seguido el valor de b
```

```
True False
```

```
'aaab' > 'ba'    # comparación de cadenas según el órden lexicográfico
```

```
False
```

El valor `True` se corresponde con el valor entero 1, mientras que el el valor `False` se corresponde con el valor entero 0. Así, es posible realizar operaciones aritméticas con valores booleanos.

```
(True + 4) * True
```

```
5
```

El tipo cadena de caracteres

- Mucha gente usa Python por sus capacidades para el tratamiento y procesamiento de cadenas.
- Se corresponde con el tipo String de otros lenguajes. Las cadenas se pueden encerrar en comillas dobles o simples.
- En Python 3.6 el tipo cadena es UNICODE por defecto.

```
objetivo = "Aprender a manejar Python"
meta = 'Ser todo un experto'
print(objetivo)
print(meta)
```

```
Aprender a manejar Python
Ser todo un experto
```

Las cadenas también se pueden comparar.

```
a = 'Ana'
b = "ANA"
a == b
```

```
False
```

Las cadenas de caracteres se pueden concatenar con el operador +.

```
b = "Me he enamorado de " + a
b
```

```
'Me he enamorado de Ana'
```

```
resultado = 9
texto = 'Suma de productos es ' + str(resultado)
texto
```

```
'Suma de productos es 9'
```

Se puede utilizar los operadores de comparación. El orden entre cadenas es el orden lexicográfico.

```
a < b
```

```
True
```

Para conocer el tamaño de una cadena, se utiliza la función `len` como se muestra a continuación:

```
a
```

```
'Ana'
```

```
len(a)
a
```

```
'Ana'
```

Cadenas de caracteres largas, muy largas, ...

Las cadenas largas de caracteres se encierran entren triple compilla doble.

```
# -*- coding: utf-8 -*-
larga = """ Los Reyes Católicos fue la denominación que recibieron los esposos
Fernando II de Aragón e Isabel I de Castilla, soberanos de la Corona de
Castilla (1474-1504) y de la Corona de Aragón (1479-1516).

Los Reyes accedieron al trono de Castilla tras la Guerra de Sucesión
Castellana (1475-1479) contra los partidarios de la princesa Juana la
Beltraneja, hija del rey Enrique IV de Castilla. En 1479 Fernando heredó el
trono de Aragón al morir su padre, el rey Juan II de Aragón. Isabel y Fernando
reinaron juntos hasta la muerte de ella en 1504. Entonces Fernando quedó
únicamente como rey de Aragón, pasando Castilla a su hija Juana, apodada
"la Loca", y a su marido Felipe de Austria, apodado "el Hermoso", Archiduque
de Austria, duque de Borgoña y conde de Flandes. Sin embargo Fernando no
renunció a controlar Castilla y, tras morir Felipe en 1506 y ser declarada
Juana incapaz, consiguió ser nombrado regente del reino hasta su muerte en 1516"""

print(larga)
```

```
 Los Reyes Católicos fue la denominación que recibieron los esposos
Fernando II de Aragón e Isabel I de Castilla, soberanos de la Corona de
Castilla (1474-1504) y de la Corona de Aragón (1479-1516).

Los Reyes accedieron al trono de Castilla tras la Guerra de Sucesión
Castellana (1475-1479) contra los partidarios de la princesa Juana la
Beltraneja, hija del rey Enrique IV de Castilla. En 1479 Fernando heredó el
trono de Aragón al morir su padre, el rey Juan II de Aragón. Isabel y Fernando
reinaron juntos hasta la muerte de ella en 1504. Entonces Fernando quedó
únicamente como rey de Aragón, pasando Castilla a su hija Juana, apodada
"la Loca", y a su marido Felipe de Austria, apodado "el Hermoso", Archiduque
de Austria, duque de Borgoña y conde de Flandes. Sin embargo Fernando no
renunció a controlar Castilla y, tras morir Felipe en 1506 y ser declarada
Juana incapaz, consiguió ser nombrado regente del reino hasta su muerte en 1516
```

Formateando cadenas

El uso del símbolo **%** seguido de uno o mas caracteres de formato permiten controlar con precisión la

construcción de cadenas. Por ejemplo, creamos una cadena:

```
texto = "Cambio: %d %s son %.2f %s"
texto % (1, "dolar" , 0.926666666 , "euros")
```

```
'Cambio: 1 dolar son 0.93 euros'
```

Operadores de formato

%s : ha de sustituirse por una cadena

%d : ha de sustituirse por un entero

%.2f : ha de sustituirse por un número con dos decimales

```
radio = 4.323343
pi = 3.1416
area = pi * radio ** 2

print ('El área de un círculo de radio %.3f es %.5f' % (radio, area))
```

```
El área de un círculo de radio 4.323 es 58.72057
```

```
print ('El área de un círculo de radio %.3f es %.5f' % (radio, pi*radio**2))
```

```
El área de un círculo de radio 4.323 es 58.72057
```

References

- Tutorial de Python. Por Guido Van Rossum
- Guía de estilo de código Python