
 No description has been provided for this image No description has been provided for this image

Python Basics I

Tipos de los datos

Python es un lenguaje fuertemente tipado. Eso significa que las variables que usamos pertenecen a un tipo de datos: numero entero (int), real (float), texto (String), u otro tipo de objetos.

¿Por qué es importante saber bien de que tipos son los datos? Porque cada tipo de dato tiene una serie propiedades y operaciones asociadas. Por ejemplo, a un texto no lo puedo sumar 5. Por lo que cuando vayamos a hacer operaciones entre ellos, tenemos que asegurarnos de que son del mismo tipo para que el resultado sea el esperado. `texto + 5` no tiene sentido y va a dar error. Parece obvio, pero hay ocasiones en las que los tipos de los datos no son los esperados.

¿Cuántos tipos de datos hay? Básicamente aquí no hay límites. En este notebook veremos los más básicos, pero en notebooks posteriores verás que puedes crearte tus propios tipos de datos mediante las **clases**. En Python hay una serie de tipos de datos básicos, que podremos usar sin importar ningun módulo externo, son los llamados *built-in Types*. Estos son los más comunes:

- **Numérico:** tenemos `int`, `float` y `complex`. Dependiendo de si es un numero entero, uno real o un complejo.
- **String:** o Cadena. Cadena de texto plano
- **Booleano:** o Lógico. `True` / `False`

¿Cómo sabemos el tipo de datos de una variable? Mediante `type(nombre_variable)`

Contenidos

- [Numéricos](#)
- [Strings](#)

- [Booleano](#)

Numéricos

[al índice](#)

```
In [1]: numero = 22  
type(numero)
```

```
Out[1]: int
```

Bien, es un **int**, un número entero. Si lo que quiero es un numero real, es decir, que tenga decimales, le añado un punto

```
In [4]: numero_real = 22.0  
print(type(numero_real))
```

```
<class 'float'>
```

Aunque no le haya añadido decimales como tal, ya le estoy diciendo a Python que esta variable es un numero real (**float**).

```
In [5]: numero_real_decimales = 22.45123  
type(numero_real_decimales)
```

```
Out[5]: float
```

Algunas operaciones básicas que podemos hacer con los numeros:

- Sumar: `+`
- Restar: `-`
- Multiplicar: `*`
- Dividir: `/`
- Elevar: `**`
- Cociente division: `//`
- Resto de la división: `%`

```
In [6]: print("Sumas/restas")  
print(1 + 2)  
print(1.0 + 2) # Fijate que cuando fuerzo a que alguno de los numeros s  
print(1 + 2.0 # Siempre que opero float e int dará como tipo resultado  
print(1-2)  
  
print("Multiplicaciones/Divisiones")  
print(2 * 2)  
print(2.0 * 2) # Ocurre lo mismo que en las sumas, cuando ponemos uno d  
print(2/2) # Al hacer la división, directamente convierte el numero en  
print(1000/10)
```

```
print("Resto de la división")
print(10/3)
print(int(10/3)) # Me quedo con el entero de la division
print(10 % 3) # Me quedo con el resto de la division
```

Sumas/restas

3
3.0
3.0
-1

Multiplicaciones/Divisiones

4
4.0
1.0
100.0

Resto de la división

3.3333333333333335
3
1

```
In [8]: n = 45356810
print("¿Es", n, "par?")
resto = n % 2
print("El resto es", resto)
if resto == 0:
    print("Es par")
else:
    print("Es impar")
```

¿Es 45356810 par?
El resto es 0
Es par

Strings

al índice

El tercer tipo de datos más común es el *String*, o cadena de texto. Hay varias maneras de declararlo:

```
In [10]: # Con comillas dobles
cadena = "Esto es una variable de tipo string"
cadena_2 = 'Esto es una variable de tipo spring en comillas simples'
print(cadena)
print(cadena_2)
```

Esto es una variable de tipo string

Esto es una variable de tipo spring en comillas simples

Si da la casualidad de que en el texto hay comillas, tenemos la posibilidad de que Python las interprete como parte del texto y no como comando de inicio/cierre de String

```
In [13]: print("String con'simples'")
print('Spring con "dobles"')
print("Spring con \"simples o dobles\"")
```

```
String con'simples'
Spring con "dobles"
Spring con "simples o dobles"
```

En ocasiones queremos poner saltos de línea o tabulaciones en los prints, o simplemente en una variable de tipo String. Para ello usamos los *escape characters* como `\n` para saltos de línea o `\t` para tabulaciones.

```
In [16]: print("Primera linea\nSegunda Linea\n\tTercera Linea, tabulada")
```

```
Primera linea
Segunda Linea
    Tercera Linea, tabulada
```

Para unir dos variables de tipo String, simplemente usamos el `+`

```
In [17]: nombre = "Antonia"
apellido = "Scott"

nombre_apellido = nombre + " " + apellido
print(nombre_apellido)
```

```
Antonia Scott
```

```
In [18]: esto_es_una_prueba = "Pues eso, que estamos jugando"
un_entero = 100
print(estilo_es_una_prueba + un_entero)
```

```
-----
---
TypeError                                Traceback (most recent call la
st)
Cell In[18], line 3
      1 esto_es_una_prueba = "Pues eso, que estamos jugando"
      2 un_entero = 100
----> 3 print(estilo_es_una_prueba + un_entero)

TypeError: can only concatenate str (not "int") to str
```

Booleano

al índice

Por último, el cuarto tipo de datos basiquísimo es el *booleano*: `True` / `False`. Para que Python reconozca este tipo de datos, hay que escribirlos con la primera letra en mayúscula

```
In [22]: False
```

Out[22]: False

```
In [23]: ya_se_de_python = False
print(type(ya_se_de_python))

<class 'bool'>
```

```
In [24]: dinerito = 5
dinerito >= 15
```

Out[24]: False

```
In [25]: dinerito = 15
cond_1 = dinerito >= 15
print(cond_1)
print(type(cond_1))
```

True
<class 'bool'>

Tipos de datos hay muchos, verás más adelante cómo crear tus propios tipos de datos mediante las clases y los objetos. Pero por ahora, quédate con los tipos de datos más simples:

- **int**: entero
- **float**: real
- **str**: cadena de texto
- **booleano**: true/false

En otros lenguajes de programación el valor numérico suele ir más desagregado dependiendo del volumen del mismo. No es lo mismo tener un `double`, que un `float`. Por suerte en Python no hay que preocuparse de eso :)

Veamos más ejemplos

```
In [26]: print(type(1.0))

<class 'float'>
```

```
In [27]: print(type(-74))

<class 'int'>
```

```
In [28]: print(type("3.45"))

<class 'str'>
```

```
In [29]: print(type(True))

<class 'bool'>
```

```
In [30]: print(type(cond_1))
```

```
<class 'bool'>
```

```
In [ ]:
```