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

Python Basics I

Contenidos

- [Funciones \(I\): *Built in*](#):-Built-in)
- [Métodos](#)

Funciones (I): *Built in*

[al índice](#)

Hay una serie de funciones internas, que vienen en el intérprete de Python. Algunas de las más comunes son:

- **Tipos:** `bool()`, `str()`, `int()`, `float()`
- **Min/Max:** `min()`, `max()`
- **print()**
- **type()**
- **range()**
- **zip()**
- **len()**
- ...

La sintaxis de la función es:

`nombre_funcion(argumentos)`

Algunas ya las hemos visto. Sin embargo, hay unas cuantas que las iremos descubriendo a lo largo de estos notebooks. Para más detalle, tienes [aquí](#) todas las funciones *built-in* de la documentación.

De momento, en lo que se refiere a funciones, vamos a ir trabajando con funciones ya hechas, pero más adelante crearemos nuestras propias funciones.

```
In [2]: # Len se usa para calcular la longitud de una variable. Ya veras que lo
print(len("Esto es una cadena de longitud..."))
```

```
# Funcion max. Tiene tantos argumentos como cantidad de números entre l
print(max(1,2,12,13,24,-5))
```

33

24

```
In [7]: cadena = "La longitud de esta cadena es"
print(cadena, len(cadena), "y no es un float", sep = "***test**", end = "
print("Esto va en otra linea")
```

La longitud de esta cadena es***test**29***test**y no es un float Esto va en otra linea

```
In [ ]:
```

Métodos

[al indice](#)

Se trata de una propiedad MUY utilizada en programación. Son funciones propias de las variables/objetos, y que nos permiten modificarlos u obtener más información de los mismos. Dependiendo del tipo de objeto, tendremos unos métodos disponibles diferentes.

Para usar un método se usa la sintaxis `objeto.metodo()`. Ponemos un punto entre el nombre del objeto y el del metodo, y unos paréntesis por si el método necesita de algunos argumentos. **Aunque no necesite de argumentos, los paréntesis hay que ponerlos igualmente.**

Veamos algunos ejemplos

String

Una variable de tipo string, tiene una serie de métodos que permiten sacarle jugo a la cadena de texto. [Aquí](#) tienes todos los métodos que podemos usar en cadenas de texto

```
In [23]: string_ejemplo = "string en minusculas"

# metodo para poner un string en mayusculas
print(string_ejemplo.upper())
string_ejemplo

# metodo todo en minusculas
print("ESTO ES UN EJEMPLO DIFERENTE".lower())
string_ejemplo2 = "ESTO ES UN EJEMPLO DIFERENTE"
print(string_ejemplo2.lower())

# metodo para reemplazar algun caracter o varios
```

```
print(string_ejemplo.replace("s","c"))
print(string_ejemplo.replace("minusculas","minúsculas"))
print(string_ejemplo.replace("s",""))

# método para convertir un string en una lista
print("Esto se va a convertir en una lista de strings".split("convertir"))

# método para conocer la posición dentro de un string
print(string_ejemplo.index("m"))
```

STRING EN MINUSCULAS

esto es un ejemplo diferente

esto es un ejemplo diferente

ctring en minuuculac

string en minúsculas

tring en minucula

['Esto se va a ', ' ' en una lista de strings']

10

In []:

Como ves, se pueden hacer muchas cosas en los Strings gracias a sus métodos. Ya verás cómo la cosa se pone más interesante cuando los tipos de los datos sean todavía más complejos.

Los métodos son una manera de abstraernos de cierta operativa. Convertir todos los caracteres de una cadena a minuscula, puede ser un poco tedioso si no existiese el método `lower()`. Tendríamos que acudir a bucles o programación funcional.



No
description
has been
provided for
this image

ERRORES en métodos

In [24]: *# Cuando un método necesita ciertos argumentos, y no se los proporciona*
string_ejemplo.replace()

```
-----
---
TypeError                                Traceback (most recent call last)
Cell In[24], line 2
      1 # Cuando un método necesita ciertos argumentos, y no se los proporcionamos
----> 2 string_ejemplo.replace()

TypeError: replace expected at least 2 arguments, got 0
```

In []:

In []: