# Paso por valor y referencia

Dependiendo del tipo de dato que enviemos a la función, podemos diferenciar dos comportamientos:

- **Paso por valor**: Se crea una copia local de la variable dentro de la función.
- **Paso por referencia**: Se maneja directamente la variable, los cambios realizados dentro de la función le afectarán también fuera.

Tradicionalmente:

- **Los tipos simples se pasan por valor**: Enteros, flotantes, cadenas, lógicos...
- **Los tipos compuestos se pasan por referencia**: Listas, diccionarios, conjuntos...

## Ejemplo de paso por valor

Como ya sabemos los números se pasan por valor y crean una copia dentro de la función, por eso no les afecta externamente lo que hagamos con ellos:

```
def doblar_valor(numero):
    numero *= 2

n = 10
doblar_valor(n)
print(n)
```

```
10
```

## Ejemplo de paso por referencia

Sin embargo las listas u otras colecciones, al ser tipos compuestos se pasan por referencia, y si las modificamos dentro de la función estaremos modificándolas también fuera:

```
def doblar_valores(numeros):
    for i,n in enumerate(numeros):
        numeros[i] *= 2

ns = [10,50,100]
doblar_valores(ns)
print(ns)
```

```
[20, 100, 200]
```

Para modificar los tipos simples podemos devolverlos modificados y reasignarlos:

```
def doblar_valor(numero):
    return numero * 2

n = 10
```

```
n = doblar_valor(n)
print(n)
```

```
20
```

Y en el caso de los tipos compuestos, podemos evitar la modificación enviando una copia:

```
def doblar_valores(numeros):
    for i,n in enumerate(numeros):
        numeros[i] *= 2

ns = [10,50,100]
doblar_valores(ns[:])  # Una copia al vuelo de una lista con [:]
print(ns)
```

```
[10, 50, 100]
```

---

*Última edición: 25 de Septiembre de 2018*