**programación funcional (functional programming)**

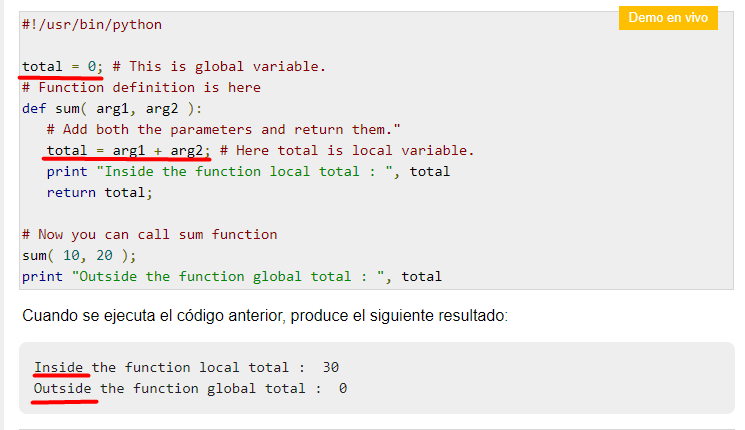
Son PARADIGMAS DE PROGRAMACIÓN:

1. PROCEDURAL (PROCEDIMENTAL): que se ejecutan de forma secuencial
2. FUNCTIONAL: En vez de ir línea a línea creamos funciones para algo en concreto y luego utilizamos esa función en aquellas líneas que nos interesen
3. OBJECT-ORIENTED: organizar el código en objetos (variables). Dentro de estos objetos meteremos la información y la lógica (funciones y methods)

ALCANCE: variables GLOBALES vs LOCALES:

Las variables que se definen dentro de un cuerpo de función tienen un alcance local y las definidas fuera tienen un alcance global.

Esto significa que solo se puede acceder a las variables locales dentro de la función en la que están declaradas, mientras que todas las funciones pueden acceder a las variables globales en todo el cuerpo del programa. Cuando llamas a una función, las variables declaradas dentro de ella entran en el ámbito. A continuación se muestra un ejemplo simple:



¡OJO, si la vamos a utilizar la variable global como una operación dentro de la función, está no funciona! En el ejemplo anterior sí que funciona porque estamos volviendo a definir total como una suma de parámetros locales que sí que existen)!

a = 1

def nombre\_funcion(b)

a = a+b

return a

LANZAMOS nombre función(8) → no funciona pq estamos intentando REFERENCIAR “a” CON “a” (cuando localmente a no existe) - Si fuese solo usarla sí que se podría, pero referenciarla consigo mismo no

a = 1

def nombre\_funcion(b)

global a

a = a+b

return a

LANZAMOS nombre función(8) → funciona pq le hemos puesto un global a (si no daría error de referencia, pq localmente no existiría)

def nombre\_funcion(b)

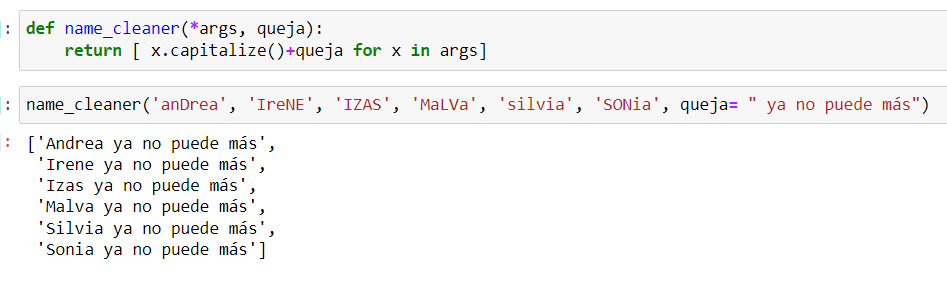
c = a+b

return c

LANZAMOS nombre función(8) → así si funcionaría pq estamos referenciando otra variable.

MÚLTIPLES ARGUMENTOS PARA UNA FUNCIÓN

**\*args** como argumento hace que valga meter cuantos argumentos queramos dentro de la función. ¡OJO, todos sus elementos los devuelve en una Tuple, o List (según esté definido el argumento)!



**\*\*kwarg** como argumento hace que valga meter cuantos argumentos CON KEY queramos dentro de la función. ¡Te los devuelve en un dic!

