Argumentos variables

Contenidos

Introducción

al indice

En los ejemplos de la sesión anterior teníamos que fijar un número concreto de argumentos para nuestras funciones, pero hay ocasiones que no tenemos seguro cuántos argumentos son. Por suerte, las funciones de Python nos aportan esa flexibilidad mediante `*`

Veamos cómo implementar una función multiplicadora con numero variable de argumentos

In [2]:
```
def multiplica(*argumentos):
    print(argumentos)
    print(type(argumentos))
multiplica(2,3,5)
multiplica(1,4,67,4)
```

```
(2, 3, 5)
<class 'tuple'>
(1, 4, 67, 4)
<class 'tuple'>
```

In [6]:
```
def multiplica(*args):
    resultado = 1
    for indice, elemento in enumerate(args):
        print("El siguiente elemento es", args[indice])
        resultado = resultado * elemento
    print("El resultado es", resultado)
    return resultado
multiplica(2,3,4)
multiplica(23,12)
```

```
multiplica()
```

```
El siguiente elemento es 2
El siguiente elemento es 3
El siguiente elemento es 4
El resultado es 24
El siguiente elemento es 23
El siguiente elemento es 12
El resultado es 276
El resultado es 1
```

Out[6]: 1

Ten en cuenta que `*args` es algo variable con X elementos. Como no sabemos a priori cuantos son, tendremos que recorrerlos con un `for`, y para cada argumento, aplicarle una operación. Por tanto, `*args` es un iterable, en concreto una **tupla**. Lo que le está dando la funcionalidad de "argumentos variables" es `*`, no `args`. Igual que ponemos `*args`, podemos poner `*argumentos`.

Puedes combinar argumentos posicionales con los `*args`

In [7]:
```
# En este ejemplo, uso el ultimo argumento para dividir todo lo que hab
def multiplica_divide(*args, div):
    resultado =1
    for i in args:
        resultado = resultado * i
    resultado = resultado / div
    return resultado
multiplica_divide(10,34,21,5)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call la
st)
Cell In[7], line 8
      6     resultado = resultado / div
      7     return resultado
----> 8 multiplica_divide(10,34,21,5)

TypeError: multiplica_divide() missing 1 required keyword-only argument:
'div'
```

In [8]:
```
# En este ejemplo, uso el ultimo argumento para dividir todo lo que hab
def multiplica_divide(*args, div):
    resultado =1
    for i in args:
        resultado = resultado * i
    resultado = resultado / div
```

```
    return resultado
multiplica_divide(10,34,21,div=5)
```

Out[8]: 1428.0

In [ ]:

In [9]:
```
# En este ejemplo, uso el ultimo argumento para dividir todo lo que hab
def multiplica_divide(div, *args):
    resultado =1
    for i in args:
        resultado = resultado * i
    resultado = resultado / div
    return resultado
multiplica_divide(5,10,34,21)
```

Out[9]: 1428.0

In [ ]:

In [10]:
```
def multiplica_divide(div, *args):
    resultado =1
    for i in args:
        resultado = resultado * i
    resultado = resultado / div
    return resultado
multiplica_divide(10,34,21,div=5)
```

```
---------------------------------------------------------------------------
---
TypeError                                 Traceback (most recent call la
st)
Cell In[10], line 7
      5     resultado = resultado / div
      6     return resultado
----> 7 multiplica_divide(10,34,21,div=5)

TypeError: multiplica_divide() got multiple values for argument 'div'
```

ERRORES con argumentos variables

Declara los argumentos variables al principio, y los fijos al final para evitar errores. Además, si los combinas, tendrás que concretar cuáles son los argumentos fijos

In [ ]:

In [ ]:

In [ ]: