

FUNCIONS: PARÀMETRES ARBITRARIS

PARÀMETRES PER OMISSIÓ, PARÀMETRES ARBITRARIS



M^a Belén Tortosa Pedrón



PARÀMETRES PER OMISSIÓ

- En Python també és possible, assignar valors per defecte als paràmetres de les funcions. Això significa, que la funció podrà ser utilitzada amb menys arguments dels que s'espera.
 - Es possible escriure funcions amb paràmetres que tenen determinats valors quan no s'especifiquen en la crida (valors per defecte).
- Exemple:

```
def saludar(nom, missatge="Benvingut"):  
    print (missatge, nom)  
  
saludar("Pöl")
```

```
C:\WINDOWS\system32\cmd.exe  
Benvingut Pöl
```



KEYWORDS COM A PARÀMETRES

- En Python, és possible cridar a una funció, passant-li els arguments esperats, com a parells de **clau=valor**:
- Exemple:

```
def saludar(nom, missatge = "Benvingut"):  
    print (missatge, nom)  
saludar(nom="Pöl", missatge="Hola")
```

```
C:\WINDOWS\system32\cmd.exe  
Hola Pöl
```

```
def saludar(nom, missatge = "Benvingut"):  
    print (missatge, nom)  
saludar(missatge="Hola", nom = "Pöl")
```

```
C:\WINDOWS\system32\cmd.exe  
Hola Pöl
```

```
def saludar(nom, missatge = "Benvingut"):  
    print (missatge, nom)  
saludar(nom = "Pöl", missatge1="Hola")
```

```
C:\WINDOWS\system32\cmd.exe  
Traceback (most recent call last):  
  File "I:\Curs2016-2017\M3\AWS1\material\codi\29_FuncionsPropies", line 3, in <module>  
    saludar(nom = "Pöl", missatge1="Hola")  
TypeError: saludar() got an unexpected keyword argument 'missatge1'
```



KEYWORDS COM A PARÀMETRES

- Podem utilitzar la funció amb el valor per defecte o no sense passar-li el valor per defecte:

```
def multiplicarCadena(cadena, n=1):  
    print (cadena * n)  
  
multiplicarCadena("Hola mon!\n")  
  
multiplicarCadena("Fent plans\n", 10)
```

EXECUCIÓ

```
ca. Administrador: C:\Window  
Hola mon?  
Fent plans  
Fent plans  
Fent plans  
Fent plans  
Fent plans  
Fent plans  
Fent plans  
Fent plans  
Fent plans
```

Important

- Els **keyword arguments** HAN de ser utilitzats després dels «paràmetres posicionals» o «**positional arguments**». A més a més, podem utilitzar la forma *parametre=valor* per a especificar el paràmetre al que li anem a assignar un valor. Exemples:

```
def escribirParametros(p1, p2="Res", p3=0):  
    print (p1)  
    print (p2)  
    print (p3)  
  
print("PRIMERA CRIDA:")  
escribirParametros(1, "Hola", 2)  
print("SEGONA CRIDA:")  
escribirParametros(p1=2, p2="Mon!", p3=3)  
print("TERCERA CRIDA:")  
escribirParametros(3, p2="!")  
print("QUARTA CRIDA:")  
escribirParametros(4, p3=5)
```

EXECUCIÓ

```
ca. Administrador: C:\Window  
PRIMERA CRIDA:  
1  
Hola  
2  
SEGONA CRIDA:  
2  
Mon!  
3  
TERCERA CRIDA:  
3  
!  
0  
QUARTA CRIDA:  
4  
Res  
5
```

Si un paràmetre té un valor per defecte, pot omitir-se



PARÀMETRES ARBITRARIS

- Es possible que una funció esperi rebre un número arbitrari (desconegut) d'arguments. Aquests arguments, arribaran a la funció en forma de tupla.
- Per definir arguments arbitraris en una funció, es posa davant del nom del paràmetre un * *(se representa per una tupla)*
- Exemple:

```
def recorrer_parametros_arbitrarios(parametro_fije, *arbitrarios):  
    print (parametro_fije)
```

```
# Los parámetros arbitrarios se corren como tuplas  
for argument in arbitrarios:  
    print (argument)
```

```
recorrer_parametros_arbitrarios('Valor 1', 'Valor 2', 'Valor 3', 'Valor 4')
```

C:\WINDOWS\system32\cmd.exe

```
Valor 1  
Valor 2  
Valor 3  
Valor 4
```



PARÀMETRES ARBITRARIS

- Si una funció espera rebre paràmetres fixes i arbitraris, **els arbitraris sempre han d'anar després de les fixes.**
- Poden fer paràmetres arbitraris coma parells de clau=valor. En aquest cas, el nom del paràmetre ha d'anar precedits per dos asterisc: ******

```
def recorre_parametres_arbitraris(parametre_fixe, *arbitraris, **kwargs):  
    print (parametre_fixe)  
    for argument in arbitraris:  
        print (argument)  
  
    # Els arguments arbitraris tipus clau, es recorren com els diccionaris  
    for clau in kwargs:  
        print ("El valor de", clau, "es", kwargs[clau])  
  
recorre_parametres_arbitraris("Fixed", "arbitrari 1", "arbitrari 2", "arbitrari 3", clau1="valor un", clau2="valor dos")
```

```
C:\ Windows\system32\cmd.exe  
Fixed  
arbitrari 1  
arbitrari 2  
arbitrari 3  
El valor de clau1 es valor un  
El valor de clau2 es valor dos
```



EXEMPLE

```
def imprimintParametres(p1, *p2, **p3):  
    print (p1)  
    for i in p2:  
        print (i)  
    for valor in p3:  
        print (valor, " : ", p3[valor])  
  
imprimintParametres("Provant", 1, 2, 3, "Provant de nou", a='Utilitzant', b='Python', c=2,d=7)
```

- Los argumentos no formales se clasifican en dos:

- ***variable1**: Se representa per una tupla.
- ****variable2**: Es representa per un diccionari.

- En aquest exemple, els paràmetres de la funció serien:

- p1 = "Provant"
- p2 = (1,2,3,"Provant de nou")
- p3 = {'a':"Utilitzant", 'b':"Python", 'c':2,7}



FUNCIÓ lambda

- Hi ha una forma de crear funcions en una línia: Aquestes són les **funcions lambda**.
Sintaxis:

variable= lambda parametre1, parametre2: operació de retorn

```
formaLambda2 = (lambda x, n: x + n, lambda x, n: x + n)
```

```
print ("Crida 1:", formaLambda2[1](1, 1))  
print ("Crida 2:", formaLambda2[1](2, 1))  
print ("Crida 3:", formaLambda2[0](1, 0))  
print ("Crida 4:", formaLambda2[0](2, 0))
```

Administrado

```
Crida 1: 2  
Crida 2: 3  
Crida 3: 1  
Crida 4: 2
```




EXAMPLE:

```
def formaLambda1(n):  
    return lambda x: x + n  
  
f, g = formaLambda1(1), formaLambda1(0)  
  
print (f(1))  
print (f(2))  
print (g(1))  
print (g(2))
```

EXECUCIÓ

A screenshot of a terminal window titled "Administrac" (likely "Administración"). The terminal displays the output of the Python code: 2, 3, 1, 2, each on a new line.

2
3
1
2