Gestion des arguments

Pour le moment, nous avons vu que les fonctions pouvaient avoir autant d'arguments que souhaité. Il est également possible d'avoir des arguments optionnels. Les arguments optionnels doivent avoir une valeur par défaut (s'ils ne sont pas renseignés, il faut que le code leur attribue quand même une valeur) et il faut qu'ils soient placés après les arguments obligatoires.

Nous avons déjà vu de tels arguments optionnels dans la fonction `print` par exemple.

La syntaxe est la suivante :

```
def f(x, y, z=0, n=1):
    return (x+y+z)**n
```

In [30]:
```
def f(x, y, z=0, n=1):
    return (x+y+z)**n

print(f(1, 1))
print(f(1, 1, z=1))
print(f(1, 1, n=2))
print(f(1, 1, 1, 3))
print(f(1, 1, z=1, n=3))
print(f(1, 1, n=3, z=1))
```

```
2
3
4
27
27
27
```

ATTENTION : POUR LES PLUS INTRÉPIDES !!!

Il est également possible d'avoir une fonction avec un nombre arbitraire d'arguments à une fonction. Pour cela il faut utiliser la convention de `python` et avoir un argument `*` (le dernier de la liste sauf s'il y a un `**`). Habituellement, la coutume veut que cet argument s'appelle `*args` mais ce n'est pas obligatoire.

Prenons l'exemple d'une fonction notée `somme` qui fait la somme de tous les paramètres qui lui sont donnés (mais on veut que le nombre de paramètres soit quelconque).

In [38]:
```
def somme(*args):
    S = 0
    for arg in args:
        # print(f"J'ajoute l'élément {arg}")
        S += arg
    return S
```

In [39]:
```
print(somme(1))
print(somme(1, 2, 3, 4))
```

```
1
10
```

In [41]:
```
somme(*list(range(1000)))
```

Out[41]:
```
499500
```