



Les fonctions

Enseignante: Mme Lamia MANSOURI

1. Déclaration



La déclaration d'une fonction se compose :

- du mot clé `def` suivi de l'identificateur de la fonction, de parenthèses entourant les paramètres de la fonction séparés par des virgules, et du caractère «deux points» qui termine toujours une instruction composée;
- d'une chaîne de documentation (ou docstring) indentée comme le corps de la fonction;
- du bloc d'instructions indenté par rapport à la ligne de définition, et qui constitue le corps de la fonction
- Le bloc d'instructions est obligatoire. S'il est vide, on emploie l'instruction `pass`.
- La valeur de retour est définie en utilisant le mot-clé `return`.

```
def nomDeLaFonction(liste de paramètres):
```

```
...
```

```
bloc d'instructions
```


```
...
```

Exemple


```
# fonction calculant la somme de i1 à i2
def sum(i1, i2):
    result = 0
    for i in range(i1, i2 + 1):
        result += i
    return result
```

- Il est possible de donner des valeurs par défaut aux arguments.
- Les valeurs par défaut sont transmises aux paramètres lorsque la fonction est appelée sans arguments.
- Les arguments prédéfinis doivent suivre les autres

```
def f(x, y=0):  
    print(x*y)
```



```
def f(x=0, y):  
    print(x*y)
```



- Il est possible aussi de retourner plusieurs valeurs

2. Appel d'une fonction



Soit cette fonction

```
def petit (a, b):  
    if (a < b):  
        d = a  
    else:  
        d = 0  
    return d
```

Passage de paramètres par position :

```
print(petit(10, 12))
```

Passer les paramètres selon les positions attendues

Passage par nom:

```
print(petit(a=10,b=12))
```

Aucune confusion possible

```
print(petit(b=12, a=10))
```


3. Variables locales et variables globales



Lorsqu'une fonction est appelée, Python réserve pour elle (dans la mémoire de l'ordinateur) **un espace de noms**. Cet espace de noms local à la fonction est à distinguer de **l'espace de noms global** où se trouvent les variables du programme principal.

variables locales à la fonction :

- Ces variables ne sont accessibles qu'à la fonction elle-même.
- Les contenus des variables locales sont stockés dans l'espace de noms local qui est inaccessible depuis l'extérieur de la fonction.

```
>>> def f():
...     x = 5
...     print(x)
...
>>> f()
5
>>> print(x)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'x' is not defined
```

3. Variables locales et variables globales



Variables globales:

- Les variables définies à l'extérieur d'une fonction sont des variables globales.
- Leur contenu est « visible » de l'intérieur d'une fonction, mais la fonction ne peut pas le modifier.

Exemple :

```
def g() :  
    a=3  
    print("Dans la fonction g : ", a)  
    return a
```

```
def f(a) :  
    a+=2  
    print("Dans la fonction f : ", a)  
    return a
```

```
a=3  
print(a)  
g()  
print(a)  
f(a)  
print(a)  
f(a+5)  
print(a)  
a=f(a+7)  
print(a)
```

```
3  
Dans la fonction g : 3  
3  
Dans la fonction f : 5  
3  
Dans la fonction f : 10  
3  
Dans la fonction f : 12  
12
```

3. Variables locales et variables globales



Utilisation d'une variable globale - global

global est une instruction Python qui indique que la variable a qui est utilisée à l'intérieur de la fonction est la même que celle qui est définie à l'extérieur de la fonction (dans l'environnement global).

```
#fonction
def modif_1(v):
    x = v

#appel
x = 10
modif_1(99)
print(x) ➔ 10
```

x est une variable locale,
pas de répercussion

```
#fonction
def modif_2(v):
    x = x + v

#appel
x = 10
modif_2(99)
print(x)
```

x n'est pas assignée ici,
l'instruction provoque
une **ERREUR**

```
#fonction
def modif_3(v):
    global x
    x = x + v

#appel
x = 10
modif_3(99)
print(x) ➔ 109
```

On va utiliser la variable
globale **x**. L'instruction
suivante équivaut à
x = 10 + 99

4. Imbrication des fonctions



Il est possible de définir une fonction dans une autre fonction. Dans ce cas, elle est locale à la fonction, elle n'est pas visible à l'extérieur.

```
#écriture de la fonction  
def externe(a):
```

```
    #fonction imbriquée  
    def interne(b):  
        return 2.0* b
```

```
    #on est dans externe ici  
    return 3.0 * interne(a)
```

```
#appel  
x = 10  
print(externe(x)) → renvoie 60  
print(interne(x)) → provoque une erreur
```

La fonction interne() est imbriquée dans externe, elle n'est pas exploitable dans le prog. principal ou dans les autres fonctions.

5. La fonction lambda



- Les lambdas Python sont de petites fonctions anonymes (essentiellement utilisé dans les calculs mathématiques).
- des fonctions extrêmement courtes car limitées à **une seule instruction**.
- **pas de return**, l'expression est automatiquement retournée
- pas d'instructions, seulement une expression (par exemple : for, if, while ne sont admis !)

syntaxe

lambda arg1, arg2,... : instruction de retour

Exemples:

$F(x)=x^2$ \longleftrightarrow $F=\text{lambda } x: x^{**}2$

$\text{Print}(F(3))$ \longrightarrow 9

$F(x,y)=x^2+y$ \longleftrightarrow $F=\text{lambda } x, y : x^{**}2 + y$

$\text{Print}(F(3,5))$ \longrightarrow 14