

# Chapitre 4 . Fonctions

On voit dans ce chapitre comment créer et utiliser des fonctions.

## 1- FONCTIONS DEJA VUES DEPUIS CE DEBUT D'ANNEE :

Dans les chapitres précédents, on a déjà utilisé plusieurs fonctions :

<pre>&gt;&gt;&gt; int(3.4)</pre>	Convertit en entier.
<pre>&gt;&gt;&gt; a = int(3.4)</pre>	3 est assigné à a
<pre>&gt;&gt;&gt; float("49.3")</pre>	Convertit en réel
<pre>&gt;&gt;&gt; vote = float("49.3")</pre>	49,3 est assigné en à vote
<pre>&gt;&gt;&gt; type(True)</pre>	Donne le type de « True »
<pre>&gt;&gt;&gt; reponse = type(True)</pre>	Assigne <class 'bool'> à reponse
<pre>&gt;&gt;&gt; input() 2023</pre>	<pre>&gt;&gt;&gt; input() 2023 '2023'</pre>
<pre>&gt;&gt;&gt; annee = input("A saisir : ")</pre>	A saisir :

Les fonctions natives de python sont documentées sur : <https://docs.python.org/fr/3/library/functions.html>

Parmi elles, une fonction intéressante :

<pre>&gt;&gt;&gt; dir()</pre>	<pre>['__annotations__', '__builtins__', '__cached__', '__doc__', '__main__', '__name__', '__package__', '__pyzo__', '__spec__', 'annee', 'reponse']</pre> <p>Sans argument, elle donne la liste des noms dans l'espace de nommage local. Avec un argument, elle essaye de donner une liste d'attributs valides pour cet objet.</p>
-------------------------------	---

Point Cours : Pour exécuter une fonction nommée par exemple *nomFonction*, on écrit :

```
nomFonction( )
```

Vide ou plusieurs arguments

L'exécution de cette fonction entraîne une exécution d'un ensemble de tâches. Lorsque celles-ci sont terminées, `nomFonction()`

prend comme valeur celle **RETOURNEE** par la fonction.

On peut mémoriser ce retour dans une variable, par exemple :

```
maVariable = nomFonction()
```

## 2- COMMENT CREER SA PROPRE FONCTION ? :

⇒ Exemple d'une fonction qui renvoie le maximum de 3 nombres :

```
cours.py
1 def maxi(a,b,c) :
2     M = a
3     if b > M :
4         M = b
5     if c > M :
6         M = c
7     return M
8
9 # Programme principal
10 print(maxi(-8,9,1))
11 max = maxi(2023,1,1)
12 print(f"Le max de 2023,1,1 est : {max}")
```

Mot clé

obligatoire

Arguments de la fonction

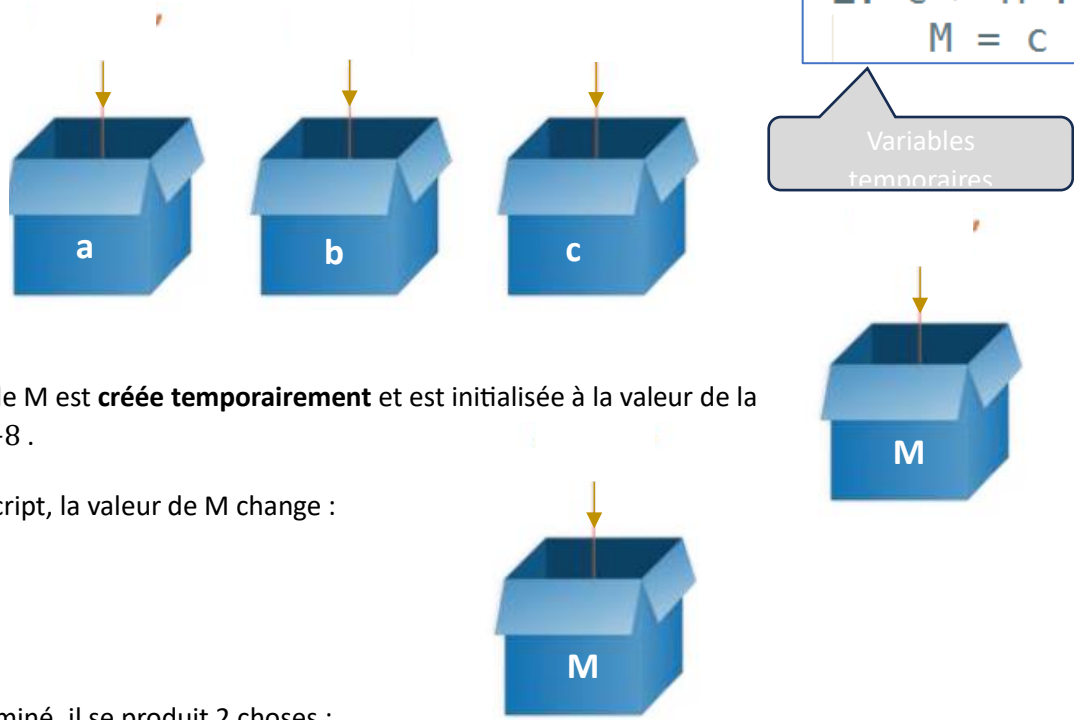
indentation

Valeur de retour

Lors de la 1<sup>ère</sup> exécution de cette fonction, `maxi(-8,9,1)` prend comme valeur 9. Avant d'en arriver là, l'appel à la fonction `maxi()` avec les arguments -8, 9 et 1 provoque l'exécution du script ci-contre avec les variables a, b, c qui sont créées temporairement dans l'espace mémoire et qui prennent automatiquement les valeurs -8, 9 et 1 :

```
M = a
if b > M :
    M = b
if c > M :
    M = c
```

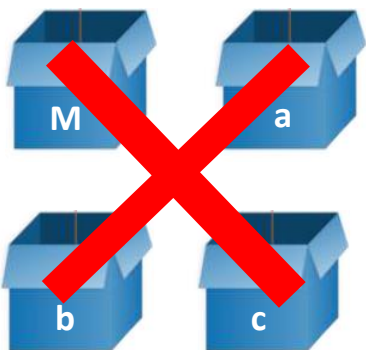
## Créations de nouvelles variables dans l'espace mémoire



Dans ce script, la variable M est **créée temporairement** et est initialisée à la valeur de la variable a, c'est-à-dire -8.

Lors de l'exécution du script, la valeur de M change :

Lorsque le script est terminé, il se produit 2 choses :



- la valeur de la variable M est **retournée** et dans le programme principal et `maxi(-8,9,1)` prend la valeur du retour, c'est-à-dire, on a `maxi(-8,9,1)` qui prend la valeur 9.
- Les variables a, b, c et M qui ont été créées par le script de la fonction sont perdues dans l'espace mémoire.



Point Cours : Les fonctions présentent les avantages :

- **D'encapsuler** du code dans des sortes de « *boîtes noires* » étanches. Les variables utilisées par ce code ne rentrent pas en CONFLIT avec celles utilisées ailleurs.
- D'être facile d'utilisation dans la mesure où l'on n'a que les entrées et sortie à gérer. Les entrées par le biais des ARGUMENTS et la sortie par celui de la VALEUR RETOURNEE.

### 3- PEUT-ON APPELER UNE FONCTION DANS UNE FONCTION ? :

On répond à cette question en étudiant cet exemple :

```

1  def maxi(a,b,c) :
2      M = a
3      if b > M :
4          M = b
5      if c > M :
6          M = c
7      return M
8
9
10 def superMaxi(a,b,c,d,e,f,g,h,i) :
11     a = maxi(a,b,c)
12     b = maxi(d,e,f)
13     c = maxi(g,h,i)
14     return maxi(a,b,c)
15
16
17 # Programme principal
18 a = superMaxi(-8,9,1,99,-888,1,0,0,7)
19 print(a)
20 a = superMaxi('do','ré','mi','fa','sol','la','si','do#','mib')
21 print(a)

```

a	b	c	M
-8			-8
	9		9
		1	9


a	b	c	M
99			99
	-888		99
		1	99

a	b	c	M
0			0
	0		0
		7	7


a	b	c	M
9			9
	99		99
		7	99

Mémoires temporaires  
ouvertes que durant  
l'exécution des fonctions

a	b	c	d	e	f	g	h	i



a



a

#### 4- EST-IL OBLIGATOIRE D'AVOIR UNE VALEUR DE RETOUR ? :

Dans certaines configurations, il n'est pas nécessaire d'avoir une valeur de retour. Par exemple :

```
from turtle import *
```

```
def traitIncline(x,y,longueur,angle,couleur) :  
    up()  
    color(couleur)  
    pensize(10)  
    goto(x,y)  
    left(angle)  
    backward(longueur/2)  
    down()  
    forward(longueur)  
    right(angle)
```

```
def croix(x,y,longueur,couleur) :  
    traitIncline(x,y,longueur,45,couleur)  
    traitIncline(x,y,longueur,135,couleur)
```



# programme principal

```
croix(0,0,100,"red")  
croix(300,300,100,"red")
```

```
croix(0,0,56, « red »)  
croix(100,100,56, « red »)
```

```
mainloop()
```

