

# Fonctions imbriquées et à plusieurs arguments - Corrigé

## NUMWORKS

Cette fiche a été rédigée par **Claire Savinas**. Elle enseigne au lycée Jean Vilar à Villeneuve-Lès-Avignon. Elle est formatrice Python sur l'académie de Montpellier.

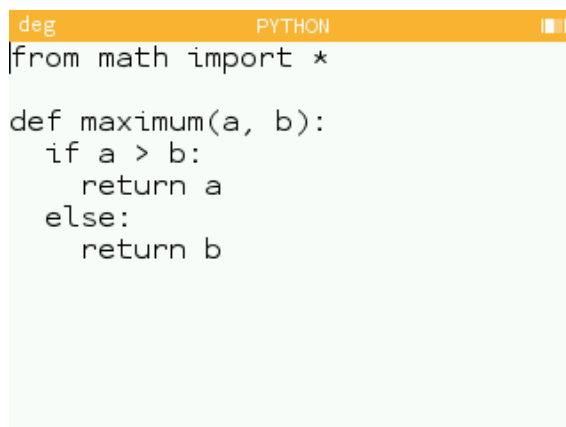
### 1 Objectifs de la séance

- programmer en utilisant une fonction à plusieurs arguments
- créer une fonction qui appelle une autre fonction
- découvrir l'utilité d'une fonction

### 2 Feuille d'activité pour les élèves


#### 2.1 Découverte d'une fonction à deux arguments

(a) Nous allons commencer par créer un nouveau script nommé `activite2.py`. Voici une fonction nommée `maximum` qui prend en entrée deux nombres et qui renvoie le maximum des deux valeurs.



```
deg PYTHON
from math import *

def maximum(a, b):
    if a > b:
        return a
    else:
        return b
```

La saisir dans le script `activite2.py` puis faire **Exécuter le script** dans le menu à droite du titre du script. La fonction ainsi créée est disponible dans le menu de la touche . Appeler cette fonction avec différentes valeurs, par exemple `maximum(12,15)`, `maximum(6,-8)`, ...

(b) Ecrire dans le même script une fonction que vous nommerez `minimum` qui prend en entrée deux nombres et qui renvoie le minimum des deux valeurs.

## 2.2 Découverte d'une fonction à plus que deux arguments

(a) Ecrire dans le même script une fonction que vous nommerez `maximum3` qui prend en entrée trois nombres et qui renvoie le maximum des trois valeurs.

(b) Ecrire dans le même script une fonction que vous nommerez `maximum4` qui prend en entrée quatre nombres et qui renvoie le maximum des quatre valeurs.

**Aide** : Vous pouvez utiliser votre fonction `maximum` et vous appuyer sur un schéma

(c) Ecrire dans le même script une fonction que vous nommerez `maximum8` qui prend en entrée huit nombres et qui renvoie le maximum des huit valeurs en utilisant la fonction `maximum4`.

**Aide** : faire un schéma avec le résultat de chaque appel aux fonctions

## 3 Corrigé de la séance

### 3.1 Découverte d'une fonction à deux arguments

```
deg PYTHON
|from math import *
|
|def maximum(a, b):
|    if a > b:
|        m = a
|    else:
|        m = b
|    return m
```

(a) Nous aurions pu proposer le script ci-contre qui utilise une variable supplémentaire.

(b) Voici deux versions de la correction.

```
deg PYTHON
|from math import *
|
|def minimum(a, b):
|    if a < b:
|        return a
|    else:
|        return b
```

Une première version qui n'utilise pas de variable supplémentaire.

```

deg PYTHON
|from math import *
|
|def minimum(a, b):
|    if a < b:
|        m = a
|    else:
|        m = b
|    return m

```

Et une seconde version qui fait intervenir une variable auxiliaire `m`.

### 3.2 Découverte d'une fonction à plus que deux arguments

(a) Il y a de nombreuses possibilités sans faire appel à la fonction `maximum` mais elles sont longues et complexes.

```

deg PYTHON
|from math import *
|
|def maximum(a, b):
|    if a > b:
|        return a
|    else:
|        return b
|
|def maximum3(a, b, c):
|    m = maximum(a, b)
|    return maximum(m, c)

```

En faisant appel à la fonction `maximum`, la nouvelle fonction est relativement simple.

```

deg PYTHON
|from math import *
|
|def maximum(a, b):
|    if a > b:
|        return a
|    else:
|        return b
|
|def maximum3(a, b, c):
|    return maximum(maximum(a,b),c)

```

Une autre version sans appel à une variable auxiliaire.

(b) De la même manière, on réutilise la fonction `maximum`.

```
deg PYTHON
from math import *

def maximum(a, b):
    if a > b:
        return a
    else:
        return b

def maximum4(a, b, c, d):
    m1 = maximum(a, b)
    m2 = maximum(c, d)
    return maximum(m1, m2)
```

Cette première version utilise deux variables auxiliaires `m1` et `m2`.

```
deg PYTHON
from math import *

def maximum(a, b):
    if a > b:
        return a
    else:
        return b

def maximum4(a, b, c, d):
    return maximum(maximum(a, b),
                  maximum(c, d))
```

Tandis que cette seconde version, n'utilise pas de variable supplémentaire.

(c) Ici, on réutilise la fonction `maximum4` qui fait à son tour appel à la fonction `maximum`.

```
deg PYTHON
def maximum(a, b):
    if a > b:
        return a
    else:
        return b

def maximum4(a, b, c, d):
    return maximum(maximum(a, b),
                  maximum(c, d))

def maximum8(a, b, c, d,
             e, f, g, h):
    return maximum(
        maximum4(a, b, c, d),
        maximum4(e, f, g, h))
```

Une version qui n'utilise pas de variable supplémentaire.

```
def maximum2(a, b):  
    return a if a > b else b  
  
def maximum4(a, b, c, d):  
    return maximum(maximum(a, b),  
                  maximum(c, d))  
  
def maximum8(a, b, c, d,  
             e, f, g, h):  
    m1 = maximum4(a, b, c, d)  
    m2 = maximum4(e, f, g, h)  
    return maximum(m1, m2)
```

Une autre version avec deux variables auxiliaires.