Fonctions imbriquées et à plusieurs arguments

Découverte d'une fonction à deux arguments

1. 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 copy ; var. Appeler cette fonction avec différentes valeurs, par exemple **maximum(12,15)**, **maximum(6,-8)**, ...

Nous aurions pu proposer le script suivant qui utilise une variable supplémentaire.

```
deg                PYTHON
from math import *

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

2. 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.

Voici deux versions de la correction. La première n'utilise pas de variable supplémentaire. La seconde fait intervenir une variable auxiliaire **m**.

```python
from math import *

def minimum(a, b):
  if a < b:
    return a
  else:
    return b
```

```python
from math import *

def minimum(a, b):
  if a < b:
    m = a
  else:
    m = b
  return m
```

Découverte d'une fonction à plus que deux arguments

1. 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.

   Il y a de nombreuses possibilités sans faire appel à la fonction **`maximum`** mais elles sont longues et complexes.
   En faisant appel à la fonction **`maximum`**, la nouvelle fonction est relativement simple. On propose à droite une autre version qui ne fait pas appel à une variable auxiliaire.

```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)
```

```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)
```

2. 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.*

   A gauche, une première version avec utilisation de de deux variables auxiliaires, **`m1`** et **`m2`**.
   A droite, une proposition de script sans variable supplémentaire.

```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)
```

```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))
```

3. 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.

```python
    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))
```

```python
    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):
  m1 = maximum4(a, b, c, d)
  m2 = maximum4(e, f, g, h)
  return maximum(m1, m2)
```