Python 5 : Les fonctions

1) Déclarer une nouvelle fonction

On peut créer de nouvelles fonctions dans Python pour ensuite les utiliser librement dans la suite du programme. On procède ainsi :

```
def somme(x,y):
    S=x+y
    return S
```

En jaune figure le nom de la fonction, c'est vous qui le définissez.
Entre parenthèses **en gras** figurent les variables de la fonction (aussi appelés **arguments** de la fonction), il peut y en avoir un seul, deux, trois, autant que vous voulez. Ils sont séparés d'une virgule.
A l'intérieur de la structure de la fonction (que l'on repère grâce à l'indentation) il y a les opérations que réalise la fonction.
Après l'instruction *return* figure ce que la fonction va renvoyer (c'est l'image de (x,y) par somme).

Ainsi, si à la suite de cette fonction on effectue l'opération :

```
print(somme(1,3))
```

Le programme va afficher 4.

Si on écrit une fonction et qu'on exécute le programme, il ne va rien se passer à l'écran car **définir une fonction créé simplement une nouvelle commande**.
Une fonction se termine toujours par return … car il faut que vous précisiez ce que va renvoyer la fonction (cela ne va pas de soi !).
On évitera de mettre des print à l'intérieur d'une fonction.

2) Quelques exemples de fonctions

Exemple 1:
Voici la fonction valeur absolue bien connue :

```
def valeurabsolue(x):
    if x>0:
        x=x
    else:
        x=-x
    return x
```

Exemple 2:
Voici une fonction qui renvoie le minimum de 2 nombres a et b rentrés en arguments.

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

Exemple 3:

On peut utiliser les fonctions pour gagner du temps dans un programme plus large.
Par exemple :
Si on a la donnée d'une suite récurrente comme ceci :
On a son premier terme $u_0 = 4$ et la formule de récurrence $u_{n+1} = f(u_n)$
Où f est la fonction définie par $f(x) = \begin{cases} 2x - 1 \; si \; x \leq 0 \\ \ln(x) \; si \; x > 0 \end{cases}$.
Si on veut calculer le terme u_{100}, on peut (on suppose avoir importé les fonctions de *math*) :

```
def f(x):                   # on commence par définir la fonction f
    if x>0:
        return log(x)
    else:
        return 2*x-1
u=4                         # on initialise u avec la valeur de u_0
for k in range(0,100):      # on va répéter la boucle 100 fois pour calculer u_100
    u=f(u)                  # on se sert de la fonction f qu'on a créée au-dessus
print(u)                    # on affiche le terme u_100
```