Définitions de fonctions

Après avoir écrit un script pour calculer quelque chose, on aimerait bien pouvoir réutiliser ce script pour d'autes calculs, comme dans l'exercice sur la suite de Syracuse du dernier TP. On pourra alors utiliser des *fonctions*. Elles se comportent comme les fonctions déjà définies dans Python, comme `print` ou `sqrt`.

On définit les fonctions avec la syntaxe suivante :

```
def nom_de_la_fonction(argument1,argument2,...):
    """
    Documentation de la fonction
    """
    instructions
    return(valeur1,valeur2,...)
```

Le choix du nom est soumis aux même règles que les noms de variables. Une fonction peut prendre autant d'arguments que nécessaire, mais ne peut renvoyer qu'une seule valeur (comme les fonctions en mathématiques). En revanche, cette valeur peut être un couple, un triplet, *etc.*

Exercice 1. Écrire une fonction qui prend en paramètre un réel, et qui renvoie son carré.

Exercice 2. Écrire une fonction qui prend en paramètre un réel x et un entier n, et qui renvoie le n-ième terme de la suite géométrique de premier terme 1 et de raison x.

Pour utiliser une fonction, il suffit de l'appeler comme les fonctions déjà connues, en tapant `nom_de_la_fonction(argument1,...)`.

On peut accéder à la documentation d'une fonction en tapant `help(nom_de_la_fonction)`.

Exercice 3. Quelle est la documentation de la fonction `sqrt` ?

Toutes les variables crées puis modifiées dans la fonction sont des variables *locales* ; elles sont détruites dès que l'appel à la fonction est terminé.

Pour pouvoir modifier des variables définies dans le script globalement, il faut les déclarer au début de la fonction, avec la commande `global nom_de_variable`.

Exercice 4. On considère la fonction suivante :

```
def f(n):
    x=n
    return(x+1)
```

Définir cette fonction, puis appeler `f(1)`. Que contient la variable `x` ?

Rajouter l'instruction `global x` au début de la fonction, puis répondre aux mêmes questions.

Exercice 5. Définir une variable n valant 0, puis définir une fonction qui calcule le cube d'un réel, et qui affiche le nombre de fois que la fonction a été appelée.

Une particularité des fonctions est qu'elles peuvent s'appeler elles-mêmes : on appelle cette propriété la récursivité. On peut par exemple calculer la factorielle avec

```
def factorielle(n):
  if n==0:
    return(1)
  else:
    return(n*factorielle(n-1))
```

Il faut évidemment faire attention à ce que le calcul de la fonction puisse s'arrêter à un moment, sans quoi on rentrera dans une boucle infinie.

Exercice 6. Réécrire la fonction de l'exercice 2 en utilisant une fonction récursive.

Exercice 7. Écrire une fonction récursive calculant le n-ième terme de la suite de Syracuse.

Exercice 8. Une façon rapide de calculer la puissance d'un nombre est d'utiliser l'*exponentiation rapide*. Cela repose sur la propriété suivante :

$$x^n = \begin{cases} \left(x^2\right)^{\frac{n}{2}} & \text{si } n \text{ pair} \\ x\left(x^2\right)^{\frac{n-1}{2}} & \text{sinon} \end{cases}$$

Écrire une fonction qui prend en paramètres un complexe z et un entier n, et qui calcule z^n par exponentiation rapide.