TP14

# Définition d'une fonction sur Scilab

Une **fonction** est un groupe d'instructions en langage Scilab qui sont placées dans un même bloc débutant par un "`function` sortie=nom(entree)" et se terminant par un "`endfunction`". Elle se présente sous la forme suivante :

```
function sortie=nom(entree)
    instructions
endfunction
```

où *nom* est le nom que l'on donne à la fonction et qui permet de l'appeler dans la console, *entree* représente les arguments à mettre en entrée de la fonction et *sortie* les arguments en sortie.

Pour définir une fonction, on utilisera l'éditeur de Scilab. Après avoir exécuté le fichier, la fonction est enregistrée par Scilab et il suffit pour l'utiliser de l'appeler par son *nom* dans la console, comme on appelle une fonction Scilab :

`-->nom(`*entree*`)`

Voyons tout de suite un premier exemple de fonction :

```
function res=exemple(n)
    res=(n+1)^2
endfunction
```

Lorsqu'il n'y a pas d'erreurs de syntaxe grossières dans la fonction entrée, Scilab affiche simplement dans la console lors de l'exécution un message du type :

```
-->exec('C:\ECE1\TP\TP13-Définition d'une fonction sur Scilab\exemple.sce', -1)
```

S'il y a une erreur, Scilab affiche un message pour la signaler et il précise en général la ligne où elle se trouve.

On peut alors utiliser la fonction que nous venons de définir comme suit dans la console :

```
-->exemple(5)
 ans  =
 
    36.
```

On remarque que Scilab renvoie alors la valeur de la variable *sortie* obtenue après exécution des instructions qui composent la fonction.

Nous avons ainsi deux possibilités pour créer une procédure sur Scilab :

- Soit en utilisant une suite d'instructions et les commandes `input` et `disp` pour pouvoir interagir avec l'utilisateur. C'est ce que nous avons fait depuis le début d'année. Par exemple, pour calculer $\sum_{k=1}^{n} k^2$ :

```
n=input('Donner une valeur de n: ')
S=0
for k=1:n do
    S=S+k^2
end
disp(S)
```

- Soit en définissant une fonction dans l'éditeur de Scilab, puis en exécutant le fichier et enfin en appelant la fonction dans la console. Par exemple, toujours pour calculer $\sum_{k=1}^{n} k^2$ :

```
function S=somme(n)
    S=0
    for k=1:n do
        S=S+k^2
    end
endfunction
```

**Remarque.** Cette deuxième méthode (définir une fonction) présente plusieurs avantages. On peut utiliser plusieurs fois la fonction dans la console sans avoir à exécuter le fichier depuis l'éditeur à chaque fois. De plus, une première fonction peut être utilisée dans une seconde en faisant appel à elle simplement par son nom.

**Exercice 1** 1. Construire une fonction `discriminant` qui, étant donné trois réels $a, b, c$, retourne le discriminant du polynôme $P(x) = ax^2 + bx + c$.

2. Construire une fonction `maximum` qui, étant donnés deux nombres réels $a$ et $b$, retourne le maximum de $a$ et $b$.

3. Construire une fonction `sommedouble` qui, étant donné un entier naturel $n$, retourne la valeur de la somme double $\displaystyle\sum_{1 \leq i \leq j \leq n} ij$.

**Exercice 2** Soit $(u_n)_{n\in\mathbb{N}}$ la suite définie par :

$$u_0 = 1 \quad \text{et} \quad \forall n \in \mathbb{N},\ u_{n+1} = \frac{u_n}{1 + u_n^2}.$$

1. Construire une fonction `u` qui, étant donné un entier naturel $n$, retourne la valeur de $u_n$.
   Calculer plusieurs termes de la suite $(u_n)_{n\in\mathbb{N}}$ et émettre une conjecture sur sa convergence.

2. (a) Montrer que, pour tout $n \in \mathbb{N}$, $u_n \geq 0$.
   (b) Montrer que la suite $(u_n)_{n\in\mathbb{N}}$ est monotone.
   (c) En déduire que la suite $(u_n)_{n\in\mathbb{N}}$ converge et déterminer sa limite $\ell$.

3. Construire une fonction `vitesse` qui, étant donné un réel $\varepsilon$, permet d'obtenir le plus petit entier naturel $n$ tel que $|u_n - \ell| \leq \varepsilon$. Cette fonction fera appel à la fonction `u`.
   Tester pour $\varepsilon = 10^{-1}$, pour $\varepsilon = 10^{-2}$. La suite converge-t-elle rapidement vers sa limite ?

**Exercice 3** 1. Construire une fonction `de` qui simule le lancer d'un dé équilibré et affiche le numéro obtenu. On utilisera les commandes :

- `rand` qui permet de générer de manière aléatoire un nombre réel compris strictement entre 0 et 1,
- `floor` qui permet d'obtenir la partie entière d'un nombre réel.

2. Construire une fonction `doublesix`, utilisant la fonction `de`, qui simule le lancer de deux dés équilibrés et retourne 1 si on a obtenu un double 6 et 0 sinon.

3. On effectue $n$ lancers de deux dés équilibrés. Construire une fonction `nombre` qui, étant donné un entier $n \geq 1$, simule l'expérience et affiche le nombre d'apparitions d'un double 6 au cours des $n$ lancers. Cette fonction utilisera la fonction `doublesix`.

4. On lance deux dés équilibrés jusqu'à obtenir un double 6. Construire une fonction `rang`, utilisant la fonction `doublesix`, qui simule l'expérience et affiche le rang d'apparition du premier double 6.

**Exercice 4** Soit $a \in \mathbb{N}^*$. On définit la suite de Syracuse associée à l'entier $a$ par :

$$v_0 = a \quad \text{et} \quad \forall n \in \mathbb{N},\ v_{n+1} = \begin{cases} \dfrac{v_n}{2} & \text{si } v_n \text{ pair,} \\ \dfrac{3v_n + 1}{2} & \text{si } v_n \text{ impair.} \end{cases}$$

1. Compléter la fonction `Syracuse` suivante pour que, étant donné $a$ et $n$, elle retourne $v_n$ :

```
function res=Syracuse(a,n)
    res=...
    for k=1:n do
        if floor(res/2)==res/2 then
            res=...
        else
            res=...
        end
    end
endfunction
```

2. Recopier et exécuter les commandes suivantes pour différentes valeurs de $a$ :

```
a=input('Donner une valeur de a : ')
for n=0:20 do
    disp(Syracuse(a,n))
end
```

Émettre une conjecture sur la suite de Syracuse.

3. On appelle temps de vol le plus petit entier $n$ tel que $v_n = 1$.

   Construire une fonction `tempsdevol` qui, étant donné $a \in \mathbb{N}$, calcule et affiche le temps de vol de la suite de Syracuse associée à $a$. Cette fonction fera appel à la fonction `Syracuse`.

*L'hypothèse mathématique selon laquelle la suite de Syracuse associée à n'importe quel* $a \in \mathbb{N}^*$ *atteint* 1 *remonte à 1928. Elle n'a depuis jamais été démontrée malgré de nombreuses recherches.*