CH 2 : Premiers pas vers la programmation

I. Introduction à la notion de programme

Le programme officiel de première année évoque un « enseignement annuel d'informatique et d'algorithmique »[1]. Commençons donc par définir brièvement les termes utilisés.

- Un algorithme est une suite (on parle aussi de séquence) **finie** d'instructions permettant de résoudre un problème particulier. Ce terme d'algorithme provient directement du nom Al Khwarizmi, mathématicien persan de la fin du IX ème siècle.
- L'informatique est la science de l'information et notamment son traitement automatisé. L'aspect qui nous intéresse est celui de la programmation *i.e.* l'écriture de programmes. Un programme informatique peut être vu comme la traduction, dans un langage compréhensible par un ordinateur, d'un algorithme.

On peut illustrer la différence entre algorithme et programme à l'aide de l'algorithme d'Euclide, introduit en cours de mathématiques. Sans entrer trop dans les détails, il s'agit, étant donné deux entiers a et b, d'effectuer une succession finie de divisions euclidiennes permettant de calculer le PGCD de a et de b. Un programme est l'implémentation de cet algorithme dans un langage informatique. Coder cet algorithme dans un langage de calculatrices c'est écrire un programme. Coder cet algorithme en **Scilab** c'est écrire un autre programme. Ces deux programmes sont différents mais sont chacun l'expression, dans un langage particulier, du même algorithme.

Ceux qui souhaitent aller plus loin peuvent consulter cet article .

II. Séquence de commandes

La première étape vers l'écriture de programmes **Scilab** consiste donc à pouvoir écrire des séquences de commandes. Jusqu'à présent, chaque commande tapée a été directement validée. Pour écrire directement une succession de commandes, il suffit de taper ces commandes à la suite, en les séparant par une virgule « , ».

```
--> sqrt(25),exp(0),%e-exp(1),floor(3.7)>3,(%e>2)&(%e<3)

   ans  =
       5.

   ans  =
       1.

   ans  =
       0.

   ans  =
       F

   ans  =
       T
```

Il existe un autre séparateur de commande : le point-virgule « ; ». Lorsqu'une commande est suivie de « ; », son résultat est calculé mais non affiché.

```
--> sqrt(25);

--> sqrt(25); exp(0)
   ans  =
       1.

--> sqrt(25); exp(0);
```

[1] En 2ème année, la formulation est moins ambitieuse : le programme évoque des « TP de mathématiques avec **Scilab** »

```
--> sqrt(25); exp(0), %e-exp(1)

   ans  =
       1.

   ans  =
       0.

--> sqrt(25), exp(0); %e-exp(1);

   ans  =
       5.
```

Il n'est pas évident, sur ces premiers exemples, que le calcul a bien lieu. Pour s'en rendre compte, il faudrait que le résultat d'un premier calcul, non affiché, soit utilisé dans un deuxième calcul que l'on souhaite afficher. Pour ce faire, on peut stocker les premiers calculs dans des variables `x` et `y` (*cf* III) et les utiliser dans un calcul suivant.

```
--> x=2, y=x+5, z=sqrt(x+y)^x+4
   x  =
       2.
   y  =
       7.
   z  =
       13.

--> x=2; y=x+5; z=sqrt(x+y)^x+4
   z  =
       13.
```

On notera que, même si le résultat de la commande `y=x+5` n'est pas affiché, le calcul est bien effectué et `y` est affecté à la valeur `7`. En résumé, l'utilisation du séparateur « ; » permet d'éviter l'affichage d'informations / calculs intermédiaires.

III. Les variables

III.1. Introduction : les variables en informatique

Les variables sont un outil essentiel en programmation. Cette notion s'articule autour de 3 principes fondamentaux :

- **stockage** : pouvoir stocker de l'information,
- **accès** : pouvoir accéder au contenu du stockage,
- **modification** : pouvoir modifier l'information stockée.

Dans la pratique, une variable est repérée par son **nom**. À ce nom est associé une **valeur** qui peut être modifiée. Comme en mathématiques, le simple fait de nommer les objets permet d'effectuer des calculs plus ambitieux.

III.2. Les variables en Scilab

III.2.a) Le mécanisme d'affectation

Créer une variable, c'est associer son nom à sa valeur. Ce mécanisme d'association est appelé **affectation** et utilise le symbole d'égalité `=`, appelé de ce fait **opérateur d'affectation**. Plus précisément, on utilisera la syntaxe suivante :

```
variable = expression
```

- `variable` est le nom que l'on souhaite donner à la variable que l'on crée ;
- `expression` représente soit directement une valeur soit un calcul permettant le calcul d'une valeur. Ce calcul pourra se faire à l'aide d'autres variables.

Définir une variable c'est donc lui associer une valeur initiale. Cette valeur peut être modifiée par la suite en utilisant la même syntaxe que pour la première définition.

Illustrons l'ensemble de ces procédés à l'aide d'appels console.

Une variable peut être associée à une valeur ou un calcul qui peut dépendre d'autres variables.

```
--> x=2
    x  =
        2.

--> y=7-2
    y  =
        5.

--> z=x+y*(x+5)-7
    z  =
        30.
```

Une variable peut être mise à jour à l'aide de sa précédente valeur.

```
--> z=z/2-5
    z  =
        10.
```

On peut demander l'affichage de la valeur d'une variable, si cette variable a été définie précédemment.

```
--> t
  !--error 4
Variable non définie : t

--> x, y, z
    x  =
        2.
    y  =
        5.
    z  =
        10.
```

Attention ! Il ne faut en **aucun cas** confondre l'opérateur d'affectation « = » avec l'opérateur de test d'égalité, noté « == ».

```
--> x==z
    ans  =
        F

--> x=z
    x  =
        10.

--> x==z
    ans  =
        V
```

III.2.b) La variable *ans*

Lors des appels console, vous avez dû remarquer l'affichage du mot *ans* à chaque fois qu'une commande est évaluée. Il s'agit en fait d'une variable par défaut qui permet de stocker la valeur du dernier résultat calculé. Le terme *ans* n'est autre que l'abréviation du mot anglais *answer*.

```
--> 3+4
    ans  =
        7.
--> ans=ans+3
    ans  =
        10.
--> ans==10
    ans  =
        T
--> ans
    ans  =
        T
```

III.2.c) Nom des variables

Dans les précédents exemples, nous avons utilisé essentiellement les lettres `x`, `y`, `z` pour nommer les variables. **Scilab** est en fait assez peu restrictif : une variable peut prendre comme nom n'importe quelle combinaison de lettres (en majuscule ou minuscule), de chiffres et de symboles « `%` », « `_` », « `#` », « `!` », « `$` », « `?` ». Ce qui signifie que tout autre symbole est interdit.

On respectera les règles suivantes :

1) il est interdit d'utiliser des symboles d'opérateur tels que `+`, `*`, `-`, `&`, `|`. Tentons par exemple de nommer une variable « `res-1` ».

```
-->res-1=3
  !--error 4
Variable non définie : res
```

Un message d'erreur apparaît : en effet, **Scilab** a interprété le nom « `res-1` » comme la commande permettant de soustraire `1` à la variable `res`, non précédemment définie.

2) par convention, on n'utilisera pas le symbole `%` comme premier symbole d'une variable. En effet, ceci est réservé aux constantes prédéfinies de **Scilab**. Il n'est d'ailleurs pas possible de redéfinir ces constantes.

```
-->%pi=2
  !--error 13
Redéfinition d'une variable permanente
```

3) on utilisera des noms **explicites** pour les variables. Il s'agit d'une règle de bonne conduite essentielle pour s'y retrouver lorsqu'on code des programmes de grande taille.

4) on utilisera, de préférence, des noms **courts** pour les variables. Ceci permet de simplifier l'écriture de programmes dans lesquels apparaissent plusieurs fois un même nom de variable.

Pour bien comprendre ce règles, voici quelques noms de variables que l'on pourra utiliser par la suite.

- `res` : pour désigner le résultat du calcul principal,
- `aux` : pour désigner une variable auxiliaire, autrement dit une variable contenant le résultat d'un calcul intermédiaire,
- `nbMax` : en accord avec la règle *4)*, on préférera utiliser les noms `nbMax` voire `nmax` à un nom comme `nombreMaximum` pour désigner une variable fixant la valeur maximale qu'une autre variable peut atteindre,
- ...

De manière générale, on peut prendre comme convention d'utiliser des abréviations d'un ou de plusieurs mots, en faisant commencer chaque nouveau mot par une majuscule.