TP 1 – Présentation de Scilab

Dans ce TP, nous allons prendre connaissance avec le logiciel Scilab. Ce logiciel est gratuit et téléchargeable sur internet.

1 Utilisation de la console

Lancer Scilab. Dans la fenêtre de la console apparaît une invite de commande `-->`. Scilab attend une instruction.

1.1 Calcul

Exercice 1. 1. Ecrire les instructions suivantes dans la console, qu'en conclure ?

```
1+1
ans+1
1+9*8
3^2
9.5+2
1/3*3
3.54D+3
1+1.e-16
ans-1
```

```
2+3*%i
3%e
3*%e
sin0
sin(0)
sin(%pi/2)
cos(%pi/2)
tan(%pi/8)
ln(12)
```

2. Taper au clavier `help log` puis calculer `log(12)` et `exp(2)`.
3. On veut calculer la racine carrée de 127. Chercher l'instruction à utiliser dans le navigateur d'aide (touche $F1$).

Remarque 1.1.

1. De manière générale, il est essentiel de savoir se servir de l'aide dans un logiciel de programmation. Plutôt que tenter au hasard de nombreuses instructions, il faut prendre le réflexe de regarder les nombreux exemples fournis dans l'aide.
2. Les nombres réels considérés par un logiciel de programmation sont des nombres décimaux ayant 16 chiffres significatifs. Ainsi la valeur de π donnée est une valeur approchée. De même, si l'on obtient un résultat égal à 10^{-16}, on peut considérer que c'est 0.
3. Pour les puissances de 10, plusieurs notations sont possibles : `D`, `d`, `e` ou `E`.

Exercice 2. Prévoir la réponse de Scilab aux commandes suivantes. Vérifier ensuite en exécutant ces commandes dans la console.

```
2*3
a=2
a
a=3 ;
a
c=a ;
```

```
c , a
a=1 , b=5
b = a+6
a = 2*a
a = a+2*b
who
```

1.2 Boucle for

La boucle `for` sert à exécuter un nombre prédéfini de fois une instruction ou une suite d'instructions. Sa syntaxe générale est :

```
for variable = valeur_depart : pas : valeur_fin
instructions
end
```

La valeur `valeur_fin` est une valeur non dépassée. Le pas est facultatif et vaut 1 par défaut.

Exercice 3. Prévoir la réponse de Scilab aux commandes suivantes. Vérifier ensuite en exécutant ces commandes dans la console.

Algorithme 1 :

```
--> for x=1:5
--> x
--> end
```

Algorithme 2 :

```
--> y=1
--> for x=0:3:12
--> y=y+x;
--> end
--> y
```

Exercice 4.
1. On considère la suite $(u_n)_{n\in\mathbb{N}}$ définie par $u_0 = 1$ et pour tout $n \in \mathbb{N}$, $u_{n+1} = 2u_n - 3$. A l'aide d'une boucle `for` , calculer u_{18}.
2. On considère la suite $(v_n)_{n\in\mathbb{N}}$ définie par $v_0 = v_1 = 1$ et pour tout $n \in \mathbb{N}$, $v_{n+2} = v_{n+1} + v_n$. A l'aide d'une boucle `for` , calculer v_{72}.
3. En se ramenant à un problème de calcul d'un terme d'une suite récurrente, calculer 7!, puis calculer la somme des carrés des 20 premiers entiers naturels non nuls.

2 Scinotes

Lorsque l'on effectue des manipulations complexes il peut devenir fastidieux de recopier plusieurs séquences d'expressions. On peut alors écrire un bloc d'expressions dans un fichier et l'exécuter en une fois : c'est un script. On peut aussi faire dépendre ce script d'arguments, et lui spécifier des objets en sortie : ce sera alors une fonction.

Pour écrire ce fichier, on utilise SciNotes, qui est un traitement de texte adapté à la programmation sous Scilab. Il est accessible dans Accessoires/SciNotes. Le fichier est sauvegardé avec le suffixe `.sce` ou `.sci` pour indiquer au système que c'est un fichier Scilab. Pour assurer une compatibilité entre les divers systèmes d'exploitations, les noms de fichiers doivent être en un seul mot (pas d'espace), sans accent, sans tiret et sans point.

2.1 Scripts

Un script est un fichier contenant une suite d'instructions Scilab, chaque instruction étant disposée sur une ligne. Il vaut mieux donner pour extension `.sce` à son fichier. Dans SciNotes, il suffit de presser la touche F5 pour que Scilab exécute le script intégralement.

Un élément important des scripts (et plus tard des fonctions) est les commentaires : tout ce qui est placé après un "double slash" ne sera pas lu par Scilab. C'est très utile pour éclaircir le sens d'une instruction compliquée !

Exemple 2.1. Le script suivant calcule le produit et la différence de deux nombres, et les enregistrent dans deux variables `p` et `d`.

```
// Calcule le produit et la différence de a et de b
a = 3;
b = 7; // Ceci n'est pas lu par Scilab.
p = a*b;
d = a-b;
```

2.2 Fonctions

Une fonction dans Scilab est l'analogue d'une fonction mathématique. Cela permet, à partir d'arguments (nombres, matrices, booléens, chaînes de caractères, etc...) de calculer une ou plusieurs sorties. On écrit une fonction dans un fichier, en lui donnant de préférence l'extension `.sci`. On devra respecter la syntaxe suivante.

```
function sortie=nom(arguments)
  // Commentaire qui explique et décrit la fonction
  Expression qui calcule la sortie en fonction des arguments
endfunction
```

Une fois la fonction écrite et en utilisant SciNotes, on la charge dans Scilab en pressant la touche F5. On peut ensuite l'utiliser dans la console de Scilab ou dans d'autres scripts ou fonctions.

Exemple 2.2. Ecrire la fonction suivante en utilisant Scinotes. Que fait-elle ?

```
function f=mystere(n)
    f=1;
    for i=1:n
    f=f*i;
    end
endfunction
```

Après avoir chargé la fonction dans Scilab, tester les commandes suivantes :

```
--> mystere(4)
--> mystere(5)
--> mystere(10)
```

Une fonction peut avoir plusieurs arguments, on les sépare alors par une virgule. De même il peut y avoir plusieurs sorties, dans ce cas on les sépare par des crochets.

Exemple 2.3. On peut coder la fonction suivante, qui associe à deux nombres leur somme et leur différence.

```
function [s,d]=somdiff(a,b)
   //  Calcule s : la somme de a et de b et d : la différence de a et de b
   s = a+b;
   d = a-b;
endfunction
```

Après avoir chargé la fonction dans Scilab, tester les deux commandes suivantes. Que remarque-t-on ?

```
--> somdiff(3,4)
--> [u,v] = somdiff(3,4)
```

Exercice 5. Écrire une fonction `diviseurs(n)` qui, à un entier $n \in \mathbb{N}^\star$, renvoie la liste des diviseurs de ce nombre puis une autre `parfaits(n)` qui donne la liste des nombres parfaits inférieurs à n.

Exercice 6.

1. On reprend les suites $(u_n)_{n\in\mathbb{N}}$ et $(v_n)_{n\in\mathbb{N}}$ de l'exercice précédent. Ecrire une fonction `calculu(n)` et une fonction `calculv(n)` qui prennent un entier $n \in \mathbb{N}$ en entrée et renvoient respectivement la valeur de u_n et v_n. Retrouver les valeurs calculées à l'exercice précédent.
2. Ecrire une fonction `harmonique(n)` qui prend un entier $n \in \mathbb{N}^\star$ en entrée et qui renvoie la valeur de $\displaystyle\sum_{k=1}^{n} \frac{1}{k}$.
3. On admet que $\displaystyle\lim_{n\to+\infty} \frac{1}{\ln n} \sum_{k=1}^{n} \frac{1}{k} = 1$. Proposer une fonction Scilab qui permet de vérifier empiriquement ce résultat.

Exercice 7.

1. Définir une fonction qui pour un entier $n \in \mathbb{N}^\star$, renvoie la valeur de $S_n = \displaystyle\sum_{k=1}^{n} k^3$.
2. Définir une fonction qui pour un entier $n \in \mathbb{N}^\star$, renvoie la valeur de $T_n = \displaystyle\sum_{k=1}^{n} k$.
3. Vérifier empiriquement que $S_n = (T_n)^2$.

Exercice 8. On considère deux suites $(u_n)_{n\in\mathbb{N}}$ et $(v_n)_{n\in\mathbb{N}}$ définies par $u_0 = 1$, $v_0 = 2$ et $\forall n \in \mathbb{N}$, $u_{n+1} = \frac{u_n+v_n}{2}$, $v_{n+1} = \sqrt{u_n v_n}$.

1. Vérifier que ces deux suites sont bien définies.
2. Ecrire un programme qui prend en entrée un entier $n \in \mathbb{N}$ et qui renvoie le couple (u_n, v_n).
3. Que valent u_{30} et v_{30} ?

Exercice 9.

1. Tester la commande `zeros(5,10)`.
2. Ecrire une fonction Scilab qui prend un entier $n \in \mathbb{N}^\star$ en entrée et renvoie la liste des k^2 pour $k \in [|1, n|]$.
3. Ecrire une fonction Scilab qui prend un entier $n \in \mathbb{N}^\star$ en entrée et renvoie un tableau à n lignes et n colonnes rempli de 0 sauf sur les deux diagonales où il y a des 1.

3 Boucles While, If-Then-Else

En programmant avec Scilab, on peut vouloir exécuter une instruction seulement si une condition est vérifiée. Pour cela on utilise la structure If-Then-Else. La syntaxe générale est la suivante :

```
if condition then
   instructions ;
end
```

On peut demander à Scilab d'éxecuter une autre instruction si la condition n'est pas vérifée. Pour cela on utilise `else`.

```
if condition then
   instructions1 ;
else
   instructions2 ;
end
```

Il existe aussi une instruction `elseif` pour enchaîner sur une autre condition.

La boucle `while` sert a exécuter un nombre non défini de fois une instruction ou une suite d'instructions. Sa syntaxe générale est

```
while test
  instructions
  ....
end
```

Remarque 3.1.

— Par souci de lisibilité, on indente les boucles. On les ferme toujours avec `end`.

— Il faut faire attention à ne pas avoir de boucle infinie. Si le critère d'arrêt `test` est toujours vérifié, Scilab va répéter les instructions indéfiniment. Il convient donc de vérifier que le programme va s'arrêter avant de le tester. Pour interrompre l'exécution d'un programme, appuyer simultanément sur les touches Ctrl et C.

Exercice 10. Ecrire une fonction `partieentiere(x)` qui détermine la partie entière du réel x.

Exercice 11.

1. On définit la suite $(u_n)_{n\in\mathbb{N}}$ par $u_0 = 1$ et pour tout $n \in \mathbb{N}$, $u_{n+1} = 2u_n$ si n est pair et $u_{n+1} = 3u_n$ si n est impair. Calculer u_{100}.
   On pourra introduire une fonction `calculdeu(n)` et utiliser la fonction `floor`.
2. On définit la suite $(v_n)_{n\in\mathbb{N}^\star}$ par $v_1 = 1$ et pour tout $n \in \mathbb{N}^\star$, $v_{n+1} = \frac{2v_n}{n}$. Alors on peut prouver que $\lim\limits_{n\to+\infty} v_n = 0$.
   Ecrire une fonction `calculrang(eps)` qui prend un réel $\varepsilon > 0$ en argument et renvoie le premier $n \in \mathbb{N}$ tel que $v_n < \varepsilon$.

Exercice 12. On considère la suite $(w_n)_{n\in\mathbb{N}^\star}$ définie par $w_1 = 1$ et $w_{n+1} = \frac{1}{n}w_n + 1$. Créer une fonction `calculdew(n)` qui permet de calculer w_n pour tout $n \in \mathbb{N}^\star$.

Modifier la fonction `calculdew` de façon à gérer les situations où le nombre n entré n'appartient pas à $\mathbb{N}^\star$.

Exercice 13 (Algorithme de Syracuse)**.** On choisit un entier naturel quelconque. S'il est pair, on le divise par 2, sinon on le multiplie par 3 et on ajoute 1. En répétant ce processus, on constate qu'on aboutit à 1 (actuellement personne n'a su le démontrer).

1. Ecrire la suite en partant de 7.
2. Ecrire une fonction `syracuse(n)` qui affiche le nombre suivant le nombre n dans la liste de Syracuse. Compléter ce programme de façon à lui faire effectuer la boucle complète jusqu'à arriver à 1. On pourra utiliser la fonction `disp`.

Exercice 14 (D'après EDHEC)**.** On considère la suite $(u_n)_{n\in\mathbb{N}}$ définie par $u_0 = 1$ et pour tout $n \in \mathbb{N}$, $u_{n+1} = \frac{e^{-u_n}}{u_n}$.

Que renvoient les scripts suivants ? Que sait-on de u_5 et u_6 ? Quelle conjecture peut-on émettre sur le comportement de la suite $(u_n)_{n\in\mathbb{N}}$?

```
u = 1
n = 0
while u > 0.00001
u = exp(-u)/u
n=n+1
end
disp(n)
```

```
u = 1
n = 0
while u < 100000
u = exp(-u)/u
n=n+1
end
disp(n)
```

Exercice 15. Créer une fonction `euclide(n,d)` qui affiche le quotient et le reste de la division euclidienne de $n \in \mathbb{N}$ par $d \in \mathbb{N}^\star$.

On rappelle que le quotient q peut être défini comme le plus grand entier k tel que $kd \leq n$ et que le reste est $r = n - qd$.

On pourra mettre en place un garde-fou traitant le cas $d = 0$.

Exercice 16. Écrire une fonction `saisiecontrolee()` qui demande la saisie d'un nombre n compris entre 1 et 100. On procède en 3 temps :

— saisie au clavier du nombre n.

— tant que ce nombre n'est pas conforme, répéter la saisie,

— afficher le message "saisie terminée".

On pourra utiliser les fonctions `disp` et `input`.

Exercice 17. Ecrire une fonction `justeprix()` qui choisit au hasard un entier entre 0 et 1000 et demande à l'utilisateur de le deviner. Si l'utilisateur se trompe, le programme doit annoncer "trop grand" ou "trop petit" et demander un nouveau nombre.

En plus des fonctions définies au exercices précédents, on pourra utiliser la fonction `rand`.