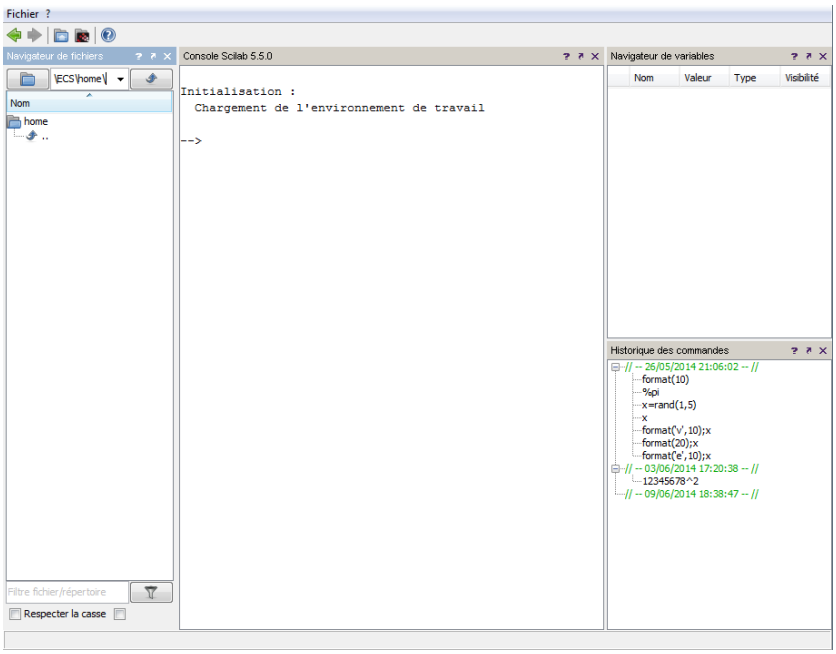
TP1 : Introduction à *Scilab*

1. Présentation du logiciel

Né au début des années 90, Scilab (de Scientific Laboratory) est un logiciel de calcul numérique crée par l'INRIA (Institut national de recherche en informatique et en automatique) et l'ENPC (École nationale des ponts et chaussées). Ce logiciel est utilisé pour des applications scientifiques très variées : simulation en probabilités, statistiques, traitement du signal et de l'image, systèmes dynamiques,... et il possède de plus l'avantage d'être un logiciel gratuit et entièrement libre depuis sa version 5.0.
Pour plus d'informations :
http://www.scilab.org/scilab/about
http://fr.wikipedia.org/wiki/Scilab

2. Lancement de *Scilab*

Lancer *Scilab* à partir de l'icône  . Une fenêtre composée de quatre zones (sous fenêtres) apparaît :



— **Le navigateur de fichiers** (à gauche) : permet principalement d'aller chercher des fichiers auparavant sauvegardés et de naviguer dans les différents répertoires.

— **La console** (au centre) : il s'agit de la plus importante des quatre fenêtres, c'est là que l'utilisateur peut interagir avec *Scilab* en tapant des commandes spécifiques. Celles-ci doivent être entrées après l'invite de commande : `-->`

— **L'historique des commandes** (bas-droite) : comme son nom l'indique, cette fenêtre permet de retrouver les commandes entrées par l'utilisateur, elles sont regroupées par dates.

— **Le navigateur de variables** (haut-droite) : très utile, cette fenêtre permet de voir le nom des variables déjà affectées ainsi que leur valeur. Cela pourra s'avérer utile lors de la gestion de certaines erreurs.

3. Premiers pas (le mode calculatrice - les opérations élémentaires)

Il est possible d'utiliser *Scilab* comme une calculatrice évoluée.

Exercice 1 :
Taper les commandes suivantes et noter les résultats et différences (pensez aussi à observer les deux fenêtres de droites - navigateur de variables et historique de commandes - et les modifications qui s'y opèrent) :

```
-->5+7
-->5+7;
-->2+3*5.2^10
-->2+3*5,2^10
```

Nous pouvons d'ores et déjà remarquer plusieurs choses :

— Il est possible de demander à *Scilab* d'effectuer un calcul sans en afficher le résultat, en ajoutant un ';' en fin de ligne. Cette nuance n'a pour l'instant que peu d'intérêt mais elle sera appréciable lors de l'exécution d'un programme censé calculer plusieurs milliers (millions ?) d'opérations, l'affichage des résultats intermédiaires peut ralentir de manière très significative le processus.

— *Scilab* peut recevoir plusieurs calculs dans une même ligne de commande à condition de les séparer par une virgule. Les résultats sont alors affichés dans l'ordre de saisie. Le point, quant à lui, fait office de virgule.

— Les priorités opératoires semblent respectées (et c'est en réalité bien le cas).

— lorsque *Scilab* renvoie un résultat c'est sous la forme $'ans ='$. ans (pour *answer*) est le nom d'une variable temporaire de *Scilab* dans laquelle est stockée le résultat du dernier calcul effectué mais non affecté par l'utilisateur. On peut aussi voir cette variable dans la fenêtre Navigateur de variables.

— La fenêtre Historique des commandes s'est remplie au fur et à mesure. Il est possible de réexécuter l'une des commandes de cette fenêtre en double cliquant dessus (Faites-en l'essai). Une autre méthode consiste à appuyer successivement sur la touche $\boxed{\uparrow}$ du clavier pour rappeler les dernières commandes exécutées (Faites-en l'essai).

Opérations usuelles :

Addition :	`+`	Division :	`/`
Soustraction :	`-`	Puissance :	$\widehat{\ }$
Multiplication :	`*`		

Fonctions usuelles :

Racine carrée :	`sqrt()`	Cosinus :	`cos()`
Module :	`abs()`	Sinus :	`sin()`
Partie entière :	`floor()`	Tangente :	`tan()`
Logarithme népérien :	`log()`	Arccos :	`acos()`
Logarithme en base 10 :	`log10()`	Arcsin :	`asin()`
Exponentielle :	`exp()`	Arctan :	`atan()`

Constantes usuelles :

π : `%pi`, e : `%e`, i : `%i`, Vrai : `%T`, Faux : `%F`

`%i` désigne le nombre complexe i. Le nombre complexe $5+3i$ sera entré dans la console *Scilab* de la manière suivante : `5+3*%i` (Faites-en l'essai)
`%T` et `%F` seront abordés au paragraphe 'Booléen et logique'.

L'aide *Scilab*
En cas de doutes sur l'utilisation ou l'existence d'une commande de *Scilab* vous pouvez taper dans la console : 'help *commande*'.
Essayez par exemple : `-->help rand`.
L'aide fournie par *Scilab* peut parfois paraître indigeste mais elle présente également de nombreux exemples très instructifs.

Exercice 2 :

(a) Pour chaque commande, prévoir sur feuille les réponses affichées par *Scilab*, puis vérifiez vos réponses à l'aide de la console :

```
-->sqrt(2^10)          -->sin(0)              -->log(%e)
-->sqrt(2^10);         -->cos(%pi)            -->floor(%e*%pi)
-->8/8*3               -->tan(3*%pi/4)        -->%e^(%i*%pi/4)
-->11-1/2              -->tan(3%pi/4)         -->floor(abs(3-2*%i))
```

(b) Taper les commandes *Scilab* qui permettent d'effectuer les calculs suivants :

$$\frac{e^{\pi}}{\pi^{e}}, \qquad 8\sqrt{5}-3\sqrt{7}, \qquad \left(\frac{2}{3}+\pi e\right)^{3}, \qquad \sqrt{\left|\sin\left(\ln\left(\frac{2^{10}}{e^{-9}}\right)\right)\right|}$$

4. Booléen et logique

Scilab peut effectuer des tests logiques sur des valeurs, dans ce cas il renvoie T (pour true - vrai) ou F (pour false - faux). Essayer les commandes suivantes :

```
-->2<3
-->%pi<%e
-->2==2.0
```

Principaux tests de bases :

inférieur :	`<`	inférieur ou égal :	`<=`	égal :	`==`
supérieur :	`>`	supérieur ou égal :	`>=`	différent :	`<>`

Connecteurs et négation :
Ces connecteurs ont la même signification qu'en mathématiques.

Et : `&`
Essayer par exemple : `-->(2<3)&(6==9)`

Ou : `|` (AltGr+6)
Essayer par exemple : `-->(2<3)|(6==9)`

Non : `~` (tilde : AltGr+2)
Essayer par exemple : `-->~(2==3)`

Exercice 3 : Prévoir les réponses de *Scilab* aux commandes suivantes.

```
-->(1>2)|(89)|(2<>5)
-->(abs(3+4*%i)>4)&((2==2)|(3==4))
-->~((%T)|(%F))
-->~((%T)&(%F))
-->~((%T)|((%F)&(abs(2-3*%i)>log(%e^4))))
```

5. Affectation de variables

Une variable peut être considérée comme un petit morceau de mémoire machine réservé par *Scilab*. Cette variable est repérée par un nom unique (sensible à la casse) et contient une valeur (réel, complexe, booléen, chaîne de caractères, vecteur,...)
Pour créer une variable dans *Scilab* il faut la déclarer et lui affecter une valeur en même temps grâce au symbole `=`, exemples :

```
-->a=8
-->z=5-3*%i
-->a*z
-->A='Bonjour tout le monde'
```

Taper les instructions précédentes et observer l'apparition de ces variables dans la fenêtre 'Navigateur de variables'.

Exercice 4 :
Exécuter les instructions suivantes. Quelles sont les valeurs finales des variables `a`, `b`, `texte1` et `texte2` ? Était-ce prévisible ?

```
-->a=%pi
-->a=%e
-->a
-->b=123
-->b=b+1
-->b
-->a=a+b
-->texte1='Bonjour '     //Attention à l'espace à la fin
-->texte1=texte1+'ceci est mon premier '
-->texte2=texte1+'TP Scilab !'
```

Remarques très importantes :

(a) le symbole `=` de *Scilab* n'a pas du tout la même signification que le symbole = des mathématiques ! Pour *Scilab* (et aussi pour la plupart des langages de programmation), écrire : `expression1=expression2`, signifie que l'on va transmettre et affecter à l'`expression1` la valeur de l'`expression2`. Dans ces conditions, lorsque l'on écrit `b=b+1`, on signifie à *Scilab* que l'on souhaite d'abord calculer la valeur de `b+1` puis transmettre cette valeur à la variable `b` qui va donc changer de valeur. `b` va donc être augmenté de 1 (on dit aussi *incrémenté*).
Par contre, en mathématiques, écrire : $b = b + 1$ est absurde puisqu'on aurait alors $0 = 1$.

(b) Pour des chaînes de caractères, le symbole `+` n'est évidemment pas l'addition, il s'agit de la *concaténation* : on colle les chaînes les unes aux autres.

(c) Pour affecter une chaîne de caractères à une variable il faut encadrer cette chaîne par des apostrophes ou des guillemets. Si la chaîne à affecter contient elle même des apostrophes ou des guillemets il faut alors doubler ces apostrophes ou ces guillemets, de plus, si la chaîne est trop longue, on peut la couper avec deux points consécutifs.

Tester et comprendre l'exemple suivant :

```
-->A='Affecter une chaine est ""simple"",..
-->je l''ai testé aujourd''hui'
```

(d) Pour effacer toutes les variables on utilise la commande `clear`.

Longueur et accès chaines :
Pour connaître le nombre exact de caractères d'une chaîne on peut utiliser la commande `length()` :

```
-->length('Bonjour tout le monde!')
```

Renverra 22 comme réponse.

Et si l'on souhaite savoir quel est le $10^{ème}$ caractère de la chaîne :

```
-->phrase='Bonjour tout le monde!'
-->part(phrase,10)
```

Renverra la lettre `o` du mot `tout`.

Exercice 5 :

(a) Créer une variable `A` contenant la chaîne de caractères suivante :
`'Les facultés de l'esprit qu'on définit par le terme analytiques '`

et une variable `B` contenant la chaîne :
`'sont en elles-mêmes fort peu susceptibles d'analyse.'`

(b) Concaténer `A` et `B` dans une variable `C`.

(c) Combien de caractères contient la chaîne `C` ? (Vous donnerez deux méthodes)

(d) Quelle est la centième lettre de la chaîne `C` ?

(e) Créer une commande qui renverra `T` ou `F` selon que la dixième et la centième lettre de `C` sont identiques ou pas.

6. Premiers programmes

Évidemment toutes ces commandes n'ont d'intérêt que si elles peuvent être enregistrées dans un fichier pour être ensuite exécutées automatiquement, sans avoir à retaper toutes les commandes et tout en donnant à l'utilisateur la possibilité de changer les valeurs de certaines variables.
Démarrer SciNotes (icône ) et taper les deux lignes du code suivant dans la nouvelle fenêtre :

```
1. a=input("Entrer un nombre : ")
2. disp(2*a, "Le double de ce nombre vaut :")
```

Exécuter ce code (icône  ou touche F5 - il faut sauvegarder au préalable) et observer.
`input` et `disp` sont deux fonctions fondamentales en *Scilab*.
La première permet une certaine interactivité avec l'utilisateur, elle affiche le message entre guillemets et attend la réponse de l'utilisateur puis elle réalise une affectation de cette valeur dans une variable.
La seconde permet d'afficher la valeur de certaine variable.
Pour plus d'informations :

```
-->help input
-->help disp
```

Exercice 6 :

(a) Réaliser un programme demande à l'utilisateur un nombre x et qui affichera en retour la valeur de $\sin(\ln(|x|+1))$.

(b) Réaliser un programme qui demande à l'utilisateur deux nombres qui seront affectés à des variables `a` et `b`, puis qui échange les valeurs de ces deux variables.

(c) Réaliser un programme qui demande un entier $n > 1$ à l'utilisateur et qui affiche ensuite un nombre entier au hasard entre 1 et n (loi uniforme sur $\{1, 2, \ldots, n\}$).
En déduire un simulateur de lancer d'un dé équilibré à 6 faces.