TP nº 1 - Introduction

Tous les sujets et les corrigés sont disponibles aux adresses suivantes :
`http://www.lif.univ-mrs.fr/~vpoupet/enseignement/matlab09.php`
`http://www.lif.univ-mrs.fr/~pvanier/?q=cours`

Exercice 1. *Premiers pas*

Matlab peut être vu comme une calculatrice extrèmement puissante. Les opérations simples peuvent être tapées directement, et l'on obtient le résultat en appuyant sur la touche "Entrée".

1. Essayez de faire quelques opérations dans l'interpréteur :

```
>> 5+5
>> 5*5
>> 5^5
>> 5/5
```

En réalité on peut faire bien plus que des petites opérations. Et pour ce faire, on va avoir besoin de la notion de *Variable* : Une variable permet de mémoriser un résultat et de le réutiliser par la suite, de manière à pouvoir automatiser certaines tâches. Afin d'*affecter* une variable, on utilise le signe $=$. Ainsi la ligne :

```
>> var1=3
```

doit être lue comme $var1 \leftarrow 3$ et non pas comme un test d'égalité.

2. Familiarisez-vous avec les variables :

```
>> var1 = 52
>> var1
>> var2 = 32;
>> var2
>> var1 * var2
>> Var1 = 12
>> var1
>> 11 = var1
```

3. A quoi sert le point virgule à la fin d'une ligne de commande ? Que remarque-t'on à propos de la gestion des majuscules/minuscules dans les noms de variables ?

Réponse : Le point virgule sert à ne pas afficher dans l'interpréteur la valeur calculée/affectée.

Pour effacer une variable, on peut se servir de la command `clear varname`, si on ne donne pas d'argument à `clear`, alors toutes les variables sont effacées.

4. Essayez par vous-même :

```
>> clear var1
>> var1
>> clear
```

Il existe une floppée de fonctions mathématiques comme par exemple `sin`, `log`, `sqrt` et bien d'autres encore. Par défaut, il y a également des variables qui sont prédéfinies comme `pi` ou `i` (nombre imaginaire pur) par exemple.

5. Testez les fonctions : `sin(pi/2)`, `sqrt(16)` ...

Exercice 2. *Informations utiles*

Vous ne connaitrez probablement jamais toutes les commandes de Matlab [1], mais ce n'est pas un problème, car vous pourrez retrouver toutes les informations nécéssaires facilement en vous servant de l'aide. Si vous vous rappelez d'une commande mais pas de comment on l'utilise, alors la commande `help commande` vous sera utile.

1. Regardez l'aide de quelques fonctions classiques :

```
>> help log
>> help mod
```

Si vous n'avez pas de nom de commande mais vous savez ce que vous cherchez, vous pouvez également utiliser la fonction recherche dde l'aide ou utiliser un des guides.

Exercice 3. *Tableaux*

Le nom "Matlab" veut dire laboratoire matriciel, et donc comme le nom l'indique, la base du logiciel sont les tableaux et les vecteurs. Un tableau permet de stocker plusieurs valeurs à la fois en pouvant acceder à chacune de manière positionnelle. Par exemple sur le tableau suivant, on peut acceder indépendamment à chaque valeur.

```
 1  2  3  4
 5  6  7  8
 9  10 11 12
 13 14 15 16
```

Pour déclarer ce tableau, il suffit de taper :

```
>> a=[1 2 3 4;5 6 7 8;9 10 11 12;13 14 15 16]
```

Puis pour accéder par exemple à la valeur dans la troisième ligne et dans la deuxième colonne, on tape `a(3,2)`. La séparation des éléments dans une ligne se fait soit par des virgules, soit par des espaces. La séparation des éléments dans une colonne se fait avec des points virgule. On peut également faire des tableaux à une "dimension" appelés vecteurs :

```
>> b=[1 2 3 4]
>> b=[1,2,3,4]
```

Pour accéder au second élément par exemple on tape `b(2)`.

Exercice 4. *Les scripts*

Afin de pouvoir réutiliser les lignes de calcul, il est utile de les mettre dans un *script*. Un script est un fichier texte que Matlab pourra lire et exécuter.

1. Ouvrez l'éditeur de scripts de Matlab soit en cliquant sur la page blanche de la barre d'outils, soit en allant dans le menu "File→New→M-file".

Créez le script suivant :

```
% Ceci est un script matlab,
% le signe "pourcent" permet de mettre des commentaires
% qui ne seront pas interpretes

disp('Salut') % disp permet d'afficher ce que l'on veut à l'écran, les '
              % permettent d'indiquer que l'on veut afficher du texte.

a = input('entrez a : ') % input demande à l'utilisateur d'entrer une
```

1. Si vous y arrivez bravo... ça sert à pas grand chose de le faire

```
                                % valeur au clavier
b = 6

b=b+a

a, b % la virgule permet de mettre plusieurs commandes sur une seule
     % ligne, elle a le meme role que la touche entrée.
```

Enregistrez le fichier et appelez le dans l'interpréteur.

2. Écrivez un programme qui demande deux valeurs `a` et `b` à l'utilisateur et qui les affiche, qui intervertit leurs contenus et qui les affiche à nouveau.

Réponse :

```
a = input('entrez a : ');
b = input('entrez b : ');
c = a;
a = b;
b = a;
disp(a);
disp(b);
```

Exercice 5. *Boucles : Automatisation d'actions*

On peut répéter des actions grâce aux boucles : la boucle `for` permet de changer la valeur d'une variable de manière régulière. La syntaxe pour la boucle `for` est la suivante :

```
for i=1:n
 disp(i);
end
```

Le code entre le `for` et le `end` est executé `n` fois : une première fois avec la variable `i` à 1, une deuxième fois avec la variable `i` à 2, etc jusqu'à `n`.

1. Écrivez un programme qui demande deux entiers a et b et qui affiche le résultat de la somme suivante :

$$\sum_{k=1}^{b} k^a$$

Réponse :

```
a = input('entrez a : ');
b = input('entrez b : ');
S=0;
for k=1:b
   S=S+k^a;
end
S
```

2. Écrivez un programme qui calcule le 1000ème terme de la suite de Fibonacci :

$$u_0 = 0 \quad u_1 = 1 \quad u_{n+2} = u_{n+1} + u_n$$

Réponse :

```
u0=0; % terme representant u_n
u1=1; % terme representant u_n+1
for k=1:999
  u2 = u1 + u0;  % calcul de u_n+2
  u0 = u1;       % preparation des variables pour la
  u1 = u2;       % etape du calcul
end
disp(u0); % u0 contient en fait à ce moment la u_n+1
```

3. Écrivez un programme qui calcule le 100ème terme de la suite suivante :

$$u_0 = 2 \quad u_1 = 1 \quad u_2 = 3 \quad u_{n+3} = 2u_{n+2} + 3u_{n+1} + u_n$$

Réponse :

```
u0=0; % terme representant u_n
u1=1; % terme representant u_n+1
u2=3; % terme representant u_n+2
for k=1:99
  u3 = 2*u2+3*u1 + u0;  % calcul de u_n+3
  u0 = u1;       % preparation des variables pour la
  u1 = u2;       % etape du calcul
  u2 = u3;
end
disp(u0);
```