République algérienne Démocratique et Populaire
Ministère de l'enseignement supérieur et de la recherche scientifique
Université de JIJEL
Faculté des Sciences Exactes et Informatique
Département d'informatique

Les Outils de Programmation pour les Mathématiques (OPM)

L1 Informatique/Mathématiques/ MI

Plan du cours

- Cours 1: Programmation: Concepts de bases.

- Cours 2: Présentation de Matlab.

- Cours 3: Commandes et Types de données de base de Matlab.

- Cours 4: Les vecteurs et matrices dans Matlab.

- Cours 5: Programmation sous Matlab

1) Structures de contrôle
   a) Structure conditionnelle
   b) Choix multiple – instruction switch
   c) Les boucles: la boucle for, la boucle while
2) Fonctions: les m-fonctions, les fonctions Inline, les fonctions anonymes.

- Cours 6: Figure et Courbes sous Matlab.

La programmation

Qu’est ce que la programmation

Un ordinateur ne sait faire que certaines opérations très basiques :

- Additionner, soustraire, multiplier, diviser des valeurs numériques
- Déplacer des données d’un endroit de sa mémoire a un autre endroit
- Sauter d’une ligne de code a une autre si une condition est vraie

La programmation consiste à **combiner ces instructions** qui agissent sur des données afin de réaliser une tâche spécifique à l’aide de l’ordinateur (tester si un nombre est premier, détecter des éléments dans une image, etc...)

Différents langages de programmation

Années 1940

. Assembleur

Langage très proche des capacités de la machine : on peut y réaliser directement les opérations que peut exécuter la machine, et on doit penser son programme de façon a ce qu'il corresponde à ces instructions.

Très fastidieux pour écrire de longs programmes, et pas très facile de relire un code

```
PlayerEndWorld:

        Ida WorldEndTimer       ;check to see if world end timer expired ;branch to leave if not
        bne EndExitOne          ;check world number
        ldy WorldNumber         ;if on world 8, player is done with game,
        cpy #World8             ;thus branch to read controller
        bcs EndChkBButton
        Ida #$00                ;otherwise initialize area number used as offset ;and level
        sta AreaNumber          number control to start at area 1 ;initialize secondary mode of
        sta LevelNumber         operation ;increment world number to move onto the next world
        sta OperMode_Task       jget area address offset for the next area ;set flag to load game
        inc WorldNumber         timer from header
        jsr LoadAreaPointer
        inc FetchNewGameTimerFlag ;set mode of operation to game mode
        Ida #GameModeValue
        sta OperMode
```

L'assembleur est toujours utilise de nos jour . Ici, un exemple du code Super Mario World sur Super Nintendo.

Différents langages de programmation

Années 1950, I960

FORTRAN, LISP, COBOL, BASIC

Langage plus moderne, avec un syntaxe proche des mathématiques modernes.

Ex : En Assembleur, pour multiplier deux valeurs (par exemple 2 et 3), il faut charger chacune des valeurs dans une case mémoire, effectuer la multiplication des deux cases mémoires, et recopier le résultat sur le disque.

En BASIC, il suffit d'écrire 2*3.

Différents langages de programmation

Années 1970

- C, Prolog, Pascal, SQL

L'ere de la programmation structurée, avec une syntaxe plus rigoureuse qui permet de plus facilement relire un code et le comprendre.

Différents langages de programmation

Années 1980

- Matlab, C++, Objective-C, Perl

Recherche de performance (programmes qui s'executent vite même si le code n'est pas optimisé) et de modularité (possibilité de facilement réutiliser des morceaux d'un programme dans un autre programme) afin de réduire le temps de développement.

Débuts de la programmation orientée objet.

Différents langages de programmation

Années 1990

Python, R, Java, PHP, JavaScript (rien a voir avec Java)

Perfectionnement des concepts de l'orienté objet, débuts de l'ere Internet : les langages incorporent beaucoup de facilité de communication réseau.

Différents langages de programmation

Années 2000 et de nos jours

D, Cuda, Go, Rust

Recherche de sécurité dans les langages, afin d' éviter les failles dans les logiciels.

Recherche de performance et facilité de programmation des ordinateurs avec des processeurs en parallèle (comme les cartes graphiques)

Nouvel engouement pour d'anciens langages revenant à la mode, grâce à des nouveaux «frameworks» disponibles (Matlab, Python, JavaScript (HTML5), ou pour le développement sur plateforme mobile (Objective-C pour Iphone, Java pour Android)

Quelques définitions supplémentaires

Langage de programmation

C'est une façon d' écrire des instructions qui seront ensuite « traduites » en opérations de base pour l'ordinateur

Les instructions ont en général une tache précise : elles forment ensemble un logiciel.

Un langage de programmation est à l'informatique ce qu'une langue est à la littérature : c'est une façon de dire les choses afin de raconter une histoire.

Quelques définitions supplémentaires

Logiciel

C'est une suite d'instructions écrites dans un langage de programmation qui permet de réaliser une ou plusieurs tâches (éditeur de texte, jeux vidéo, lecteur vidéo, navigateur internet, ...)

La plupart du temps, possède une interface graphique pour faciliter les interactions avec l'utilisateur

Un logiciel est à l'informatique ce qu'un livre est à la littérature : c'est une histoire écrite dans une langue.

Tout comme un livre peut être traduit dans différentes langues, un logiciel peut être traduit dans différents langages de programmation (ex : Angry Birds ecrit en Java pour Android, et Objective-C pour I OS)

Quelques définitions supplémentaires

Framework

Un framework est une collection de sous-programmes, écrits dans un langage spécifique, permettant de réaliser des tâches complexes plus simplement.

Ex : En C++, pour multiplier deux matrices A et B, il faut le faire comme en maths -parcourir les lignes de l'une, les colonnes de l'autre, et faire des produits et des sommes. Avec le framework BOOST, on écrit prod(A,B)

Un framework permet donc de réutiliser le travail effectué par d'autres pour son propre programme, et perdre moins de temps en évitant d' écrire ce que d'autres ont déjà écrit.

Classification des langages de programmation

- **Le domaine d'application** : gestion, éducation, intelligence artificielle

- **La technologies visée** : réseau , Base de données, web

- **La façon d'aborder un problème** : utilisation des procédures , des objets,

Paradigmes de programmation

Les Paradigmes des langages de programmation sont :

-Programmation **Impérative /Procédural.** (Pascal, Cobol, fortran, C, ….)

-Programmation **Déclarative**

-Programmation **O.O** (Java, C++, simula, …)

-Programmation **Fonctionnelle.**

Paradigmes de programmation

Programmation Impérative :

un programme impératif est une séquence d'instructions qui spécifie étape par étape les opérations à effectuer pour obtenir à partir des entrées un résultat (la sortie).

Dans la programmation impérative, un programme peut modifier l'état de la mémoire et peut interagir avec des effets de bord (affichage à l'écran, lecture au clavier, etc.).

Paradigmes de programmation

Programmation déclarative

Dans cette forme de programmation on décrit la tâche que le programme doit accomplir au lieu de décrire comment le programme doit accomplir cette tâche.

Paradigmes de programmation

Programmation fonctionnelle

Ce paradigme prend son origine dans le langage mathématique traitant des fonctions. Une fonction, dans son incarnation informatique, accepte des données et produit des données. Une fonction peut être soit *primitive*, soit formée par une composition de fonctions.

Il n'y a pas de séparation entre données et programmes : une fonction est un objet de *première classe* sur lequel d'autres fonctions peuvent opérer. Nous ne rentrerons pas plus en détail dans les caractéristiques précise de la programmation fonctionnelle.

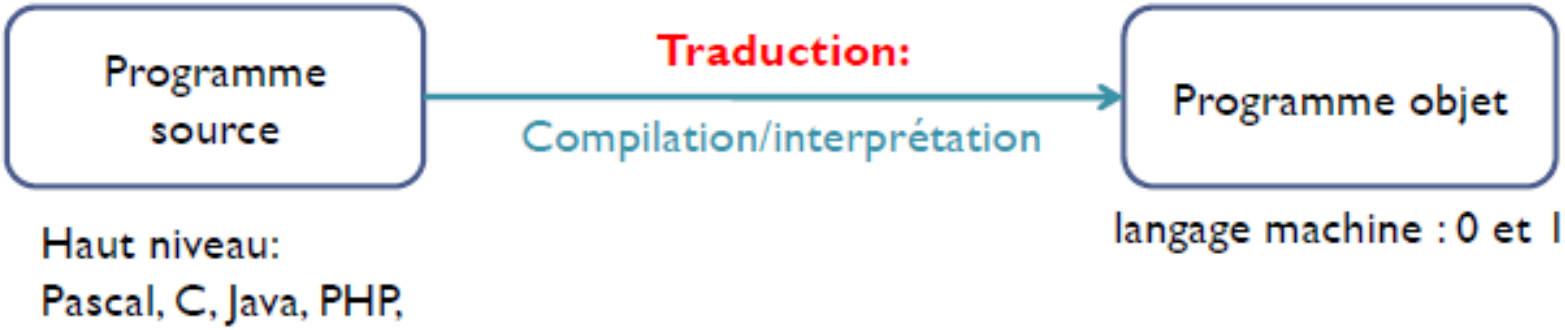
Paradigmes de programmation

Programmation Orientée Objet

Programmer avec un langage oreitné-objet amène le programmeur à identifier les *acteurs* qui composent un problème, puis à déterminer ce qu'est et ce que doit savoir chaque acteur. En regroupant les aspects communs puis en les spécialisant, le programmeur établit une hiérarchie de classe.

Exemples de langages de programmation orientée objet :
C++, C#, Java, Python, etc ...

Code source vs code machine



Code source vs code machine

- Programme source, code source :
  - Programme écrit dans un langage.

- Code machine, code exécutable :
  - Programme dans un langage machine
  - Directement exécutable par la machine

Code source vs code machine

Langage machine : utilisé par le processeur , il est constitué d'une suite de 0 et de 1. Le langage machine n'est pas compréhensible par l'être humain.

L'assembleur : est le premier langage informatique qui ait été utilisé. Celui-ci est très proche du langage machine mais reste compréhensible pour des développeurs.

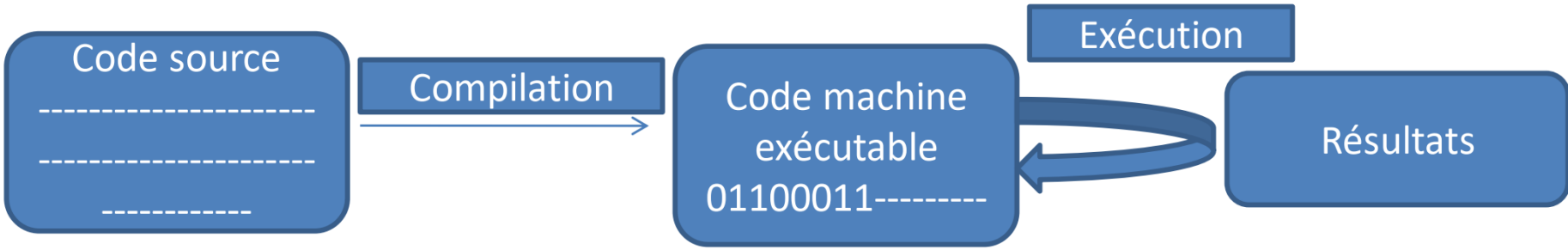
Code source vs code machine

Compilateur : Logiciel chargé de traduire le code source en langage machine. Un programme écrit dans un langage dit « **compilé** » va être traduit une fois pour toutes par un **compilateur**, afin de générer un nouveau fichier qu'on appelle fichier **exécutable** (programme objet).

- Convertir un code source en un code exécutable,
- Une fois le code source compilé, le programme " l'exécutable" peut être exécuté autant de fois que nécessaire.

Code source vs code machine

Problème: Un programme compilé sous une plateforme **X ne peut être** exécuté sur toutes les autres plateformes.
(Exemple programme écrit en C++)



Code source vs code machine

Interpréteur : est un programme chargé de décoder chaque instruction du langage et d'exécuter les actions correspondantes instruction par instruction.

- Ne génère pas le fichier en code machine.
- Un interpréteur quand à lui, effectue le même travail de vérification du code que le compilateur, mais ne génère pas le code exécutable dans un fichier, il interprète le code ligne par ligne et les exécutent immédiatement.

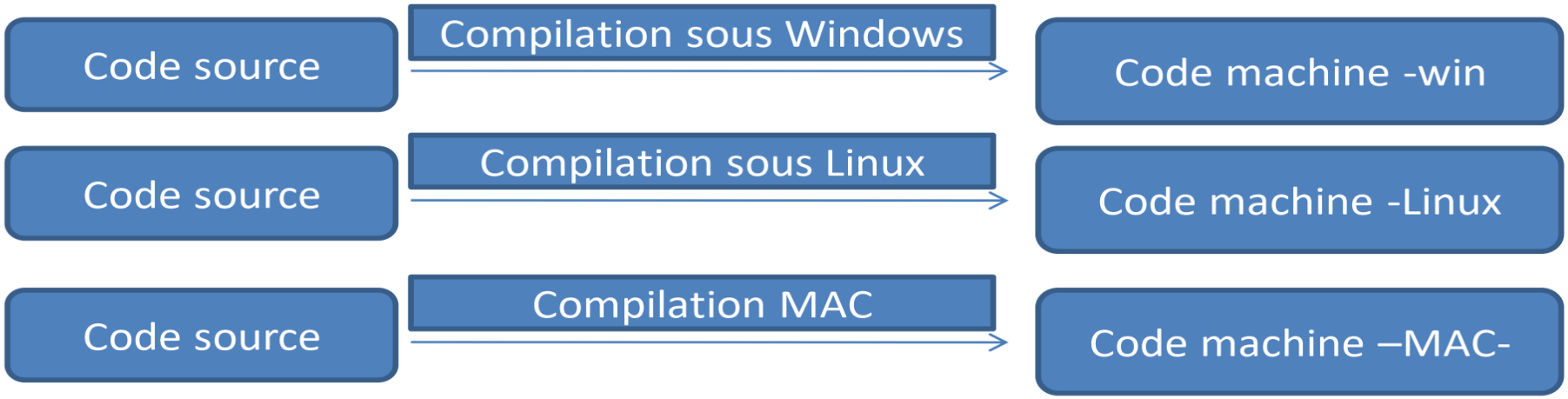
Code source vs code machine

Interpréteur

Un programme écrit dans une plateforme **X peut être exécuté sur toutes les** autres plateformes (Exemple programme écrit en Python, Java)

Langage compilés vs langage interprété

-Langages compilés : dépendant de la plateforme



– Langages interprétés : indépendant de la plateforme

Exemple de langages de programmation

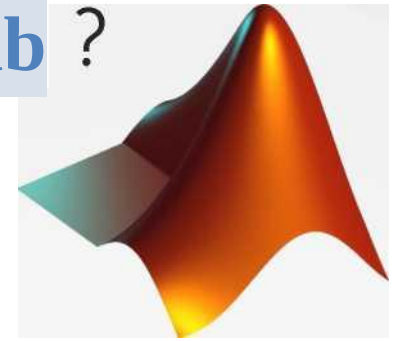
Langage	Domaine d'application principal	Compilé/interprété
ADA	Le temps réel	Langage compilé
BASIC	Educatif	Langage interprété
C	Programmation système	Langage compilé
C++	Programmation système objet	Langage compilé
Cobol	Gestion	Langage compilé
Fortran	Calcul	Langage compilé
MATLAB	Calcul mathématique (numérique)	Langage interprété
Mathematica	Calcul mathématique (formel	Langage interprété
Pascal	Enseignement	Langage compilé
PHP	Programmation web	Langage interprété
Prolog	Intelligence artificielle	Langage interprété
Perl	Traitement de chaînes de caractères	Langage interprété

Remarque:

- MATLAB n'est pas le seul environnement de calcul scientifique;
- Il Existe d'autres dont les plus importants sont **MAPLE** et **MATHEMATICA**.
- Il existe même des logiciels libres qui sont des clones de MATLAB comme **SCILAB** et **OCTAVE**.

Présentation de Matlab

Qu'est-ce que Matlab ?



Matlab (MATrix LABoratory) est un langage de programmation orienté calcul scientifique, pensé pour rendre le calcul matriciel simple à programmer et efficace en temps.

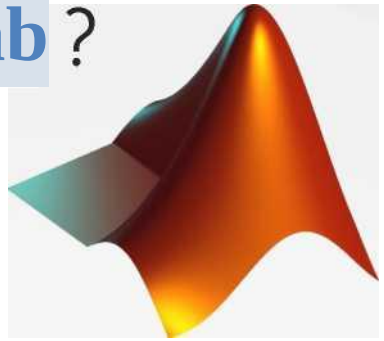
Matlab est un logiciel (payant) proposant une interface graphique vers un éditeur de code en Matlab, et un outil de debugging pour exécuter des programmes en mode pas a pas.

MATLAB comprend :

- De nombreuses fonctions graphiques,
- Un système puissant d'opérateurs s'appliquant à des matrices,

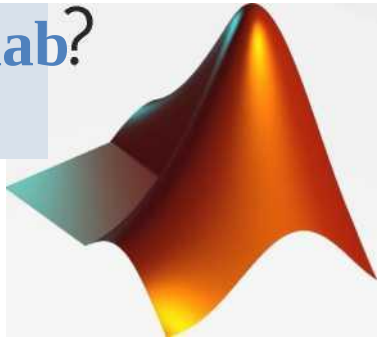
Un langage de programmation extrêmement simple à utiliser.

Matlab ?



En termes de vitesse d'exécution, les performances de Matlab sont inférieures à celle obtenues avec un langage de programmation classique. L'emploi de MATLAB devrait donc être restreint à des problèmes peu gourmands en temps calcul, mais dans la plupart des cas, il présente une solution élégante et rapide à mettre en œuvre.

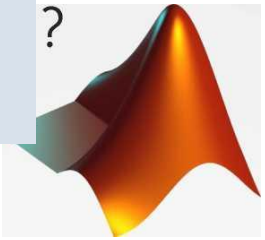
Domaines d'application de Matlab?



A l'origine MATLAB était conçu pour faire principalement des calculs sur les vecteurs et les matrices d'où son nom '**Matrix Laboratory',** mais par la suite **il a était amélioré et** a**ugmenté** pour pouvoir traiter beaucoup plus de domaines.

- Domaines d'ingénierie
- Domaines de la recherche scientifique,
- Etablissements d'enseignement supérieur.

Point fort de matlab



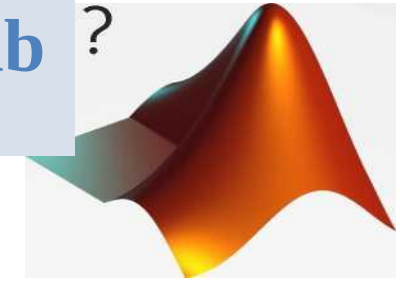
- forte et simple interaction avec l'utilisateur
- Sa richesse fonctionnelle :

➢ Réaliser des manipulations mathématiques complexes en écrivant peu d'instructions.

➢ Evaluer des expressions, dessiner des graphiques et exécuter des programmes classiques.

➢ Permet l'utilisation directe de plusieurs milliers de fonctions prédéfinie.

Point fort de matlab ?



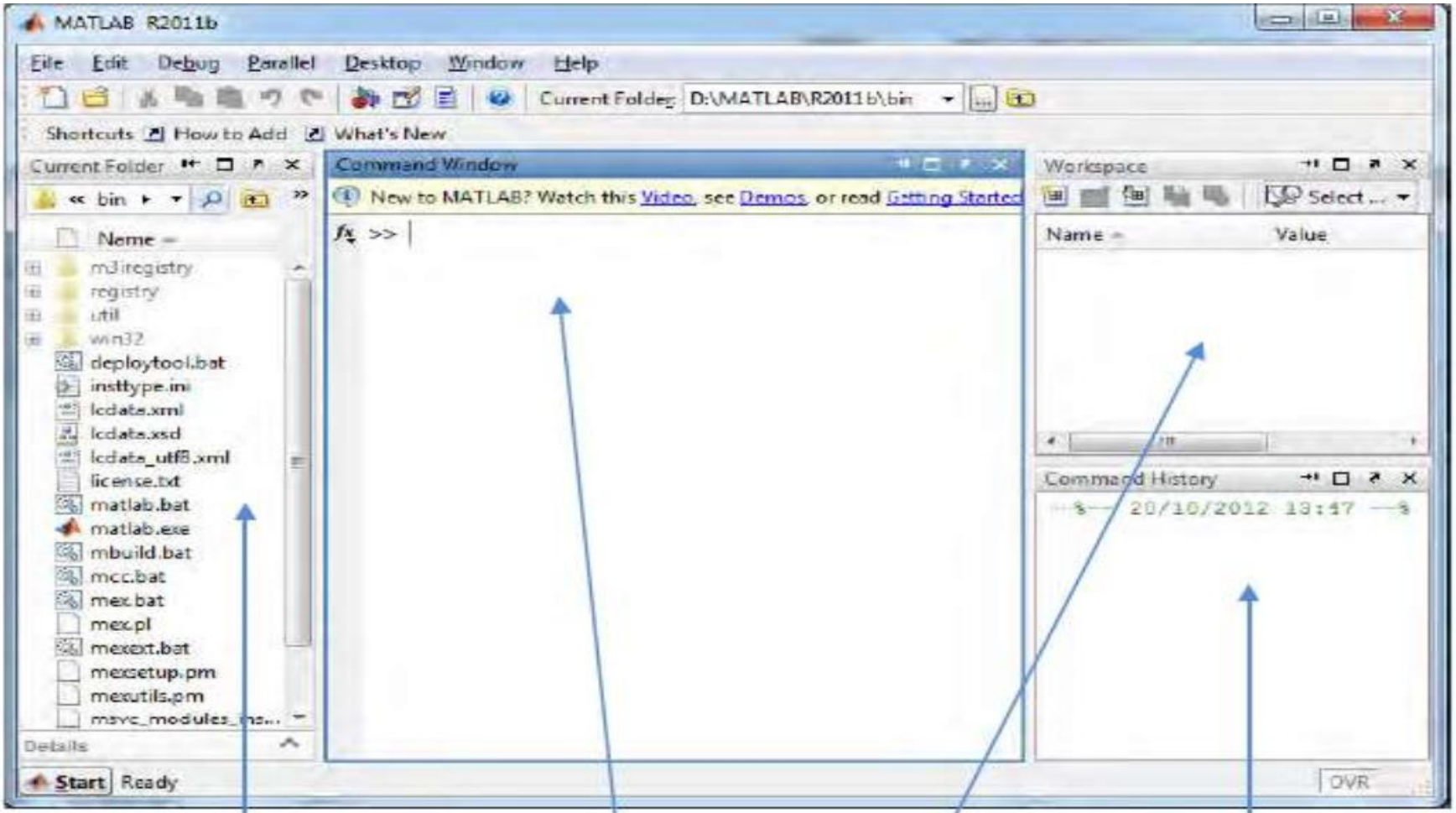
• La simplicité de son langage de programmation :un programme écrit en MATLAB est plus facile à écrire et à lire comparé au même programme écrit en C ou en PASCAL.

• Sa manière de tout gérer comme étant des matrices, ce qui libère l'utilisateur de s'occuper de typage de données et ainsi de lui éviter les problèmes de transtypage.

Présentation de I'interface

L'interface de Matlab se compose de plusieurs zones

Présentation de I ’interface



Présentation de I ’interface

Chaque zone possède un objectif précis :

Le menu regroupe des commandes de base de Matlab, comme enregistrer, afficher les préférences, etc...

L’explorateur de fichiers permet de visualiser ses fichiers scripts et de les ouvrir pour les éditer.

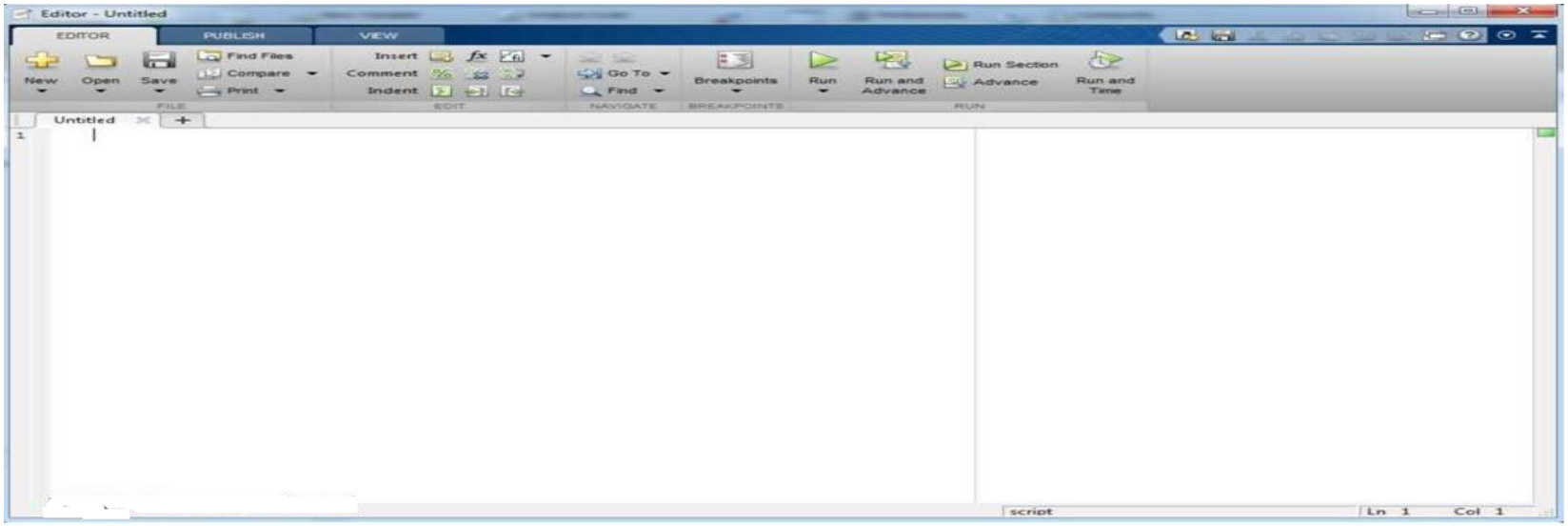
La zone de commande permet d’écrire des commandes et de visualiser leur résultat

La zone des variables permet de visualiser toutes les variables en mémoire à l’instant présent (leur nom ainsi que visualiser leur contenu).

L’historique permet de visualiser 1’historique des commandes précédemment exécutées.

Présentation de I ’interface

On peut écrire des commandes simples dans Matlab. Cependant, quand on souhaite écrire un programme complet, on utilise l’éditeur de script Matlab :



✓ On tape le programme au préalable dans un fichier en utilisant l'éditeur intégré.

✓ Une fois le script enregistré, on peut l'exécuter en tapant son nom dans la fenêtre Matlab

Matlab à la maison

Pour travailler sur Matlab chez soi, il existe plusieurs solutions

. Acheter une version étudiante de Matlab

En trouver une sur Internet... 

S'inscrire, sur Coursera, au cours **Introduction to Programming with MATLAB** (https://www.coursera.org/learn/matlab) qui est gratuit et permet d'obtenir un accès temporaire a Matlab Online (Matlab dans un navigateur internet)

Installer le logiciel Octave (https://www.gnu.org/software/octave) qui est une sorte de clone de Matlab

Les commandes de base en Matlab

Le moyen le plus simple d'utiliser MATLAB est d'écrire directement dans la fenêtre de commande (Command Window) juste après le curseur (promp) **>>**

Par exemple, on écrit la commande

```
>> 3+6
```

Puis on clique sur la touche Entrer pour voir le résultat

```
ans =
     9
```

Ici, **ans** (pour answer) est une variable qui contient toujours le résultat de la dernière opération réalisée.

Les variables (simples) en Matlab

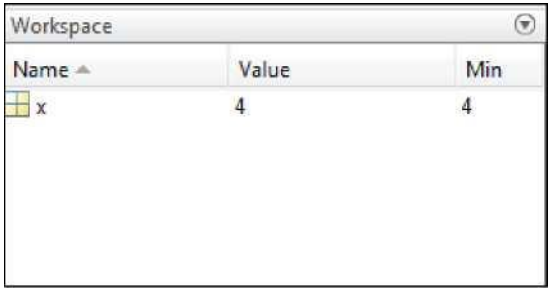
Comme dans la plupart des langages de programmation (comme le C), il est possible de Stocker des valeurs numériques dans des variables.

Pour déclarer une variable et lui attribuer une valeur, il suffit d'utiliser le symbole égal :

```
>> x = 4
```

Cette commande crée une variable nommée x, et lui attribue la valeur 4.

La variable apparait alors dans la zone des variables (workspace):

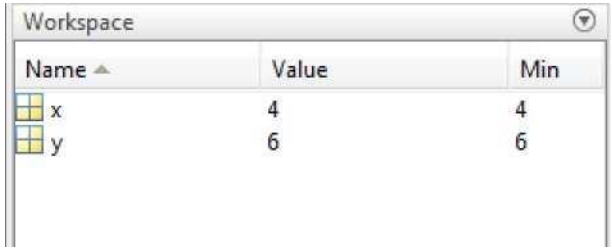


Les variables (simples) en Matlab

II est possible d'utiliser des variables à la place de valeurs numériques dans les opérations .

Exemple:

```
>> y = x+2
```



Contrairement au langage C il est nécessaire de déclarer une variable avant de lui attribuer une valeur, en Matlab, on attribue directement une valeur à la variable pour la créer.

Les variables (simples) en Matlab

Pour afficher la valeur contenue dans une variable, on utilise le mot clef **disp,** ou bien on peut écrire directement le nom de la variable :

```
>> disp(y)
     6
```

```
>> y

y =
     6
```

Lors du TP, on a parle souvent de « mots clefs ». En réalité, le terme exact est « fonction » : on parlera désormais de la fonction **disp,** tout comme les fonctions **cos, sqrt,** etc...

Les variables (simples) en Matlab

Exemple1: Pour déclarer une variable a, on dit simplement à quoi elle est égale.

```
>> a=1.2


a=
       1.2000
```

On peut maintenant inclure la variable a dans de nouvelles expressions mathématiques, pour en définir une nouvelle variable b :

```
>> b = 5*a^2+a


b=
       8.4000
```

Les variables (simples) en Matlab

Exemple1(suite)

On peut maintenant inclure les deux variables a et b dans une nouvelle expression mathématiques, comme suit:

```
>> c = a^2 + b^3/2

c=
       297.7920
```

On a maintenant trois variables a, b et c. Ces variables ne sont pas affichées en permanence à l'écran. Mais pour voir le contenu d'une variable, rien de plus simple, on tape son nom :

```
>> b


b=
       8.4000
```

ou on double-click sur son nom dans le workspace.

Les variables (simples) en Matlab

Remarque 1:

Si nous voulons qu'une expression soit calculée mais sans afficher le résultat, on ajoute un point-virgule '**;**' à la fin de l'expression comme suit :

```
>> 5+6 ;
>>
```

Remarque 2:

Il est possible d'écrire plusieurs expressions dans la même ligne en les séparant par des virgules ou des points virgules.

Exemple1:

```
>> 5+6, 2*5-1, 12-4
ans  = 11
ans = 9
ans = 8
```

Exemple 2

```
>> 5+6; 2*5-1, 12-4;

>> ans = 9
```

Les variables (simples) en Matlab

Remarque importante:

Les identificateurs (nom de variable et de fonctions) doivent respecter les règles suivantes:

- ✓ débute nécessairement par une lettre, éventuellement suivie de lettres, de chiffres ou du caractère souligné (_).

- ✓ ne contient ni blancs, ni caractères de ponctuation, ni caractères accentués, ni opérateurs arithmétiques (-, +, ...), etc.
- ✓ sa longueur est inférieure ou égale à (63) caractères.

- ✓ en Matlab, les majuscules sont distinguées des minuscules (on dit qu'ils sont *case-sensitive*).

Les variables (simples) en Matlab

La fonction *input*

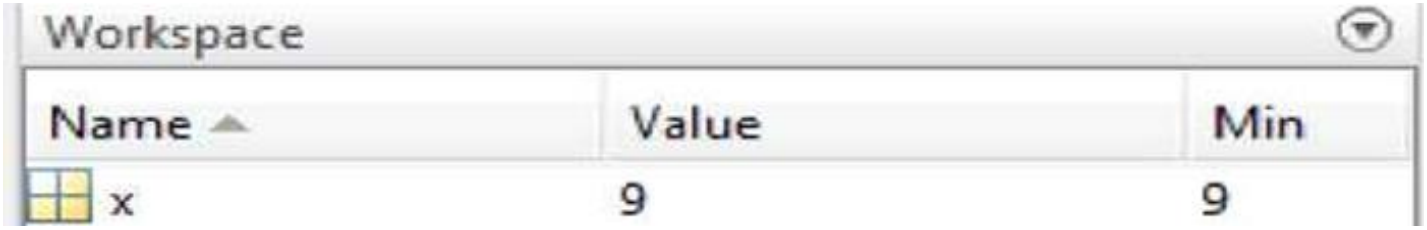
Pour initialiser une variable avec une valeur entrée au clavier par l'utilisateur, on utilisera la fonction **input:**

Pour les valeurs numériques: ***n* = input('*message*')**

- ✓ Affiche le message *message*,
- ✓ et affecte la valeur numérique entrée au clavier à la variable *n*.

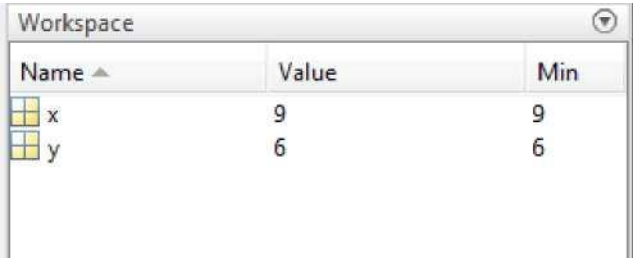
Exemple:

```
>> x = input('Entrez une valeur : ');
Entrez une valeur : 9
```



Les variables en Matlab

Pour voir la liste des variables utilisées, soit on regarde dans la fenêtre **'Workspace'** soit on utilise les commandes « **whos** » ou « **who** »



- **whos** donne une description détaillée (le nom de la variable, son type et sa taille).
- **who** donne juste les noms des variables.

Les variables en Matlab

Exemple: « **whos** » « **who** »

Par exemple, on a utilisé 3 variables **a, u et v et une quatrième sans nom (ans):**

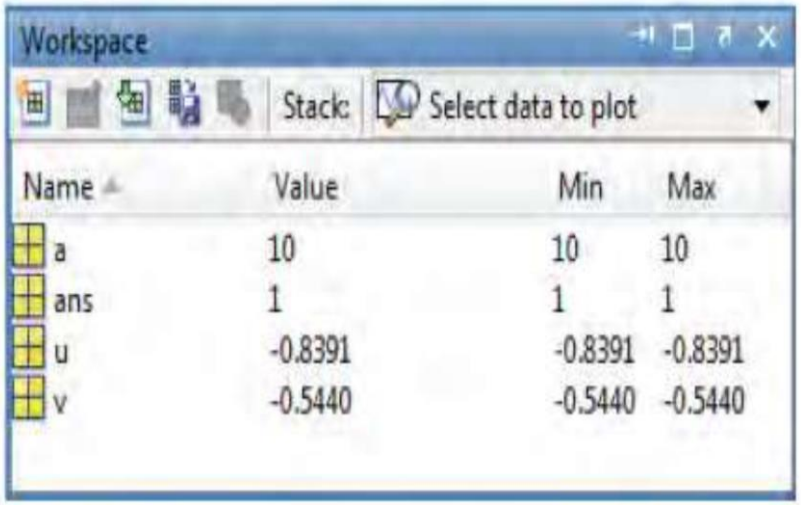
```
>> who
      Your variables are:
      a    ans  u    v

>> whos
      Name        Size              Bytes  Class     Attributes
      a           1x1                   8  double
      ans         1x1                   8  double
      u           1x1                   8  double
      v           1x1                   8  double
```

Les variables en Matlab

Exemple: « **whos** » « **who** » **(suite)**

L'utilisation des deux commandes « **whos** » et « **who** » peut être omise car des informations sur les variables sont visibles directement dans la fenêtre workspace.



Les types de données

- **Rappel:** En Matlab:

✓ toute variable est une matrice d'éléments d'un type donné.

✓ on ne définit pas de manière explicite le type d'une variable.

- Les principaux types de données utilisés par Matlab sont: réel, complexe, chaîne de caractères et logique.

- Il n'y a pas de type entier à proprement parler.
- Le type logique a deux valeurs :

✓ ***true*** (vrai) représenté par 1.

✓ ***false***(faux) représenté par 0

Les types de données

Exemple

```
>> x=3;
>> y=2*i;
>> z=4;
>> rep='oui';
>> x==z
ans =
     0
>> eq = (x ==z-1)
eq =
     1
>> whos
  Name          Size                Bytes  Class     Attributes
  ans           1x1                     1  logical
  eq            1x1                     1  logical
  rep           1x3                     6  char
  x             1x1                     8  double
  y             1x1                    16  double    complex
  z             1x1                     8  double
```

Les types de données

Les fonctions suivantes permettent de déterminer le type d'une variable:

✓***class(a)*** : retourne le nom de la classe à laquelle appartient la variable *a* (le type de *a*) .

```
>> s = 'bonjour';
>> class(s)
ans =
     char
```

✓ ***ischar(a)*** retourne 1 si *a* est de type chaîne de caractères et 0 sinon.

✓ ***islogical(a)*** retourne 1 si *a* est de type logique et 0 sinon.

✓ ***isreal(a)*** retourne :

• 1 si *a* est réel ou de type chaîne de caractères.

• 0 sinon (*a* est complexe à partie imaginaire non nulle ou n'est pas une matrice de valeurs réelles ou de caractères).

Les types de données

Exemple:

```
>> a = 2; b = 3;
>> c=a+b*i;
>> d = a==b;
>> rep='non';
>> A = [1 2 3; 4 5 6];
>> ischar(rep)
ans =
     1
>> ischar(a)
ans =
     0
>> islogical(a)
ans =
     0
>> islogical(d)
ans =
     1
```

```
>> isreal(a)
ans =
     1
>> isreal(rep)
ans =
     1
>> isreal(A)
ans =
     1
>> isreal(c)
ans =
     0
```

Les nombres en MATLAB

- Une notation décimale conventionnelle, avec un point décimal facultatif '.'

- Le signe '+' ou '–' pour les nombres signés.

- La notation scientifique utilise la lettre 'e' pour spécifier le facteur d'échelle en puissance de 10.

- Les nombres complexes utilise les caractères 'i' et 'j' (indifféremment) pour designer la partie imaginaire.

Les nombres en MATLAB

Le type	Exemples
Entier	5 -83
Réel en notation décimale	0.0205 3.1415926
Réel en notation scientifique	1.60210e-20 6.02252e23 (1.60210×10^{-20} et 6.02252×10^{23})
Complexe	5+3i -3.14159j

Le type réel (les nombres réels)

Comme dans tous les langages de programmation, les nombres réels s'écrivent avec un point pour séparer la partie entière de la partie décimale, et non pas une virgule.

```
>> 2.7 * 3.9

ans =

    10.5300
```

Les nombres réels

MATLAB utilise toujours les nombres réels (double précision) pour faire les calculs, ce qui permet d'obtenir une précision de calcul allant jusqu'aux 16 chiffres significatifs.

La commande	Signification
format short	Affiche les nombres avec 4 chiffres après la virgule
format long	Affiche les nombres avec 15 chiffres après la virgule
format bank	Affiche les nombres avec 2 chiffres après la virgule
format rat	Affiche les nombres sous forme d'une ration (a/b)

Les nombres réels

Exemple:

```
>> 8/3
        ans =
               2.6667

>> format long

>> 8/3
       ans =
              2.66666666666667
>> format bank

>> 8/3
       ans =
              2.67
>> format short

>> 8/3
       ans =
              2.6667
```

Les nombres réels

Exemple:

```
>> 7.2*3.1
       ans =
             22.3200

>> format rat

>> 7.2*3.1
       ans =
             558/25

>> 2.6667
ans =
   26667/10000
```

Les nombres réels

La fonction **vpa** peut être utilisé afin de forcer le calcul de présenter plus de décimaux significatifs en spécifiant le nombre de décimaux désirés.

Exemple

```
>> sqrt(2)
      ans =
            1.4142
```

```
>> vpa(sqrt(2),50)
      ans =
            1.4142135623730950488016887242096980785696718753769
```

Les opérations de base

Les opérations de base dans une expression sont résumés dans le tableau suivant :

L'opération	La signification
+	L'addition
-	La soustraction
*	La multiplication
/	La division
\	La division gauche (ou la division inverse)
^	La puissance
'	Le transposé
(et)	Les parenthèses spécifient l'ordre d'évaluation

On peut combiner ces symboles en une seule commande. Les règles de priorité usuelles entre operateurs sont alors appliquées.

Les priorités des opérations dans une expression

L'évaluation d'une expression s'exécute de gauche à droite en considérant la priorité des opérations indiquée dans le tableau suivant :

Les opérations	La priorité (1=max, 4=min)
Les parenthèses (et)	1
La puissance et le transposé ^ et '	2
La multiplication et la division * et /	3
L'addition et la soustraction + et -	4

Par exemple:

5+2*3 = 11
2*3^2 = 18

Le type complexe

Les nombres complexes

- Les nombres complexes peuvent être écrits sous forme cartésienne $a + bi$.
- L'unité imaginaire est désignée par *i* ou *j*.
- Les fonctions ***imag***, ***real***, ***abs***, ***angle*** permettent de passer aisément de la forme polaire à la forme cartésienne et réciproquement.

✓ ***imag*** et ***real*** retournent la partie imaginaire et la partie réelle

✓ **abs** et ***angle*** retournent le module et l'argument.

Les nombres complexe

Exemple

```
>> a = 2; b = 3;
>> sqrt(-1)
ans =
          0 + 1.0000i
>> c = a+b*I
c =
   2.0000 + 3.0000i
>> d = 1+2*I
d =
   1.0000 + 2.0000i
>> c*d
ans =
  -4.0000 + 7.0000i
>> x = a*exp(b*i)
x =
  -1.9800 + 0.2822i
```

```
>> real(c)
ans =
      2
>> imag(c)
ans =
      3
>> abs(x)
ans =
    2.0000
>> angle(x)
ans =
      3
```

Le type logique

- Un résultat de type logique est retourné par certaines fonctions ou dans le cas de certains *tests* (évaluation des expressions logique).
- Le type logique (***logical***) possède 2 valeurs: **vrai** (*true*) représenté par 1 et **faux** (*false*) représenté par 0.
- Une valeur numérique non nulle est considérée comme *true*.

```
>> a=0; b=3;
>> if(a) disp('true'), else disp('false'), end
false
>> if(b) disp('true'), else disp('false'), end
true
```

Le type logique

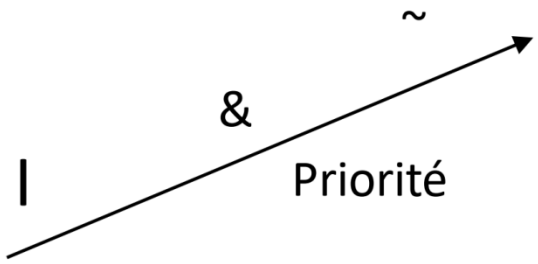
Exemple

```
>> a=2; b = 3;
>> a<b
ans =
     1
>> a==3
ans =
     0
>> c=a==(b-2)
c =
     0
>> isreal(a)
ans =
     1
>> isreal(a*i)
ans =
     0
>> x=123;
>> y=exp(log(x));
>> test=(x==y)
>> if test, disp('x = y'), else ('x est different de y'), end
ans =
x est different de y
>> whos
Name    Size   Bytes  Class    Attributes
 a      1x1        8  double
 ans    1x20      40  char
 b                 8  double
 c      1x1        1  logical
 test   1x1        1  logical
 x      1x1        8  double
 y      1x1        8  double
>> format long
>> exp(log(123))
ans =
    1.22999999999999e+002
```

Le type logique

1) Les opérateurs logiques

&	et (x & y)
|	ou (x | y)
~	non (~x)



2) Les opérateurs de comparaison

==	égal à (x == y)
>	strictement plus grand que (x >y)
<	strictement plus petit que (x <y)
>=	plus grand ou égal à (x ≥ y)
<=	plus petit ou égal à (x ≤ y)
~=	different de (x ~= y)

Le type logique

Exemple

```
>> a=2; b = 3;
>> a==b
ans =
     0
>> b>3
ans =
     0
>> a~=2
ans =
     0
>> c = (a>2)|(b<=3)
c =
     1
>> c = (a>2)||(b<=3)
c =
     1
```

```
>> c = ~(a>2)|(b<=3)
c =
     1
>> c = ~((a>2)|(b<=3))
c =
     0
>> c = (a>2)&&(b<=3)
c =
     0
>> c = ~(a>2)&(b<=3)
c =
     1
>> ~1
ans =
     0
>> ~0
ans =
     1
```

Le type chaine de caractères

- Une donnée de type chaîne de caractères (*char*) est définie par des apostrophes simples (’): 'Hello World’

- Une variable de type chaîne de caractères est traitée comme un vecteur ligne dont chaque élément est un caractère:
  - ✓ il est possible de manipuler chaque caractère de la chaîne en faisant référence à sa position dans la chaîne.
- Pour concaténer des chaines de caractères, on les représente comme des éléments d’un vecteur ligne.
- Les deux apostrophes '' permettent de définir la chaîne vide.

```
>> s1 = 'bonjour';
s2 = ', il fait';
s3 = ' beau aujourd''hui';
>> %s4 = [s1 s2 s3]
>> s4 = [s1, s2, s3]
s4 =
bonjour, il fait beau aujourd'hui
>> s4(1) % accès pour écriture
         % le premier élément à l'indice 1
ans =
    b
>> s4(1) = 'B'   % accès pour modification
s4 =
Bonjour, il fait beau aujourd'hui

% On peut aussi utiliser l'opérateur d'énumération :
>> s4(10:16)
ans =
    il fait
```

```
% rappel
>> disp(s1)
Bonjour
>> s5 = input('Saisir un
   texte: ','s');
Saisir un texte:
```

Fonctions générales relatives aux chaînes de caractères

Fonctions	Description
length(string)	Retourne le nombre de caractères de la chaîne string
upper (string)	Convertit la chaîne string en majuscules
lower (string)	Convertit la chaîne string en minuscules
strcat(s1,s2…)	Concatène horizontalement les chaînes s1, s2...

- ✓ La fonction ***isempty (s)*** permet de tester si la chaîne ***s*** est vide ou non.
- ✓ La fonction ***strcmp (s1, s2)*** permet de tester si les deux chaines ***s1*** et ***s2*** sont égales ou non.

Le type chaine de caractères

Exemple 1

```
>> c1 = ''; c2 = 'bonjour';
>> c3 = 'Bonjour';
>> c4 = ['bon' 'jour'];
>> isempty(c1)
ans =
     1
>> isempty(c2)
ans =
     0
>> strcmp(c2, c3)
ans =
     0
>> strcmp(c2, c4)
ans =
     1
>> length(c1)
ans =
     0
>> length(c2)
ans =
     7
```

63

Le type chaine de caractères

Exemple 2

```
>> strcat('MAT','LAB')                    %Concaténer deux chaînes de caractères
ans = MATLAB

>> strcat('TP',' Fnct',' Mat','lab')      %Concaténer plusieurs chaînes de caractères
ans=TP Fnct Matlab

>> upper('tpFNCTmatlab')                  %Conversion en majuscules
ans = TPFNCTMATLAB
```

```
>> lower('tpFNCTmatlab')                  %Conversion en minuscules
ans = tpfnctmatlab

>> x='abcde';                             %Calculer le nombre de caractères de la chaîne
>> length(x)
ans = 5
```

Le type chaine de caractères

Remarque :

Pour l'affectation d'une chaîne de caractères à une variable, la chaîne doit être délimitée par des apostrophes (‘). Si la chaîne contient elle-même des apostrophes, le mécanisme d'échappement consiste à dédoubler les apostrophes dans la chaîne. Exemple :

```
>>ch1 = 'Sciences et ingénierie de l''environnement'

ch1 = Sciences et ingénierie de l'environnement
```

Les principales constantes

MATLAB définit les constantes suivantes :

La constante	Sa valeur
pi	π=3.1415...
exp(1)	e=2.7183...
i	$=\sqrt{-1}$
j	$=\sqrt{-1}$
Inf	∞
NaN	Not a Number (Pas un numéro)
eps	$\varepsilon \approx 2 \times 10^{-16}$.

Les principales fonctions

Voici un tableau qui résume des fonctions fréquemment utilisées :

La fonction	Sa signification
sin(x)	le sinus de x (en radian)
cos(x)	le cosinus de x (en radian)
tan(x)	le tangent de x (en radian)
asin(x)	l'arc sinus de x (en radian)
acos(x)	l'arc cosinus de x (en radian)
atan(x)	l'arc tangent de x (en radian)
sqrt(x)	la racine carrée de x → $\sqrt{.}$
abs(x)	la valeur absolue de x → $\|x\|$
exp(x)	= e^x
log(x)	logarithme naturel de x → $\ln(x)=\log_e(x)$
log10(x)	logarithme à base 10 de x → $\log_{10}(x)$
imag(x)	la partie imaginaire du nombre complexe x
real(x)	la partie réelle du nombre complexe x
round(x)	arrondi un nombre vers l'entier le plus proche
floor(x)	arrondi un nombre vers l'entier le plus petit → $\min\{n \mid n \leq x, n \text{ entier}\}$
ceil(x)	arrondi un nombre vers l'entier le plus grand → $\max\{n \mid n \geq x, n \text{ entier}\}$

Les principales commandes

MATLAB offre beaucoup de commandes. Nous nous contentons pour l'instant d'un petit ensemble:

La commande	Sa signification
who	Affiche le nom des variables utilisées
whos	Affiche des informations sur les variables utilisées
clear x y	Supprime les variables x et y
clear, clear all	Supprime toutes les variables
clc	Efface l'écran des commandes
exit, quit	Fermer l'environnement MATLAB
format	Définit le format de sortie pour les valeurs numériques `format long` : affiche les nombres avec 15 chiffres après la virgule `format short`: affiche les nombres avec 04 chiffres après la virgule `format bank` : affiche les nombres avec 02 chiffres après la virgule `format rat` : affiche les nombres sous forme d'une ration (a/b)

Exercice

Créer une variable x et donnez-la la valeur 2, puis écrivez les expressions suivantes :

- $3x^3-2x^2+4x$
- $\frac{e^{1+x}}{1-\sqrt{2x}}$
- $|\sin^{-1}(2x)|$
- $\frac{\ln(x)}{2x^3} - 1$

Exercice (Solution)

```
>> x=2 ;

>> 3*x^3-2*x^2+4*x ;

>> exp(1+x)/(1-sqrt(2*x)) ;

>> abs(asin(2*x)) ;

>> log(x)/(2*x^3)-1 ;
```