Types composés – chaînes de caractères

- **Chaînes de caractères**
  - introduites entre guillemets " ou apostrophes '
  - beaucoup de langages (Matlab compris) discernent entre " et '
- Dans beaucoup de langages (Matlab compris) une chaîne de caractères est considérée un tableau d'une ligne (vecteur horizontal) à données du type alphanumérique
  - pour combiner plusieurs chaînes :

```
> t1 = "abc"; t2 = "def"
> t3 = [t1, t2]
t3 = abcdef
```

- Caractères spéciaux (communs à beaucoup de langages)
  - sont introduits avec la barre inversée (backslash) \
  - souvent (Matlab) reconnus seulement entre les " et non pas entre les '
  - \n fin de ligne (écran et fichiers Linux)
  - \r\n fin de ligne (fichiers Windows)
  - \t tabulation (fin de « colonne »)
  - \\ la barre inversée même

Texte dans un programme

- Par défaut : considéré corps du programme
  - alors : nom d'une variable, nom d'une fonction, valeur du type numérique...
  - a = sin(3.14)
- Entre guillemets ou apostrophes : valeur du type chaîne de caractères
  - alors : texte que l'interpréteur/compilateur n'essaye pas de comprendre, qui peut néanmoins être traité par l'ordinateur (mémorisé, transformé, affiché...)
  - t = "Message a afficher"
- Après ou entre les caractères spécifiques : commentaire
  - alors : note destinée au programmeur, donc pas à l'ordinateur qui ne peut pas s'en servir d'aucune manière et va l'ignorer entièrement
  - Matlab : à commencer du pour cent % (Octave : ou croisillon #) jusqu'à la fin de la ligne
  - a = sin(3.14) % 3.14=pi radians correspond a 180 degres
  - ou à commencer d'une ligne contenant (seulement) %{ jusqu'à une ligne }%
  - dans d'autres langages (exemples) : après l'instruction Rem (Basic), après le caractère ' (Visual Basic), entre les caractères /* et */ (C)

Affectation

- **Opérateur de l'affectation =**
  - nombre = 0.56
  - texte = 'alpha'
  - v2 = [0 5 2]
  - matrice = [2 3 4; 5 8 10; -1 -5 9]
- Attention au symbole « = »
  - Trompeur si on compare Matlab avec pseudo-code
  - Il y a des langages ou l'affectation est marquée autrement tandis que = désigne la relation de l'égalité

Langage	Affectation (action)	Égalité (relation)
Pseudo-code	a ← 2	a = 2
Pascal	a := 2	a = 2
Matlab, C	a = 2	a == 2
Visual Basic	a = 2	a = 2

- C'est une action
  - Ce n'est pas donc une relation (déclaration/constatation d'un fait persistant, durable)
  - même si dans beaucoup de langages notée à l'aide d'un opérateur
- S'effectue une fois précisément au moment où l'ordinateur y arrive
- On ne peut pas associer deux variables de façon permanente
  - a = 2 # a devient égal 2
    b = a # b devient égal 2
    a = 3 # a devient égal 3
    # b reste égal 2 ≠ a

Gestion d'identifiants

- Si la variable avec le nom donné n'existe pas, elle est créée, c'est-à-dire :
  - son *nom* (« étiquette ») est ajouté au **tableau d'identifiants** (noms, symboles) reconnus
  - son *type* est aussi mémorisé dans ce tableau
  - une place (« boîte ») dans la mémoire est réservée pour garder sa *valeur*
  - l'*adresse* (emplacement, « numéro de la boîte ») de cet espace est aussi mémorisé dans le tableau
- Liste d'identifiants en Matlab
  - who
    affiche une liste de noms de variables définies
  - whos
    ajoute de l'information sur taille de matrice, occupation de la mémoire et classe de variable
- Suppression en Matlab
  - clear *nom*
    supprime la variable ou la fonction portant le nom *nom*
  - l'entrée correspondante est effacée du tableau d'identifiants
  - la place réservé dans la mémoire devient libre et peut ensuite être occupée par une autre variable

Opérateurs arithmétiques

- **Opérateurs arithmétiques** scalaires
  - addition + soustraction -
  - multiplication * division /
  - puissance ^ ou ** (il y a des langages où ça n'existe pas comme opérateur)
  - il y en a plus dans d'autres langages, p. ex. modulo (reste) mod (qui en Matlab est une fonction et non pas un opérateur)
- Matriciels de Matlab
  - matriciels classiques : + - * / \ ^ ** ' (transposition)
    - A / B est équivalent à A * B^-1
    - A \ B est équivalent à A^-1 * B (mais c'est calculé plus vite avec \)
  - terme à terme : + - .* ./ .\ .** .^ – c'est ça qui est nécessaire dans beaucoup d'algorithmes de traitement de données
  - matrice inverse : A^-1 ou inv(A)
  - les tailles des opérandes doivent obéir aux règles de mathématique !
- Combinés affectation et arithmétique (C, Matlab)
  - a+=4 est équivalent à a=a+4 (par analogie : -= *= /=)
  - a++ est équivalent à a=a+1 (par analogie : --)

Interaction avec l'utilisateur (ou avec soi-même lors du développement du programme)

- Affichage le plus simple : *nom_de_la_variable*
  - > a
    a = 5
  - utile pour vérifier le contenu d'une variable à l'étape donnée de l'exécution
- Affichage : disp
  - disp(a)
    affiche la valeur de la variable a
  - disp("Message")
    affiche le texte « Message »
  - Pour afficher un texte suivi d'une valeur, il faut deux *disp* séparés
    disp("Le prix moyen vaut :")
    disp(prix_moy)
- Demande d'une entrée : input
  - prixAchat=input("Entrez le prix d'achat de l'article : ");
    affiche le texte, attend l'entrée d'une valeur et l'enregistre dans la variable *prix_achat*
  - mot=input("Entrez un mot : ", "s");
    pour des chaînes de caractères (variables du type alphanumérique)

Formatage simple de résultats numériques

- format *format*
  - affecte les résultats d'instructions affichés automatiquement
  - et ce que produit *disp*
- *format* peut être :
  - short
    5 chiffres
    > pi
    3.1416
  - long
    15 chiffres
    > pi
    3.14159265358979
  - short e
    5 chiffres + puissance
    > pi
    3.1416e+000
  - long e
    15 chiffres + puissance
    > pi
    3.14159265358979e+000
  - bit
    binaire (base 2)
    > 3
    ans = 11
  - hex
    hexadécimal (base 16)
    > 255
    ans = FF
- Valeurs complexes sont affichées comme
  - 4 + 2i

1.234e+5
veut dire
$1{,}234 \cdot 10^5$

3. Fonctions et portée de variables

Fichiers M, programmes, fonctions
Définition de fonctions, arguments et résultats
Appel de fonctions, passage d'arguments, renvoi de résultats
Portée et espaces de variables

Fichiers M

- Tout programme et fonction Matlab doit être enregistré(e) dans un fichier portant l'extension .m
  - des fichiers programmes sont toujours plein texte, c.-à-d. sans formatage
- Fonction – on va s'y concentrer en TP :
  - commence par la commande *function* et finit par *endfunction*
  - peut posséder (et normalement possède) des arguments **passés** au moment de son **appel** ainsi que des résultats **renvoyés** à la fin de son **exécution**
- Programme :
  - un ensemble quelconque d'instructions (ne commençant par *function*)
  - ne peut pas posséder d'arguments ni de résultats explicites
  - on peut lui fournir des données d'entrée par des fichiers, par l'espace de variables de base ou par la ligne de commandes (sur l'écran)
  - par analogie, lui, il peut fournir des données de sortie
  - son exécution prend place dans l'espace de variables de base
- Pour *lancer un programme* ou *appeler (invoquer) une fonction*
  - on entre juste son nom propre (sans l'extension .m)

Noms (désignations) des fonctions

- Convention générale largement adoptée en génie informatique :
  - Le nom d'une fonction doit être composé d'un verbe (à l'infinitif en français) décrivant ce que la fonction réalise, suivi de substantifs (et possiblement adjectifs) expliquant quelle donnée cela concerne (p. ex. *calculerprixmoyen*)
  - À titre d'exception, une fonction, surtout à fonctionnalité simple ou basique, qui renvoie exactement *un* résultat peut être appelée après ce résultat donc avec un ou plusieurs substantifs ou adjectifs (p. ex. *moyenne*)
- Différentes fonctions ou versions avec un rôle similaire (en TP)
  - S'il n'est pas possible de les distinguer à l'aide d'un substantif ou adjectif, compléter le nom avec le numéro de l'exercice / sous-point (p. ex. *moyenne3a*)
- Exigences de Matlab
  - Le nom ne peut être composé que des caractères autorisés (comme variables)
  - Le nom d'un fichier M (sauf l'extension .m qui y est ajoutée) doit être identique avec celui de la fonction dont il contient la définition
- Convention de Matlab
  - Minuscules seulement sans séparateurs de mots (p. ex. *calculercotisation*)
  - Les minuscules sont en même temps plus sûres du côté système des fichiers

Répertoire de travail

- Tout logiciel (donc l'interpréteur Octave aussi), dans un moment donné, travaille dans un répertoire (dossier) défini
  - C'est là qu'il va chercher des fichiers (y compris fichiers programmes et fonctions Matlab), enregistrer des résultats...
  - L'adresse d'un répertoire est appelé *chemin*
    - p. ex. *c:\users\patricia\My Documents* dans le système Windows 7
    - il est affiché en haut de la fenêtre Explorateur Windows
- Afin d'être trouvable pour Octave, un fichier M doit se trouver dans le répertoire de travail en cours
  - pwd pour vérifier le répertoire de travail en cours
  - cd(*chemin*) pour changer du répertoire de travail en cours
    - cd('h:\octave');
    - Le *chemin* est un texte donc il faut utiliser les " ou ' (avec ", il faut remplacer chaque \ par \\, \ étant un caractère spécial)
- ou dans un répertoire contenu dans la variable *path*
  - addpath(*chemin*) pour ajouter un répertoire à la variable *path*
  - rmpath(*chemin*) pour effacer un répertoire de la variable *path*

Fonctions, arguments, résultats (1)

- Sans arguments ni résultat renvoyé
  - function *nom*
    instructions
    endfunction
  - Cas le plus simple ; puisqu'il manque du renvoi, dans beaucoup de langages ce n'est pas considéré une *fonction* mais un élément à part : *procédure*
  - C'est utile quand :
    - on veut raccourcir le code du programme principal (le rendre plus lisible) – on place un ensemble de commandes dans une fonction séparée
    - un ensemble d'actions est répété – modification plus simple
  - C'est possible quand :
    - les actions à exécuter sont toujours exactement les mêmes ou
    - elles dépendent de variables globales ou de données qui se trouvent hors le programme (p. ex. dans un fichier)
  - Exemple
    - function afficherbienvenue
      disp("Bienvenue dans le système Infores")
      endfunction

Fonctions, arguments, résultats (2)

- Avec des arguments mais sans résultat renvoyé
  - function *nom*(*argument_1*, *argument_2*, ...)
    instructions (où on se sert des argument_x)
    endfunction
  - Cas plus fréquent car normalement
    - le fonctionnement de programmes est (doit être) paramétrable
    - les **paramètres de l'exécution d'une fonction** sont gardées dans des variables locales dans la mémoire
    - ils sont **communiqués = passées** à une fonction en tant que ses **arguments = données d'entrée**
  - Aucun effet de l'exécution de la fonction n'est sauvegardé (à moins que dans une variable globale ou p. ex. dans un fichier)
  - Exemple
    - function afficherbienvenue(prenom)
      disp(["Bienvenue ", prenom, " dans le système Infores"])
      endfunction
  - **Indentation** : les instructions qui appartiennent à la fonction, sont décalées vers la droite par rapport à la ligne de début et à la ligne de fin de la fonction

Portée et espaces de variables

- Un **espace de variables** est un milieu dans lequel des variables sont reconnues et alors peuvent être utilisées
  - Chaque espace possède son propre tableau d'identifiants de variables
- En Matlab il existe :
  - l'**espace de base** – là où vous entrez des commandes en ligne (« invite des commandes », la fenêtre principale Octave) + fichiers M programmes
  - les **espaces des fonctions** – pour chaque fonction, au moment de son chaque appel (et non pas déclaration ou définition), est créé un nouveau, propre espace de variables avec un tableau d'identifiants vide
- Par défaut, une variable est seulement reconnue par l'interpréteur dans l'espace dans lequel elle avait été définie ; on l'appelle alors **variable locale** où **variable à portée locale**
  - Dans l'espace de base ne sont pas reconnues les variables définies dans les fonctions
  - Dans l'espace d'une fonction particulière ne sont pas reconnues les variables définies dans l'espace de base ni dans les espaces d'autres fonctions