TP2 – VARIABLES ET STRUCTURES DE DONNEES DE BASE

1 – VARIABLES ET AFFECTATION

Quand on programme, il est utile de pouvoir stocker de l'information dans des variables. Une ***variable*** est une chaîne de caractères et de nombres associée avec un morceau d'information. L'opérateur d'affectation, dénoté par le symbole "=", est l'opérateur qui est utilisé pour affecter des valeurs aux variables dans MATLAB. La ligne >> **x=1** prend la valeur connue, **1**, et affecte cette valeur à la variable avec le nom "**x**". Après avoir exécuté cette ligne, vous verrez une nouvelle variable apparaître dans la fenêtre de l'espace de travail. Jusqu'à ce que la valeur soit changée ou la variable supprimée, le caractère **x** se comporte comme la valeur **1**.

Ex1: Affecter la valeur **2** à la variable **y**. Multiplier **y** par **3** pour montrer que la variable **y** se comporte comme la valeur **2**.

L'espace de travail (*Workspace*) est une abstraction pour l'espace dans la mémoire de l'ordinateur étant utilisé pour stocker des variables. Pour maintenant, il est suffisant de savoir que la fenêtre des commandes a son propre espace de travail dont le contenu est rendu visuellement disponible dans la fenêtre de l'espace de travail. Vous pouvez voir une liste de toutes les variables dans l'espace de la fenêtre des commandes en utilisant la fonction ***whos***.

Notez que les affectations vont toujours à gauche : cela signifie que la valeur de droite du signe "=" est affectée à la variable à gauche du signe "=". Donc, >> **1 = x** générera une erreur. L'opérateur d'affectation est toujours le dernier dans l'ordre des opérations relatif aux opérateurs mathématiques, logiques et de comparaison.

Il y'a certaines restrictions sur les noms des variables. Ils peuvent contenir seulement des caractères alphanumériques (lettres et chiffres) aussi bien que des soulignés. Cependant, le premier caractère doit être une lettre. La longueur maximum du nom d'une variable est de 255 caractères. Les espaces à l'intérieur du nom d'une variable ne sont pas permis, et les noms des variables sont sensibles à la case (par exemple, x et X seront considérés comme des variables différentes). Vous pouvez supprimer une variable de l'espace de travail en utilisant la fonction ***clear***. En tapant >> **clear x**, la variable x sera éliminée de l'espace de travail. En tapant >> **clear** ou >> **clear all**, toutes les variables seront supprimées de l'espace de travail. En tapant >> **clc**, l'écran sera nettoyé, mais aucune de vos variables ne sera enlevée.

En programmation, les variables sont associées à une valeur d'un certain type. Il y'a plusieurs types de données qui peuvent être affectés à des variables. Un type de données est une classification du type d'information qui est stockée dans une variable. Les types de données de base que nous allons utiliser sont : ***logical, double, char, struct, et cell***. Une description formelle de ces types de données sera donnée dans les sections suivantes.

Tout d'abord, nous donnons un bref aperçu sur les matrices. Une matrice ou un tableau peut être vu comme une table rectangulaire de valeurs, pas nécessairement des valeurs numériques. Un élément d'une matrice est une unité d'information contenue dans une matrice. Un indice d'une matrice est une adresse à l'intérieur de ce tableau. Pour les tableaux à une dimension, l'indice est un entier positif dénotant la position de l'élément en considération. Pour les tableaux à deux dimensions, l'indice est une paire d'entiers positifs qui dénote la ligne et la colonne de l'élément en considération.

Dans MATLAB, chaque valeur est considérée comme une matrice. Les mots sont définis comme une matrice de lettres. Même un seul nombre est considéré comme une matrice 1×1.

Remarque : Un point-virgule peut être utilisé après qu'une variable soit créée pour éviter l'affichage. Par exemple, >> **x = 2 ;** ne montrera pas l'affectation résultante sur l'écran, mais l'affectation à **x** est toujours exécutée. Vous pouvez vérifier cela en regardant dans la fenêtre de l'espace de travail.

2 – LES TABLEAUX DOUBLE

Ex2 : Affecter la valeur **1** à la variable **x**. Vérifier que **x** est un **double** en utilisant la fonction **class**.

Vous pouvez construire des tableaux double dans MATLAB en utilisant des crochets [ ]. Le terme technique pour la fonction fournie par les crochets est appelée concaténation (nom) ou concaténer (verbe). Il est commun de séparer les colonnes par des virgules et les lignes par des points-virgules à l'intérieur d'une concaténation. Vous pouvez créer un tableau vide en plaçant des crochets autour de rien du tout.

Ex3 : Créer les tableaux suivants : $\mathbf{x} = (1 \quad 4 \quad 3)$ $\mathbf{y} = \begin{pmatrix} 1 & 4 & 3 \\ 9 & 2 & 7 \end{pmatrix}$

Remarque : On peut écrire la matrice **y** de différentes façons :

>> y = [x ; 9, 2, 7] ou >> y = [[1, 4, 3] ; [9, 2, 7]] ou y = [[1 ; 9], [4 ; 2], [3 ; 7]]

Ex4 : Créer un tableau vide. Vérifier que c'est un **double** en utilisant la fonction **class**.

Notez que MATLAB toujours résout d'abord les crochets les plus internes. La ligne :
>> [[1 ; 9], [4 ; 2], [3 ; 7]] concaténera [1 ; 9] avant les crochets externes. Si vous essayez de concaténer des matrices qui ne vont pas ensemble (par exemple, elles ne forment pas un rectangle), vous obtiendrez une erreur vous disant que la concaténation horizontale ou verticale est incorrecte.

Plusieurs fois, nous aimerions connaître la taille ou la longueur d'un tableau. La fonction **size** est appelée sur un tableau **M** est retourne un tableau **1 × 2** où le premier élément est le nombre de lignes dans la matrice **M** et le second élément est le nombre de colonnes dans **M**. La fonction **size** vous permet aussi de spécifier la taille de seulement une dimension en utilisant l'entrée **size(M,dim)**. Ainsi size(M,1) retournerait le nombre de lignes dans M, et size(M,2) retournerait le nombre de colonnes dans la matrice M. La fonction **length** est appelée sur un tableau M et retourne le nombre d'éléments dans la matrice M si M est à une dimension. Si M a plus d'une dimension, alors elle retourne la longueur de la plus grande dimension. Il est conseillé de n'utiliser **length** que sur des tableaux à une seule dimension.

Ex5 : Calculer la taille et la longueur des matrices **x** et **y** précédentes. Utiliser la fonction size pour obtenir seulement le nombre de lignes dans **y** et seulement le nombre de colonnes dans **y**.

Pour générer des tableaux qui sont en ordre et régulièrement espacés, il est utile d'utiliser l'opérateur "**:**" de cette façon :

Valeur du début : pas : valeur finale (si le pas n'est pas spécifié, sa valeur par défaut est 1).

Par exemple, **z = 1 : 1 : 2000** (ou bien **z = 1 : 2000**) génèrera le vecteur **z = [1 2 3 … 2000]**. Des pas négatifs ou non entiers peuvent aussi être utilisés. Ainsi, **z = 2 : - .5 : 0** donne **z = [2 1.5 1 0.5 0]**. Si l'incrément rate la dernière valeur, il s'étendra seulement jusqu'à la valeur juste avant la valeur finale. Par exemple, **x = 1 : 2 : 8** donne **x = [1 3 5 7]**.

Des fois par exemple, on a besoin d'un tableau qui commence à 1, se termine à 8, et a exactement 15 éléments. Pour ceci, vous pouvez utiliser la fonction **linspace**. Ainsi, **linspace(a,b,n)** génère un tableau de **n** élément également espacés commençant à **a** et se terminant à **b**.

Ex6 : Utiliser la fonction **linspace** pour générer un tableau commençant à 3, se terminant à 9, et contenant 10 éléments.

Un autre type de tableau qui est hautement structuré est une matrice dans laquelle chaque élément est un **0** ou un **1**. Pour ce but, les fonctions **zeros** et **ones** sont utiles. Ces fonctions prennent en entrées le nombre de lignes et de colonnes et retournent une matrice de 0 ou de 1 de dimension spécifié par les entrées.

Ex7 : Générer une matrice de **1** de dimension **5 × 3** et une matrice **3 × 5** de **0**.

Ex8 : Créer la matrice M en utilisant les fonction **zeros**, **ones**, et les opérateurs : , [ ] et ;.

$$M = \begin{pmatrix} 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 2 & 3 & 4 & 5 & 6 \\ 2 & 4 & 6 & 8 & 0 & 0 \\ 8 & 7 & 2 & 5 & 9 & 0 \end{pmatrix}$$

Dans MATLAB, les indices d'un tableau sont dénotés par des parenthèses attachées au nom de la variable. Si A est une matrice, nous pouvons obtenir l'élément dans la ligne **l** et la colonne **c** en utilisant la notation **A(l , c)**. Ceci est référé comme l'indexation d'une matrice. Vous pouvez spécifier un tableau d'indices pour obtenir de multiples éléments d'un tableau. En d'autres termes, l et c peuvent être des tableaux, et vous pouvez utiliser les opérateurs de création de tableau à l'intérieur de l'indexation. Si vous voulez une ligne ou une colonne entière, vous pouvez abréger cette opération avec l'opérateur **:** . Si vous voulez aller à la fin d'un tableau tout en indexant, vous pouvez utiliser le mot **end**.

Ex9 : Trouver l'élément de la deuxième ligne, troisième colonne en utilisant l'indexation de la matrice **M**.

Ex10 : Extraire de **M**, les éléments dans la troisième ligne et la première et troisième colonne en utilisant l'opérateur [ ].

Ex11 : Extraire tous les éléments de la quatrième ligne de **M** en utilisant le mot **end** ou le raccourci **:**.

Ex12 : Soit **A = [7 3 9 2 4 5]**, extraire le troisième élément de **A** ; puis le troisième, cinquième et sixième élément de **A** ; et enfin le troisième, quatrième et cinquième élément de **A**.

Vous pouvez réaffecter une valeur d'un tableau en utilisant l'indexation du tableau et l'opérateur d'affectation. Vous pouvez réaffecter à plusieurs éléments un seul nombre en utilisant l'indexation du tableau sur le côté gauche. Vous pouvez aussi réaffecter à plusieurs éléments d'un tableau plusieurs valeurs aussi longtemps que le nombre d'éléments étant assignés et le nombre de valeurs assignées est le même.

Ex13 : Soit **A = 1 : 6**. Réaffecter 7 au quatrième élément de **A**. Réaffecter **1** au premier, second et troisième élément de **A**. Réaffecter **9, 8 et 7** respectivement aux second, troisième et quatrième éléments de **A**.

L'arithmétique de base est définie pour les tableaux. Tout d'abord, soit **a** un scalaire et **M** une matrice.

M + a, M - a, M * a et **M / a** additionne **a** à chaque élément de **M**, soustrait **a** de chaque élément de **M**, multiplie chaque élément de **M** par **a**, et divise chaque élément de **M** par **a**, respectivement.

Ex14 : Soit $\mathbf{M} = \begin{pmatrix} \mathbf{1} & \mathbf{2} \\ \mathbf{3} & \mathbf{4} \end{pmatrix}$. Ajouter et soustraire **2** de la matrice **M**. Multiplier et diviser **M** par **2**. Elever au carré chaque élément de **M**.

Il y'a deux types différents de multiplications (et de divisions) de matrices. Il y'a **la multiplication de matrices élément par élément** et **la multiplication de matrices standard**. La multiplication élément par élément est dénotée **.***. Pour des matrices M et P de même taille, M.*P prend chaque élément de M et le multiplie par chaque élément correspondant de P. La même chose est vraie pour **./** et **.^**.

Ex15 : Soit $\mathbf{M} = \begin{pmatrix} \mathbf{1} & \mathbf{2} \\ \mathbf{3} & \mathbf{4} \end{pmatrix}$ et $\mathbf{P} = \begin{pmatrix} \mathbf{1} & \mathbf{3} \\ \mathbf{2} & \mathbf{4} \end{pmatrix}$. Calculer M **.*** P, M **./** P et M **.^** P .

La transposée d'une matrice M est une matrice P, où P(i,j) = M(j,i). En d'autres termes, la transposée permute les lignes et les colonnes de M. Vous pouvez transposer une matrice en utilisant l'opérateur apostrophe, '.

Toutes les fonctions arithmétiques construites dans MATLAB, tel que **sin**, peuvent prendre des **tableaux** comme arguments d'entrée. La sortie est la fonction évaluée pour chaque élément du tableau d'entrée. Une fonction qui prend un tableau comme entrée et effectue la fonction sur lui est dite être **vectorisée**.

Ex16 : Calculer la racine carrée du vecteur **x = [1 4 6 9]**.

Les opérations logiques sont seulement définies entre un scalaire et un tableau et entre deux tableaux de même taille. Entre un scalaire et un tableau, l'opération logique est conduite entre le scalaire et chaque élément du tableau. Entre deux tableaux, l'opération logique est conduite élément par élément.

Ex17 : Vérifier quels sont les éléments du vecteur **x = [1 2 4 5 9 3]** qui sont supérieurs à **3**. Vérifier quels sont les éléments dans **x** qui sont plus grands que les éléments correspondants dans le vecteur **y = [0 2 3 1 2 3].** (**Rép. :** 0 0 1 1 1 0, 1 0 1 1 1 0).

MATLAB peut indexer les éléments d'un tableau qui satisfont une expression logique.

Ex18 : Créer une variable **y** qui contient tous les éléments du vecteur **x** qui sont strictement supérieur à **3**. Affecter à toutes les valeurs de **x** qui sont plus grand que **3**, la valeur **0**.
(**Rép. :** >> y = x (x >3), >> x (x > 3) = 0).

3 – LES TABLEAUX CHAR

Char est un type de données pour stocker des caractères alphanumériques. Un tableau de **char**, habituellement à une dimension, est appelé une **chaîne**. Les chaînes sont assemblées en utilisant des apostrophes sur les deux côtés, mais des crochets peuvent aussi être utilisés pour concaténer des chaines.

Ex19 : Affecter le caractère **S** à la chaine **s**. Affecter la chaîne **Hello World** à la variable **w**. Vérifier que **s** et **w** ont le type **char** en utilisant la fonction **class**.

Noter qu'un espace vide, ' ', entre Hello et World est aussi un caractère. N'importe quel symbole peut être un caractère, même ceux qui ont été réservés pour les opérateurs.

Ex20 : Affecter le caractère **+** à la variable **p**. Vérifier que **p** ne se comporte pas comme l'addition en tapant la commande >> **1 p 2**.

Remarque : Les nombres peuvent aussi être exprimés comme caractères. Par exemple, **x** = '**123**' signifie que **x** est la **chaîne 123** et non le nombre **123**.

Ex21 : Ajouter **1** à la chaîne **1**. Ajouter **1** à la chaîne **123**.

Ex22 : Créer la chaîne **L'école**. Le caractère apostrophe **'** est représenté par deux apostrophes.

Les chaînes peuvent être concaténées ensemble, verticalement ou horizontalement, en utilisant les crochets juste comme les tableaux **double**.

Ex23 : Créer les chaînes **s1='Hello'** et **s2='World'** et utiliser la concaténation de **s1** et **s2** pour former la chaîne **s3 = 'Hello World'**. Souvenez-vous de l'espace entre les mots !

Les tableaux char peuvent être indexés de la même manière que les tableaux **double**, incluant l'indexation de tableaux. Une chaîne vide peut être créée avec deux apostrophes ' '.

Ex24 : Créer une chaîne vide. Vérifier que la chaine vide est de type **char**.

La fonction **sprintf** écrit de nouvelles données à une chaîne pré formatée. Vous pouvez y insérer des chaînes avec **%s**, des entiers avec **%d**, ou des nombres plus génériques avec **%f** et avec **%g**. Vous pouvez aussi contrôler le nombre de chiffres insérés dans la chaîne formatée.

Ex25 : **s** = **sprintf** ('C''est le TP d''%s et il y a %d étudiants','informatique', 15)