Cours Apl 03 : Un peu de vocabulaire, notions fondamentales.

- Variables, constantes :

Une variable, comme en mathématiques, est le contraire d'une constante.
Par exemple dans X=5
X est une variable de valeur 5.
X peut prendre des valeurs différentes, mais 5 sera toujours 5.

- Fonctions primitives ou utilisateur :

Une fonction dite "primitive" est une fonction naturellement fournie dans Apl. Ces fonctions font partie des briques de base du langage Apl.
Par exemple l'addition est une primitive : +

Une fonction fait une action et rend généralement un résultat.
2 x 3 calcule 2 multiplié par 3 et rend donc 6.
Vous pouvez le tester en le tapant en en faisant "Entrée".
Apl étant un langage de programmation, vous pouvez assembler des fonctions primitives pour écrire vos propres fonctions.

Votre première fonction : Moyenne

Entrez la commande suivante :

```
)ed Moyenne
```

Un nouvel écran s'ouvre (l'éditeur), vous permettant de saisir les instructions qui composeront votre fonction de calcul d'une moyenne.
Modifiez la première ligne de manière à obtenir ceci :

```
r← Moyenne d
```

Ce qui signifie qu'on affectera à r le résultat de la fonction "Moyenne".
Les valeurs dont on calculera la moyenne sont portées par l'argument droit qu'on nommera "d".
r et d ne sont pas des noms obligatoires, mais il peut être intéressant d'adopter une norme et de l'appliquer dans toutes les fonctions.

Entrez alors cette expression sur la ligne suivante :

```
r← (+/,d)÷⍴,d
```

Elle signifie que le résultat de la fonction est la somme des éléments de d divisée par son nombre d'éléments.
Pour sortir de l'éditeur de fonction en validant votre saisie, utilisez la touche "escape".

- On appelle espace de travail ou Workspace, les variables (données), écrans, et fonctions que vous avez écrits, tout comme un document excel peut se composer de feuilles avec des données, et éventuellement des fonctions Visual Basic.

- Types des variables :

Concernant les types de variables, Apl a le mérite d'être très simple.
Il n'existe en effet que 2 types :

- **Alphabétiques** : le mot 'bonjour' est de type alphabétique.
  On ne peut pas faire d'opérations arithmétiques sur ce type d'objets.
  'deux' + 'un' provoquera un message d'erreur "Domain Error" indiquant que la fonction "plus" n'est pas utilisée avec des arguments de type approprié.

- **Numériques** : les variables numériques sont composées exclusivement de nombres.

- Dimensions des objets et variables :

Apl étant un langage très puissant pour manipuler des objets de dimensions diverses, il sera intéressant de les parcourir préalablement.

- la structure la plus simple est le "**scalaire**". C'est un objet sans dimension. Il n'a ni longueur, ni largeur, ...
  Par exemple le nombre 12 est un scalaire.

  - En ajoutant une dimension, on obtient un **vecteur**. Par exemple on pourrait avoir une variable Mois dont le contenu serait une suite de nombres de 1 à 12.
    ```
    Mois ← 1 2 3 4 5 6 7 8 9 10 11 12
    ```
    Mois est un vecteur numérique de dimension 12.
    Pour la connaître, on tape simplement :
    ```
          ⍴Mois
    ```
    12

  - Une dimension de plus et on passe aux **matrices**, l'équivalent d'une feuille de tableur.
    Par exemple, pour créer une matrice des 3 meilleurs élèves avec en 1ère colonne le prénom et en 2ème leur moyenne, on procèdera comme suit :
    ```
    Tierce ← 3 2 ⍴ 'Ennio' 17 'Thérèse' 16 'Dominique' 15
    ```

    Si on veut voir le résultat, on tape le nom de la variable suivi de Entrée et son contenu s'affiche dessous.

    ```
          Tierce
    Ennio      17
    Thérèse    16
    Dominique  15
    ```

Quand nous avons créé la variable Tierce, nous avons mis les prénoms entre ' ' afin d'indiquer à Apl qu'il s'agissait de "constantes" alphabétiques et pas de noms de variables dont on aurait voulu utiliser le contenu.

La dimension de Tierce est 3 2.
Pour connaitre la dimension d'une variable on utilise la fonction ⍴ (Rho) avec en argument droit la variable.

```
      ⍴ Tierce
3 2
```

On a également utilisé ⍴ pour constituer Tierce, mais cette fois avec 2 arguments (mode dyadique). L'argument gauche indique la dimension souhaitée, et l'argument droit porte les données qui rempliront la variable.

Attention !
Apl est sensible à la casse.
Non pas qu'il soit particulièrement fragile, mais pour lui un "A" et un "a" sont aussi différents qu'un "a" et un "z". Donc si pour afficher Tierce, vous tapez tierce, apl vous répondra "Value Error", ce qui signifie que l'objet auquel vous faites référence lui est inconnu, qu'il ne correspond à aucune valeur connue.

- Avec 3 dimensions (plans, lignes, colonnes) on passe aux **cubes**.
- Au delà, on entre dans le domaine des **hyper cubes**.

- Fonctions Monadiques, Dyadiques, arguments.

Nous venons de voir que la fonction ⍴(rhô) a été utilisée de 2 manières.
Avec un seul argument à droite (mode monadique), elle rend la dimension de l'argument.
Avec 2 arguments (mode dyadique) elle rend un objet dont la taille est donnée en argument gauche, et est rempli avec ce qui est dans l'argument droit.
Exemple : 3 ⍴ 5 rend
5 5 5

- Toutes les primitives Apl possèdent les 2 modes de fonctionnement.
- Si une fonction n'a qu'un argument, il est forcément à droite.

- Affectation

Les fonctions rendent généralement un résultat. Si celui-ci est affecté à une variable ou utilisé comme argument droit par une fonction immédiatement à gauche,aucun résultat n'est affiché dans la session. En revanche, si il n'est ni affecté, ni réutilisé, Apl l'affiche dans la session.

Exemples :

```
      2 + 3
6
      res ← 4 × 4          rien ne s'affiche
      2 + res ← 4 × 4
18
```

Apl a calculé 4 × 4, l'a affecté à res, auquel il a ajouté 2. Comme le résultat de 2 + res n'est affecté à rien, Apl l'a affiché dans la session.

Travaux pratiques :

1. Créez le vecteur Monvec composé des nombres 10 20 30 40 et affichez-en la taille.

2. Créez le vecteur Coeff 2 1 10 100 et multipliez le par Monvec.

3. En utilisant la fonction Moyenne, calculez la moyenne de Monvec. Si vous n'aviez pas écrit la fonction, écrivez-la comme indiqué dans le cours.

4. Qu'indique ρ ρ Monvec ?

5. Que se passe-t-il si on fait :
3 x Monvec

puis :

3 x 'Monvec'
Pourquoi cette dernière expression ne fonctionne-t-elle pas ?

6. Créez une matrice Mnum1 de 3 lignes et 4 colonnes composée des nombres suivants :

10	20	30	40
1	2	3	4
1	1	1	1

7. Affichez la taille de Mnum1.

8. Affichez la moyenne de Mnum1.

9. Affichez la taille du mot 'Bonjour'

10. Additionnez Monvec et Coeff.

11. Sauvez votre travail dans un workspace 'pratique-apl'
)wsid c:\Mes documents\pratique-apl
)save

Solutions :

1. Créez le vecteur Monvec composé des nombres 10 20 30 40 et affichez-en la taille en utilisant la fonction ⍴.

```
 Monvec←10 20 30 40
⍴Monvec
```

2. Créez le vecteur Coeff 2 1 10 100 et multipliez le par Monvec.

```
 Coeff  ←2 1 10 100
Monvec x Coeff
```

3. En utilisant la fonction Moyenne, calculez la moyenne de Monvec.

```
Moyenne Monvec
```

4. En utilisant la fonction ⍴ affichez la taille de Monvec.

```
⍴Monvec
```

5. Que se passe-t-il si on fait :
3 x Monvec
on obtient Monvec avec chacun de ses éléments multipliés par 3.
30 60 90 120

puis :

3 x 'Monvec'
Pourquoi cette dernière expression ne fonctionne-t-elle pas ?
On a un message d'erreur : Domain Error
En effet, dans ce cas, 'Monvec' n'est pas un nom de variable mais une simple chaîne de caractères, puisqu'il est placé entre quotes.
Et en Apl, multiplier une chaîne par un nombre n'a aucun sens.

6. Créez une matrice Mnum1 de 3 lignes et 4 colonnes composée des nombres suivants :

10	*20*	*30*	*40*
1	*2*	*3*	*4*
1	*1*	*1*	*1*

```
 Mnum1 ←3 4⍴10 20 30 40 1 2 3 4 1 1 1 1
```

7. Affichez la taille de Mnum1.

```
⍴Mnum1
```

8. Affichez la moyenne de Mnum1.

```
Moyenne Mnum1
```

9. Affichez la taille du mot 'Bonjour'

```
⍴'Bonjour'
```

10. Additionnez Monvec et Coeff.

```
Monvec + Coeff
```