Cours Apl 06 : Fonctions arithmétiques et mathématiques

Commençons par les 4 opérations de base : + - x ÷

A travers les quelques exemples suivants, vous allez commencer à percevoir la puissance du langage Apl.

- Comment **additionner** les éléments de 2 matrices de taille identique, telles que Mnum1 et Mnum3 (voir travaux pratiques) ?

C'est très simple, il n'y a qu'à demander :

```
      Mnum1 + Mnum3
11 22 33 44
 6  8 10 12
10 11 12 13
```

- Vous pouvez également tester les autres opérations.

- Pour faire une opération entre 2 objets il est nécessaire que leurs dimensions soient identiques ou au moins "compatibles".

Par exemple `Mnum1 ÷ 10` donnera

```
1   2   3   4
0.1 0.2 0.3 0.4
0.1 0.1 0.1 0.1
```

Dans cet exemple, le scalaire 10 s'est "propagé" sur tous les éléments de Mnum1.
Ce comportement paraît en effet logique.

Par contre, à la place d'Apl, que feriez-vous si on vous demandait :

Mnum1 x 2 1 ⍴ 10

Moi, je ne comprendrais pas ce qu'on me demande. Que faut-il multiplier entre 2 matrices de dimensions totalement différentes ?

Et bien, Apl non plus ne comprendra pas cette demande.
Et si vous lui posez la question, il vous répondra un magnifique "Lenght Error", ce qui signifie que les dimensions des objets ne sont pas compatibles.

Par contre, un vecteur et une matrice peuvent tout à fait interragir pour peu que le vecteur fasse le même nombre de lignes ou de colonnes que la matrice.

Par exemple créez le vecteur suivant :

```
      Vnum1 ← 10 100 1000
```

On peut tenter d'ajouter une colonne à Mnum1 :

```
      Mnum1,Vnum1
10 20 30 40   10
 1  2  3  4  100
 1  1  1  1 1000
```

On peut également multiplier la première ligne par 10, la deuxième par 100 et la troisième par 1000.
Pour cela il faut précisez à Apl qu'on travaille sur la 1ère dimension. En effet, on applique chaque élément de Vnum1 à une ligne de Mnum1.

```
      Mnum1×[1]Vnum1
 100  200  300  400
 100  200  300  400
1000 1000 1000 1000
```

On pourrait aussi appliquer un coefficient différent sur chaque colonne. Par exemple 10%, 15% et 20%

```
      Mnum1×[1].1 .15 .20
1      2   3    4
0.15 0.3 0.45 0.6
0.2  0.2 0.2  0.2
```

- Fonctions d'arrondis, minimum et maximum

2 fonctions primitives assurent ces différentes tâches.

- **Plafond (⌈)** en mode monadique arrondi un nombre à l'entier supérieur.
En mode dyadique, cette fonction rend le maximum entre les arguments gauche et droit.
Les arguments gauche et droit doivent avoir les mêmes dimensions, sauf si l'un d'entre eux est un scalaire, auquel cas il pourra se propager sur l'autre argument.
Exemples :

```
        ⌈ 3.5 ¯2.5 2 1.1
4 ¯2 2 2

      8 ⌈ 3 10 12 ¯4
8 10 12 8
```

Cette fonction, contrairement à ce que son nom laisse penser, est fort utile pour définir des planchers puisqu'elle rend la plus grande valeur.
Imaginons par exemple qu'un généreux élu décide que dans chaque foyer fiscal on considérera qu'il y a au moins une part et demi, au lieu d'une actuellement.
Si on appelle Nbp le nombre de parts, on dit que son plancher est à 1.5 et donc en Apl on écrit :

```
Nbp ← Nbp ⌈ 1.5
```

- la fonction **plancher (⌊)** fait les mêmes choses mais vers le bas (c'est elle que l'on utilise pour plafonner des valeurs).
Exemple :

```
      1 3 3 ⌊ 4 2 1
1 2 1
```

- Autres fonctions :

- Modulo : |
Cette fonction rend le reste de D divisé par G
Exemple :

```
      2 | 5
1
      3 | 9
0
```

Pour savoir si un nombre est pair, il suffit de vérifier que le reste de sa division par 2 est égal à 0.

```
      2|10
0
```

- Il existe un certain nombre d'autres fonctions que vous trouverez dans l'aide en ligne (factorielle, inversion de matrice, logarithmes, etc ...)

Travaux pratiques :

0. Chargez votre Ws de travaux pratiques :
)load c:\Mes documents\pratique-apl

1. Affichez les dimensions de Mnum1, Mnum2 et Mnum3.
Quelles matrices sont directement compatibles en terme de dimensions ?

2. Multipliez Mnum2 par la première ligne de Mnum1. Plus précisément, la première ligne de Mnum2 par la première ligne, première colonne de Mnum1, la deuxième ligne par la première ligne, deuxième colonne de Mnum1, etc ...

3. Affichez une matrice représentant le reste de la division par 2 de chaque élément de Mnum1.

4. Affichez Mnum2 en plafonnant chacun de ses éléments à 35.
Affichez Mnum2 avec 25 comme valeur plancher

5. Arrondissez les éléments de ce vecteur à l'entier inférieur : 10.5 3.8 4.2 1.1 1.9

6. Comment arrondir à l'entier le plus proche ?
L'algoritme est le suivant :
- on ajoute 0.5 à chaque élément
- on arrondi à l'entier inférieur
Arrondissez le vecteur précédent à l'entier le plus proche.

7. Pour arrondir à n chiffres après la virgule, on procède comme pour le point précédent mais on commence par déplacer la virgule de n positions à droite en multipliant par 10 puissance n (x 10 * n) et en ramenant en final la virgule de n positions à gauche par l'opération inverse (x 10 * ¯n)
Calculez Mnum1÷Mnum3, puis affichez le arrondi à 0, 1, puis 2 chiffres après la virgule.

8. Amateurs de loto, soyez heureux, Apl a été conçu pour vous !
Apl comporte en effet une fonction permettant de tirer G nombres parmi D.
Par exemple 2 ? 10 rend 2 nombres compris entre 1 et 10.
Votre mission : Affichez un vecteur de 6 nombres tirés parmi 49.
Recommencez autant de fois que nécessaire pour aller déposer vos grilles.

9. Constituez une matrice de coefficients tirés au hasard, de dimension 3 lignes, 4 colonnes avec des coefficients pouvant aller de .01 à 1 et multipliez la par Mnum3.
Arrondissez le résultat de ce calcul à 1 chiffre après la virgule.

10. Affichez une matrice représentant les plus grandes valeurs entre chaque élément de Mnum1 et de Mnum3.

11. Sauvez votre travail :
)save

Solutions :

1. Affichez les dimensions de Mnum1, Mnum2 et Mnum3.

```
      ⍴Mnum1
3 4
      ⍴Mnum2
4 6
      ⍴Mnum3
3 4
```

Quelles matrices sont directement compatibles en terme de dimensions ?
Mnum1 et Mnum3 car leurs dimensions sont identiques.

2. Multipliez Mnum2 par la première ligne de Mnum1. plus précisément, la première ligne de Mnum2 par la première ligne, première colonne de Mnum1, la deuxième ligne par la première ligne, deuxième colonne de Mnum1, etc ...

```
      Mnum2×[1],1↑[1]Mnum1
```

3. Affichez une matrice représentant le reste de la division par 2 de chaque élément de Mnum1.

```
      2|Mnum1
```

4. Affichez Mnum2 en plafonnant chacun de ses éléments à 35.

```
      Mnum2⌊35
```

Affichez Mnum2 avec 25 comme valeur plancher.

```
      Mnum2⌈25
```

5. Arrondissez les éléments de ce vecteur à l'entier inférieur : 10.5 3.8 4.2 1.1 1.9

```
      ⌊10.5 3.8 4.2 1.1 1.9
```

6. Comment arrondir à l'entier le plus proche ?
L'algorithme est le suivant :
- on ajoute 0.5 à chaque élément
- on arrondi à l'entier inférieur
Arrondissez le vecteur précédent à l'entier le plus proche.

```
      ⌊.5+10.5 3.8 4.2 1.1 1.9
```

*7. Pour arrondir à n chiffres après la virgule, on procède comme pour le point précédent mais on commence par déplacer la virgule de n positions à droite en multipliant par 10 puissance n (x 10 * n) et en ramenant en final la virgule de n positions à gauche par l'opération inverse (x 10 * ¯n)*
Calculez Mnum1÷Mnum3, puis affichez le arrondi à 0, 1, puis 2 chiffres après la virgule.

```
        Mnum1÷Mnum3
```

Arrondir à 0 chiffre après la virgule :

```
⌊.5+Mnum1÷Mnum3
```

Arrondir à 1 chiffre après la virgule :

```
.1×⌊.5+10×Mnum1÷Mnum3
```

Arrondir à 2 chiffres après la virgule :

```
(10*¯2)×⌊.5+(10*2)×Mnum1÷Mnum3
```

Décomposons par groupes d'opérations de droite à gauche :
- `Mnum1÷Mnum3` : division elle-même
- `(10*2)×` : on pousse la virgule de 2 rangs à droite
- `⌊.5+` : on ajoute 0.5 et on arrondi à l'entier inférieur
- `(10*¯2)×` : on ramène la virgule à sa position initiale.

8. Affichez un vecteur de 6 nombres tirés parmi 49.
Recommencez autant de fois que nécessaire pour aller déposer vos grilles.

```
        6?49
        6?49
        6?49
```

9. Constituez une matrice de coefficients tirés au hasard, de dimension 3 lignes, 4 colonnes avec des coefficients pouvant aller de .01 à 1 et multipliez la par Mnum3.

```
        Mnum3×3 4ρ.01×12?100
```

Arrondissez le résultat de ce calcul à 1 chiffre après la virgule.

```
.1×⌊.5+10×Mnum3×3 4ρ.01×12?100
```

10. Affichez une matrice représentant les plus grandes valeurs entre chaque élément de Mnum1 et de Mnum3.

```
        Mnum1⌈Mnum3
```