10 - Séquences, listes, ensembles (exemples)

1. Séquences

Voici un exemple de *concaténation* de séquences :

```
> s :=1,2,4,8 : t :=s,16 : u :=t,s ;
```

$$u := 1, 2, 4, 8, 16, 1, 2, 4, 8$$

On forme la séquence de tous les k, à partir de $k = 1.7$, de 1 en 1, sans dépasser 8.3 :

```
> seq(k,k=1.7..8.3) ;
```

$$1.7, 2.7, 3.7, 4.7, 5.7, 6.7, 7.7$$

Voici la séquence des $k(k+1)$, de $k = 1$ à $k = 5$:

```
> restart : seq(k*(k+1),k=1..5) ;
```

$$2, 6, 12, 20, 30$$

On reprend le même exemple, mais en donnant au préalable la valeur 10 à la variable `k`.
On voit que le fonctionnement de `seq` n'en est pas pour autant perturbé.

```
> k :=10 : seq(k*(k+1),k=1..5) ;
```

$$2, 6, 12, 20, 30$$

Après un `restart`, on forme encore la même séquence, cette fois avec l'opérateur `$` :

```
> restart : k*(k+1) $ k=1..5 ;
```

$$2, 6, 12, 20, 30$$

Nouvel essai, après avoir donné une valeur à la variable k.

Maple répond ici par un message d'erreur.

En effet, il évalue l'expression en "`110 $ 10=1..5`" avant d'évaluer l'opérateur `$`.

Contrairement à l'instruction `seq`, l'opérateur `$` ne *protège* donc pas l'indice de sommation.

Il aurait fallu écrire `'k*(k+1)' $ 'k'=1..5`.

Si on écrit `k*(k+1) $ 'k'=1..5`, on obtient la séquence $110, 110, 110, 110, 110$.

```
> k :=10 : k*(k+1) $ k=1..5 ;
Error, wrong number (or type) of parameters in function $
```

L'opérateur `$` est plus indiqué pour créer rapidement ce genre de séquence :

```
> $3..7 ; 7$5 ;
```

$$3, 4, 5, 6, 7$$

$$7, 7, 7, 7, 7$$

Voici comment calculer le nombre d'éléments dans une séquence :

```
> restart : s :=a,e,d,c,e,c,a,a,b,c ; nops([s]) ;
```

$$s := a, e, d, c, e, c, a, a, b, c$$

$$10$$

Voici maintenant comment calculer le nombre d'éléments *différents* dans une séquence s.

On convertit d'abord s en un ensemble pour en éliminer tous les doublons.

```
> restart : s :=a,e,d,c,e,c,a,a,b,c ; nops({s}) ;
```

$$s := a, e, d, c, e, c, a, a, b, c$$

$$5$$

2. Listes

On place une liste dans la variable `L`. On voit que les éléments restent dans l'ordre défini par l'utilisateur et qu'il peut y avoir des doublons. L'instruction `convert(L,multiset)` permet de savoir combien de fois chaque élément figure dans cette liste.

```
> restart : L :=[a,b,c,d,c,a,c] : nops(L) ; op(L) ; convert(L,multiset) ;
```

$$7$$

$$a, b, c, d, c, a, c$$

$$[[c, 3], [d, 1], [a, 2], [b, 1]]$$

L'instruction `sort` trie les listes de nombres par ordre croissant.

Voici comment les trier par ordre décroissant :