Chapitre II : Structures de données Maples

Commandes op et whattype

La commande op permet de renvoyer les opérandes d'un objet Maple.

```
> op(3/2);
```

$$3, 2 \qquad \textbf{(1.1)}$$

```
> op(x^2);
```

$$x, 2 \qquad \textbf{(1.2)}$$

```
> op(x^2+1);
```

$$x^2, 1$$

$$x^2, 1$$

La commande whattype(...) permet de connaitre le type d'un objet Maple.

```
> whattype(2.);
```

float

```
> whattype(2/3);
```

fraction

```
> whattype(sqrt(3));
```

`^`

```
> whattype(Pi);
```

symbol

Intervalles

MAPLE permet de définir des intervalles sous la forme de deux termes séparés par au moins deux points.
Comme tout objet MAPLE un intervalle peut avoir un nom (ne pas utiliser les noms int, Int ou I qui sont des noms protégés de MAPLE qui renverrait un message d'erreur).

```
> J1:=2..3;
```

$J1 := 2..3$

```
> J2:=a..b;
```

$J2 := a..b$

```
> whattype(J1);
```

..

```
> op(J1);
```

$2, 3$

```
> op(J2);
```

$$a, b$$

```
> Int:=x..y;
```

```
Error, attempting to assign to `Int` which is protected
```

Les intervalles ne sont pas utilisés directement en général, ils apparaissent dans d'autres commandes.

```
> int(x^2+1,x=2..3);
```

$$\frac{22}{3}$$

```
> int(x^2 + 1, x = 2.0 ..3);
```

$$7.333333333 \qquad \textbf{(2.1)}$$

```
> int(x^2 + 1, x = a ..b);
```

$$\frac{1}{3} b^3 - \frac{1}{3} a^3 + b - a \qquad \textbf{(2.2)}$$

La commande Int est une forme inerte, elle permet uniquement l'affichage du symbole de l'intégration.

```
> Int(x^2+1,x=a..b)=int(x^2+1,x=a..b);
```

$$\int_a^b \left(x^2 + 1\right) dx = \frac{1}{3} b^3 - \frac{1}{3} a^3 + b - a$$

Suites

Une suite "sequence" est une suite d'objets MAPLE séparés par des virgules.

```
> suite:=a,b,1,1,3,a*x+1,b,exp(x)-y,{};
```

$$suite := a, b, 1, 1, 3, a\,x + 1, b, e^x - y, \{\ \}$$

```
>
```

On peut multiplier une suite par une constante

```
> -3*suite;
```

$$-3\,a, -3\,b, -3, -3, -9, -3\,a\,x - 3, -3\,b, -3\,e^x + 3\,y, -3\ \{\ \}$$

On peut inclure une suite dans une autre suite.

```
> x,y,suite,15;
```

$$x, y, a, b, 1, 1, 3, a\,x + 1, b, e^x - y, 15$$

```
> suite:=suite,suite;
```

$$suite := a, b, 1, 1, 3, a\,x + 1, b, e^x - y, a, b, 1, 1, 3, a\,x + 1, b, e^x - y$$

```
> suite^2;
```

$$\left(a, b, 1, 1, 3, a\,x + 1, b, e^x - y, \{\ \}\right)^2 \qquad \textbf{(3.1)}$$

Les suites Maple ont le type **exprseq** comme le montre la commande whattype.

```
> whattype(suite);
```

$$exprseq$$

On peut extraire le n-ième élément d'une suite s avec la notation **s[n]**

```
> suite[1];
```

$$a$$

On peut également utiliser un intervalle pour accéder à une partie des éléments d'une suite.

```
> suite[2..6];
```

$$b, 1, 1, 3, a\,x+1$$

On ne peut pas affecter un élément d'une suite.

```
> suite[3]:=alpha;
Error, invalid assignment (a, b, 1, 1, 3, a*x+1, b, exp(x)-y,
a, b, 1, 1, 3, a*x+1, b, exp(x)-y)[3] := alpha; cannot assign
to an expression sequence
```

Génération automatique de suites, fonction seq

La fonction seq permet la création automatique de suites avec une boucle indexée par un entier.

```
> s:=seq(i^2,i=0..10);
```

$$s := 0, 1, 4, 9, 16, 25, 36, 49, 64, 81, 100$$

```
> op(x^2+1);
```

$$x^2, 1$$

```
> whattype(%);
```

$$exprseq$$

```
> op(op(x^2+1)[2]);
```

$$1$$

Ensembles (set)

Un ensemble est une structure de données de MAPLE définie syntaxiquement par une suite encadrée par les accolades: {.,.,.}.
Conformément à la définition des ensembles, l'ordre d'entrée des éléments peut être changé par MAPLE puisqu'il est sans importance et les éléments répétés sont considérés comme un élément unique.

```
> restart:
  E:={x,y,z,x};
```

$$E := \{x, y, z\}$$

Les ensembles possèdent le type **set** :

```
> whattype(E);
```

$$set$$

On peut transformer un ensemble en une suite de ses éléments il suffit de lister ses opérandes grace à la commande **op**

```
> E:=op(E);
```

$$E := z, x, y$$

```
> whattype(E);
```

$$exprseq$$

On peut accéder aux éléments d'un ensemble, il faut tenir compte que le classement des éléments est

celui de Maple et non celui de l'affectation.

```
> G:={x,y,z,u,v};
```

$$G := \{u, v, x, y, z\}$$

```
> G[1];
```

$$u$$

Comme pour les suites on ne peut pas affecter l'élément d'un ensemble.

```
> G[1]:=x;
Error, cannot reassign the entries in a set
```

L'opérateur **in** permet d'écrire des relations d'appartenance, la commande **evalb** permet d'évaluer la valeur booléenne d'une expression logique, elle retourne true pour vrai, false pour faux, et FAIL pour échec.

```
> evalb(x in E);
```

$$true$$

Sous Ensembles

On peut créer des ensembles en respectant leurs propriétés, entiers, rationnels, réels, la commande SetOf(option) permet de créer un ensemble particulier.

```
> Pi in SetOf(integer);
```

$$\pi \in SetOf(integer)$$

```
> evalb(%);
```

$$false$$

```
> evalb(1 in SetOf(integer));
```

$$true$$

```
> evalb(11/3 in SetOf(integer));
```

$$false$$

```
> Q:=SetOf(rational);
```

$$Q := SetOf(rational)$$

```
> evalb(Pi in Q);
```

$$false$$

```
> evalb(11/3 in Q);
```

$$true$$

```
> evalb(1. in Q);
```

$$false$$

La commande logique **subset** teste si un ensemble est un sous ensemble d'un autre ensemble et retourne la valeur logique True si c'est vrai et False si c'est faux

```
> E:={x,y,z};G:={y};
```

$$E := \{x, y, z\}$$

$$G := \{y\}$$

```
> G subset E;
```

true

```
> E subset G;
```

false

Opérations sur les ensembles

Les commandes **union,intersect,minus** permettent d'effectuer respectivement l'union l'intersection et la différence entre ensembles.

```
> restart:
  E:={x,y,z,t,u,v}; G:={alpha,beta,t};
```

$$E := \{x, y, z, u, t, v\}$$

$$G := \{t, \alpha, \beta\}$$

```
> E union G;
```

$$\{x, y, z, u, t, v, \alpha, \beta\}$$

```
> E intersect G;
```

$$\{t\}$$

```
> E minus G;
```

$$\{x, y, z, u, v\}$$

```
> G minus E;
```

$$\{\alpha, \beta\}$$

```
> Z union Q;
```

$$Z \cup Q$$

```
> M:={(x+1)^2-x^2-2*x-1,1,2,3,4};
  N:={0,1,2,3,4};
```

$$M := \{1, 2, 3, 4, (x+1)^2 - x^2 - 2x - 1\}$$

$$N := \{0, 1, 2, 3, 4\}$$

```
> M union N;
```

$$\{0, 1, 2, 3, 4, (x+1)^2 - x^2 - 2x - 1\}$$

```
> M intersect N;
```

$$\{1, 2, 3, 4\}$$

```
> M minus N;
```

$$\{(x+1)^2 - x^2 - 2x - 1\}$$

```
> N minus M;
```

$$\{0\}$$

Listes (list)

Une liste est une structure de données de MAPLE représentée syntaxiquement par une suite

d'éléments encadrée par des crochets: [.,.,.]. Contrairement aux ensembles, l'ordre des éléments est important et les répétitions sont préservées. Ces objets jouent, comme les ensembles, un rôle essentiel dans MAPLE.

```
> L:=[x,y,z,x];
```

$$L := [x, y, z, x]$$

```
> whattype(L);
```

$$list$$

```
> L := op(L);
```

$$L := x, y, z, x \quad \textbf{(7.1)}$$

```
> whattype(L);
```

$$exprseq \quad \textbf{(7.2)}$$

Comme pour les ensembles on peut transformet une suite en liste

```
> U:=0,1,2,3,4;
```

$$U := 0, 1, 2, 3, 4$$

```
> whattype(U);
```

$$exprseq \quad \textbf{(7.3)}$$

```
> U:=[U];
```

$$U := [0, 1, 2, 3, 4]$$

```
> whattype(U);
```

$$list$$

On peut transformet une liste en suite en listant ses opérandes.

```
> U:=op(U);
```

$$U := 0, 1, 2, 3, 4$$

```
> whattype(U);
```

$$exprseq$$

Question : comment faire pour ajouter à une liste un nouvel élément ?

```
> V:=[seq(i^2,i=0..4)];
```

$$V := [0, 1, 4, 9, 16]$$

On peut coupler la commande **seq** avec la commande **in** pour créer des suites qui ne prennent leurs indices dans un sous ensemble prédéfinis.

```
> L1:=[7,9,11,13];
```

$$L1 := [7, 9, 11, 13]$$

```
> W:=[seq(i^2,i in L1)];
```

$$W := [49, 81, 121, 169]$$

Contrairement aux suites et aux ensembles on peut modifier un élément d'une liste.

```
> L[2]:=alpha;
```

$$L_2 := \alpha$$

```
> L;
```

$$[x, \alpha, z, x]$$

Tri des éléments d'une suite.
Le tri des éléments d'une liste se fait par la commande **sort** , elle possède entre autre les options `<`, `>` **,numeric**,**lexorder**,**length.**
L'option **numeric** indique le tri dans l'ordre ascendant des éléments de type **numeric.**
Les options `<` et `>` permettent le tri dans l'ordre ascendant et descendant respectivement.
L'option **lexorder** permet de trier les éléments d'une liste de type chaine de caractère dans l'ordre lexicographique.
L'option length permet de trier les éléments d'une liste de type chaine de caractère selon la taille.

```
> restart:
  L:=[infinity,-5,3,0,4];
```

$$L := [\infty, -5, 3, 0, 4]$$

```
> sort(L,numeric);
```

$$[-5, 0, 3, 4, \infty]$$

```
> sort(L,`<`);
```

$$[-5, 0, 3, 4, \infty]$$

```
> sort(L,`>`);
```

$$[\infty, 4, 3, 0, -5]$$

```
> L:=[x,xy,y,z,a,b];
```

$$L := [x, xy, y, z, a, b]$$

```
> sort(L,lexorder);
```

$$[a, b, x, xy, y, z]$$

```
> sort(L,length);
```

$$[x, y, z, a, b, xy]$$

```
> L:=[x,xy,-5,z,3,2,1];
```

$$L := [x, xy, -5, z, 3, 2, 1]$$

```
> sort(L);
```

$$[-5, 1, 2, 3, x, xy, z]$$

```
> sort(L,lexorder);
Error, invalid input: a string/name list is expected for sort
method `lexorder`
> sort(L,`>`);
Error, invalid input: a numeric list is expected for sort
method ``>``
```

Addition & soustraction de listes.
On peut additionner et soustraire des listes.
On peut également multiplier et diviser les éléments d'une liste par une constante de type **numeric**, faire attention au sens des opérations.

```
> L:=[1,2,3,4];M:=[-12,3,7,8];N:=[1,2,3];U:=[x,y,z];B:=[{},{{},
  {0}},{{},{0},{1}}];
```

$$L := [1, 2, 3, 4]$$

$$M := [-12, 3, 7, 8]$$

$$N := [1, 2, 3]$$

$$U := [x, y, z]$$

$$B := [\{\}, \{\{\}, \{0\}\}, \{\{\}, \{0\}, \{1\}\}]$$

```
> L+M;
```

$$[-11, 5, 10, 12]$$

```
> L+N;
```

```
Error, adding lists of different length
```

```
> N+U;
```

$$[x+1, y+2, z+3]$$

```
> B+N;
```

$$[1+\{\}, 2+\{\{\}, \{0\}\}, 3+\{\{\}, \{0\}, \{1\}\}]$$

```
> L/5;
```

$$\left[\frac{1}{5}, \frac{2}{5}, \frac{3}{5}, \frac{4}{5}\right]$$

```
> 3*U;
```

$$[3x, 3y, 3z]$$

```
> 5*B;
```

$$[5\{\}, 5\{\{\}, \{0\}\}, 5\{\{\}, \{0\}, \{1\}\}]$$

La commande map

La commande map applique une opération de calcul ou une autre commande Maple aux opérandes d'un objet Maple.
La syntaxe est la suivante
map(fonction Maple,expression); le résultat est que la fonction Maple insérée à gauche sera appliquée aux opérandes de l'expression Maple à droite.

```
> J:=-infinity..infinity;
```

$$J := -\infty .. \infty$$

```
> map(exp,J);
```

$$0 .. \infty$$

```
> f:=x->x^2/(x^2+1);
```

$$f := x \rightarrow \frac{x^2}{x^2+1}$$

```
> map(f,J);
```

$$undefined .. undefined$$

```
> S:=[1,2,3,4];
```

$$S := [1, 2, 3, 4]$$

```
> map(cos,S);
```

$$[\cos(1), \cos(2), \cos(3), \cos(4)]$$

Les commandes expand et simplify s'appliquent automatiquement à tous les éléments d'une liste sans avoir besoin de la commande map.

```
> L:=[(x+1)^2-x^2-2*x-1,1,2];
```

$$L := \left[(x+1)^2 - x^2 - 2\,x - 1,\ 1,\ 2 \right]$$

```
> expand(L);
```

$$[0,\ 1,\ 2]$$

Les Tables

Une table est une liste indexée, à la différence d'une liste simple qui est indexée par des entiers, la table peut être indexée par d'autres valeurs que des entiers et non nécéssairement numériques.
On peut créer une table par la syntaxe **table**([var1=val1,var2=val2....]) où vari sont les variables d'indexations et vali sont les valeurs indexées.

```
> Cours:=table([Analyse=Mercredi,Algèbre=Dimanche,Informatique=
  Lundi]);
```

$$Cours := table([Algèbre = Dimanche,\ Informatique = Lundi,\ Analyse = Mercredi])$$

```
> Cours[Analyse];
```

$$Mercredi$$

Une table ne s'affiche pas automatiquement, on peut l'afficher soit en forçant son évaluation soit par un ordre d'affichage (print).

```
> Cours;
```

$$Cours$$

```
> print(Cours);
```

$$table([Algèbre = Dimanche,\ Informatique = Lundi,\ Analyse = Mercredi])$$

Lors de l'indexation de nom de variable par l'utilisateur une table est automatiquement crée sous le même nom.

```
> u[1]:=1;u[2]:=1;u[3]:=2;u[4]:=3;u[5]:=5;
```

$$u_1 := 1$$

$$u_2 := 1$$

$$u_3 := 2$$

$$u_4 := 3$$

$$u_5 := 5$$

```
> u;
```

$$u$$

```
> print(u);
```

$$table([1 = 1,\ 2 = 1,\ 3 = 2,\ 5 = 5,\ 4 = 3])$$

```
>
```