Physique Numérique (SMP6)

TP2 : Séquences, ensembles, listes, fonctions, représentations graphiques

Rappel :
Il est fortement recommandé d'utiliser la fonction restart en début de chaque exercice pour libérer les variables de leur affectation.

Insertion de textes et de commentaires :
Sur une ligne de commande Maple : le dièse (#) permet d'insérer un commentaire à la suite d'une commande : Maple n'interprète pas ce qui suit le dièse.
Entre deux lignes de commandes : on peut insérer du texte (non interprété par Maple, ex : Exercice N°1) : le raccourcis "T" ou taper "Ctrl+T" change le prompt en mode texte.
Insertion de lignes de commandes : on regardera avec profit l'effet de "Ctrl+J" et "Ctrl+K".

Le but de ce TP est d'étudier des types d'objets fondamentaux : les séquences, les listes et les ensembles. Au cours du TP dernier, nous avons manipulé des variables en leur affectant une donnée simple (un réel, une fraction, un complexe...). Il peut être intéressant de manipuler un grand nombre d'objets en les regroupant.

Il existe plusieurs façons de regrouper des objets :

Les Séquences :

Définition Maple : Une séquence est une collection ordonnée d'expressions, séparées par des virgules. Ces expressions peuvent être du même type (par exemple des nombres entiers) ou de type différents (ex : une fonction suivie d'une variable, d'un réel, d'un entier et d'une chaîne de caractères).

Pour construire une séquence, il existe essentiellement deux méthodes :

1ère méthode : on construit à la main la séquence en écrivant à la suite les éléments de la séquence séparés par des virgules :

```
> restart;l:=cos, x, evalf(sqrt(3)), 2011, 'bonjour'; # un exemple
  de séquence
```

$$l := \cos, x, 1.732050808, 2011, bonjour \quad \textbf{(1)}$$

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

$$exprseq \quad \textbf{(2)}$$

Les séquences sont d'utilisation constante en Maple : il est important de bien savoir les manipuler.

```
> l[1];l[2];l[5];l[7];
```

$$\cos$$

$$x$$

bonjour

```
Error, invalid subscript selector
```

L'erreur provient du fait qu'il n'existe pas de 7ième élément à la séquence l.

```
> l[3..5];
```

1.732050808, 2011, *bonjour* **(3)**

La séquence vide se nomme NULL

```
> k:=NULL;
```

k := **(4)**

```
> b:=NULL; l;
```

b :=

cos, *x*, 1.732050808, 2011, *bonjour* **(5)**

Pour mettre bout à bout (concaténer) deux séquences, il suffit de les écrire en les séparant par des virgules :

```
> m:=Pi, t, s, u;
```

m := π, *t*, *s*, *u* **(6)**

```
> n:=l,NULL,m;
```

n := cos, *x*, 1.732050808, 2011, *bonjour*, π, *t*, *s*, *u* **(7)**

Remarque : les séquences sont des structures à lecture seule : il n'est pas possible de modifier une séquence existante en affectant l'un de ses éléments :

```
> n[3];
```

1.732050808 **(8)**

```
> n[3]:=0;
Error, invalid assignment (cos, x, 1.732050808, 2011, bonjour,
Pi, t, s, u)[3] := 0; cannot assign to an expression sequence
```

Par contre, rien ne nous empêche de créer une nouvelle séquence à partir de morceaux de la précédente.

Deuxième méthode : dans le cas usuel où la séquence est issue d'un terme générique (dépendant d'un paramètre k), il existe une commande permettant de construire cette séquence de manière automatique. Il s'agit de la commande **seq** :

```
> restart; l:=seq(k*Pi, k=0..7);
```

l := 0, π, 2 π, 3 π, 4 π, 5 π, 6 π, 7 π **(9)**

Remarque : évidemment, le paramètre k utilisé ici est une variable muette ; toute autre lettre peut convenir ! Par contre attention aux bornes de l'intervalle !

```
> t:=seq(cos(k*a), k=12..15); oups:=seq(cos(k*a), k=15..12);
```

t := cos(12 *a*), cos(13 *a*), cos(14 *a*), cos(15 *a*)

oups := **(10)**

La commande seq est très générale et TRES IMPORTANTE.
Je vous laisse méditer sur (les possibles variations sur) les exemples suivants :

```
> seq(cat("le ",k,"ème élément"), k=0..5);
```

"le 0ème élément", "le 1ème élément", "le 2ème élément", "le 3ème élément", "le 4ème élément", "le 5ème élément" **(11)**

```
> seq(seq(i^2-j,j=1..7),i=1..6);
```

(12)

$$0, -1, -2, -3, -4, -5, -6, 3, 2, 1, 0, -1, -2, -3, 8, 7, 6, 5, 4, 3, 2, 15, 14, 13, 12, 11, 10, 9, 24, 23, 22, 21, 20, 19, 18, 35, 34, 33, 32, 31, 30, 29 \quad \textbf{(12)}$$

Les listes :

Définition Maple : Une liste est une suite ordonnée d'expressions séparées par des virgules, notée entre crochets. Ce sont des structures à lecture seule.

Remarque importante :
On se rappellera avec profit qu'**une liste est exactement une "séquence entre crochets"**. Autrement dit, les opérations usuelles sur les séquences s'appliquent aux listes, modulo la mise entre crochets.

```
> restart;L:=[seq(1+(1/i), i=1..15)];whattype(L);
```

$$L := \left[2, \frac{3}{2}, \frac{4}{3}, \frac{5}{4}, \frac{6}{5}, \frac{7}{6}, \frac{8}{7}, \frac{9}{8}, \frac{10}{9}, \frac{11}{10}, \frac{12}{11}, \frac{13}{12}, \frac{14}{13}, \frac{15}{14}, \frac{16}{15}\right]$$

$$list \quad \textbf{(13)}$$

```
> L[4];L[7..13];
```

$$\frac{5}{4}$$

$$\left[\frac{8}{7}, \frac{9}{8}, \frac{10}{9}, \frac{11}{10}, \frac{12}{11}, \frac{13}{12}, \frac{14}{13}\right] \quad \textbf{(14)}$$

Pour obtenir la séquence des éléments constituant la liste (ie. enlever les crochets) on utilise la commande *op* :

```
> op(L);
```

$$2, \frac{3}{2}, \frac{4}{3}, \frac{5}{4}, \frac{6}{5}, \frac{7}{6}, \frac{8}{7}, \frac{9}{8}, \frac{10}{9}, \frac{11}{10}, \frac{12}{11}, \frac{13}{12}, \frac{14}{13}, \frac{15}{14}, \frac{16}{15} \quad \textbf{(15)}$$

Le nombre d'éléments d'une liste est *nops(L)* ; *member* teste la présence d'éléments dans une liste.

```
> nops(L);
```

$$15 \quad \textbf{(16)}$$

```
> member(4/3, L); member(Pi, L);
```

$$true$$

$$false \quad \textbf{(17)}$$

La **concaténation** de deux listes est évidente si on se souvient qu'**une liste est une séquence entre crochets** : on enlève les crochets, on fusionne les deux séquences, et on remet les crochets :

```
> K:=[a,b,c];
```

$$K := [a, b, c] \quad \textbf{(18)}$$

```
> M:=[op(L), op(K)];
```

$$M := \left[2, \frac{3}{2}, \frac{4}{3}, \frac{5}{4}, \frac{6}{5}, \frac{7}{6}, \frac{8}{7}, \frac{9}{8}, \frac{10}{9}, \frac{11}{10}, \frac{12}{11}, \frac{13}{12}, \frac{14}{13}, \frac{15}{14}, \frac{16}{15}, a, b, c\right] \quad \textbf{(19)}$$

Remarque : On pourrait se demander quel est l'intérêt de distinguer la notion de séquence et de liste : je vous laisse méditer sur l'exemple suivant :

```
> restart; L:=seq(seq(i+j, i=1..5), j=1..4);
```

$$L := 2, 3, 4, 5, 6, 3, 4, 5, 6, 7, 4, 5, 6, 7, 8, 5, 6, 7, 8, 9 \quad \textbf{(20)}$$

```
> M:=[seq([seq(i+j, i=1..5)], j=1..4)];
```

$$M := [[2, 3, 4, 5, 6], [3, 4, 5, 6, 7], [4, 5, 6, 7, 8], [5, 6, 7, 8, 9]] \quad \textbf{(21)}$$

```
> L[2]; M[2];
```

$$3$$

$$[3, 4, 5, 6, 7] \quad \textbf{(22)}$$

Conclusion : une séquence de séquences est une séquence dans laquelle les séquences séquencées ne peuvent plus être distinguées après concaténation. Dans une liste de listes, les sous-listes sont conservées dans leur intégrité.
De nombreuses fonctions utilisent la notion de liste. Nous aurons l'occasion de les manipuler dans les exercices.

Les ensembles :

Définition Maple : Un ensemble est une collection non ordonnée d'expressions toutes différentes ; il se note à l'aide d'une séquence entre accolades. La collection est non ordonnée ; l'ordre des éléments donné en entrée n'est pas toujours respecté par Maple. Les expressions sont toutes différentes : Maple élimine les doublons.

```
> restart; L:={a,a,b,c,a,b,e};whattype(L);
```

$$L := \{a, b, c, e\}$$

$$set \quad \textbf{(23)}$$

Le principe est le même que pour les listes : ce sont des séquences entre accolades. Cependant les opérations usuelles sont connues de Maple : intersection, union, différence.

```
> L minus {a, c};
```

$$\{b, e\} \quad \textbf{(24)}$$

```
> L intersect {c,f,e}; L union {c,f,e};
```

$$\{c, e\}$$

$$\{a, b, c, e, f\} \quad \textbf{(25)}$$

```
> nops(L); op(L);
```

$$4$$

$$a, b, c, e \quad \textbf{(26)}$$

Le dernier objectif de ce TP est d'introduire les notions usuelles en mathématiques de **fonctions** (numériques ou non) et d'**expressions**.
Maple fait bien évidemment la distinction entre ces deux notions : une fonction *f* est l'objet mathématique qui à un élément *x* associe l'unique élément *f(x)*.

f est la fonction, *f(x)* est une expression en *x*.

Cette distinction est d'autant plus importante que Maple utilise des commandes différentes selon qu'on manipule des fonctions ou des expressions. Par exemple, pour le calcul de la dérivée d'une fonction *f*, *D(f)* sera la fonction dérivée *f'*, alors que *diff(f(x), x)* sera l'expression *f'(x)*. Evidemment *(D(f))(x)* rend le même résultat que *diff(f(x),x)*.

Les expressions et les fonctions :

Attention : une fonction n'est pas une expression !

```
> restart;A:=x^2+x+1; # ceci est une expression en x
```

$$A := x^2 + x + 1 \quad \textbf{(27)}$$

Si on veut évaluer A en x=3, il faut attribuer une valeur à x.

```
> x:=3; A;
```

$$x := 3$$
$$13 \quad \textbf{(28)}$$

Pour plusieurs valeurs de x, il faut définir une fonction ; il y a pour cela plusieurs méthodes :

La flèche : elle est très proche de la notation mathématique : la syntaxe est nom_fonction := nom_variable -> expression_de_la_variable

```
> restart;f:=x-> x^2+x+1; f(3);f(2);
```

$$f := x \to x^2 + x + 1$$
$$13$$
$$7 \quad \textbf{(29)}$$

```
> f(x); # f(x) est une expression en x
```

$$x^2 + x + 1 \quad \textbf{(30)}$$

```
> g:= (x,y)-> x^y;# fonction de plusieurs variables
```

$$g := (x, y) \to x^y \quad \textbf{(31)}$$

```
> h:=x->[cos(x), sin(x)]; #fonction qui renvoie une liste
```

$$h := x \to [\cos(x), \sin(x)] \quad \textbf{(32)}$$

La fonction unapply : syntaxe : unapply(expression, séquence_noms_variables) :

```
> restart; f:=unapply(x^2+x+1,x);
```

$$f := x \to x^2 + x + 1 \quad \textbf{(33)}$$

```
> g:=unapply([x^y, x+y],x,y);
```

$$g := (x, y) \to [x^y, x + y] \quad \textbf{(34)}$$

Il faut remarquer qu'on peut définir sous Maple des fonctions d'objets bien plus généraux que des nombres :
Par exemple : des fonctions sur les listes, les séquences, les ensembles !

```
> restart;
> f:=L-> [op(L), Pi];# fonction qui, à une liste, rajoute Pi en
  dernier élément
```

$$f := L \to [op(L), \pi] \quad \textbf{(35)}$$

```
> f([1,2,3]);
```

$$[1, 2, 3, \pi] \quad \textbf{(36)}$$

```
> g:=(E,F)->E union F minus {0};
```

$$g := (E, F) \to (E \cup F) \setminus \{0\} \quad \textbf{(37)}$$

```
> g({0, 1, 3, Pi}, {4, 7, 1, log(2)});
```

$$\{1, 3, 4, 7, \pi, \ln(2)\} \quad \textbf{(38)}$$

La composition des fonctions :

La composition des applications (notation g o f en mathématiques) se fait à l'aide de @

```
> restart; f:=exp@ln;
```

$$f := \exp@\ln \quad \textbf{(39)}$$

```
> f(x);
```

$$x \quad \textbf{(40)}$$

Les itérées nièmes s'obtiennent de la manière suivante :

```
> g:=x->x+1;
```

$$g := x \to x + 1 \quad \textbf{(41)}$$

```
> (g@@5)(x);
```

$$x + 5 \quad \textbf{(42)}$$

Les fonctions add et mul

Elles permettent de calculer des sommes et des produits. Leur syntaxe est la suivante :
add(f(k), k=a..b) calcule la somme des f(k) pour k de a à b,
mul(f(k), k=a..b) calcule le produit des f(k) pour k de a à b.

On peut remarquer la grande analogie de cette syntaxe avec celle de la comande seq vue plus haut.

```
> add(cos(x)^i,i=0..7);
```

$$1 + \cos(x) + \cos(x)^2 + \cos(x)^3 + \cos(x)^4 + \cos(x)^5 + \cos(x)^6 + \cos(x)^7 \quad \textbf{(43)}$$

La fonction map

La fonction map permet, entre autres, d'évaluer une fonction (d'une variable) en plusieurs valeurs qui sont les éléments d'un ensemble ou d'une liste.

```
> restart; f:=x->2*sin(x);
```

$$f := x \to 2 \sin(x) \quad \textbf{(44)}$$

```
> E:={a,b,c,d,e}; L:=[a,b,c,d,e];
```

$$E := \{a, b, c, d, e\}$$
$$L := [a, b, c, d, e] \quad \textbf{(45)}$$

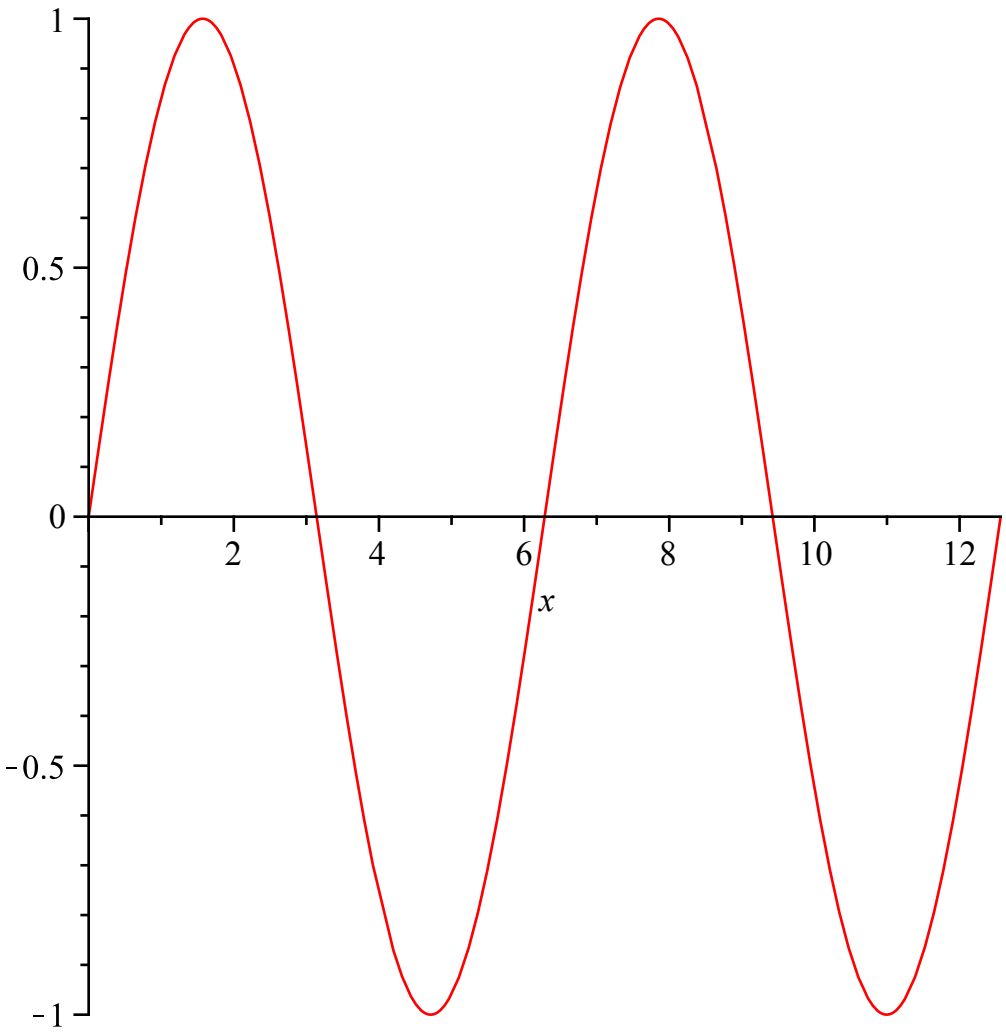
```
> map(f,E); map(f,L);
```

$$\{2 \sin(a), 2 \sin(b), 2 \sin(c), 2 \sin(d), 2 \sin(e)\}$$
$$[2 \sin(a), 2 \sin(b), 2 \sin(c), 2 \sin(d), 2 \sin(e)] \quad \textbf{(46)}$$

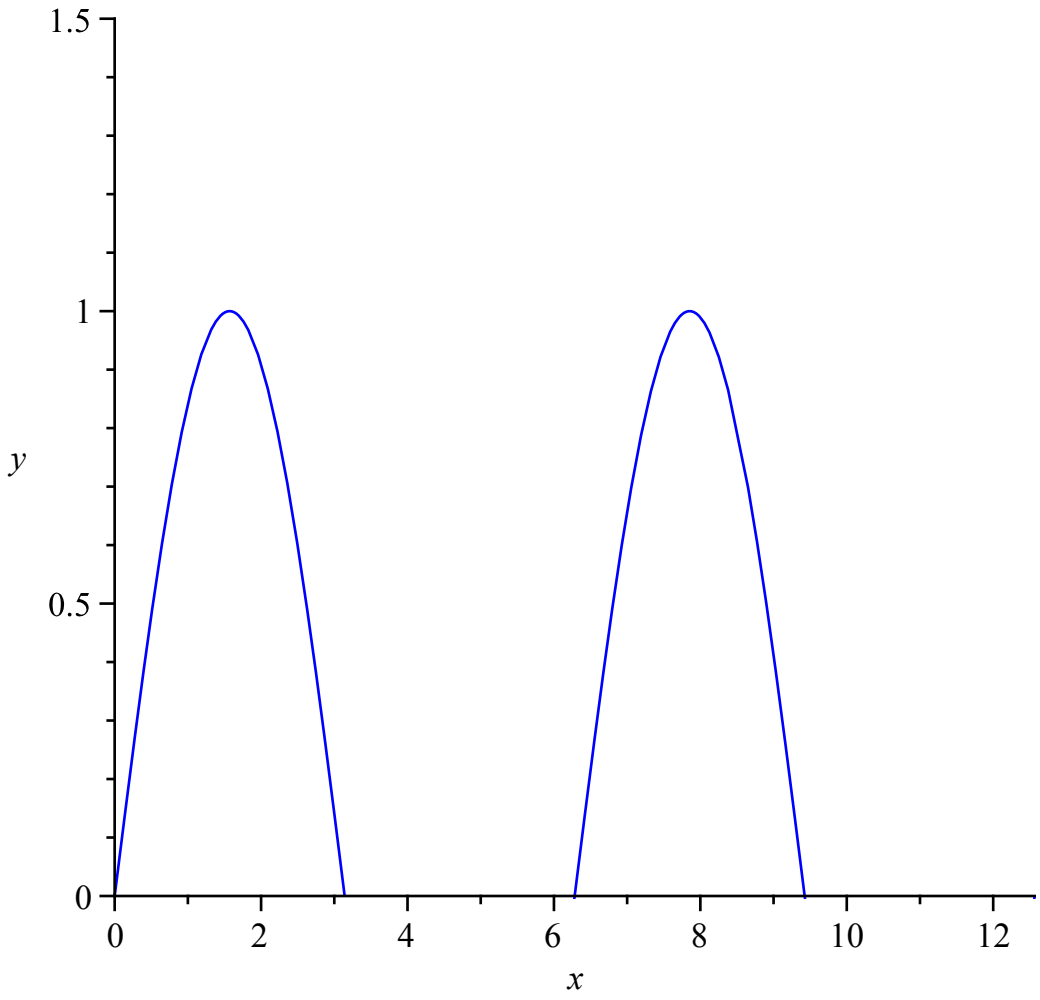
Les représentations graphiques

Nous allons voir comment tracer le graphique d'une fonction f. On utilise pour cela la fonction plot, aux multiples options (couleur, titre, axes etc.). On regardera l'aide pour les exercices correspondants. Voir les exemples suivants :

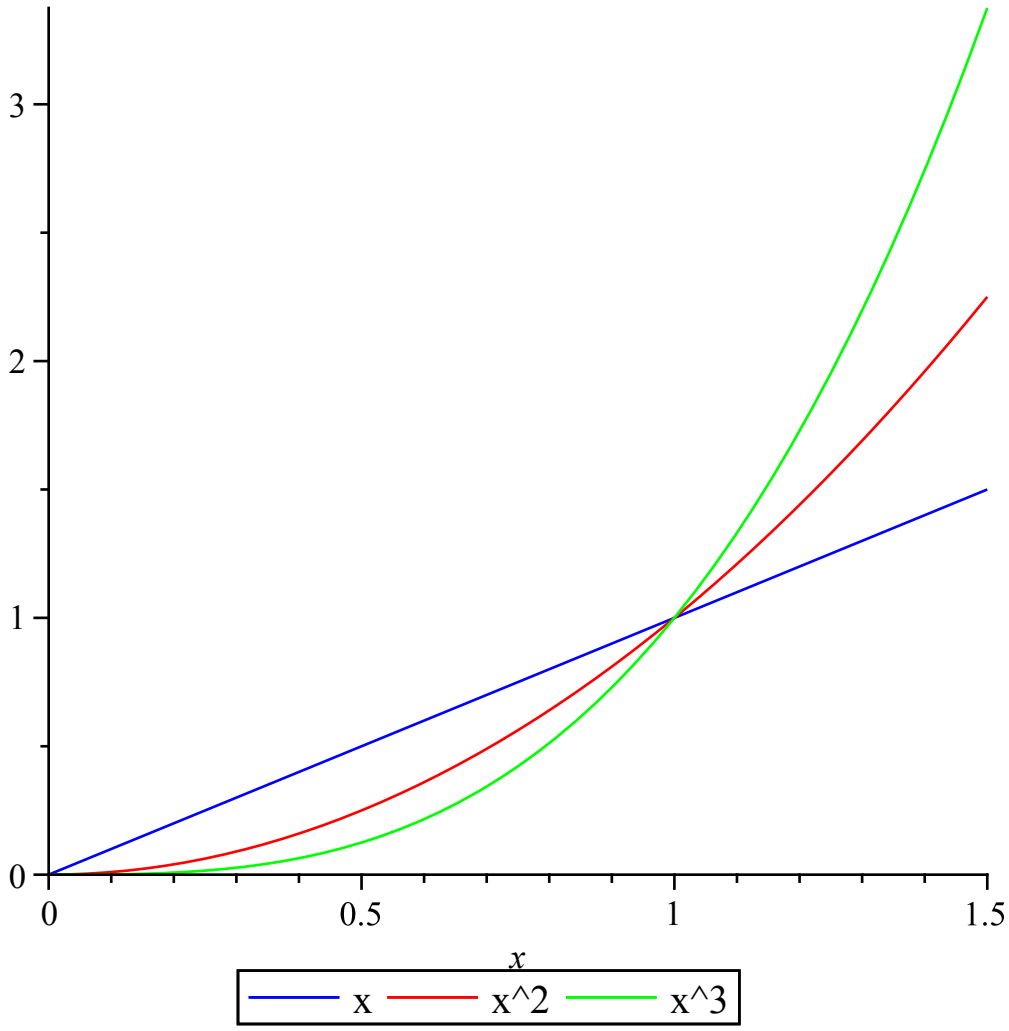
```
> restart; plot(sin(x),x=0..4*Pi);
```




```
> plot(sin(x), x=0..4*Pi, y=0..1.5, color=blue,title="Fonction
  Sinus");
```




```
> plot([x, x^2, x^3], x=0..1.5, color=[blue, red, green], legend=
  ["x", "x^2", "x^3"]);
```



Pour le reste, se référer à l'aide de *plot* ! Il existe d'autres fonctions avancées permettant de tracer d'autres types de graphiques. La plupart sont contenues dans la bibliothèques plots qu'il faut charger avec la commande *with*

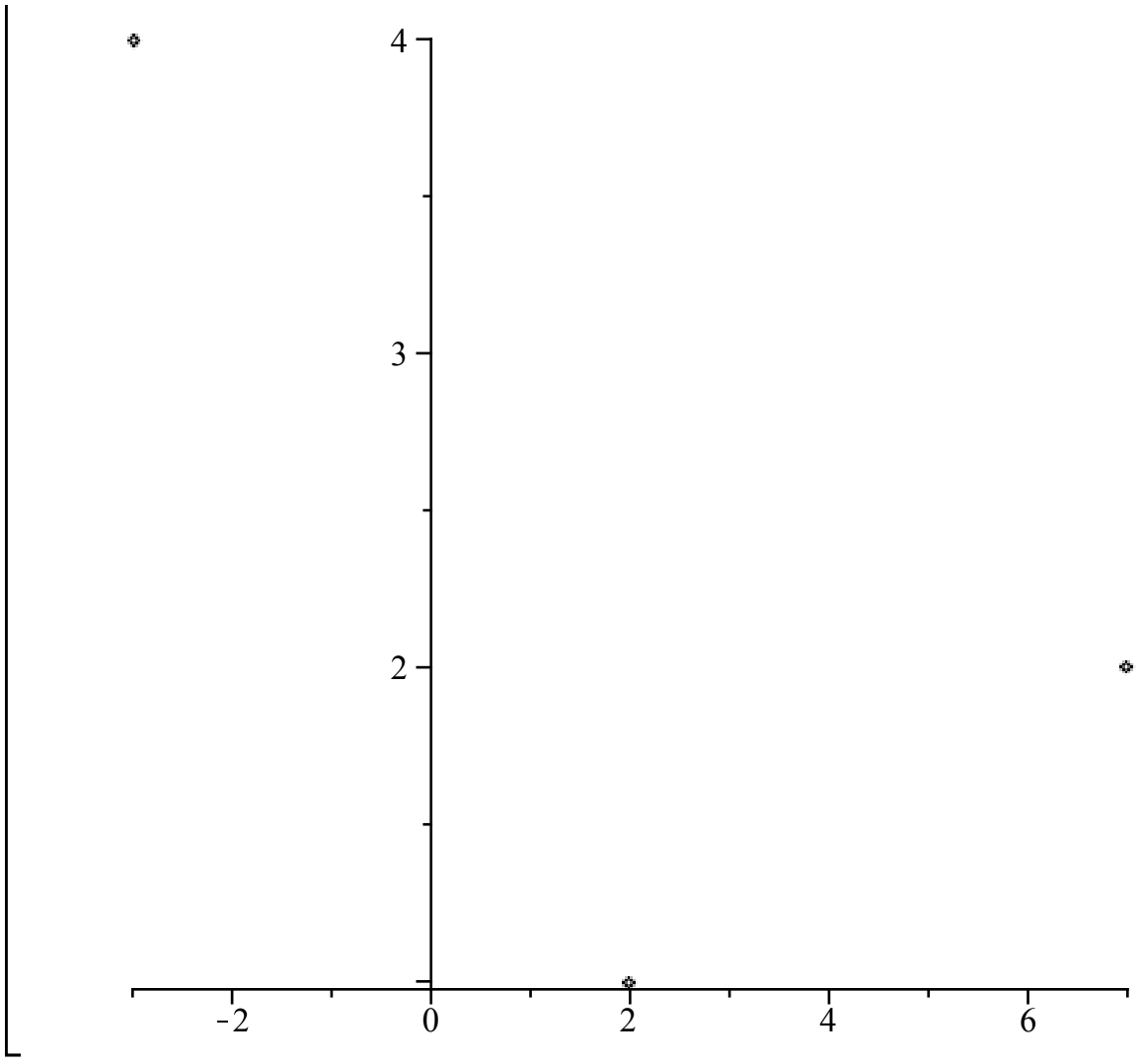
```
> with(plots);
```

[*animate*, *animate3d*, *animatecurve*, *arrow*, *changecoords*, *complexplot*, *complexplot3d*, *conformal*, *conformal3d*, *contourplot*, *contourplot3d*, *coordplot*, *coordplot3d*, *densityplot*, *display*, *dualaxisplot*, *fieldplot*, *fieldplot3d*, *gradplot*, *gradplot3d*, *graphplot3d*, *implicitplot*, *implicitplot3d*, *inequal*, *interactive*, *interactiveparams*, *intersectplot*, *listcontplot*, *listcontplot3d*, *listdensityplot*, *listplot*, *listplot3d*, *loglogplot*, *logplot*, *matrixplot*, *multiple*, *odeplot*, *pareto*, *plotcompare*, *pointplot*, *pointplot3d*, *polarplot*, *polygonplot*, *polygonplot3d*, *polyhedra_supported*, *polyhedraplot*, *rootlocus*, *semilogplot*, *setcolors*, *setoptions*, *setoptions3d*, *spacecurve*, *sparsematrixplot*, *surfdata*, *textplot*, *textplot3d*, *tubeplot*] **(47)**

On remarquera en particulier les commandes *animate, implicitplot, pointplot* (cf l'aide !).

```
> ?plots, pointplot #on spécifie à Maple qu'il faut, dans l'aide,
  chercher pointplot dans plots
```

```
> pointplot({[2,1], [-3,4], [7,2]});
```


4
3
2
-2
0
2
4
6