1 Utilitaires standards

1.1 Parenthèses et priorités d'opérateurs

L'éditeur Maple, qui est du type « pleine page » demande d'entrer les expressions mathématiques en ligne, ce qui est parfois délicat quand l'expression est un tant soit peu complexe...
Heureusement, si on place le curseur dans une expression mathématique et qu'on clique sur le « x » à gauche au dessus de la fenètre, l'expression s'affiche en mode mathématique.
Malheureusement, on ne peut plus la travailler dans ce mode...
Les priorités des opérateurs sont les priorités habituelles. Ce qui donne, dans l'ordre et de gauche à droite en cas d'égalité :

- **
- * et /
- + et -

Donnons un exemple :

```
> z:=sqrt((x+1)**2/(x**2*(x-1)**3));
```

$$z := \sqrt{\frac{(x+1)^2}{x^2\,(-1+x)^3}}$$

1.2 Simplifications

L'utilitaire de « simplification » le plus simple est la fonction `simplify`. On observera sur les exemples qui suivent que « simplifier » est parfois discutable...

```
> p:=(x+1)**2-x**2-2*x;
```

$$p := (x+1)^2 - x^2 - 2\,x$$

```
> simplify(p);
```

$$1$$

```
> q:=(1-x**12)/(1-x);
```

$$q := \frac{1-x^{12}}{1-x}$$

```
> simplify(q);
```

$$x^{11}+x^{10}+x^9+x^8+x^7+x^6+x^5+x^4+x^3+x^2+x+1$$

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

$$r := \frac{x+2}{-1+x} + \frac{x}{x+1}$$

```
> simplify(r);
```

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

```
> s:=4**(1/2);
```

$$s := \sqrt{4}$$

```
> simplify(s);
```

$$2$$

```
> t:=cos(5*a)-cos(3*a);
```

$$t := \cos(5\,a) - \cos(3\,a)$$

```
> simplify(t);
```

$$16\cos(a)^5 - 24\cos(a)^3 + 8\cos(a)$$

1.3 Développements

La fonction `expand` qui permet de développer toutes sortes d'expressions.
Signalons aussi la fonction `collect(p,x)` qui réunit les termes en `x` dans `p`. Elle développe donc d'abord l'expression.

```
> expand((x+1)*(x-3));
```
$$x^2 - 2\,x - 3$$
```
> expand((x+1)/(x-3));
```
$$\frac{x}{x-3} + \frac{1}{x-3}$$
```
> expand(sin(a+b));
```
$$\sin(a)\cos(b) + \cos(a)\sin(b)$$
```
> expand(cos(2*a));
```
$$2\cos(a)^2 - 1$$
```
> expand(ln(x/(x-1)**2));
```
$$\ln(\frac{x}{(-1+x)^2})$$

1.4 Combinaisons

La fonction `combine` est une sorte de réciproque de la précédente. Elle possède beaucoup d'options dont on ne parle pas ici.

```
> combine(4*sin(a)**3);
```
$$-\sin(3\,a) + 3\sin(a)$$
```
> combine((a**b)**3);
```
$$a^{(3\,b)}$$
```
> combine(exp(a)**2*exp(b));
```
$$e^{(2\,a+b)}$$
```
> combine(exp(sin(a)*cos(b))*exp(cos(a)*sin(b)));
```
$$e^{\sin(a+b)}$$

1.5 Substitutions

La fonction `subs` permet de sustituer une variable libre par une expression quelconque dans une expression quelconque. On peut ainsi par exemple avoir la valeur d'une expression pour une valeur particulière d'une variable sans modifier celle-ci.

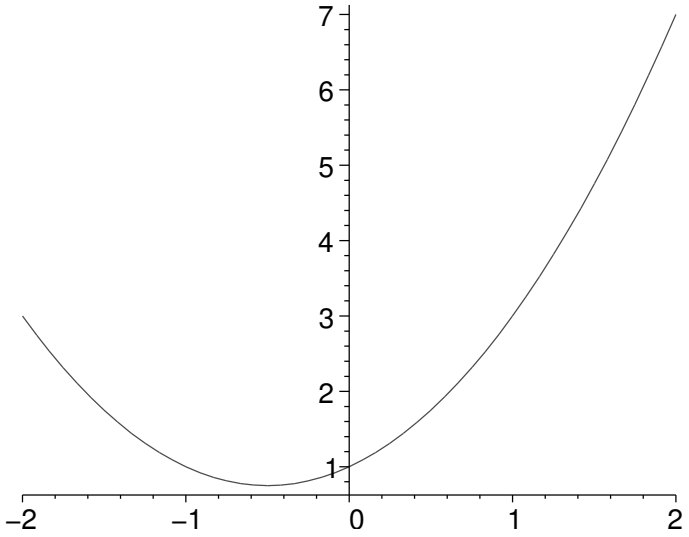
```
> subs(x=2,x**2+x+1);
```
$$7$$
```
> subs(x=a**(1/3),3*x*ln(x**3));
```
$$3\,a^{(1/3)}\ln(a)$$
```
> subs(sin(x)=y,sin(x)/sqrt(1-sin(x)));
```
$$\frac{y}{\sqrt{1-y}}$$

1.6 Fonctions et expressions

On ne confondra pas une « fonction » comme la fonction `f` définie ci-dessous, et une « expression » comme ici `y` qui est une expression de la variable libre `x`. Notez bien que ce sont deux choses différentes qui s'utilisent de façons différentes...

```
> f:=x->x**2+x+1;
```
$$f := x \to x^2 + x + 1$$
```
> f;
```
$$f$$
```
> f(t);
```
$$(\cos(5\,a) - \cos(3\,a))^2 + \cos(5\,a) - \cos(3\,a) + 1$$
```
> f(a);
```
$$a^2 + a + 1$$
```
> y:=x**2+x+1;
```
$$y := x^2 + x + 1$$
```
> y;
```
$$x^2 + x + 1$$

On peut avoir facilement un bon tracé de la fonction :

```
> plot(y,x=-2..2);
```


On ne mélangera pas une fonction et une expression :

```
> y-f;
```
$$x^2 + x + 1 - f$$
```
> y-f(x);
```
$$0$$

2 Procédures

2.1 Ecriture et utilisation

C'est la fonction `proc` qui permet de créer une fonction, encore appelée procédure, en complément du procédé simplifié déjà vu au paragraphe 1.6.
On distinguera clairement deux phases :

- Il s'agit de donner un nom à une séquence de calcul en lui indiquant les paramètres utilisés, c'est la première phase d'**écriture** de la procédure.
- Ensuite, en utilisant le nom donné auparavent, avec des valeurs des paramètres, on fera éxécuter la séquence de calcul sans avoir besoin de la réécrire, c'est la phase d'**utilisation** de la procédure.

Dans l'utilisation d'une procédure, il n'y a pas de différence visible entre une fonction Maple d'origine et une procédure (ou fonction) que vous avez créée... On dit qu'on procède par extension du langage.

La syntaxe générale est :

```
> nom:=proc(paramètre(s))
> local variable(s) locale(s);
> corps de la procédure
> variable(s) « résultat »
> end;
```

Les différentes variables sont éventuellement séparées par des virgules.

2.2 Une première procédure

On va maintenant créer une procédure qui calcule, pour un terme donné, le terme suivant de la suite

$$u_{n+1} = \frac{u_n + \dfrac{a}{u_n}}{2}$$

On a donc deux paramètres : le terme donné de la suite et a.

a) Ecriture de la procédure

```
> suivant:=proc(u,a) local v; v:=u; v:=(v+a/v)/2; v end;
```

$$suivant := \mathbf{proc}(u, a)\ \mathbf{local}\ v;\ v := u\ ;\ v := 1/2 * v + 1/2 * a\,/v\ ;\ v\ \mathbf{end\ proc}$$

b) Utilisation de cette procédure

```
> suivant(2,3);
```

$$\frac{7}{4}$$

3 A retenir

1) Les opérations mathématiques usuelles : $+\ -\ *\ /\ **$ et l'utilation des parenthèses.

2) Les opérateurs :

- **a)** `simplify` qui « simplifie » tous types d'expressions
- **b)** `expand` qui développe tous types d'expressions
- **c)** `combine` qui regroupe au mieux tous types d'expressions

3) L'opérateur `subs` qui permet de remplacer une variable libre par une expression quelconque, ou simplement une valeur, dans une expression quelconque.

4) La notion de **fonction** qu'il ne faut pas confondre avec la notion d'**expression**, ainsi que `->` qui permet de créer une fonction simple.

5) La notion de **procédure** avec :

- **a)** la distinction entre le moment où on **écrit, crée** une procédure et le moment où on l'**utilise**
- **b)** `proc ... end;` qui permet de créer une procédure
- **c)** la notion de **variable locale** indispensable pour écrire une procédure
- **d)** et la notion de **paramètre** donnés en entrée à la procédure.