Syntaxe Maple

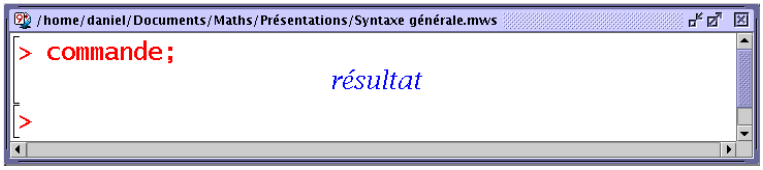
Un bref survol de la syntaxe générale de *Maple* pour donner une vue d'ensemble de ses fonctions.

Notez bien les *conventions de couleurs* dans toutes ces pages :

> **syntaxe Maple**
> *instructions génériques* : à remplacer par de la syntaxe *Maple* appropriée
> {*instructions optionnelles*} facultatives (les "{}" **ne font pas partie** de la syntaxe !)
> *sortie* : résultat fourni par *Maple*

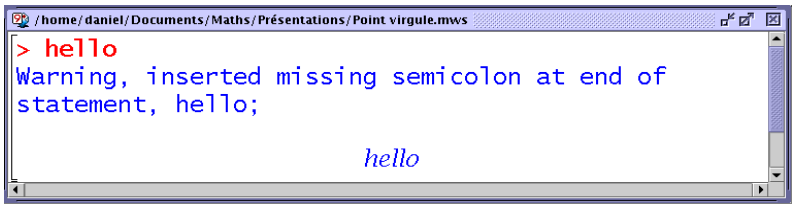
I. Interface

Un calcul en *Maple* se présente toujours sous l'aspect suivant :



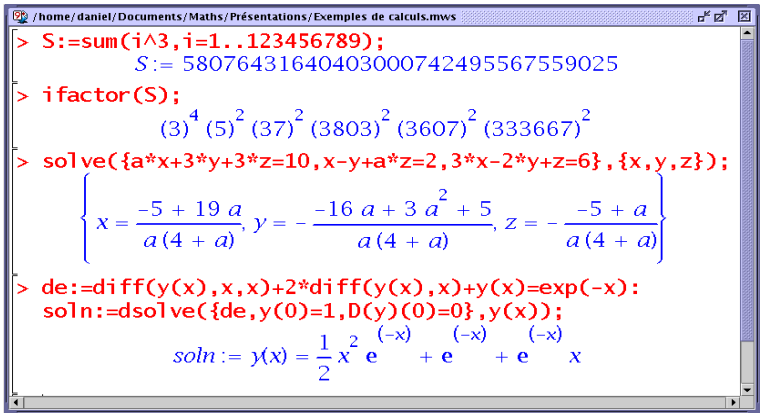
- La commande est entrée à l'invite ([>) en syntaxe *Maple*, et terminée par un point-virgule.
- *Maple* affiche son résultat centré et en notation mathématique.

Si le point-virgule est omis, *Maple* l'ajoutera pour vous mais affichera un avertissement :



II. Exemples de calculs

Voici quelques calculs que *Maple* exécute rapidement, sans programmation nécessaire :



III. Aide

Différents cas peuvent se présenter :

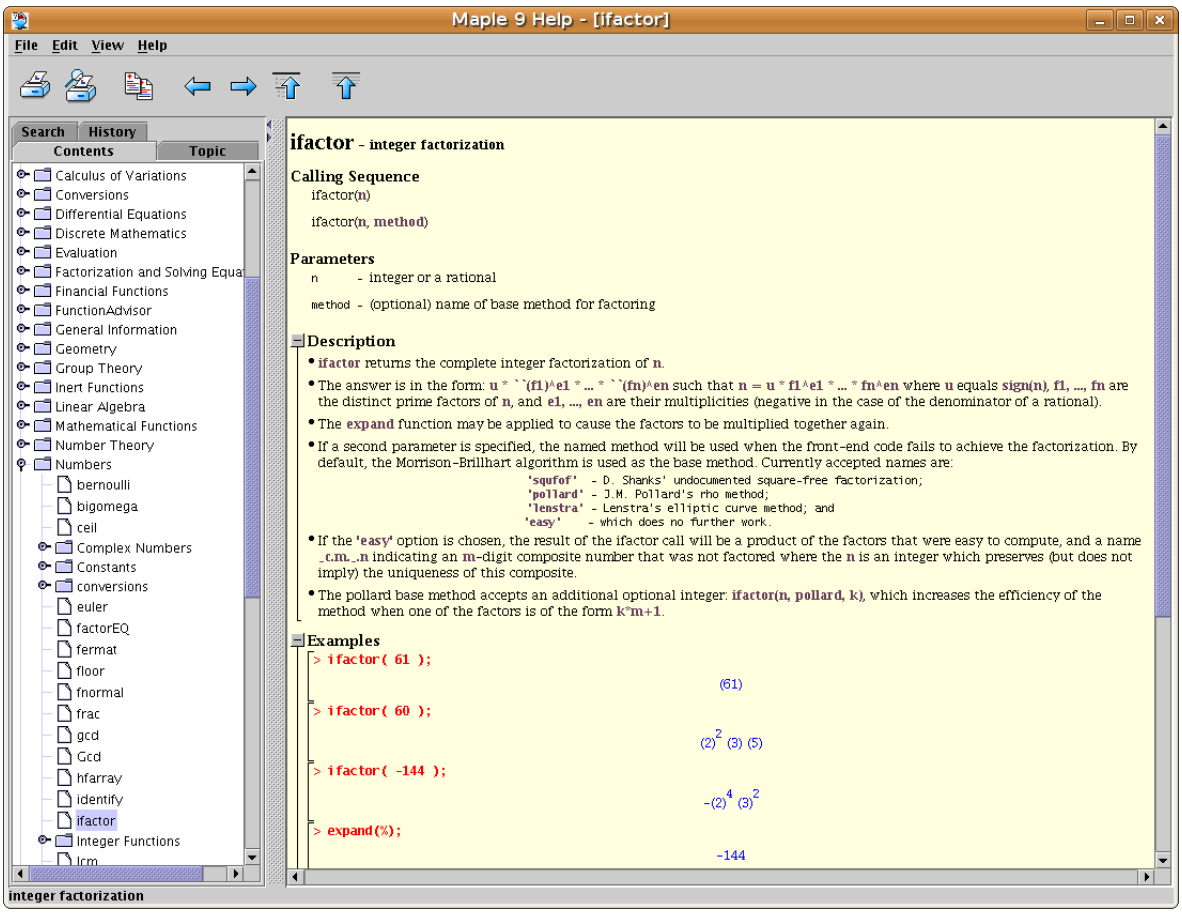
1. On connaît le nom de la fonction cherchée (on souhaite juste un rappel du détail de la syntaxe).

Il suffit de le taper directement dans l'interface, précédé d'un " **?** " (pas besoin de " **;** " dans ce cas).

On est conduit directement à la page d'aide de la commande (ci-après).

2. Le nom de la fonction est inconnu

Il faut alors le retrouver par raffinements progressifs dans le navigateur d'aide de *Maple* (Help > Introduction > Mathematics)



IV. Affectation

Elle prend la forme générale suivante :

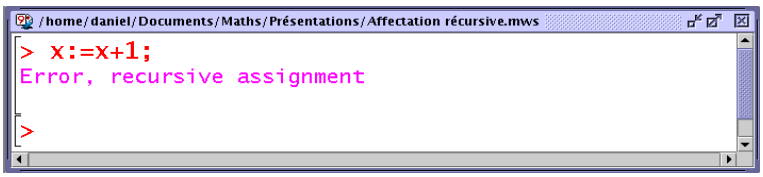
variable **:=** *expression* **;**

Noter la dissymétrie : au membre de gauche, un nom de variable ; à droite, une expression[1].

Signalons qu'une variable peut être vide en *Maple*. Ce n'est pas équivalent à un contenu nul ! Cela peut poser problème dans le cas d'une affectation du type **x := x+1 ;**

- si **x** a un contenu numérique, celui-ci est augmenté d'une unité : pas de problème.
- mais si **x** est vide, *Maple* déclenche son mécanisme de protection contre une spirale incessante d'évaluations :

1 Voir le poly. "Expressions et fonctions".



Plus de détails dans le poly. "Évaluation".

V. Arithmétique

Les opérations algébriques de base sont généralement notées classiquement en *Maple* :

- addition : **+** ; soustraction : **-** ;
- multiplication : ***** (attention ! Elle n'est **jamais implicite**) ;
- division : **/** ;
- exponentiation : **^** ou ****** ;
- résultat précédent **%** ([2]) ;
- π : **Pi** (et non **pi**, qui est la lettre grecque !), *e* : **exp(1)** (ni **e** ni **E** !), i : **I** (et non **i** !).

VI. Fonctions intégrées

Qu'elle soit intégrée à *Maple* ou définie par l'utilisateur, l'appel à une fonction prend toujours la même forme :

nom(*argument*{*s*}) ;

Parmi la grande quantité de fonctions que connaît *Maple*, citons :

- exponentielles et logarithmes : **exp**, **ln**, **log10**, **log[b]** (logarithme en base *b*) ;
- racine carrée : **sqrt** ;
- valeur absolue (ou module d'un complexe) **abs** ;
- fonctions circulaires **sin**, **cos**, **tan** ;
- fonctions hyperboliques **sinh**, **cosh**, **tanh** ;
- fonctions circulaires inverses **arcsin**, **arccos**, **arctan** ;
- fonctions hyperboliques inverses **arcsinh**, **arccosh**, **arctanh**.

VII. Approximation

L'approximation numérique est contrôlée par la commande **evalf** et la variable **Digits**.
La syntaxe générale est

evalf(*expression*{, *chiffres*}) ;

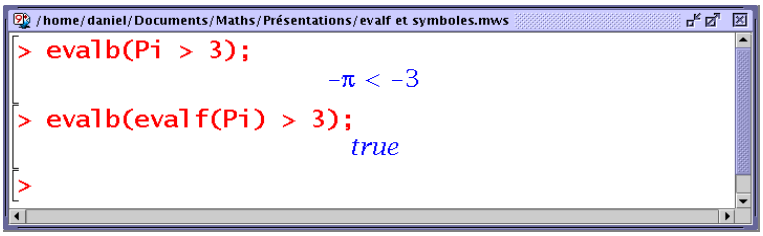
L'approximation est obtenue à la précision par défaut (10 décimales) si elle n'est pas spécifiée.

Cette précision par défaut est contenue dans la variable **Digits** (D majuscule !), que l'on peut modifier.

---

2 **Chronologiquement !** (et non selon l'ordre de la feuille) **%%** et **%%%** donnent aussi accès à l'avant-dernier et à l'antépénultième résultat, mais cela ne va pas plus loin.

Attention aux symboles et expressions "non numériques[3] " (**Pi** ...) :



III. Tests

Donnons simplement la syntaxe générale :

```
  if condition1 then
      instruction1a { ; instruction1b ; ...}
{ elif condition2 then
      instruction2a { ; instruction2b ; ...}}
  ...

{ else
      instructionNa { ; instructionNb ; ...}}
  end ;
```

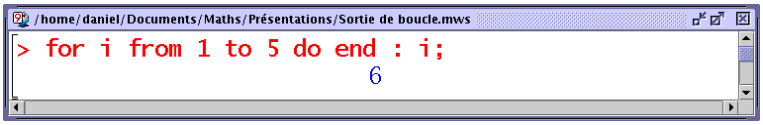
Notons que les clauses **elif** et **else** sont *facultatives.*

IX. Boucles définies

Dans le cas où le nombre d'itérations est connu à l'avance, on emploiera la syntaxe suivante :

```
{for indice}{from début}{by pas} to fin
do instruction1 { ; instruction2 ; ...}
end ;
```

Seule la clause **to** est obligatoire ! (**to 10** suffirait à répéter 10 fois une instruction).
La valeur par défaut de *début* et *pas* est 1.
Notons la valeur de sortie de *indice* :



C'est la première valeur *extérieure* à l'intervalle défini pour la boucle.

X. Boucles indéfinies

Si la poursuite de l'itération dépend de la persistance d'une condition, on codera :

```
  while condition
  do instruction1 { ; instruction2 ; ...}
```

3 Voir le poly. "Types (semi-)numériques".

```
end ;
```

Maple permet de *combiner* les deux types de boucles précédents[4] (**for** *indice* **while** *condition* ...).

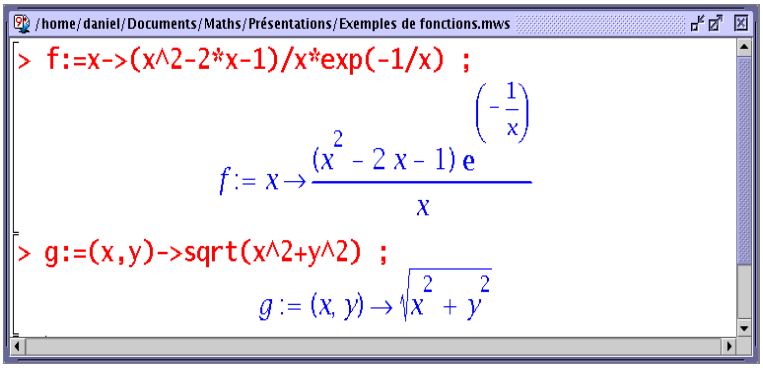
XI. Fonctions simples

Une fonction "simple" peut être définie à l'aide de la syntaxe abrégée suivante :

```
nom := {(}argument{s)} - > expression ;
```

(Bien noter que la flèche "**- >**" est en fait composée de *deux caractères distincts* : un tiret et un symbole "supérieur".)

En voici deux exemples :



Les parenthèses ne sont là que pour assurer le groupage des arguments, s'ils sont plusieurs.

On ne peut pas récupérer un résultat précédent pour en faire une fonction à l'aide de la syntaxe naïve "**f := x - > %**" ([5]).
Il faut utiliser **unapply**. (plus de détails à dans le poly. "Expressions et fonctions")

XII. Procédures

Contentons-nous de donner ici la syntaxe générale :

```
   nom := proc(paramètre{s})
{  local séquence de variables locales ; }
{  global séquence de variables globales ; }
{  options séquence d'options ; }
   instruction1{ ;
    ...
   instructionN }
   end ;
```

Les différents éléments sont détaillés dans le poly. "Fonctions et procédures".

---

4 Ce qui va faire hurler les puristes !
5 La *portée* du " **%** " est limitée à la définition de la fonction, voir "Expressions et fonctions".