Variables et fonctions avec le logiciel Maple

I Variables

Dans le logiciel Maple, les variables ont essentiellement deux fonctionnalités :

- jouer le rôle de variables mathématiques, d'inconnues...
- stocker des informations.

Par défaut, une variable ne contient pas d'information. Elle est traitée comme une variable mathématique. Si elle contient une information, elle est remplacée par cette information dans les expressions où elle apparaît.

I.1 Affecter une valeur

L'affectation d'une valeur à une variable permet de stocker une donnée à une variable. Le symbole utilisé sous Maple pour réaliser une affectation n'est pas = mais :=.

```
[ > x:=2;
                                     x := 2
```

Nous venons de stocker le nombre 2 dans la variable **x**. Dorénavant, la variable **x** sera remplacée par la valeur 2 dans les expressions où elle apparaîtra :

```
[ > 2*x+1;
                                     5
```

```
[ > x^2+x+1;
                                     7
```

Une variable ne contient qu'une seule donnée. Si vous faites une seconde affectation à une variable, la donnée contenue dans la variable est **perdue**.

1.

```
[ > x:=2;
                                     x := 2
```
2.

```
[ > x:=3;
                                     x := 3
```
3.

```
[ > x;
                                     3
```

La valeur contenue dans une variable est celle issue de la dernière affectation effectuée et non la dernière affichée sur votre feuille de calcul. Vous pouvez constater ce fait en ré-exécutant la première commande puis la troisième commande.

On peut enlever l'affectation de la variable **x** :

```
[ > x:='x';
                                     x := x
```

Maintenant, la variable **x** ne contient plus de donnée.

Vous pouvez affecter à **y** une expression en fonction de **x**. Si **x** ne contient aucune donnée au moment de l'affectation et que vous stockez par la suite une valeur dans la variable **x**, la valeur de **y** évoluera en fonction de **x** :

```
[ > y:=x^2+x;
    x:=2;
    y;
                                     y := x^2 + x
                                     x := 2
                                     6
```

```
[ > x:=3;
    y;
                                     x := 3
                                     12
```

Par contre, si x contient une donnée, la valeur de y n'évoluera pas en fonction de x :

```
[ > x:=2;
  y:=x^2+x;
  y;
                                     x := 2
                                     y := 6
                                     6

  > x:=3;
  y;
                                     x := 3
                                     6
```

On peut utiliser la valeur stockée dans une variable pour redéfinir cette même variable :

```
[ > x:=2;
  x:=x+1;
                                     x := 2
                                     x := 3
```

Par contre, si x ne contient aucune donnée, cela génère un message d'erreur :

```
[ > x:='x';
  x:=x+1;
                                     x := x

  Error, recursive assignment
```

Le type d'une variable est la nature de la donnée qu'elle contient. On peut connaître le type d'une variable grâce à la fonction `whattype`.

```
[ > x:='x';          # réinitialisation de la variable x
  whattype(x);
                                     x := x
                                     symbol

  > x:=2;
  whattype(x);
                                     x := 2
                                     integer

  > x:=2.;
  whattype(x);
                                     x := 2.
                                     float
```

On remarque que pour Maple, `2.` n'est pas considéré comme un entier mais comme un nombre flottant ; les nombres `2.` et `2` ne sont pas stockés de la même façon.

Résumé : `nom:=expression;` Affecte l'expression à la variable `nom`.
`nom:='nom';` Réinitialise la variable `nom`.
`restart;` Réinitialise toutes les variables.
`whattype(nom);` Renvoie le type de la variable `nom`.

I.2 Faire une hypothèse

Il est possible de faire des hypothèses sur les variables utilisées. Cela altère le comportement des instructions de Maple :

```
[ > x:='x';
  simplify(sqrt(x^2));
                                     x := x
                                     csgn(x) x
```

La fonction `csgn` est la fonction cosigne (une généralisation de la notion de signe adaptée aux nombres complexes). Si l'on suppose $x > 0$, Maple simplifie le cosigne :

```
[ > assume(x>0);
  simplify(sqrt(x^2));
                                     x ~
```

On peut également spécifier le type d'une variable :

```
> n:='n';
   simplify(floor(n+1/2));           # La fonction floor est la fonction partie entière.
                                     n := n
                                     floor (n + 1/2)
```

À nouveau, sans hypothèse, aucune simplification n'est possible. Si n est entier, on peut simplifier la partie entière :

```
> assume(n::integer);
   simplify(floor(n+1/2));           n ~
```

Remarquons deux points :

1. La fonction `assume` ne renvoie aucun résultat.
2. Maple nous signale qu'une hypothèse a été faite sur une variable en ajoutant un symbole $\sim$ après la variable lorsqu'elle apparaît dans le résultat d'un calcul.

On peut ajouter une nouvelle hypothèse à notre variable :

```
> simplify(sqrt((n-2)^2));          |n ~ -2|
```

La variable étant seulement entier, Maple ne peut simplifier la valeur absolue. Si l'on suppose de plus $n > 2$, une simplification se produit :

```
> additionally(n>2);
   simplify(floor(n));
   simplify(sqrt((n-2)^2));          n ~
                                     n ~ -2
```

Nous pouvons savoir quelles sont les hypothèses qui ont été faites sur nos variables :

```
> about(n);
   about(x);
   about(bidon);

Originally n, renamed n~:
   is assumed to be: AndProp(integer,RealRange(3,infinity))

Originally x, renamed x~:
   is assumed to be: RealRange(Open(0),infinity)

bidon:
   nothing known about this object
```

<u>Résumé</u> :	<code>assume(condition);</code>	Fait l'hypothèse exprimer par <code>condition</code> .
	<code>assume(nom::type)</code>	Fait l'hypothèse que la variable <code>nom</code> a pour type <code>type</code> .
	<code>additionally(condition)</code>	Ajoute l'hypothèse exprimer par <code>condition</code> .
	<code>additionally(nom::type)</code>	Ajoute l'hypothèse que la variable <code>nom</code> a pour type <code>type</code> .
	<code>about(var)</code>	Énonce les hypothèses faites sur la variable <code>var</code> .
	<code><=</code>	symbole $\leq$
	<code>>=</code>	symbole $\geq$

Exercice 1. *Un calcul interminable*

Calculer une valeur approchée de $T = \tan^6\left(\frac{\pi}{12}\right) + \tan^5\left(\frac{\pi}{12}\right) + \tan^4\left(\frac{\pi}{12}\right) + \tan^3\left(\frac{\pi}{12}\right) + \tan^2\left(\frac{\pi}{12}\right) + \tan\left(\frac{\pi}{12}\right)$.
On proposera une suite d'instructions aussi courte que possible.

Exercice 2.

1. On pose : $z = 1 + e^{i\frac{\pi}{3}}$.
Donner le module et un argument de z .
2. Donner la forme algébrique et le module de $Z = 1 + z + z^2$.

Exercice 3.

Soit $A = \frac{1-z^2}{1+z}$. On pose : $z = x + iy$ sous forme algébrique.

1. Donner la forme algébrique de A puis exprimer la partie réelle de A en fonction de x et de y .
2. Calculer la partie réelle de A lorsque $x = \frac{1}{2}$ et $y = \frac{1}{3}$ puis lorsque $x = 2$ et $y = 3$.

Exercice 4. Simplification

Simplifier l'expression suivante : $\sqrt{x^4 - 8x^3 + 22x^2 - 24x + 9}$ sous l'hypothèse $x \geq 3$ puis pour $1 \leq x \leq 3$.

Exercice 5. Intervertir deux valeurs

Exécutez les instructions suivantes :

```

> x:=v1;
  y:=v2;

```

$$x := v1$$

$$y := v2$$

Proposer un bloc d'instructions permettant d'intervertir le contenu des variables x et y sans exploiter les variables $v1$ et $v2$.

Indication : il sera nécessaire d'utiliser une variable auxiliaire.

II Fonctions

II.1 Définition

Nous pouvons créer les fonctions de notre choix :

```

> f:=x->3*sin(x)+4*cos(3*x);
  g:=unapply(3*sin(x)+4*cos(3*x),x);

```

$$f := x \rightarrow 3 \sin(x) + 4 \cos(3x)$$

$$g := x \rightarrow 3 \sin(x) + 4 \cos(3x)$$

Nous pouvons ensuite les évaluer en n'importe quel point :

```

> f(Pi/4);
  g(Pi/4);
  f(Pi/6);
  g(Pi/6);

```

$$-\frac{\sqrt{2}}{2}$$

$$-\frac{\sqrt{2}}{2}$$

$$\frac{3}{2}$$

$$\frac{3}{2}$$

Dans l'exemple précédent, les fonctions f et g sont identiques mais elles sont définies de façons différentes.

Voyons un autre exemple :

```

> y:=4;
  f1:=x->x*y;
  f2:=unapply(x*y,x);
  f1(4);
  f2(4);
  y:=2;
  f1(4);
  f2(4);

```

$$y := 4$$

$$f1 := x \rightarrow xy$$

$$f2 := x \rightarrow 4x$$

$$16$$

$$16$$

$$y := 2$$

$$8$$

$$16$$

Ici, les fonctions $f1$ et $f2$ sont distinctes. $f1$ est la fonction qui, à x , associe xy . La valeur renvoyée par $f1$ dépend de la valeur de y . $f2$ est, quant à elle, la fonction qui, à x , associe $4x$. Le comportement de $f2$ ne dépend pas de la valeur de y . La fonction `unapply` évalue donc les variables auxilliaires lorsqu'elle crée une fonction tandis que `->` ne le fait pas.

On peut créer des fonctions de plusieurs variables :

```

> g:=(x,y) -> x^2*y^3+4*x*y;

```

$$g := (x, y) \rightarrow x^2y^3 + 4xy$$

On peut l'évaluer en un point :

```

> g(3,1);

```

$$21$$

<u>Résumé :</u> <code>x->Expr</code>	fonction associée à l'expression <code>Expr</code> en la variable <code>x</code> (sans évaluation des autres variables)
<code>unapply(Expr, x)</code>	fonction associée à l'expression <code>Expr</code> en la variable <code>x</code> (avec évaluation des autres variables)
<code>f(x)</code>	Evalue la fonction <code>f</code> en <code>x</code> (cas d'une fonction d'une variable)
<code>(var₁, ..., var_n)-> Expr</code>	fonction associée à l'expression <code>Expr</code> en les variables <code>var₁, ..., var_n</code> (sans évaluation des autres variables)
<code>unapply(Expr, [var₁, ..., var_n])</code>	fonction associée à l'expression <code>Expr</code> en les variables <code>var₁, ..., var_n</code> (avec évaluation des autres variables)
<code>f(x₁, x₂, ..., x_n)</code>	Evalue la fonction <code>f</code> en <code>(x₁, x₂, ..., x_n)</code> (cas d'une fonction de plusieurs variables)

II.2 Opérations de l'analyse

II.2.a Addition, multiplication... composition...

Créons deux fonctions :

```
> h1:=x->ln(x)/x;
  h2:=x->exp(3*x);
```

$$h1 := x \rightarrow \frac{\ln(x)}{x}$$

$$h2 := x \rightarrow e^{3x}$$

On peut additionner ces deux fonctions :

```
> H1:=h1+h2;
  H1(x);
```

$$H1 := h1 + h2$$

$$\frac{\ln(x)}{x} + e^{3x}$$

les multiplier :

```
> H2:=h1*h2;
  H2(x);
```

$$H2 := h1 h2$$

$$\frac{\ln(x)e^{3x}}{x}$$

les diviser :

```
> H3:=h1/h2;
  H3(x);
```

$$H3 := \frac{h1}{h2}$$

$$\frac{\ln(x)}{xe^{3x}}$$

les composer :

```
> H4:=h1@h2;
  H4(x)
```

$$H4 := h1@h2$$

$$\frac{\ln(e^{3x})}{e^{3x}}$$

Ainsi, H_4 est la composée de h_1 par h_2 , i.e. $H_4 = h_1 \circ h_2$ et $H_4(x) = h_1(h_2(x))$.

II.2.b Calcul de limites

Nous pouvons calculer des limites :

```
> limit(sin(x)/x, x=0);
  limit(ln(x)/(x-1), x=1);
  limit(ln(x)/x, x=infinity);
```

1
1
0

<u>Résumé :</u> <code>limit(Expr, var=a)</code>	limite de l'expression <code>Expr</code> lorsque <code>var</code> tend vers <code>a</code>
<code>limit(Expr, var=infinity)</code>	limite de l'expression <code>Expr</code> lorsque <code>var</code> tend vers $+\infty$
<code>limit(Expr, var=-infinity)</code>	limite de l'expression <code>Expr</code> lorsque <code>var</code> tend vers $-\infty$

II.2.c Dérivation

a) Dériver une expression :

Maple peut dériver les expressions. Il faut cependant lui spécifier la variable par rapport à laquelle on dérive.

```
> diff(sin(x)/x,x);
diff(sin(x)/x,y);
```

$$\frac{\cos(x)}{x} - \frac{\sin(x)}{x^2}$$

$$0$$

On peut calculer les dérivées successives :

```
> diff(exp(x)*sin(x),x);           # première dérivée
diff(exp(x)*sin(x),x$2);         # dérivée seconde
diff(exp(x)*sin(x),x$3);         # dérivée troisième
```

$$e^x \sin(x) + e^x \cos(x)$$

$$2e^x \cos(x)$$

$$2e^x \cos(x) - 2e^x \sin(x)$$

On peut dériver par rapport à plusieurs variables successivement :

```
> diff(sin(x*y),x,y);           # par rapport à x puis par rapport à y
diff(sin(x*y),y,x);           # par rapport à y puis par rapport à x
```

$$-\sin(xy)xy + \cos(xy)$$

$$-\sin(xy)xy + \cos(xy)$$

b) Dériver une fonction :

Maple sait aussi dériver les fonctions :

```
> D(f);
```

$$x \rightarrow 3 \cos(x) - 12 \sin(3x)$$

Le résultat est cette fois-ci une fonction. Nous pouvons calculer la dérivée de f en $\frac{\pi}{4}$:

```
> D(f)(Pi/4);
```

$$-\frac{9\sqrt{2}}{2}$$

On peut également stocker ce résultat dans une variable fp et l'utiliser comme une fonction :

```
> fp:=D(f);
fp(Pi/4);
```

$$fp := x \rightarrow 3 \cos(x) - 12 \sin(3x)$$

$$-\frac{9\sqrt{2}}{2}$$

On peut également calculer les dérivées successives :

```
> (D@@3)(f);           # Dérivée troisième
```

$$x \rightarrow -3 \cos(x) + 108 \sin(3x)$$

Pour dériver les fonctions de plusieurs variables, il faut spécifier le numéro de la variable et non le nom de la variable utilisée lors de la création de la fonction.

```
> D[1](g);           # Dérivée par rapport à la première variable
D[2](g);           # Dérivée par rapport à la seconde variable
```

$$(x,y) \rightarrow 2xy^3 + 4y$$

$$(x,y) \rightarrow 3x^2y^2 + 4x$$

On peut dériver par rapport à la première variable puis par rapport à la seconde de g :

```
> D[1,2](g);
```

$$(x,y) \rightarrow 6xy^2 + 4$$

On peut dériver trois fois par rapport à la seconde variable puis une fois par rapport à la première :

```
> D[2$3,1](g);
```

$$(x,y) \rightarrow 12x$$

<u>Résumé :</u> diff(Expr,var)	calcule la dérivée de l'expression Expr par rapport à la variable var
diff(Expr,var ₁ ,...,var _n)	calcule la dérivée de l'expression Expr par rapport aux variables (var ₁ ,...,var _n)
D(f)	fonction dérivée de la fonction en une variable f
D[i](f)	fonction dérivée de la fonction f par rapport à la i ^{ème} variable
D[1\$n](f) ou (D@@n)(f)	n ^{ème} fonction dérivée de la fonction f
D[i ₁ ,...,i _n](f)	fonction dérivée de f par aux i ₁ ^{ème} , ..., i _n ^{ème} variables

II.2.d Intégration

Pour calculer une primitive d'une expression, il faut préciser la variable d'intégration :

```
> int(exp(x)*sin(x),x);
```

$$-\frac{1}{2}e^x \cos(x) + \frac{1}{2}e^x \sin(x)$$

On peut calculer une intégrale en précisant les bornes de la façon suivante :

```
> int(x^2*exp(x),x=0..1);
```

$$-2 + e$$

On obtient ainsi la valeur de $\int_0^1 x^2 e^x dx$.

Résumé : `int(Expr, var)` calcule une primitive de l'expression `Expr` par rapport à la variable `var`
`int(Expr, var=a..b)` calcule l'intégrale de l'expression `Expr` par rapport à la variable `var` entre `a` et `b`

Exercice 6. Calcul de limite

Calculer $\lim_{x \rightarrow 0} x^n \ln(x)$ où $n \in \mathbb{N}^*$.

Exercice 7. Une fonction

On considère la fonction f telle que : $f(x) = \left(1 + \frac{1}{x}\right)^x$.

1. Calculer la limite de f en $+\infty$.
2. Calculer une valeur approchée de $f(10^9)$ puis de $f(10^{10})$. Commenter.

Exercice 8. Étude d'une fonction

On considère f la fonction définie sur $\mathbb{R}$ par : $f(x) = \frac{e^x}{e^x + 1}$

1. Créer la fonction f .
2. Calculer les limites de f en $-\infty$ et en $+\infty$.
3. Calculer la dérivée de f puis simplifier l'expression de f' . Que dire du signe de la fonction f' ?
4. Que pouvez-vous déduire des deux questions précédentes ?
5. Résoudre grâce à la fonction `solve` l'équation d'inconnue x : $f(x) = y$ (E_y).
6. La solution à l'équation (E_y) est unique. Créer g la fonction qui, à y , associe cette solution.
7. Calculer $(g \circ f)(x)$ et $(f \circ g)(y)$.
8. Comparer $g'(f(x))$ à $f'(x)$.
9. Tracer les graphes de f et g en exécutant la commande :
`plot([f(x),g(x),x],x=-5..5,y=-5..5,color=[red,green,blue],scaling=constrained);`
Quel lien observez-vous entre le graphe de f en rouge et le graphe de g en vert ?

Exercice 9. Volume du ballon d'une montgolfière

Nous voulons calculer le volume du ballon d'une montgolfière. Ce ballon a été conçu de telle sorte que sa forme soit obtenue par la révolution autour de l'axe des abscisses du graphe de la fonction f définie sur $[0, 2]$ par :

$$f(x) = \sqrt{2-x} e^{x-1}$$

1. Créer la fonction f .
2. (a) Le graphe de f est obtenu par la commande : `plot(f,0..2,scaling=constrained);`
(b) La forme du ballon est obtenue par la commande :
`plot3d([x,f(x)*cos(theta),f(x)*sin(theta)],x=0..2,theta=-Pi..Pi,scaling=constrained,axes=normal);`
3. Calculer le volume du ballon.