CHAPITRE 2: LES ÉLÉMENTS DE BASE DE MAPLE

LES VARIABLES:

Une ***variable*** informatique est l'adresse , dans la mémoire de l'ordinateur , d'un emplacement permettant de stocker un objet d'un type donné .
Exemples: AB, aB , X1, _var , x[a] sont des noms de variables MAPLE .

Si l'on affecte à une variable une valeur , la variable est dite ***assignée***.
Si elle n'est associée à aucune valeur , la variable est dite ***non assignée***.

La fonction ***restart*** permet de réinitialiser toutes les variables de la feuille de calcul.

```
> restart;
```

var est une variable assignée, dont la valeur est 3 :

```
> var:=3;
```

$$var := 3$$

La variable *a* et la variable indicée x_a sont non assignées :

```
> a,x[a];
```

$$a, x_a$$

Une autre manière d'affecter une valeur à une variable est d'utiliser la fonction ***assign***:

```
> assign(nom,10-var);
> nom;
```

$$7$$

Pour désaffecter une variable assignée, lui affecter son nom écrit entre deux accents aigus :

```
> nom:='nom':nom;
```

$$nom$$

On peut aussi utiliser la fonction ***evaln*** (évaluer en un nom) :

```
> var:=evaln(var):var;
```

$$var$$

LES CONSTANTES:

true , ***false*** (valeurs booléennes vrai , faux) , ***I*** (complexe de carré -1) , ***Pi*** (nombre pi) , ***infinity*** (plus l'infini) sont des constantes MAPLE.
E (base des logarithmes népériens) n'est pas une constante MAPLE : faire ***exp(1).***

```
> eval(exp(1))=evalf(exp(1));
```

$$\mathbf{e} = 2.718281828$$

LES EXPRESSIONS :

Elles sont évaluées de la gauche vers la droite , avec la priorité usuelle des opérateurs.
Pour afficher une expression, utiliser les fonctions ***lprint*** ou ***print*** :

```
> lprint(1-2/a+(a+1)/(a-5));
1-2/a+(a+1)/(a-5)
> print(1-2/a+(a+1)/(a-5));
```

$$1-\frac{2}{a}+\frac{a+1}{a-5}$$

Toute expression a un ***type*** que l'on peut obtenir grâce à la fonction ***whattype*** :

Parmi les types possibles figurent : + (somme) , * (produit) , ^ (puissance) , = (égalité) , <> (non égalité) , < , <= (inégalité au sens strict ou large) , and , or , not (et,ou,non) , integer (nb entier) , fraction (nb rationnel) , float (nb réel en virgule flottante) , complex (nb complexe) , numeric (numérique) , symbol (symbole), string (chaîne de caractères) , list (liste) , set (ensemble) , table (table) , array(tableau) , function (fonction) , name (nom) , .. (intervalle).

```
> whattype(a+b),whattype(a>b),whattype(-1.2245),whattype(a^7),what
  type((a and b) or (c or d));
```

+, <, float, ^, or

```
> whattype(x-2/y),whattype(a/b),whattype(`MAPLE`),whattype("MAPLE"
  ),whattype({a,b});
```

*+, *, symbol, string, set*

On peut tester si une expression a un type donné en utilisant la fonction ***type*** :

```
> type(x,name),type(a+3*x-7/5,list);
```

true, false

Autres fonctions :

```
> expr:=x-3*y+5/(x+y);
```

$$expr := x-3\,y+\frac{5}{x+y}$$

nops(expression) donne le nombre d'opérandes de l'expression:

```
> nops(expr);
```

3

op(i,expression) donne le i-ème opérande de l'expression:

```
> op(2,expr);
```

$-3\,y$

op(i..j,expression) donne les opérandes de l'expression, du i-ème au j-ème:

```
> op(2..3,expr);
```

$$-3\,y,\ 5\,\frac{1}{x+y}$$

Transformation des expressions :

Regrouper certains termes selon une ou plusieurs variables avec ***collect(expression,v)*** ; *v* est une variable ou
une liste de variables :

```
> expr:=x^3*ln(x)+3*x^2+ln(x)*x-x:collect(expr,ln(x));
```

$$(x^3+x)\ln(x)+3x^2-x$$

```
> expr:=x^2+5*x-25*y+48*(x-2*y+7)+y^3:collect(expr,[x,y]);
```

$$x^2+53x-121y+336+y^3$$

Transformer en un seul terme avec ***combine(expression,noms)*** :
Le paramètre ***noms*** est optionnel et à choisir par exemple parmi : exp , ln , trig , power .

```
> combine(2*Sum(i^2,i=1..n)+Sum(i^3,i=1..n));
```

$$\sum_{i=1}^{n}(2i^2+i^3)$$

```
> combine(exp(sin(a)*cos(b))/exp(cos(a)*sin(b)),[trig,exp]);
```

$$e^{\sin(a-b)}$$

Simplifier une expression avec ***normal(expression)*** , simplifier une expression contenant des radicaux avec ***radsimp(expression)*** :

```
> normal( 1/x+x/(x+1) );
```

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

```
> normal( 1/x+x/(x+1), expanded );
```

$$\frac{x+1+x^2}{x^2+x}$$

```
> radsimp((x-1)*sqrt(x*x-2*x+1)*(x+1));
```

$$(x-1)^2(x+1)$$

Ordonner une expression avec ***sort(expression,v)***; *v* est une variable ou une liste de variables :

```
> sort(expr,y);
```

$$y^3-121y+53x+336+x^2$$

Ordonner selon un critère avec ***sort(liste,paramètre)*** ; ***liste*** est une liste de valeurs ,
paramètre = `<` ou `>` (ordre croissant, décroissant) , lexorder ou string (ordre lexicographique) .

```
> liste1:=[12,-2,5,1]:liste2:=[s,q,a,g,h,p,z,j,l,m,o,b,i]:
> sort(liste1,`>`),sort(liste2,lexorder);
```

$$[12, 5, 1, -2], [a, b, g, h, i, j, l, m, o, p, q, s, z]$$

Transformer une expression selon le modèle forme avec ***convert(expression,forme)*** :

```
> sin(2*x)=convert(sin(2*x),tan);
```

$$\sin(2\,x) = 2\,\frac{\tan(x)}{1+\tan(x)^2}$$

Faire des substitutions avec ***subs(eq1,eq2,...,eqn,expression):***

```
> expr;
  subs(x=lambda,y=2*t,expr);
  expr;
```

$$y^3 - 121\,y + 53\,x + 336 + x^2$$

$$8\,t^3 - 242\,t + 53\,\lambda + 336 + \lambda^2$$

$$y^3 - 121\,y + 53\,x + 336 + x^2$$

On constate que *expr* n'a pas changé.

```
> subsop(1=w,2=-t,4=mu,expr);
```

$$w - t + 53\,x + \mu + x^2$$

L'opérande n°1 a été remplacé par *w*, l'opérande n°2 a été remplacé par $-t$ et le n°4 par μ.

On peut également utiliser ***use ... in .. end use*** :

```
> use x=4*rho, y=h, z=mu in
    (x+y-z+3)^7
  end use;
```

$$(4\,\rho + h - \mu + 3)^7$$

Appliquer une fonction à tous les opérandes avec ***map(fonction,expression)***;

```
> map(x->x^2,expr);
```

$$y^6 + 14641\,y^2 + 2809\,x^2 + 112896 + x^4$$

Isoler une expression dans une équation avec ***isolate(equation,expression)***;

```
> expr:=-x^3+2*(sqrt(y-sin(x))+1):isolate(expr,y);
```

$$y = \left(\frac{1}{2}x^3 - 1\right)^2 + \sin(x)$$

Obtenir le membre gauche ou droit d'une égalité avec ***lhs(expression);*** ou ***rhs(expression);***

```
> lhs(%);rhs(%%);
```

$$y$$

$$\left(\frac{1}{2}x^3 - 1\right)^2 + \sin(x)$$

LES FONCTIONS:

Pour définir une ***fonction*** , on peut utiliser les syntaxes :

nom := var -> expr; pour une fonction d'une variable var
nom := (var1,var2,...,varN) -> expr; pour une fonction de N variables var1,var2,...,varN .

```
> f:=x->exp(x)-x;
```

$$f := x \to \mathbf{e}^x - x$$

```
> f:=(x,y,z,t)->x^2-3*y+4*z-t;
```

$$f := (x, y, z, t) \to x^2 - 3\,y + 4\,z - t$$

La fonction ***unapply***(expr,var1,var2,...,varN) permet de transformer l'expression expr en une fonction des N variables var1,var2,...,varN:

```
> expr:=u^2-3*v+4*w : f:=unapply(expr,u,v,w);
```

$$f := (u, v, w) \to u^2 - 3\,v + 4\,w$$

Composition des fonctions: l'opérateur @ , l'opérateur de composition itérée @@

```
> f:=u->u^3:g:=t->2*t+1:
  (f@g)(a);(g@f)(a);
```

$$(2\,a + 1)^3$$

$$2\,a^3 + 1$$

f@@N correspond à f o f o . . . o f (N fois)

```
> (f@@3)(u);
```

$$u^{27}$$

```
> (D@@2)(f);     # fonction dérivée seconde de f
```

$$u \to 6\,u$$

LES TYPES DE BASE:

LES TYPES NUMÉRIQUES

On peut utiliser la fonction ***assume*** pour imposer une condition:
Exemple: on impose à r la condition $0 < r$ (r est alors affiché suivi d'une tilde: $r\sim$)

```
> assume(r,positive);r;
  sign(r);
```

$$r\sim$$

$$1$$

La fonction ***about*** permet d'obtenir une information sur une variable:

```
> about(r);
Originally r, renamed r~:
  is assumed to be: RealRange(Open(0),infinity)
```

LES ENTIERS : ce sont des objets de type integer .
Un entier peut être *positive (>0)* , *negative* (<0), *nonneg* (>=0) , *posint* (entier>0) , *negint* (entier<0) , *nonnegint* (entier>=0) , *even* (pair) , *odd* (impair) , *primeint* (premier).
Ce que l'on peut tester avec ***type*** ou ***is***.

```
> type(-15,negint), is(13,even);
```

true, false

On peut sélectionner des éléments répondant à une condition booléenne avec ***select(condition,v);***
Exemple: sélectionner dans une liste de nombres ceux qui sont entiers >0

```
> select(x->type(x,posint),[-1/2,-7,Pi,3/4,0.25,18,2^5]);
```

[18, 32]

Quelques fonctions utilisant les entiers:

```
> q:=iquo(89,-12);r:=irem(89,-12);
```

q := -7

r := 5

(quotient et reste de la division euclidienne de a par b : a =bq +r , avec |r|<|b| , ici 89 = (-12) (-7) + 5)

54 est-il premier?

```
> isprime(54);
```

false

nombre premier suivant , nombre premier précédent:

```
> nextprime(215),prevprime(215);
```

223, 211

Le 17 ème nombre premier:

```
> ithprime(17);
```

59

Décomposition en facteurs premiers:

```
> ifactor(117208);
```

$(2)^3$ $(7)^2$ (13) (23)

Le ppcm et le pgcd de plusieurs entiers:

```
> ilcm(20,25,15);igcd(20,25,15);
```

300

5

Le plus petit ou le plus grand:

```
> min(712,56,100,25,125),max(712,56,100,25,125);
```

25, 712

mod (*a* modulo *n* donne le reste de la division de *a* par *n*, ici 31=4*7+3) :

```
> 31 mod 4;
```

3

rand(n) rend une fonction (appelée ici hasard) calculant un entier aléatoire entre 0 et $n-1$.

```
> hasard:=rand(100):h:=hasard();
```

$h := 81$

Fonction ***sign***: cette fonction rend : -1 si $n < 0$, 1 si $0 \le n$

```
> n:=-7;sign(n);
```

$n := -7$

-1

Fonction valeur absolue:

```
> abs(n);
```

7

LES RATIONNELS : ce sont des objets de type *fraction* (ou *rational*) .
Un rationnel peut être *positive* (>0) , *negative* (<0) , *nonneg* (>=0) , ce qui peut être testé par ***type*** ou ***is*** (comme pour les entiers).

<u>Quelques fonctions utilisant les rationnels:</u>

```
> q:=21*(144/124);numer(q);denom(q);
```

$$q := \frac{756}{31}$$

756

31

Comme pour les entiers , on a aussi les fonctions ***assume*** , ***min*** , ***max*** , ***sign*** , ***abs.***

LES REELS : un réel de type *float* , a une ***mantisse*** et un ***exposant***.
<u>Exemple:</u> 1234567=0.1234567*10^7 . La ***mantisse*** est 0.1234567 , et ***l'exposant*** 7 .
Le nombre s'écrit Float(0.1234567,7);

```
> evalf(Float(0.1234567,7));
```

$0.1234567\ 10^{7}$

L'utilisation du point **.** déclenche l'affichage en calcul flottant dans le second membre:

```
> 1/9=1.0/9;
```

$$\frac{1}{9} = 0.1111111111$$

Digits est une variable d'environnement représentant le nombre de décimales affichées (10 par défaut).

```
> Digits:=30:evalf(exp(1));
```

2.71828182845904523536028747135

<u>Quelques fonctions utilisant les réels:</u>

abs , ***sqrt*** (racine carrée) , ***exp*** , ***ln*** , ***log10*** , ***log[a]*** (logarithme de base a) , ***sin , cos , tan , cot*** (fonctions circulaires) , ***sinh , cosh , tanh , coth*** (fonctions hyperboliques) , ***arcsin ,arccos*** , ***arctan , arccot*** (fonctions circulaires réciproques) , ***arcsinh , arccosh , arctanh , arccoth*** (fonctions hyperboliques réciproques).
La fonction ***partie entière***: ***floor(x)*** est le plus grand entier inférieur ou égal à *x*.

```
> sqrt(1250);ln(exp(7));arctan(1);
```

$$25\sqrt{2}$$

$$7$$

$$\frac{1}{4}\pi$$

```
> floor(-7.98);floor(7.98);
```

$$-8$$

$$7$$

La fonction Gamma d'Euler:

```
> GAMMA(x)=Int(t^(x-1)*exp(-t),t=0..infinity);
```

$$\Gamma(x) = \int_0^{\infty} t^{(x-1)}\, \mathbf{e}^{(-t)}\, dt$$

LES COMPLEXES : de type *complex* , sont constitués de deux réels , la partie réelle et la partie imaginaire .

```
> z:=1-5*I*sqrt(2):z,Re(z),Im(z);
```

$$1 - 5\,I\sqrt{2},\ 1,\ -5\sqrt{2}$$

evalc : fonction d'évaluation des nombres complexes

```
> z:=evalc((1+exp(1)^(I*Pi/6))/(1-I));
```

$$z := \frac{1}{4} + \frac{1}{4}\sqrt{3} + \left(\frac{3}{4} + \frac{1}{4}\sqrt{3}\right) I$$

```
> Digits:=10:evalf(z);
```

$$0.6830127020 + 1.183012702\,I$$

```
> map(evalc,polar(a+I*b));
```

$$\mathrm{polar}(\sqrt{a^2 + b^2},\ \arctan(b, a))$$

Quelques fonctions utilisant les complexes:

abs(z) , argument(z) , conjugate(z) : module , argument dans **]-Pi,Pi]** , conjugué de z .

```
> z:=1+sqrt(3)*I:abs(z),argument(z),conjugate(z);
```

$$2,\ \frac{1}{3}\pi,\ 1 - \sqrt{3}\,I$$

AUTRES TYPES:

LE TYPE BOOLEAN :

Les booléens sont de type ***boolean*** et peuvent prendre 2 valeurs : ***false*** et ***true*** .
Si MAPLE ne parvient pas à résoudre un problème , il peut rendre le résultat ***FAIL*** .

Les opérateurs logiques sont : ***not*** (non) , ***or*** (ou) , ***and*** (et) , ***xor*** (ou exclusif) , ***implies*** (implication).
Une expression booléenne peut être évaluée par la fonction ***evalb*** :

```
> t:=10:
  evalb(t*t>0),evalb(isprime(t) and t>0);
```

true, false

```
> evalb(evalf(exp(1))<evalf(Pi));
```

true

LES TYPES NOM (name), SYMBOLE (symbol), et INDEXÉ (indexed)

Une expression Maple est de type indexé, si elle définie à partir d'indices ou d'index. Elle est alors de
type **indexed**.
Exemple: a_x est une variable indexée, contrairement à la variable *nom_var*

```
> type(a[x],`indexed`);
```

true

```
> type(nom_var,`indexed`);
```

false

Un nom, de type **name**, est une expression Maple définie à partir d'une lettre, suivie d'autres lettres
ou de chiffres, ou d'autres symboles, comme _ ,ou si elle est de type indexé.
Un nom peut aussi être formé en entourant une séquence de caractères par des accents graves ` `.

```
> type(a[x],`name`);
```

true

```
> type(nom_var,`name`);
```

true

```
> type(a/b,`name`);
```

false

```
> type(`12345`,`name`);
```

true

```
> type(12345,`name`);
```

false

Un symbole, de type **symbol,** est une expression Maple qui est un nom, mais n'est pas de type indexé.

```
> type(a[x],`symbol`);
```

false

```
> type(nom_var,`symbol`);
```

true

```
> type(a/b,`symbol`);
```

false

Quelques fonctions utilisant ces types:
length(*c*) retourne la longueur de *c* (*c* étant un nom ou un symbole) .
L'opérateur de ***concaténation*** || ou la fonction ***cat*** permettent de rassembler 2 noms ou symboles.

```
> length(nom_var);
```

7

```
> nom_var || a;
```

nom_vara

```
> cat(nom_var,a);
```

nom_vara

LE TYPE CHAÎNE DE CARACTÈRES (string)

Les chaînes de caractères, de type **string**, se notent entre guillemets " " :
par exemple , "MAPLE " et "7 pour Linux et Windows" sont 2 chaînes de caractères .
L'opérateur de ***concaténation*** || permet de rassembler 2 chaînes de caractères :

```
> ch:="MAPLE " || "7 pour Linux et Windows";
```

ch := "MAPLE 7 pour Linux et Windows"

On peut aussi utiliser la fonction ***cat*** :

```
> ch:=cat("MAPLE ","7 pour Linux et Windows");
```

ch := "MAPLE 7 pour Linux et Windows"

Quelques fonctions utilisant les chaînes de caractères:
length(*chaine*) retourne la longueur de la chaîne de caractères *chaine*.
substring(*chaine,a..b*) retourne la sous-chaîne constituée par les caractères de *chaine* situés entre les positions *a* et *b*.

```
> length(ch);
```

29

```
> substring(ch,9..18);
```

"pour Linux"

À propos des guillemets et accents:
Ne pas confondre , pour une variable *x* , `*x*` et '*x*' : *x* et `*x*` désignent le même nom de variable alors
que '*x*' représente la variable non assignée de nom *x*.
À ne pas, bien sûr, confondre avec "x" qui est la chaîne de caractères ne comportant qu'un caractère: x.

```
> x:=3:    x+1,`x`+1,'x'+1;
```

```
"x";
```

$$4, 4, x + 1$$

"x"

Exercice corrigé 2:

Ecrire une fonction ***F*** qui à ***n*** entier naturel non nul , associe le produit des ***n*** premiers nombres premiers.

```
> F:=n->product(ithprime(k),k=1..n);
```

$$F := n \to \prod_{k=1}^{n} \text{ithprime}(k)$$

```
> F(5);
```

$$2310$$

En effet, F(5) = 2.3.5.7.11 = 2310.

Travail dirigé 2:

TD 2.1:
Ecrire une fonction ***F*** qui à ***n*** entier naturel non nul , associe la somme des ***n*** premiers entiers naturels non nuls.

TD 2.2:
Soit l'expression *expr* définie par :

$$expr = (x - 3\,y + 2\,z)\,(x + y - 1)\,(1 - y + z)$$

1° De quel type est cette expression ?

2° Développer *expr*.

3° Ordonner *expr* suivant les puissances décroissantes de *x* .

4° Substituer dans *expr* : *x+1* à *x* , *y-2* à *y* , z^2 à *z* .

TD 2.3:
1° Définir la fonction P qui à *x* associe :

$$x^4 + (3 - 4\,a)\,x^3 + (-9\,a + 6\,a^2 + 2)\,x^2 + (-4\,a^3 - 4\,a + 9\,a^2)\,x + 2\,a^2 - 3\,a^3 + a^4$$

2° Factoriser P(x) et calculer ses racines x .

3° Développer P(x) et regrouper ses termes suivant les puissances décroissantes de a .

4° On prend désormais $a = -1$:

a) représenter graphiquement P pour x entre -1 et 1.

b) calculer l'image de $1 + \sqrt{3}$ par P .

5° Calculer l'intégrale de P entre -1 et 1.