Introduction à Maple

1 Utilisation

En Maple, une commande se finit toujours par ; ou :. Si elle se finit par :, Maple n'affiche pas le résultat. Il est possible de regrouper plusieurs commandes sur la même ligne.

```
 > 5+5; 4*4; 3-3:
                                    10
                                    16
 >
```

La commande % permet de rappeler le dernier résultat (affiché ou non), %% rappelle l'avant dernier résultat.

```
 > 5+5;
                                     10
 > %*2;
                                     20
 >
```

Pour multiplier 2 expressions, il faut toujours mettre un * entre les deux, idem pour la division : /.

```
 > 4*4;
                                     16
 >
```

Dans Maple comme dans tout langage de programation, il possible d'attribuer une valeur à une variable. Attention, le nom d'une variable commence toujours par une lettre et peut ensuite être suivie de chiffres. ATTENTION : les majuscules ont une importance.

```
 > toto23 := 8; ToTo23 := 6;
                                toto23 := 8
                                ToTo23 := 6

> toto23; ToTo23; toto23 * 2;
                                     8
                                     6
                                    16
>
```

Une fois assignée, une variable conserve sa valeur jusqu'à la fin de la session Maple. Il est possible de réinitialier toutes les variables par la commande restart ou une seule avec unassign.

```
 > toto := 2: titi := 3:
 > toto; unassign('titi'); titi;
                                     2
                                   titi
 > restart;
 > toto;
                                   toto
```

Afin de rendre plus lisible un programme ou une feuille de calcul, il est impératif d'ajouter un certain nombre de commentaires tout au long des calculs. Un commentaire commence par # et va jusqu'à la fin de la ligne. Les commentaires sont ignorés par Maple.

```
> toto := 2; #on met la valeur 2 à toto
                                    toto := 2
>
```

La commande ? permet d'acceder à l'aide de Maple. Pour l'utiliser, taper "? commande".

```
> ? restart;
restart
clear internal memory

Calling Sequences
     restart
[...]
```

2 Calcul numérique et calcul symbolique

On désigne par calcul numérique une suite d'oppérations arithmétiques mettant en jeu des valeurs numériques (*ie* des nombres).

Il se différencie du calcul symbolique qui autorise l'utilisation de symboles, définis ou non (x, cos, sin, $\int$, ...)

Généralement le calcul numérique donne un résultat *approché*, tandis que le calcul symbolique donne un résultat *exact*. Le calcul symbolique est beaucoup plus puissant que le calcul numérique.

```
> x^2+x^3 - (2*x + x^2);  #ceci est un calcul symbolique
                                     3
                                    x  - 2 x
> x := Pi; cos(3*x/2); # ceci est un calcul numérique
                                       0
>
```

Maple essaie par défaut de rendre un résultat exact et donc pas nécessairement sous forme numérique. Pour le forcer à évaluer les résultats exacts, il y plusieurs méthodes :

- evalf
- marquer un nombre sous forme "flottante" (0.5 au lieu de 1/2)

Le nombre de chiffres après la virgule est défini par la variable Digits. Par défaut elle est remise à 10 par la commande restart ou à chaque nouvelle feuille de calcul.

```
> 1/2 + 2; evalf(%); 0.5 + 2;
                                      5/2
                                  2.500000000
                                      2.5
> evalf(cos(1));
                                 0.5403023059
> Digits := 40: evalf ( cos(1));
                  0.5403023058681397174009366074429766037323
```

Maple dispose de nombreux outils permettant de travailler sur des expressions formelles. Voici par exemple 3 commandes fondamentales :

- expand(expr, expr1, expr2, ..., exprn) : développe les expressions
- simplify(expr) : tende de simplifier la formule.
- factor(expr) : factorise l'expression.

```
> simplify (exp( 3+b*log(a)));
                                          b
                                    exp(3) a
> expand((x+1)*(x+2));
                                   2
                                  x  + 3 x + 2
> factor(5* x^3 - 22*x^2 + 18*x-1);
                                        2
                            (x - 1) (5 x  - 17 x + 1)
```

Maple connaît beaucoup de fonctions mathématiques, ainsi que de nombreuses valeurs.
On peut citer par exemple les fonctions trigonométriques, le logarithme, l'exponentielle...

```
 > cos(Pi);
                                        -1
 > arctan(1);
                                        Pi
                                        ----
                                        4
 > evalf(exp(10));
                    22026.465794806716516957900645284244366 35
 > log[10](100);
                                         2
 > (1+I)^4;
                                        -4
```

Maple peut aussi résoudre des systèmes d'équations en utilisant la commande solve.

```
 > solve(x^3+x^2+x +1= 0, x);
                                    -1, I, -I
```

Pour tracer une courbe, il faut utiliser la commande plot (expr, intervalle).

```
> plot(x^3+x+1, x=0..1);
```

[graphe]

```
> plot(cos(x/(5+x))+3*x-2, x=0..1, y=-1..5);
```

[graphe]

3 Exercices

Exercice 1. Prise en main

Tapez les lignes suivantes et examinez le résultat :

```
z :=2*x+1 ;
evalf(z) ;
x := 5;
u :=z**2 ;
evalf(z) ;
diff(u,x) ;
```

```
unassign('x');
diff(u,x);
evalf(z);

restart ;
evalf(z);
diff(u,x);
```

Exercice 2. Prise en main - 2

```
convert(247,binary);
convert(1023,hex);
a:=sqrt(3);
a*a;
b:=2^(2/3);
b^3;
sin(Pi/12) ;
evalf(a);
Digits:=30;
evalf(a);
evalf (Pi, 40) ;
sqrt(-1);
evalc(ln(1+I));
evalf(ln(1+I),20);
I^I ;
v :=(-1)**(1/3) ;
evalf(v);
```

Exercice 3. Utilisation de `Digits`

- Afficher le résultat de $\cos(\frac{3\pi}{5})$ de façon exacte puis avec 7 chiffres après la virgule
- Affichez le même résultat avec 100 chiffres après la virgule.

Exercice 4. Trigo

- Développer $\cos(2x) + \sin(2x)$
- Montrer les formules de trigonométrie vues en cours : (exemple : $cos(a)sin(b) = \frac{1}{2}(sin(a+b) - sin(a-b))$, $cos(a)cos(b) = \frac{1}{2}(cos(a+b) + cos(a-b))$)

Exercice 5. Utilisation du graphisme

Tracer $cos(2) + sin(x/2)$ pour x variant entre -5 et 4.

Tracer $3x^3 - 4x + 10$ pour x variant entre -2 et 1.
- Combien ce polynôme a-t-il de racines réelles ?
- Trouver une approximation de sa (ses) racine(s) a 10^-2 près.

Exercice 6. Programmation

Faites un programme qui retourne la valeur 1.

Faites un programme qui retourne la valeur du nombre divisée par 2.

Implémenter l'algorithme d'Euclide vu plus haut.

4 Programmation

Rappel : comment faire une procédure

En Maple, ce que nous appelons une procédure est une fonction. Elle s'écrit de la façon suivante :

```
ma_fonction := proc(arg1, arg2, ... , argn)
  # Des lignes de Maple
end;
```

`arg1, arg2, ... argn` sont les arguments de la procédure et le résultat rendu est le dernier résultat calculé.

Exemple de procédure `fac` prenant un argument `n` et rendant la factorielle de `n` :

```
fac := proc(n)
   if n = 0
     then 1
     else n*fac(n-1)
   fi;
end;
fac(10);
                                    3628800
```

Séquences et listes

En informatique, on est souvent amener à manipuler un grand nombre de variables. Si on dispose des notes des 48 élèves d'une classe, on peut les stocker "bêtement" :

```
note1 := 18; note2 := 5, note3 := 9.5; ... note48 := 20;
```

Mais il existe des manières bien plus pratique en Maple : les listes et les séquences. Une séquence est une suite de chiffres, une liste est en gros une séquence entourée de crochets. On peut accéder au $i^{\text{ième}}$ élément de la liste `note` en tapant `note[i]`. On peut construire des séquences très facilement à l'aide de la commande `seq`. Dans la vie de tous les jours, la différence entre une liste et la séquence est la possibilité de faire des listes de listes.

Exemple :

```
note := 18,5,9.5,20: note[1];
                                      18

liste_des_ds := [ [note], [5,6,1,20], [6,11,8.4,14] ]:
liste_des_ds[2][3];
                                       1


#Utilisation de seq:
nombres := [seq( i^2, i=1..10)];
                      1, 4, 9, 16, 25, 36, 49, 64, 81, 100

#Pour appliquer une fonction à l'ensemble des éléments d'une liste:
traitement_nombre := seq( f(nombres[i]), i=1..nops(nombres));
   traitement_nombre :=
    f(1), f(4), f(9), f(16), f(25), f(36), f(49), f(64), f(81), f(100)

seq( [seq(i,i=1..j)], j=1..5);
              [1], [1, 2], [1, 2, 3], [1, 2, 3, 4], [1, 2, 3, 4, 5]
```