Séquences, Listes et Ensembles

Maple connaît plusieurs types de variables adaptées aux traitements des suites :
les **séquences** : à peu près n'importe quoi, ordonné et séparé par des virgules,
les **listes** : à peu près l'équivalent d'une suite au sens mathématique,
les **ensembles** : à peu près équivalent au sens mathématique.

```
> restart : # bonne idée de toujours commencer ainsi.
> s1 := solve(x^2+x+1);
```

$$s1 := -\frac{1}{2}+\frac{1}{2}I\sqrt{3}, -\frac{1}{2}-\frac{1}{2}I\sqrt{3}$$

Lorsque solve fournit plusieurs solutions, Maple retourne une séquence formée des solutions : l'ordre est arbitraire et peut
changer si vous réevaluez l'expression !

```
> whattype(s1); # comme son nom l'indique !
```

exprseq

```
> s1[1];
```

$$-\frac{1}{2}+\frac{1}{2}I\sqrt{3}$$

On accède aux éléments de la séquence avec l'opérateur d'indexation [].
On peut définir soi-même une séquence, ou la prolonger...

```
> s2 := 'Pi',3,14,16;
```

$$s2 := \pi, 3, 14, 16$$

```
> x := s2[2];
```

$$x := 3$$

```
> s1 := 0,s1;
```

$$s1 := 0, -\frac{1}{2}+\frac{1}{2}I\sqrt{3}, -\frac{1}{2}-\frac{1}{2}I\sqrt{3}$$

Il existe même une instruction pour construire des séquences, l'instruction seq(f(i),i=1..n) qui construit la séquence des f(i) où
f(i) est une expression dépendant de i, pas nécéssairement une fonction au sens Maple, pour i variant de 1 à n.
Rappel : 1..n désigne un intervalle.

```
> seq((i+1)/i,i=1..5);
```

$$2, \frac{3}{2}, \frac{4}{3}, \frac{5}{4}, \frac{6}{5}$$

Le deuxième argument de seq peut être de la forme i=s où s est une séquence...

```
> p := 2,3,5,7,11 : seq((2^i)-1,i=p);
```

3, 7, 31, 127, 2047

```
> s4 := solve(x^4+1);
```

$$s4 := \frac{1}{2}\sqrt{2}+\frac{1}{2}I\sqrt{2}, -\frac{1}{2}\sqrt{2}-\frac{1}{2}I\sqrt{2}, \frac{1}{2}\sqrt{2}-\frac{1}{2}I\sqrt{2}, -\frac{1}{2}\sqrt{2}+\frac{1}{2}I\sqrt{2}$$

```
> s5 := seq(x^4+1,x=s4);
```

s5 :=

$$\left(\frac{1}{2}\sqrt{2}+\frac{1}{2}I\sqrt{2}\right)^4+1, \left(-\frac{1}{2}\sqrt{2}-\frac{1}{2}I\sqrt{2}\right)^4+1, \left(\frac{1}{2}\sqrt{2}-\frac{1}{2}I\sqrt{2}\right)^4+1, \left(-\frac{1}{2}\sqrt{2}+\frac{1}{2}I\sqrt{2}\right)^4+1$$

C’est bien ce qu’on avait demandé, mais on aimerait simplifier !
On verra plus élégant tout à l’heure.

```
> simplify(s5[1]);
```

$$0$$

On peut définir un ensemble comme une séquence entre { }

```
> e1 := {a,b,c,a,d,e,a};
```

$$e1 := \{c, a, d, b, e\}$$

Remarquer que Maple a supprimé les doublons et réorganisé les choses à sa façon !

```
> whattype(e1); e1[2];
```

$$set$$

$$a$$

Maple connait la réunion union, l’intersection intersect et la différence minus

```
> e2 := e1 union {a,b,g,h}; e2 minus e1;
```

$$e2 := \{c, a, d, g, b, e, h\}$$

$$\{g, h\}$$

La dernière structure Maple est la liste, à peu de choses près une séquence, ou encore un ensemble ordonné.
Elle se déclare entre crochets

```
> L1 := [1,4,9,16]; whattype(L1) ;
```

$$L1 := [1, 4, 9, 16]$$

$$list$$

On accède aux éléments d’une liste comme pour une séquence avec [] ou en utilisant op(numero,liste).

```
> L1[2];  op(3,L1);
```

$$4$$

$$9$$

Deux instructions très utiles : nops() donne le nombre d’éléments de la liste, op() la séquence correspondante

```
> nops(L1); op(L1);whattype(");
```

$$4$$

$$1, 4, 9, 16$$

$$exprseq$$

Pour convertir une séquence en un ensemble, ou une liste, simplement l’écrire entre {} ou [].

```
> ep := {p}; lp := [p];
```

$$ep := \{3, 2, 7, 5, 11\}$$

$$lp := [2, 3, 5, 7, 11]$$

Il est impossible d’ajouter directement un (ou plusieurs) élément(s) à une liste ; la méthode consiste à la transformer en
séquence, prolonger la séquence et la retransformer en liste.

```
> L1 := [op(L1),25,36];
```

$$L1 := [1, 4, 9, 16, 25, 36]$$

Souvent, pour Maple, liste et ensemble sont interchangeables, mais ce n’est pas toujours vrai.
Revenons sur le calcul de s5 : une solution plus élégante est d’utiliser l’instruction map qui applique aux éléments d’une liste une fonction (au sens Maple du terme)

```
> map(simplify,[s5]);
```

$$[0, 0, 0, 0]$$

```
> map(x->x^2,[seq(i,i=1..12)]);
```

$$[1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144]$$

Essayer de comprendre la place des parenthèses et des crochets ! et remarquer que le premier opérande est bien une fonction
Maple.

Bien sûr tous ces types peuvent être utilisés dans des procédures, comme argument et comme valeur de retour...
Lorsque vous écrivez une procedure ayant, par exemple un argument de type liste, il est possible d'obliger Maple
à vérifier le type de l'argument.

```
> myproc := proc(L::list) 'ok c'est une liste' end;
```

myproc := **proc**(*L*::*list*) *'ok c'est une liste'* **end proc**

```
> myproc(L1);
```

ok c'est une liste

```
> myproc(e1);
Error, myproc expects its 1st argument, L, to be of type list, but received {c,
a, d, b, e}
> myproc([e1]);
```

ok c'est une liste

On peut bien entendu vérifier tous les types que Maple connaît...
Un dernier aspect des fonctions et procédures Maple est la récursivité : une procedure peut s'appeler elle-même.
L'exemple classique est la fonction factorielle :

```
> fact := proc(n::integer)
  if (n <= 0) then 1 else n*fact(n-1); fi;
  end;
```

fact := **proc**(*n*::*integer*) **if** $n \le 0$ **then** 1 **else** $n * \mathrm{fact}(n-1)$ **end if end proc**

```
> fact(5),fact(-5);
```

120, 1

On va construire les polynômes de Tchebicheff de manière récursive ; ils sont définis par cos(nt)=Tn(cos(t)) et vérifient lar relation de récurrence Tn(x)+Tn-2(x)=2xTn-1(x).

```
> Tche := proc(n) global cpt;
  cpt := cpt+1;
  if n=0 then 1
  elif n=1 then x
  else expand(2*x*Tche(n-1)-Tche(n-2))
  fi;
  end;
```

Tche := **proc**(*n*)

global *cpt*;

cpt := *cpt* + 1;

if $n = 0$ **then** 1 **elif** $n = 1$ **then** *x* **else** expand$(2*x*\mathrm{Tche}(n-1) - \mathrm{Tche}(n-2))$ **end if**

end proc

Essayez de comprendre pourquoi cpt est globale et ce qu'elle fait !

```
> cpt :=0 : Tche(12) ; cpt;
```

768398401

465

Manifestement on n'a pas libéré x !

```
> x := 'x' : cpt :=0 : Tche(25) ; cpt;
```

$288358400\,x^{21} - 2600\,x^{3} + 80080\,x^{5} - 1144000\,x^{7} + 16777216\,x^{25} - 104857600\,x^{23}$

$$-317521920\,x^{15}+9152000\,x^{9}+25\,x+146227200\,x^{13}-458752000\,x^{19}+466944000\,x^{17}$$
$$-45260800\,x^{11}$$

242785

```
> sort(""); # afin d'ordonner le polynôme.
```

$$16777216\,x^{25}-104857600\,x^{23}+288358400\,x^{21}-458752000\,x^{19}+466944000\,x^{17}$$
$$-317521920\,x^{15}+146227200\,x^{13}-45260800\,x^{11}+9152000\,x^{9}-1144000\,x^{7}+80080\,x^{5}$$
$$-2600\,x^{3}+25\,x$$

Attention aux instructions x := x+1, si x n'est pas initialisé avant d'entrer dans la procédure : Maple essaye de définir x
de façon récursive...plantage sauvage garanti et risque de perte définitive de votre travail si pas sauvegardé !!

```
> destroy := proc(n) global xx; xx := xx + 1 ; print(n,xx) ; end;
```

destroy := **proc**(*n*) **global** *xx*; *xx* := *xx* + 1; print(*n*, *xx*) **end proc**

```
> destroy(5);
Error, (in destroy) too many levels of recursion
> xx := 3 :destroy(5);
```

5, 4

Si on veut Tche(26) on mettra un peu plus de temps encore... alors que Maple à déjà calculé les 25 premiers.
On va l'obliger à s'en souvenir en ajoutantoption remember.

```
> restart:
> Tche2 := proc(n) global cpt;
  option remember ;
  if n=0 then 1
  elif n=1 then x
  else expand(2*x*Tche2(n-1)-Tche2(n-2))
  fi;
  end;
```

Tche2 := **proc**(*n*)

global *cpt*;

option *remember*;

if $n = 0$ **then** 1 **elif** $n = 1$ **then** *x* **else** expand(2∗*x*∗Tche2($n-1$) − Tche2($n-2$)) **end if**

end proc

```
> debut := time() : Tche2(25) : time()-debut;
```

.085

```
> debut := time() : Tche2(26) : time()-debut;
```

0

Spectaculaire, car les appels récursifs profitent eux aussi de la table !
On peut voir la table de remember d'une procédure à l'aide de

```
> op(4,eval(Tche2)): # mettez un point virgule pour la voir.
```