Règles de substitution

L'opération consistant à attribuer dans une expression une valeur à un symbole peut être réalisée avec *Mathematica* par l'application de *règle de substitutions*.

Appliquer par exemple la règle `y->1` à l'expression `x^2+2*x*y+y^2-x^3*y` aura pour effet de substituer, **temporairement**, la valeur 1 à la variable `y`.

Pour appliquer une régle `reg` à une expression `expr`, on écrit `expr/.reg`. L'expression qui précède le symbole `/.` est alors évaluée en tenant compte de la substitution indiquée.

Évaluer les entrées suivantes :

```
p=x^2+2*x*y+y^2-x^3*y
p/.y->1
p
```

On peut aussi remplacer `y` par une autre expression :

```
p/.y->Sin[a]
```

On peut aussi appliquer successivement plusieurs règles,

```
x^2+y/.x->1/.y->0
```

On peut aussi appliquer indépendamment plusieurs règles (le résultat de l'évaluation est alors une *liste* de résultats)

```
x^2+y^2/.{{x->1},{y->2}}
```

Vous avez pu déjà pu rencontrer des situations où *Mathematica* utilise des règles. Par exemple, la fonction `Solve` fournit des résultats sous formes de règles.

```
reg=Solve[x^3==8,x]
```

Pour obtenir directement les valeurs, solutions de l'équation $x^3 = 8$, on pourra écrire :

```
x/.reg
```

ou pour obtenir la valeur d'une fonction f en ces valeurs, on écrira

```
f[x]/.reg
```

Prévoir le résultat des évaluations suivantes :

```
trinomSol=Solve[a*x^2+b*x+c==0,x]
a*x^2+b*x+c/.trinomSol
Simplify[%]
systemSol=Solve[{2*x+3*y==7,3*x-y==2},{x,y}]
{2*x+3*y==7,3*x-y==2}/.systemSol
```

1. Calculer la dérivée seconde par rapport à t de $\frac{x}{\exp(y(a-t))}$ et vérifier qu'elle vaut xy^2 quand $t = a$.

Itérations paramétrées, conditionnelles (boucles) et les fonctions Do, While, Table

L'itération paramétrée, de syntaxe `Do[expr,iter]`, évalue `expr` autant de fois qu'indiqué par `iter`. L'*itérateur* `iter` peut prendre plusieurs formes. (voir l'aide pour les détails sur `iter`)

Évaluer les entrées suivantes :

```
Do[Print[i^2],{i,5}]
Do[Print[i^2],{i,6,0,-2}]
Do[Print[{i,j}],{i,4},{j,i-1}]
Do[Print[i^2],{i,6,0,-2}]
```

Ou encore :

```
Clear[a];
a=2;
Do[a=2*a,{2}]
a
```

1. On considère la suite $(a_n)_{n\geq 0}$ définie par $a_0 = 1,\ a_{n+1} = \frac{1}{2}(a_n + \frac{2}{a_n})$.
   Évaluer les entrées suivantes :
   ```
   x=1;
   Do[x=1/2*(x+2/x),{2}]
   x
   ```
   Que représente la dernière valeur prise par la variable `x` ? Évaluer
   ```
   Do[x=1/2*(x+2/x),{3}]
   ```
   Est-ce que la valeur prise par `x` est maintenant le réel a_3 ?
   Programmer une fonction nommée `racine2[n_]` qui calcule le terme a_n :
   ```
   racine2[n_]:=(x=1;Do[...];x)
   ```

Il est plus judicieux dans la construction précédente d'utiliser la fonction `Module` qui introduira `x` comme *variable locale.*

```
racine2bis[n_]:=Module[{x=1},...]
```

De manière analogue, programmer une fonction `suiteRacine2[n_]` qui donne les $n+1$ premiers termes de la suite (a_n) sous la forme d'une liste (on utilisera la fonction `Append` qui permet de modifier une liste) :

```
suiteRacine2[n_]:=Module[{x=1,l={1}},
    Do[(x=1/2*(x+2/x);l=...]
```

Il est possible de construire une telle liste à l'aide de la fonction `Table`. Ainsi, il suffit d'écrire :

```
suiteRacine3[n_]:=Module[{x=1},
    Table[x=1/2*(x+2/x),{n}]]
```

2. *(Extrait oraux Centrale 2007)*
   Soit $(x_n)_{n\geq 1}$ définie par $x_1 = x$ et $\forall n \in \mathbb{N}^*$, $x_{n+1} = x_n + n/x_n$.
   (a) Programmer une fonction `exo2[x_,n_]` qui donne le terme de rang n de la suite (x_n).
   (b) Dresser la liste des 10 premiers termes de la suite (x_n). (avec ou sans la fonction `Table`)
3. *(Suite de Fibonacci)*
   Soit $(u_n)_{n\geq 1}$ la suite numérique à valeurs réelles définies par $u_1 = 1,\ u_2 = 1,$ et $\forall n \in \mathbb{N},\ u_{n+2} = u_{n+1} + u_n$. Programmer une fonction permettant de calculer u_n puis une fonction permettant d'afficher la liste des n premiers termes de la suite.

L'itération conditionnelle (Tant que ...faire ...) s'emploie lorsqu'on ne connaît pas à l'avance le nombre fois où l'`expr` doit être évaluée. La syntaxe de la fonction `While` est `While[cond,expr]`. La condition `cond` est testée et si le résulat est `True`, `expr` est évaluée, ceci répétitivement jusqu'à que `cond` ne soit plus évaluée `True`.

1. La suite (a_n) définie plus haut est convergente vers $\sqrt{2}$ (*admis*). Déterminer à l'aide de *Mathematica* le plus petit entier n vérifiant $|a_n^2 - 2| < 10^{-20}$.

Exercices divers

1. *(Extrait oraux 2011)*
   Soit $(u_n)_{n\geq 1}$ la suite définie par :
   $$u_1 = 0,\ u_2 = 2,\ u_3 = 3,\ \text{et } u_n = u_{n-2} + u_{n-3}.$$
   On désire montrer que pour tout p premier, p divise u_p.
   (a) Construire une fonction *Mathematica* qui calcule u_n.
   (b) Vérifier la propriété pour les 1000 premiers nombres premiers.
   (c) À suivre ...
2. Déterminer et représenter graphiquement l'ensemble des nombres complexes z tels que $z^3 - z^2 + z - 1$ soit imaginaire pur.
3. Déterminer la fraction de dénominateur minimal dans l'intervalle $\left]\frac{19}{94}, \frac{17}{76}\right[$.
4. *(Extrait oraux Centrale 2007)*
   On considère une suite réelle $(u_n)_{n\geq 0}$ vérifiant $u_{n+2} = (n+1)u_{n+1} - (n+2)u_n$. Calculer à l'aide de *Mathematica* les 10 premiers termes de la suite quand $u_0 = u_1 = -1$.