PYTHON diapo 14

Calcul formel
et calcul littéral

Roblet [2]

dernière MAJ le 04/01/20

Diapo 14 Calcul formel et calcul littéral

1ère ligne

Pour chaque calcul formel ou littéral, on commencera par importer la bibliothèque `sympy` avec la commande suivante :

```
>>> from sympy import *
```

2ème ligne

Puis il faudra définir les variables utilisées comme étant des variables formelles avec la commande `symbols( )` :

```
>>> x=symbols('x')
>>> x,y=symbols('x,y')
```

Diapo 14 Calcul formel et calcul littéral

Cliquez directement sur le chapitre qui vous intéresse.

développer
`expand( )`

factoriser
`factor( )`

simplifier
`simplify( )`

substituer
$A(x)$`.subs( , )`

résoudre une équation
`solveset( )`
→ donne les solutions sous forme d'un ensemble

résoudre une équation
`solve( )`
→ donne les solutions sous forme d'une liste

résoudre un système d'équations
`solve( )`

Diapo 14 Calcul formel et calcul littéral

développer
expand( )

1/2

Exemples

```
>>> from sympy import *
>>> x=symbols('x')
>>> expand((x+3)**2)
x**2 + 6*x + 9
```

```
>>> from sympy import *
>>> x,y=symbols('x,y')
>>> expand((x+y)**3)
x**3 + 3*x**2*y + 3*x*y**2 + y**3
```

Diapo 14 Calcul formel et calcul littéral

développer
expand( )

2/2

Remarque

On peut aussi développer $(1+\sqrt{5})^2$:

```
>>> from sympy import *
>>> expand((1+sqrt(5))**2)
2*sqrt(5) + 6
```

Calculs effectués

$$(1+\sqrt{5})^2 = 1^2 + 2 \times 1 \times \sqrt{5} + (\sqrt{5})^2$$

$$= 1 + 2\sqrt{5} + 5$$

$$\boxed{(1+\sqrt{5})^2 = 2\sqrt{5} + 6}$$

Diapo 14 Calcul formel et calcul littéral

factoriser
`factor( )`

1/2

Exemples

```
>>> from sympy import *
>>> x=symbols('x')
>>> factor(4*x**2-9)
(2*x - 3)*(2*x + 3)
```

```
>>> from sympy import *
>>> x=symbols('x')
>>> factor(-x**2+2*x-1)
-(x - 1)**2
```

Diapo 14 Calcul formel et calcul littéral

factoriser
`factor( )`

2/2

Quand il n'y a pas de factorisation possible dans $\mathbb{R}$:

```
>>> from sympy import *
>>> x=symbols('x')
>>> factor(3*x**2+4*x+5)
3*x**2 + 4*x + 5
```

Diapo 14 Calcul formel et calcul littéral

simplifier
simplify( )

1/2

`simplify` permet de simplifier une expression littérale.

Python réduit automatiquement sans `simplify` :

```
>>> from sympy import *
>>> x,y=symbols('x,y')
>>> 2*x+3*x
5*x
```

Par contre l'expression $\frac{6x+10}{2x}$ est inchangée :

```
>>> (6*x+10)/(2*x)
(6*x + 10)/(2*x)
```

Avec `simplify`, on a obtenu $\frac{6x+10}{2x} = 3 + \frac{5}{x}$:

```
>>> simplify((6*x+10)/(2*x))
3 + 5/x
```

Diapo 14 Calcul formel et calcul littéral

simplifier
`simplify( )`

A noter qu'il n'y a pas toujours d'équivalence entre les expressions, comme ici avec la valeur interdite $x = 0$:

```
>>> simplify((2*x+x*y)/x)
y + 2
```

Diapo 14 Calcul formel et calcul littéral

substituer
$A(x).\text{subs}(\ ,\ )$

1/3

Syntaxe

Dans cette expression...

... on veut remplacer ça...

... par ça.

$$A(x).\text{subs}(\ ,\ )$$

Exemple On veut calculer $x^2 - 3$ pour $x = 2$.

```
>>> from sympy import *
>>> x,y=symbols('x,y')
>>> expression=x**2-3
>>> expression.subs(x,2)
1
```

ou

```
>>> from sympy import *
>>> x,y=symbols('x,y')
>>> (x**2-3).subs(x,2)
1
```

```
>>> from sympy import *
>>> x,y=symbols('x,y')
>>> subs(x**2-3,x,2)
    *** message d'erreur ***
```

Diapo 14 Calcul formel et calcul littéral

substituer
$A(x)$.subs(,)

Exemple On veut calculer $2x - y + 5$ pour $x = 1$ et $y = 3$.

```
>>> from sympy import *
>>> x,y=symbols('x,y')
>>> expression=2*x-y+5
>>> expression.subs([(x,1),(y,3)])
4
```

Exemple On veut remplacer x par y dans l'expression $x^2 - 3$.

```
>>> from sympy import *
>>> x,y=symbols('x,y')
>>> expression=x**2-3
>>> expression.subs(x,y)
y**2 - 3
```

Diapo 14 Calcul formel et calcul littéral

substituer
$A(x)$.subs(,)

3/3

Remarque importante

Lorsqu'on applique la commande subs à une expression, celle-ci ne change pas.

Exemples

```
>>> from sympy import *
>>> x,y=symbols('x,y')
>>> expression=x+1
>>> expression.subs(x,5)
6
>>> expression
x + 1
>>> expression.subs(x,y)
y + 1
>>> expression
x + 1
```

Diapo 14 Calcul formel et calcul littéral

résoudre une équation
`solveset( )`
→ donne les solutions sous forme d'un ensemble

1/4

Pour résoudre une équation du type $A(x) = 0$,
on peut utiliser la commande `solveset(A(x),x)`
ou `solveset(A(x))`

Exemples

Si on veut résoudre $x^2 + 2x - 3 = 0$:

```
>>> from sympy import *
>>> x=symbols('x')
>>> solveset(x**2+2*x-3,x)
{-3, 1}
```

ou

```
>>> from sympy import *
>>> x=symbols('x')
>>> solveset(x**2+2*x-3)
{-3, 1}
```

Diapo 14 Calcul formel et calcul littéral

résoudre une équation
`solveset( )`
→ donne les solutions sous forme d'un ensemble

Remarque 1

La solution est du type « set » (= « ensemble » en anglais).

Un « set » est une collection d'objets qui ne sont ni ordonnés ni indexés, et dans laquelle il n'y a pas de répétition.

Si l'on veut accéder à un élément dans un « set », on peut par exemple utiliser une boucle `for + in` .

Exemple

On reprend l'exemple précédent, et on le modifie pour n'afficher que les solutions positives :

Exemple précédent

```
>>> from sympy import *
>>> x=symbols('x')
>>> solveset(x**2+2*x-3)
{-3, 1}
```

```
>>> from sympy import *
>>> x=symbols('x')
>>> solutions=solveset(x**2+2*x-3)
>>> for k in solutions:
...     if k>0:
...         print(k)
...
1
```

Diapo 14 Calcul formel et calcul littéral

résoudre une équation
`solveset( )`
→ donne les solutions sous forme d'un ensemble

Remarque 2

`solveset` peut renvoyer un ensemble infini d'éléments, contrairement à `solve` .

Exemple On veut résoudre $\sin(x) = 0$.

```
>>> from sympy import *
>>> x=symbols('x')
>>> solveset(sin(x))
ImageSet(Lambda(_n, 2*_n*pi), Integers()) U ImageSet(Lambda(_n, 2*_n*pi + pi), Integers())
>>> solve(sin(x))
[0, pi]
>>>
```

`solve` ne donne que 2 solutions : 0 et π.

On ne rentrera pas dans le détail des fonctions « lambda » de Python, mais `solveset` renvoie ici la réponse $\{2n\pi, n \in \mathbb{N}\} \cup \{2n\pi + \pi, n \in \mathbb{N}\}$.

résoudre une équation
`solveset( )`
→ donne les solutions sous forme d'un ensemble

Remarque 3

L'équation $3x^2 + 4x + 5 = 0$ n'a pas de solution dans $\mathbb{R}$, mais en a deux dans $\mathbb{C}$.

```
>>> from sympy import *
>>> x=symbols('x')
>>> solveset(3*x**2+4*x+5,x)
{-2/3 - sqrt(11)*I/3, -2/3 + sqrt(11)*I/3}
```

Avec Sympy, « i » est noté « I ».
Pour la notation standard de « i » dans Python, se référer au diaporama « Complexes ».

Par défaut, `solveset` donne les solutions dans $\mathbb{C}$, mais on peut ne demander que les solutions dans $\mathbb{R}$ en rajoutant `domain=S.Reals` :

```
>>> solveset(3*x**2+4*x+5,x,domain=S.Reals)
EmptySet()
```

∅ « empty set » signifie en anglais « ensemble vide ».

Diapo 14 Calcul formel et calcul littéral

résoudre une équation
`solve( )`

→ donne les solutions sous forme d'une liste

Pour résoudre une équation du type $A(x) = 0$,

on peut utiliser la commande `solve(A(x),x)`

ou `solve(A(x))`

Exemples

Si on veut résoudre $x^2 + 2x - 3 = 0$:

```
>>> from sympy import *
>>> x=symbols('x')
>>> solve(x**2+2*x-3,x)
[-3, 1]
```

ou

```
>>> from sympy import *
>>> x=symbols('x')
>>> solve(x**2+2*x-3)
[-3, 1]
```

Diapo 14 Calcul formel et calcul littéral

résoudre un système d'équations
`solve( )`

1/4

Exemple

On veut résoudre le système $\begin{cases} 2x + 5y - 8 = 0 \\ -x + 3y - 7 = 0 \end{cases}$.

```
>>> from sympy import *
>>> x,y=symbols('x,y')
>>> solve((2*x+5*y-8,-x+3*y-7),(x,y))
    *** message d'erreur ***
>>> solve((2*x+5*y-8,-1*x+3*y-7),(x,y))[x]
-1
>>> solve((2*x+5*y-8,-1*x+3*y-7),(x,y))[y]
2
```

$x = -1$

$y = 2$

Diapo 14 Calcul formel et calcul littéral

résoudre un système d'équations
`solve( )`

Voici une autre syntaxe pour résoudre le système précédent :

```
>>> from sympy import *
>>> x,y=symbols('x,y')
>>> solutions=solve((2*x+5*y-8,-1*x+3*y-7),(x,y))
>>> solutions[x],solutions[y]
(-1, 2)
```

résoudre un système d'équations
`solve( )`

Lorsque le système n'admet pas de solution, un message d'erreur apparaît.

Exemple On veut résoudre le système $\begin{cases} 2x+y-7=0 \\ -2x-y-8=0 \end{cases}$.

```
>>> from sympy import *
>>> x,y=symbols('x,y')
>>> solutions=solve((2*x+y-7,-2*x-y-8),(x,y))
>>> solutions[x],solutions[y]
    *** message d'erreur ***
```

Diapo 14 Calcul formel et calcul littéral

résoudre un système d'équations
`solve( )`

Lorsque le système admet une infinité de solutions,

un message d'erreur apparaît.

Exemple On veut résoudre le système $\begin{cases} 2x+y-7=0 \\ -2x-y+7=0 \end{cases}$.

```
>>> from sympy import *
>>> x,y=symbols('x,y')
>>> solutions=solve((2*x+y-7,-2*x-y+7),(x,y))
>>> solutions[x],solutions[y]
    *** message d'erreur ***
```