

Introduction à Sage

Thierry Dumont
Institut Camille Jordan.

Tipaza, le 13 Janvier 2013.

Sage

<http://sagemath.org>

- Sage is a *free open-source* mathematics software system *licensed under the GPL*. It *combines* the power of many existing open-source packages into a common *Python*-based interface.

Sage

<http://sagemath.org>

- Sage is a *free open-source* mathematics software system *licensed under the GPL*. It *combines* the power of many existing open-source packages into a common *Python*-based interface.
- Mission : Creating a viable free open source alternative to Magma, Maple, Mathematica and Matlab.

Logiciels libres

- Un logiciel libre est un logiciel dont l'utilisation, l'étude, la modification et la duplication en vue de sa diffusion sont permises, techniquement et légalement. Ceci afin de garantir certaines libertés induites, dont le contrôle du programme par l'utilisateur et la possibilité de partage entre individus.
- Ces droits peuvent être simplement disponibles (cas du domaine public) ou bien établis par une licence, dite « libre », basée sur le droit d'auteur. Les « licences copyleft » garantissent le maintien de ces droits aux utilisateurs même pour les travaux dérivés.

Logiciels libres

- 1 la liberté d'exécuter le programme, pour tous les usages ;
- 2 la liberté d'étudier le fonctionnement du programme et de l'adapter à ses besoins ;
- 3 la liberté de redistribuer des copies du programme (ce qui implique la possibilité aussi bien de donner que de vendre des copies) ;
- 4 la liberté d'améliorer le programme et de distribuer ces améliorations au public, pour en faire profiter toute la communauté.

License : GNU General Public License : GNU GPL (entre autres).

Sage

À l'origine : *Software for Algebraic and Geometric Experimentations*

À présent : la Sauge.

Projet initié et géré par William Stein, Université de l'état de Washington, Seattle.

À l'origine : *Software for Algebraic and Geometric Experimentations*

À présent : la Sauge.

Projet initié et géré par William Stein, Université de l'état de Washington, Seattle.

Faire travailler de manière *transparente* des outils (bibliothèques, logiciels) pour obtenir un outil général.

Large communauté de développeurs.

[composants.pdf](#)


```
sage: K = RealDoubleField()
sage: A = Matrix(K, [[1,3,2], [1,4,2], [0,1,20]])
sage: B = vector(K, [1,2,3])
sage: X = A\B
```

RealDoubleField() : nombres flottants double précision habituels => LAPACK (et ATLAS).

```
sage: K = RealDoubleField()
sage: A = Matrix(K, [[1,3,2], [1,4,2], [0,1,20]])
sage: B = vector(K, [1,2,3])
sage: X = A\B
```

RealDoubleField() : nombres flottants double précision habituels => LAPACK (et ATLAS).

sage: K = IntegerRing() ou sage: K = RationalField() : IML (Integer Matrix Library) et LinBox.

s-all-2

Décomposition LU ou forme échelon : il faut être dans un anneau intègre
(contre exemple : $\mathbb{Z}/4\mathbb{Z}$).

Décomposition LU ou forme échelon : il faut être dans un anneau intègre (contre exemple : $\mathbb{Z}/4\mathbb{Z}$).

Utiliser un algorithme :

- spécialisé (lapack etc).
- générique : fonctionne dans tout ensemble muni de $x / + -$ et qui est un anneau intègre : on doit pouvoir tester cette propriété.

Décomposition LU ou forme échelon : il faut être dans un anneau intègre (contre exemple : $\mathbb{Z}/4\mathbb{Z}$).

Utiliser un algorithmes :

- spécialisé (lapack etc).
- générique : fonctionne dans tout ensemble muni de $x / + -$ et qui est un anneau intègre : on doit pouvoir tester cette propriété.

Comment réaliser ce programme ? Python et la programmation orientée objet.

Construire la voiture à partir des pièces.

Utilisation comme calculatrice

calculatrice

formelAnalyse

graphiques -navigateur-

flottant

Polynômes

Sage permet de calculer dans les anneaux de polynômes $A[x]$, leurs quotients $A[x]/\langle P(x) \rangle$, les corps de fractions rationnelles $K(x)$ ou encore les anneaux de séries formelles $A[[x]]$ pour toute une gamme d'anneaux de base.

Polynômes

Sage permet de calculer dans les anneaux de polynômes $A[x]$, leurs quotients $A[x]/\langle P(x) \rangle$, les corps de fractions rationnelles $K(x)$ ou encore les anneaux de séries formelles $A[[x]]$ pour toute une gamme d'anneaux de base.

Polynômes et expressions formelles.

Poly

Un peu de Python

```
2*3; 3*4; 4*5           # un commentaire, 3 résultats
6
12
20
a, b = 10, 20 # (a, b) = (10, 20) et [10, 20] possibles
a, b = b, a
```

Typage certain, par le contexte.

Un peu de Python : algorithmique

```
for k in [1..5]:  
    ...print 7*k # bloc qui contient une seule instruction  
7  
14  
21  
28  
35
```

Les blocs sont matérialisés par des indentations.

Un peu de Python : algorithmique

```
S = 0 ; k = 0           #           La somme S commence à 0
while e^k <= 10^6:     #           e^13 <= 10^6 < e^14
...   S = S + k^2      #           ajout des carrés k^2
...   k = k + 1
...                                     # une ligne vide est nécessaire
```

Un peu de Python : algorithmique

```
u = 6 ; n = 0
while u != 1:  # test "différent de" <> aussi possible
...     if u % 2 == 0:# l'opérateur % est le reste euclidien
...         u = u/2
...     else:
...         u = 3*u+1
...     n = n+1
...
```

Un peu de Python : algorithmique, fonctions

```
def fct2 (x, y):  
...     return x^2 + y^2  
...
```

Les procédures sont génériques.

Un peu de Python : algorithmique, fonctions

```
def fact2 (n):  
...     if n == 0: return 1  
...     else: return n*fact2(n-1)  
...
```

Un peu de Python : algorithmique, listes

```
L=[11,12,13]
len(L)
L+[1,2,3]
2*L
U=[1..4]
LL=map(cos,L)
map(float,LL)
map (lambda t: cos(t), [0, pi/6, pi/4, pi/3, pi/2])
LL.reverse()
U.reverse()
U.sort()
L = [[2, 2, 5], [2, 3, 4], [3, 2, 4], [3, 3, 3]]
```

lambda fonction.

Un peu de Python : algorithmique, dictionnaire

```
D={}; D['un']=1; D['deux']=2; D['trois']=3; D['dix']=10  
D['deux'] + D['trois']
```


Un peu de numérique

hilbert.py

Python, encore...

Python s'interface facilement avec du C, du C++ ou du Fortran.

Langage orienté objet sophistiqué :

- classes,
- héritage multiple,
- polymorphisme.

Surcharge des opérateurs usuels : (+, -, *, /, \).

Généricité.

[classes.py](#)

Un peu de Python : introspection

```
print isinstance(a,B)  
  
    print isinstance(a,A)
```

B dérive de A.

Un peu de Python : polymorphisme

```
class A:
    def ecrire_message(self):
        print "je suis un A."

class B(A):
    def ecrire_message(self):
        print "je suis un B."
```

B dérive de A.

Un exercice numérique : calcul des coefficients d'une méthode de Runge–Kutta implicite

rk.pdf