Le projet Sage

Thierry Dumont
ICJ

5 Octobre 2011

The Sage Project :
Unifying Free Mathematical Software to Create a Viable Alternative to Magma, Maple, Mathematica and MATLAB.

The Sage Project :
Unifying Free Mathematical Software to Create a Viable Alternative to Magma, Maple, Mathematica and MATLAB.

Liens :

- site : `http://www.sagemath.org/`

The Sage Project :
Unifying Free Mathematical Software to Create a Viable Alternative to Magma, Maple, Mathematica and MATLAB.

Liens :

- site : `http://www.sagemath.org/`
- livre en français :

Calcul mathématique avec Sage

Casamayou Alexandre, Connan Guillaume, Dumont Thierry, Fousse Laurent, Maltey Francois, Meulien Matthias, Mezzarobba Marc, Pernet Clément, Thiéry Nicolas, Zimmermann Paul.

`http://sagebook.gforge.inria.fr/`

Motivations

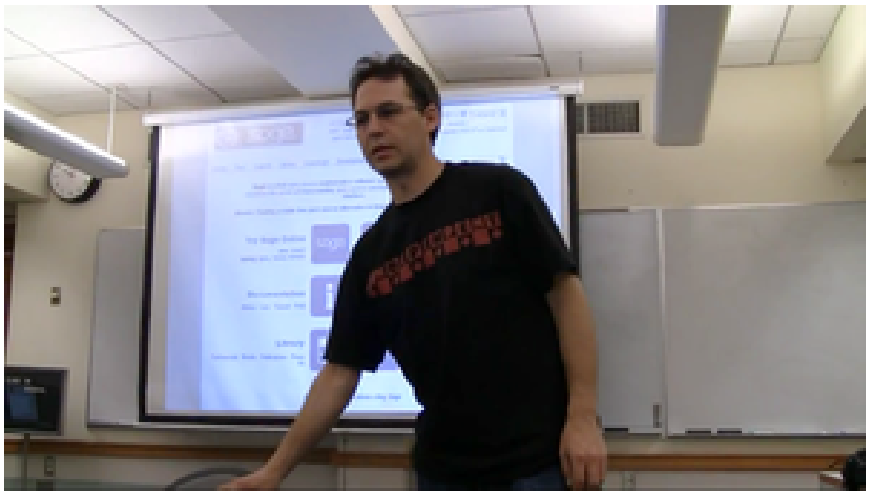
- Éthique : peut-on faire de la science avec des boites noires ?

Motivations

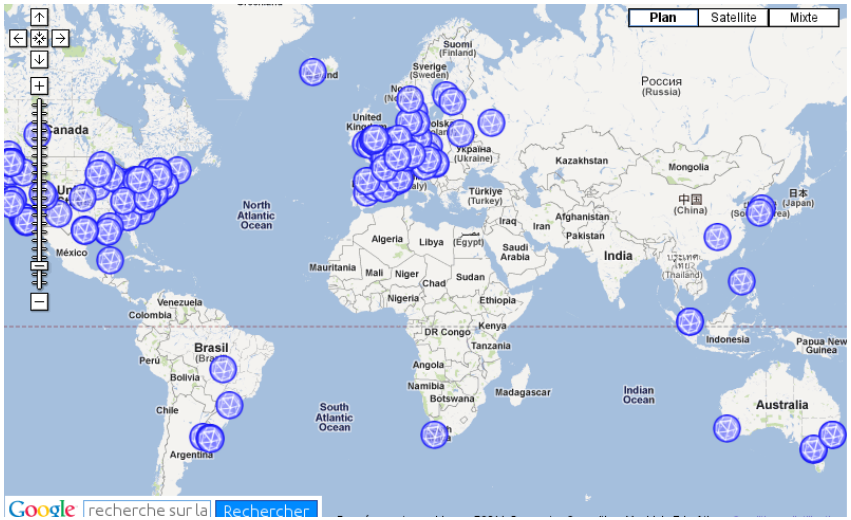
- Éthique : peut-on faire de la science avec des boites noires ?
- Économique : prix des licences.

Motivations

- Éthique : peut-on faire de la science avec des boites noires ?
- Économique : prix des licences.
- Pérennité des développements.



William Stein, Univ. Washington



Carte des développeurs

(Contre-)Exemple

MuPAD-combinat : F. Hivert et N. Thiéry (2000− >) meilleur système de calcul en combinatoire algébrique.

(Contre-)Exemple

MuPAD-combinat : F. Hivert et N. Thiéry ($2000- >$) meilleur système de calcul en combinatoire algébrique.

Écoutons W. Stein :

They [MuPAD] also have promised to release the code source of the library under a well known open-source license, some day." In 2008, MuPAD was in- stead purchased by MathWorks (makers of MATLAB), so MuPAD is no longer available as a separate product, and will probably never be open source. Instead it now suddenly costs 3000 dollars (commercial) or 700 dollars(academic).

(Contre-)Exemple

MuPAD-combinat : F. Hivert et N. Thiéry (2000− >) meilleur système de calcul en combinatoire algébrique.

Écoutons W. Stein :

They [MuPAD] also have promised to release the code source of the library under a well known open-source license, some day." In 2008, MuPAD was in- stead purchased by MathWorks (makers of MATLAB), so MuPAD is no longer available as a separate product, and will probably never be open source. Instead it now suddenly costs 3000 dollars (commercial) or 700 dollars(academic).

Le groupe MuPAD-combinat a migré vers Sage.

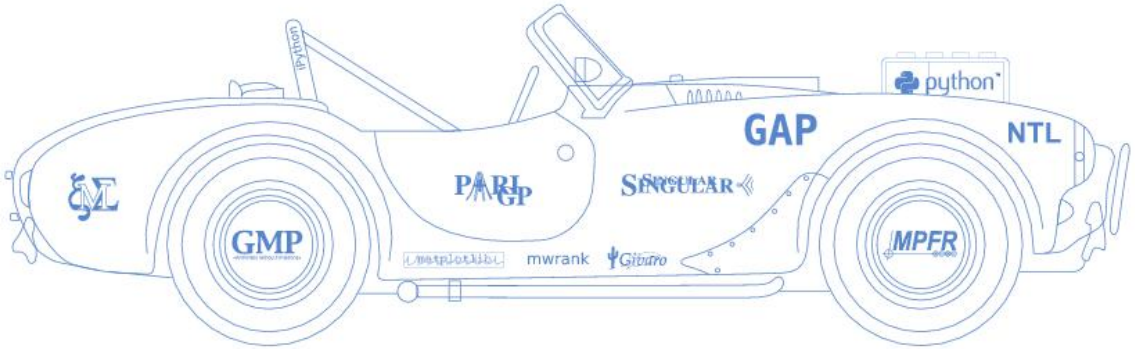
It has been such a relief during the last two years not to have this Damocles sword on our head !
– Nicolas Thiéry

Que faire ?

- logiciel libre.

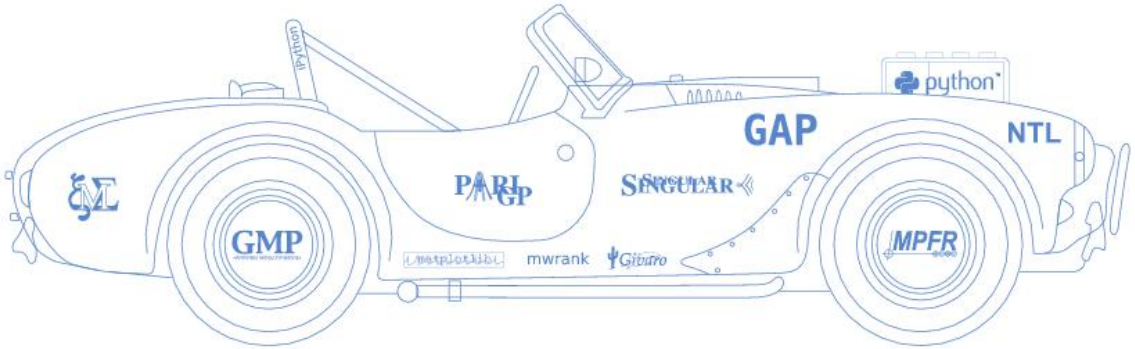
Que faire ?

- logiciel libre.
- ne pas réinventer la roue, mais construire la voiture :



Que faire ?

- logiciel libre.
- ne pas réinventer la roue, mais construire la voiture :



AC Cobra 1960-1970.

Que faire ?

- faire collaborer de manière transparente des logiciels libres existants.
  Selon le problème à résoudre, on doit automatiquement être aiguillé vers le bon logiciel : l'utilisateur ne doit rien voir.
- faire parler le même langage à tous les logiciels fédérés.
- pas de nouveau langage ! pas de langage–Sage !

Que faire ?

- faire collaborer de manière transparente des logiciels libres existants.
  Selon le problème à résoudre, on doit automatiquement être aiguillé vers le bon logiciel : l'utilisateur ne doit rien voir.
- faire parler le même langage à tous les logiciels fédérés.
- pas de nouveau langage ! pas de langage–Sage !

et le langage est :

Que faire ?

- faire collaborer de manière transparente des logiciels libres existants.
  Selon le problème à résoudre, on doit automatiquement être aiguillé vers le bon logiciel : l'utilisateur ne doit rien voir.
- faire parler le même langage à tous les logiciels fédérés.
- pas de nouveau langage ! pas de langage–Sage !

et le langage est : **Python** .

Packages..

Algebra	GAP, Maxima, Singular
Algebraic geometry	Singular
Arbitrary precision arithmetic	MPIR, MPFR, MPFI, NTL, mpmath
Arithmetic geometry	PARI/GP, NTL, mwrank, ecm
Calculus	Maxima, SymPy, GiNaC
Combinatorics	Symmetrica, Sage-Combinat
Linear algebra	ATLAS, BLAS, LAPACK, NumPy, LinBox, IML, GSL
Graph theory	NetworkX
Group theory	GAP
Numerical computation	GSL, SciPy, NumPy, ATLAS
Number theory	PARI/GP, FLINT, NTL
Statistical computing	R, SciPy

Packages..

Command-line shell	IPython
Database	ZODB, Python pickles, SQLite
Graphical interface	Sage Notebook, jsmath
Graphics	Matplotlib, Tachyon3d, GD, Jmol
Interactive programming language	Python
Networking	Twisted

Mercurial.

Sage est :

1. un distribution auto contenue (spkg).
   faibles dépendances.
   Python est intégré (c.a.d. : le source est présent dans la distribution), Lisp aussi, ainsi que lapack, atlas etc...=> compilation particulièrement simple, mais longue !

Sage est :

1. un distribution auto contenue (spkg).
   faibles dépendances.
   Python est intégré (c.a.d. : le source est présent dans la distribution), Lisp aussi, ainsi que lapack, atlas etc...=> compilation particulièrement simple, mais longue !
2. une collection d'interfaces unifiées.
3. une bibliothèque (au dessus des autres).

Installation

Tourne sous Linux, OsX, Solaris etc, mais **pas directement** sous Wxx. Distribution binaire ou source (2,5 GB nécessaires).

Installation

Tourne sous Linux, OsX, Solaris etc, mais **pas directement** sous Wxx. Distribution binaire ou source (2,5 GB nécessaires).

Packages nécessaires pour compiler le source :

gcc	(Version 4.0.1 or later)
g++	(Version 4.0.1 or later)
gfortran	(Version 4.0.1 or later)
make	(For Solaris or OpenSolaris,GNU make)
perl	(Version 5.8.0 or later)
ranlib	
tar	(For Solaris or OpenSolaris, GNU tar)
ssh-keygen	(Needed to run the notebook in secure mode)
latex	(Highly recommended, though not strictly required)
ImageMagick	recommended
ffmpeg	recommended
dvipng	recommended

Interaction :

- ligne de commande,
- notebook :
  Permet :
  1. une interaction locale,
  2. de réaliser des serveurs.

  Interaction “moderne” (Ajax, JSMath, Java...).
  Le notebook est autonome (il est son propre serveur web).

On pourrait développer d'autres systèmes d'interaction.

Interaction :

- ligne de commande,
- notebook :
  Permet :
  1. une interaction locale,
  2. de réaliser des serveurs.

  Interaction “moderne” (Ajax, JSMath, Java...).
  Le notebook est autonome (il est son propre serveur web).

On pourrait développer d'autres systèmes d'interaction.

Expérience à Lyon1 : depuis 3 ans, 3 serveurs (2x4 cœurs, 32 gb.) ; jusqu'à 200 étudiants connectés simultanément.

Python : avantages... et inconvénients

Avantages :

- interprété,
- typé
- orienté objet : classes, héritage, polymorphisme...
- surcharge des opérateurs
- généricité
- introspection

Python : avantages... et inconvénients

Avantages :

- interprété,
- typé
- orienté objet : classes, héritage, polymorphisme...
- surcharge des opérateurs
- généricité
- introspection
- interfaçage facile
- documentation incorporée.

L'énorme quantité de bibliothèques interfacées avec Python permet de mélanger calcul, réseau, bases de données, traitement de documents etc.

Python : avantages... et inconvénients

Avantages :

- interprété,
- typé
- orienté objet : classes, héritage, polymorphisme...
- surcharge des opérateurs
- généricité
- introspection
- interfaçage facile
- documentation incorporée.

L'énorme quantité de bibliothèques interfacées avec Python permet de mélanger calcul, réseau, bases de données, traitement de documents etc.

Inconvénients

- interprété, donc lent... mais il y a Cython .

Classes

Représenter des objets : exemple type des matrices :

- un tableau
- le nombre de lignes
- le nombre de colonnes

Mais les matrices peuvent être pleines ou creuses par exemple symétriques ou non symétriques...

Cacher la structure interne à l'utilisateur (le programmeur), qui n'accèdent à l'objet que par des « méthodes »

```
class Matrix:
    "Documentation (docstring)"
    def __init__(self,n,m):
        self.nlig=n
        self.ncol=m

    def nb_rows(self):
      """
        return number of rows
      """
       return self.m

    def nb_col(self):
       return self.n
```

```
def transpose(self):
    """
    Returns the transpose of self, without changing sel
    """
    --instructions ici---
```

On aura aussi, quand c'est possible :

```
def __latex__(self):
  ---instructions---
  ......
  ......
  return matrice_ecrite_en_latex
```

On aura alors :

```
Q=Matrix(10,3)
print Q.nb_col()
k=Q.nb_rows()
Q.transpose()
```

et puis :

```
s=Q.__latex__()
```

... et d'autres méthodes : exemple : sérialisation.

En résumé,toutes les manipulations se font avec les méthodes de la classe.

On gagne en abstraction.

Héritage et polymorphisme

Raffinement de concepts :
Matrices $->$ Matrices symétriques.
La classe des matrices symétriques hérite de celle des matrices.

```
class SymetricMatrix(Matrix):
```

Comme le stockage est différent, on va sûrement redéfinir `__init__`... mais aussi `transpose(self)`... etc.

```
m1=Matrix(4,4)
m2=SymetricMatrix(4,4)
s=[m1,m2]
for i in range(len(s)):
    s[i].transpose()
```

polymorphisme.

Surcharge des opérateurs

Dans le cas déjà vu des matrices, on aimerait pouvoir faire :

```
A=Matrix[3,3]
A[1,2]=4
```

Surcharge des opérateurs

Dans le cas déjà vu des matrices, on aimerait pouvoir faire :
A=Matrix[3,3]
A[1,2]=4

Tous les opérateurs usuels peuvent être (re)définis
$(=, ==, <, <=, *, +, -, /, [\ ]$ etc.)
on peut donc par exemple définir la multiplication $*$ entre matrices, etc...

Surcharge des opérateurs

Dans le cas déjà vu des matrices, on aimerait pouvoir faire :
A=Matrix[3,3]
A[1,2]=4

Tous les opérateurs usuels peuvent être (re)définis
$(=, ==, <, <=, *, +, -, /, [\ ]$ etc.)
on peut donc par exemple définir la multiplication $*$ entre matrices, etc...

Là, on commence à se dire qu'on va pouvoir faire des mathématiques .

Surcharge des opérateurs

Dans le cas déjà vu des matrices, on aimerait pouvoir faire :
A=Matrix[3,3]
A[1,2]=4

Tous les opérateurs usuels peuvent être (re)définis
$(=, ==, <, <=, *, +, -, /, [\ ]$ etc.)
on peut donc par exemple définir la multiplication $*$ entre matrices, etc...

Là, on commence à se dire qu'on va pouvoir faire des mathématiques .

... et bien justement :

Un peu de maths...

1. Ensemble $\mathbb{Z}$ des entiers relatifs : $\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots$
   opérations $+, x$, mais on ne sait pas toujours diviser
   $(x * 5 = 1\,?)$:
   $\mathbb{Z}$ est un anneau .

Un peu de maths...

1. Ensemble $\mathbb{Z}$ des entiers relatifs : $\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots$
   opérations $+, x$, mais on ne sait pas toujours diviser
   $(x * 5 = 1\,?)$:
   $\mathbb{Z}$ est un anneau .
2. Ensemble $\mathbb{Q}$ des rationnels.
   si $a \neq 0$ alors $x * a = 1$ a une solution (Ex : $x * \frac{2}{3} = 1$).
   $\mathbb{Q}$ est un corps (de même que $\mathbb{R}$ et $\mathbb{C}$).

Un peu de maths...

1. Ensemble $\mathbb{Z}$ des entiers relatifs : $\ldots, -3, -2, -1, 0, 1, 2, 3, \ldots$
   opérations $+, x$, mais on ne sait pas toujours diviser
   $(x * 5 = 1\,?)$:
   $\mathbb{Z}$ est un anneau .
2. Ensemble $\mathbb{Q}$ des rationnels.
   si $a \neq 0$ alors $x * a = 1$ a une solution (Ex : $x * \frac{2}{3} = 1$).
   $\mathbb{Q}$ est un corps (de même que $\mathbb{R}$ et $\mathbb{C}$).

Autres exemples rigolos :

Exemple : $\mathbb{Z}/3\mathbb{Z}$:

```
Z3=IntegerModRing(3)
x=Z3(2)
y=Z3(1)
```

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

x	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

On peut toujours résoudre $a * x = 1$ (si $a \neq 0$).
On a donc bien un corps .

$\mathbb{Z}/4\mathbb{Z}$:

```
Z4=IntegerModRing(4)
x=Z4(2)
y=Z4(3)
```

+	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

x	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

Généricité

Exemple :

```
def prod(A,B):
    return A*B
```

fonctionne pour tout couple d'objets `A, B` tel que le produit `*` est défini ; par exemple :

- ensemble de nombres
  (`QQ, RDF, ZZ, IntegerModRing(5),...`)
- matrices,
- matrice x vecteur,
- polynômes.
- ...
- toute classe dans laquelle l'opérateur `*` est défini .

Généricité

Mais ceci peut poser problème :

```
def div(A,B):
    return A/B
```

C'est à dire : trouver `X` tel que `B*X=A`.

Selon les ensembles auxquels A et B appartiennent, le résultat peut :

1. exister :
   Exemple : A et B $\in \mathbb{Q}$ (et $B \neq 0$).

Selon les ensembles auxquels A et B appartiennent, le résultat peut :

1. exister :
   Exemple : A et B $\in \mathbb{Q}$ (et $B \neq 0$).
2. ne pas exister, mais exister... ailleurs :
   - A, B entiers : il faut une coercion, car le résultat n'est pas entier : $\mathbb{N}, \mathbb{N} \to \mathbb{Q}$,

Selon les ensembles auxquels A et B appartiennent, le résultat peut :

1. exister :
   Exemple : A et B $\in \mathbb{Q}$ (et $B \neq 0$).
2. ne pas exister, mais exister... ailleurs :
   - A, B entiers : il faut une coercion, car le résultat n'est pas entier : $\mathbb{N}, \mathbb{N} \to \mathbb{Q}$,
   - A, B polynômes : $A = x^4 + 3x^2 + 1$ et $B = x^8 + 2$

Selon les ensembles auxquels A et B appartiennent, le résultat peut :

1. exister :
   Exemple : A et B $\in \mathbb{Q}$ (et $B \neq 0$).
2. ne pas exister, mais exister... ailleurs :
   - A, B entiers : il faut une coercion, car le résultat n'est pas entier : $\mathbb{N}, \mathbb{N} \to \mathbb{Q}$,
   - A, B polynômes : $A = x^4 + 3x^2 + 1$ et $B = x^8 + 2$

   $$A/B = \frac{x^4 + 3x^2 + 1}{x^8 + 2}$$

   est une fraction rationnelle.

Selon les ensembles auxquels A et B appartiennent, le résultat peut :

1. exister :
   Exemple : A et B $\in \mathbb{Q}$ (et $B \neq 0$).
2. ne pas exister, mais exister... ailleurs :
   - A, B entiers : il faut une coercion, car le résultat n'est pas entier : $\mathbb{N}, \mathbb{N} \to \mathbb{Q}$,
   - A, B polynômes : $A = x^4 + 3x^2 + 1$ et $B = x^8 + 2$

   $$A/B = \frac{x^4 + 3x^2 + 1}{x^8 + 2}$$

   est une fraction rationnelle.
3. ne pas exister.

```
Z=IntegerModRing(4)
a=Z(3)
b=Z(2)
a/b
ZeroDivisionError: Inverse does not exist.
```

Donc les opérateurs doivent implanter les coercions nécessaires (et raisonnables) et/ou déclencher des “traps” quand c’est nécessaire.

Du générique plus sérieux

Résoudre un système linéaire :

$$AX = B$$

par la méthode de Gauss.

Méthode de Gauss

$$\begin{array}{rrrcr} x & +10y & -3z & = & 5 \\ 2x & -y & +2z & = & 2 \\ -x & +y & +z & = & -3 \end{array}$$

Méthode de Gauss

$$\begin{aligned} x &+10y &-3z &= &5 \\ 2x &-y &+2z &= &2 \\ -x &+y &+z &= &-3 \end{aligned}$$

$$\begin{aligned} x &+10y &-3z &= &5 \\ &-21y &+8z &= &-8 \\ &11y &-2z &= &2 \end{aligned}$$

Méthode de Gauss

$$\begin{aligned} x &+10y &-3z &= 5 \\ 2x &-y &+2z &= 2 \\ -x &+y &+z &= -3 \end{aligned}$$

$$\begin{aligned} x &+10y &-3z &= 5 \\ &-21y &+8z &= -8 \\ &11y &-2z &= 2 \end{aligned}$$

$$\begin{aligned} x &+10y &-3z &= 5 \\ &-21y &+8z &= -8 \\ & &46z/21 &= 13/21 \end{aligned}$$

Méthode de Gauss

$$\begin{aligned} x &+10y &-3z &= 5 \\ 2x &-y &+2z &= 2 \\ -x &+y &+z &= -3 \end{aligned}$$

$$\begin{aligned} x &+10y &-3z &= 5 \\ &-21y &+8z &= -8 \\ &11y &-2z &= 2 \end{aligned}$$

$$\begin{aligned} x &+10y &-3z &= 5 \\ &-21y &+8z &= -8 \\ & &46z/21 &= 13/21 \end{aligned}$$

La méthode de Gauss fonctionne, dès que les coefficients de A et B sont dans un corps (ou un anneau intègre , moyennant une modification).

La méthode de Gauss fonctionne, dès que les coefficients de A et B sont dans un corps (ou un anneau intègre , moyennant une modification).
sage : $Y = A \backslash B$

La méthode de Gauss fonctionne, dès que les coefficients de A et B sont dans un corps (ou un anneau intègre , moyennant une modification).

sage : $Y = A \backslash B$

Mais comment s'assurer qu'on est bien dans un corps ?

Python permet l'introspection

Introspection

Exemple :

```
>m=Matrix(10,3)
>isinstance(m,Matrix)
True
```

Introspection

Exemple :

```
>m=Matrix(10,3)
>isinstance(m,Matrix)
True
>isinstance(m,SymetricMatrix)
False
```

Introspection

Exemple :

```
>m=Matrix(10,3)
>isinstance(m,Matrix)
True
>isinstance(m,SymetricMatrix)
False
>m=SymetricMatrix(3,3)
>isinstance(m,SymetricMatrix)
True
```

Introspection

Exemple :

```
>m=Matrix(10,3)
>isinstance(m,Matrix)
True
>isinstance(m,SymetricMatrix)
False
>m=SymetricMatrix(3,3)
>isinstance(m,SymetricMatrix)
True
>isinstance(m,Matrix)
True
```

Introspection

Exemple :

```
>m=Matrix(10,3)
>isinstance(m,Matrix)
True
>isinstance(m,SymetricMatrix)
False
>m=SymetricMatrix(3,3)
>isinstance(m,SymetricMatrix)
True
>isinstance(m,Matrix)
True
```

Application à Sage : les ensembles sont représentés par des classes qui dérivent (éventuellement) de la classe des anneaux , ou des corps .
La classe des corps dérive de celle des anneaux etc...

Exemple :
class Integer(Ring) :
....

montrer les résolutions de systèmes !

Application :

sage : $Y = A \backslash B$

Sage se demande : *scratch*, *scratch*

Application :

sage : $Y = A \backslash B$

Sage se demande : *scratch, scratch*

Est-ce que les coefficients de A et B sont bien dans un corps ? Si oui, je sais faire...

Application :

sage : $Y = A \backslash B$

Sage se demande : *scratch, scratch*

Est-ce que les coefficients de A et B sont bien dans un corps ? Si oui, je sais faire...

Ensuite je vais appeler la *bonne* méthode.

Interfaces entre Python et le reste du monde

1. l'interface du pauvre : pyexpect
   Lance un exécutable, échange avec lui des chaînes de caractères : on communique par l'entrée-sortie standard du logiciel.
   Dans Sage, n'est plus utilisée que pour Maxima.

Interfaces entre Python et le reste du monde

1. l'interface du pauvre : pyexpect
   Lance un exécutable, échange avec lui des chaînes de caractères : on communique par l'entrée-sortie standard du logiciel.
   Dans Sage, n'est plus utilisée que pour Maxima.
2. C/C++ : Swig
3. C++ Boost-Python.

Interfaces entre Python et le reste du monde

1. l'interface du pauvre : pyexpect
   Lance un exécutable, échange avec lui des chaînes de caractères : on communique par l'entrée-sortie standard du logiciel.
   Dans Sage, n'est plus utilisée que pour Maxima.
2. C/C++ : Swig
3. C++ Boost-Python.
4. Cython.

Cython

- Python est trop lent.

Perdre un peu de la versatilité de Python pour avoir un langage compilé.

```
def mysum ( N ):
    s = 0
    for k in xrange (1 , N ): s += k
    return s
```

devient par exemple :

```
def mysum ( N ):
    cdef int k
    cdef long long s=0
    for k in xrange (1 , N ): s += k
    return s
```

Cython

Ce code peut être transcrit en C par Cython, puis compilé... et appelé par Python ! Typiquement 50 à 100 fois plus rapide.

Cython

Ce code peut être transcrit en C par Cython, puis compilé... et appelé par Python ! Typiquement 50 à 100 fois plus rapide.
Conséquence :

- une grande partie de la bibliothèque Sage est en Cython,

Cython

Ce code peut être transcrit en C par Cython, puis compilé... et appelé par Python ! Typiquement 50 à 100 fois plus rapide.
Conséquence :

- une grande partie de la bibliothèque Sage est en Cython,
- Cython fabrique du C, et donc peut servir à interfacer Python avec C.

Cython

Ce code peut être transcrit en C par Cython, puis compilé... et appelé par Python ! Typiquement 50 à 100 fois plus rapide.
Conséquence :

- une grande partie de la bibliothèque Sage est en Cython,
- Cython fabrique du C, et donc peut servir à interfacer Python avec C.

Développez en Python, puis passez en Cython ce qui est “number crunching”.
Toujours la même idée : Python est lent mais agile, on le réserve pour les boucles les plus externes.

Retour sur le Notebook et l'avenir de Sage

Avantages :

- Serveur autonome (pas besoin d'Apache) ;
- tout est en Python => facile à patcher ; par exemple pour ajouter une identification sur un annuaire ldap (installer Python-Ldap dans Sage et écrire un petit script).
- Interaction relativement agréable.

Retour sur le Notebook et l'avenir de Sage

Avantages :

- Serveur autonome (pas besoin d'Apache) ;
- tout est en Python => facile à patcher ; par exemple pour ajouter une identification sur un annuaire ldap (installer Python-Ldap dans Sage et écrire un petit script).
- Interaction relativement agréable.

Inconvénients

- Serveur autonome : performances ?
- difficilement multi-machines.
- sécurité ?
  https, ok, mais bon...
  Pas de vrais comptes utilisateurs (stockage dans des répertoires appartenant au même utilisateur Unix).

Développements en cours (version 5)

- Identification : openID.
- couche abstraite pour le stockage.
  Sera branchée sur une base de données autonome *NoSql* :

  Exemple : MongoDB, CouchDB

  MongoDB : JSon
  CouchDB : écrite en Erlang.