Introduction à

Marc Mezzarobba

Groupe de travail Pequan, 16 octobre 2014

Qu'est-ce que Sage ?

« *Une alternative libre viable à Magma, Maple, Mathematica et Matlab* »

« *Construire la voiture au lieu de réinventer la roue* »

Essayer Sage

http://sagemath.org/

GNU GPL

http://sagenb.org/

http://cloud.sagemath.com/

Qu'est-ce que Sage ?

1. Une distribution
2. Une bibliothèque Python
3. Un système interactif

Une distribution

```
$ wget http://mirror/sagemath/src/sage-6.3.tar.gz
  && tar xfz sage-6.3.tar.gz && cd sage-6.3 && make
$ ./sage
```

ATLAS • *boehm_gc* • cddlib • cephes • cliquer
cvxopt • Cython • *ECL* • eclib • ecm • *f2c* • fplll
FLINT • GAP • gfan • Givaro • GLPK • GSL
IML • IP[y]: ipython • LAPACK • lcalc
LinBox • matplotlib • Maxima • M4RI
MPC • MPFI • MPFR • mpmath • networkx • NTL
NumPy • PALP • PARI GP • PolyBoRi • PPL • Pynac
python • R • SciPy • SINGULAR • symmetrica
sympow • tachyon • zn_poly • + *d'autres...*

+ ∼ 60 paquets optionnels   + ∼ 60 paquets expérimentaux

Une bibliothèque Python

```
sage.algebras
sage.calculus
sage.categories
sage.coding
sage.combinat
sage.crypto
sage.databases
sage.finance
sage.functions
sage.geometry
sage.graphs
sage.groups
sage.homology
sage.interfaces
sage.lfunctions
sage.logic
sage.matrix
sage.modular
sage.modules
sage.monoids
sage.numerical
sage.parallel
sage.plot
sage.rings
sage.sat
sage.schemes
sage.sets
sage.stats
sage.symbolic
...
```

- S'appuie sur les logiciels tiers embarqués
- $\simeq$ 600 000 lignes de code spécifique (hors doc + tests)

Une bibliothèque Python

```
sage.algebras
sage.calculus
sage.categories
sage.coding
sage.combinat
sage.crypto
sage.databases
sage.finance
sage.functions
sage.geometry
sage.graphs
sage.groups
sage.homology
sage.interfaces
sage.lfunctions
sage.logic
sage.matrix
sage.modular
sage.modules
sage.monoids
sage.numerical
sage.parallel
sage.plot
sage.rings
sage.sat
sage.schemes
sage.sets
sage.stats
sage.symbolic
...
```

- S'appuie sur les logiciels tiers embarqués
- $\simeq$ 600 000 lignes de code spécifique (hors doc + tests)

Une bibliothèque Python

`sage.algebras`
`sage.calculus`
`sage.categories`
`sage.coding`
`sage.combinat`
`sage.crypto`
`sage.databases`
`sage.finance`
`sage.functions`
`sage.geometry`
`sage.graphs`
`sage.groups`
`sage.homology`
`sage.interfaces`
`sage.lfunctions`
`sage.logic`
`sage.matrix`
`sage.modular`
`sage.modules`
`sage.monoids`
`sage.numerical`
`sage.parallel`
`sage.plot`
`sage.rings`
`sage.sat`
`sage.schemes`
`sage.sets`
`sage.stats`
`sage.symbolic`
`...`

- S'appuie sur les logiciels tiers embarqués
- $\simeq$ 600 000 lignes de code spécifique (hors doc + tests)

Une bibliothèque Python

- sage.algebras
- sage.calculus
- sage.categories
- sage.coding
- sage.combinat
- sage.crypto
- sage.databases
- sage.finance
- sage.functions
- sage.geometry
- sage.graphs
- sage.groups
- sage.homology
- sage.interfaces
- sage.lfunctions
- sage.logic
- sage.matrix
- sage.modular
- sage.modules
- sage.monoids
- sage.numerical
- sage.parallel
- sage.plot
- sage.rings
- sage.sat
- sage.schemes
- sage.sets
- sage.stats
- sage.symbolic
- ...

- S'appuie sur les logiciels tiers embarqués
- $\simeq$ 600 000 lignes de code spécifique (hors doc + tests)

Une bibliothèque Python

```
sage.algebras
sage.calculus
sage.categories
sage.coding
sage.combinat
sage.crypto
sage.databases
sage.finance
sage.functions
sage.geometry
sage.graphs
sage.groups
sage.homology
sage.interfaces
sage.lfunctions
sage.logic
sage.matrix
sage.modular
sage.modules
sage.monoids
sage.numerical
sage.parallel
sage.plot
sage.rings
sage.sat
sage.schemes
sage.sets
sage.stats
sage.symbolic
...
```

- S'appuie sur les logiciels tiers embarqués
- $\simeq$ 600 000 lignes de code spécifique (hors doc + tests)

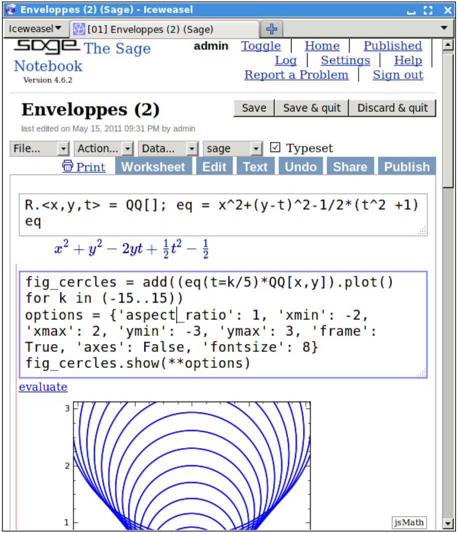
Une bibliothèque Python

```
sage.algebras
sage.calculus
sage.categories
sage.coding
sage.combinat
sage.crypto
sage.databases
sage.finance
sage.functions
sage.geometry
sage.graphs
sage.groups
sage.homology
sage.interfaces
sage.lfunctions
sage.logic
sage.matrix
sage.modular
sage.modules
sage.monoids
sage.numerical
sage.parallel
sage.plot
sage.rings
sage.sat
sage.schemes
sage.sets
sage.stats
sage.symbolic
...
```

- S'appuie sur les logiciels tiers embarqués
- $\simeq$ 600 000 lignes de code spécifique (hors doc + tests)

Un système interactif

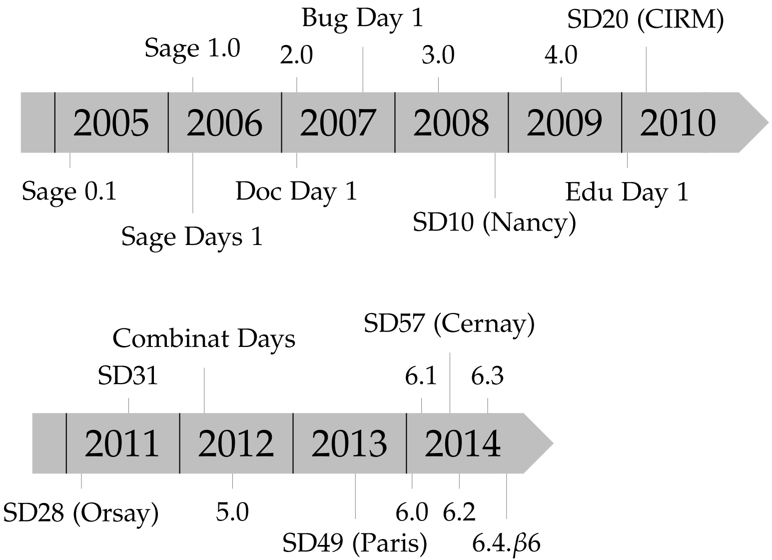
```
~$ sage
----------------------------------------------------------------------
| Sage Version 4.6.2, Release Date: 2011-02-25                       |
| Type notebook() for the GUI, and license() for information.        |
----------------------------------------------------------------------
sage: taylor(exp(x), x, 0, 5)
1/120*x^5 + 1/24*x^4 + 1/6*x^3 + 1/2*x^2 + x + 1
sage:
sage: MatrixSpace(RR,5,3).random_element()
[-0.570390764900653  0.521446993576251 -0.950894560265950]
[-0.942431942330060  0.254122819002693  0.916721924359961]
[-0.195702504102615 -0.350489870318781 -0.214359534055980]
[ 0.487076746020482  0.461116221981387 -0.665179594662514]
[ 0.180194930460366  0.616390883848273 -0.389309976296204]
sage:
sage: import urllib2
sage: f = urllib2.urlopen("http://sagemath.org/")
sage: f.read(121)
'<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"\n"http
://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">'
sage:
```



Qu'est-ce que Sage ?

1. Une distribution
2. Une bibliothèque
3. Un système interactif
4. Une communauté

Histoire


Bug Day 1
SD20 (CIRM)
Sage 1.0
2.0
3.0
4.0
2005
2006
2007
2008
2009
2010
Sage 0.1
Doc Day 1
Edu Day 1
SD10 (Nancy)
Sage Days 1
SD57 (Cernay)
Combinat Days
SD31
6.1
6.3
2011
2012
2013
2014
SD28 (Orsay)
5.0
6.0
6.2
SD49 (Paris)
6.4.β6

Développement et communauté

Langages principaux : Python + Cython

Développeurs faciles à trouver, bibliothèques riches, glu facile, pas trop inefficace

Outils : git + trac, listes de diffusion, wiki...

Sage Days, financement NSF et autres

Bugs nombreux mais code facile d'accès

Sage sera ce que vous en ferez !

Sage comme calculatrice

```
>>> 1+1

    2

>>> factorial(50)

    30414093201713378043612608166064768844377641568960512000000000000

>>> s = add(1/n for n in (1..30))

>>> s

    9304682830147/2329089562800

>>> s.n()

    3.99498713092039

>>>
```

Python

```
>>> a = -2/3

>>> type(a)

    <type 'sage.rings.rational.Rational'>

>>> dir(a)[140:160]

    ['_singular_',
    '_singular_init_',
    '_sub_',
    '_sympy_',
    '_test_category',
    '_test_eq',
    '_test_nonzero_equal',
    '_test_not_implemented_methods',
    '_test_pickling',
    '_tester',
    'abs',
    'absolute_norm',
    'additive_order',
    'base_extend',
    'base_ring',
    'cartesian_product',
    'category',
    'ceil',
    'charpoly',
    'conjugate']

>>> a.abs(), abs(a)

    (2/3, 2/3)

>>> def myfact(n):
        res = 1
        for k in srange(1, n+1):
            res = res*k
        return res

>>> myfact(50)

    30414093201713378043612608166064768844377641568960512000000000000

>>>
```

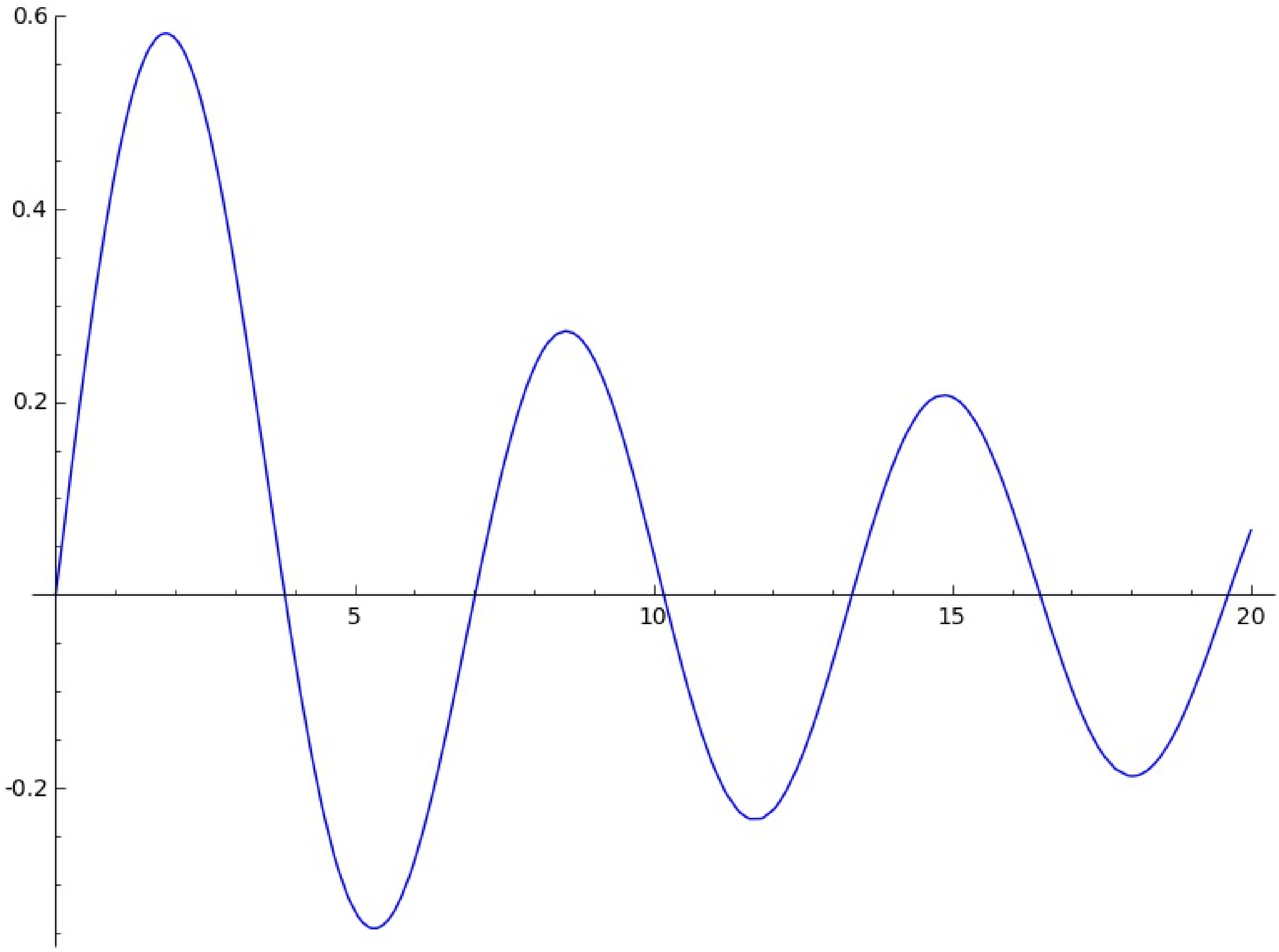
Calcul symbolique à la Maple

```
>>> x, y = var('x', 'y')
    u = cos(x)*sin(y)
    u

    cos(x)*sin(y)

>>> diff(u, x)

    -sin(x)*sin(y)
```

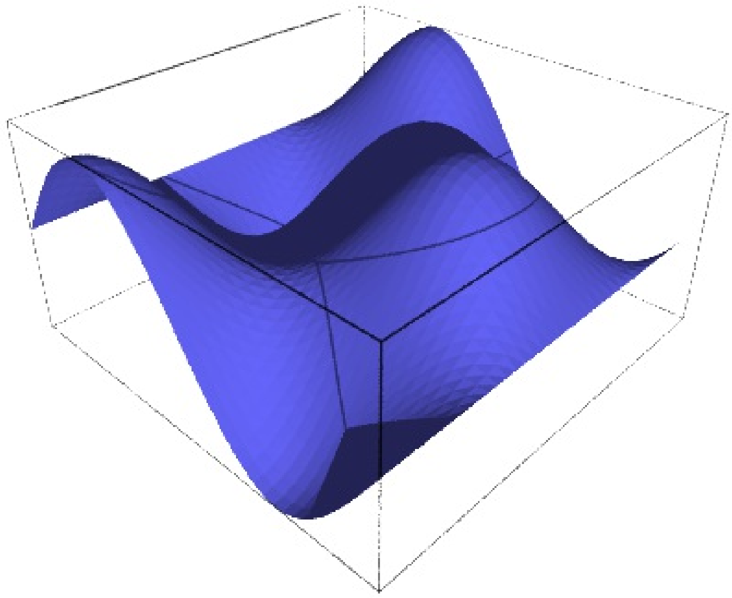
```
>>>   u.derivative(y)

      cos(x)*cos(y)

>>>   u.series(x, order=5)

      (sin(y)) + (-1/2*sin(y))*x^2 + (1/24*sin(y))*x^4 + Order(x^5)

>>>   u(x=1)

      cos(1)*sin(y)

>>>   u(x=sqrt(2), y=1).n(prec=1000)

      0.131222094408801673541794663949435572581514799978818130494047391392750052887318298141843711494656
      042885537573273761743793267666056826212117989865256371450531190922976819604345254006924670564411 33
      781847596004378690230103677152626720456712901449325329568055173470137249434273963974796978559495 22
      11331345
>>>
```

Graphiques

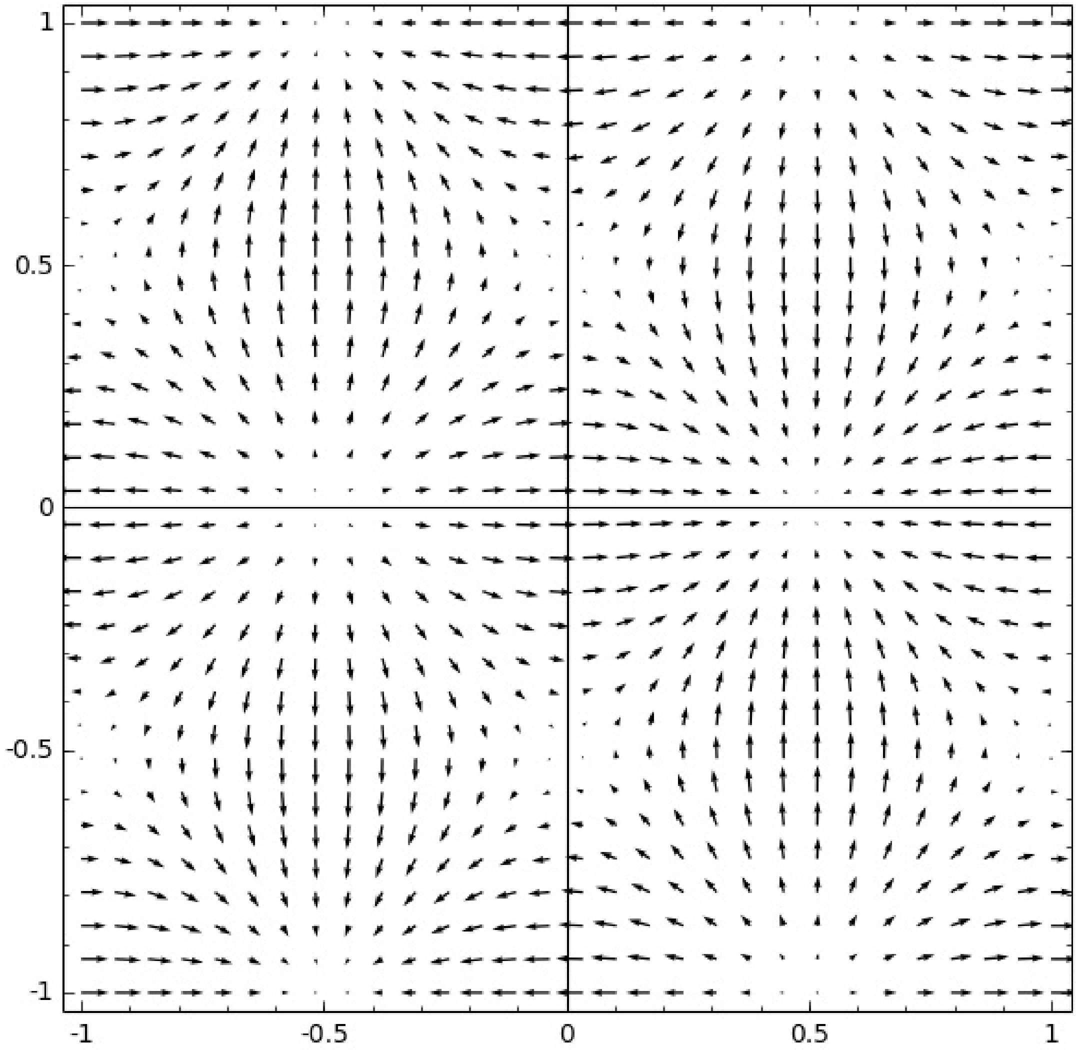
```
>>>   plot(Bessel(1,'J'), 0, 20)
```



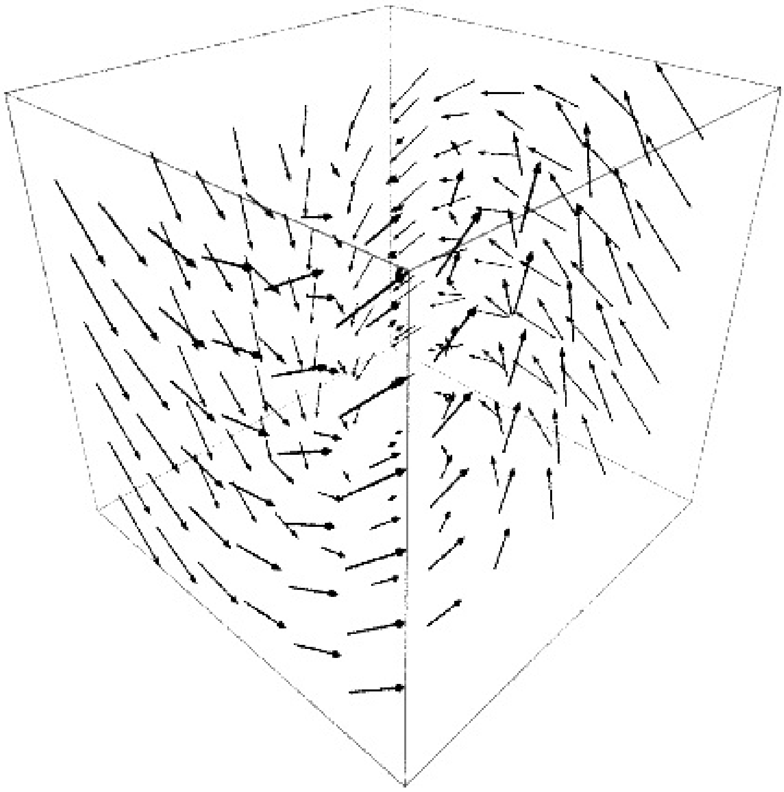
```
>>>   u, v = var('u', 'v')
      f = sin(pi*u) * cos(pi*v)
      plot3d(f, (u, -1, 1), (v, -1, 1))
```



```
>>> plot_vector_field(f.gradient(), (u,-1,1), (v,-1,1), aspect_ratio=1,
    plot_points=30)
```



```
>>>   x,y,z=var('x y z')
      myplot = plot_vector_field3d(
      (-y, 0, x), (x,-2,2), (y,-2,2), (z,-2,2),
      colors='black', plot_points=6)
      show(myplot, viewer="tachyon")
```



```
>>>
```

Éléments et parents

```
>>>   type(1)
```

<type 'sage.rings.integer.Integer'>

```
>>>   parent(1)
```

Integer Ring

```
>>>   ZZ
```

Integer Ring

```
>>>   parent(1) is ZZ
```

True

```
>>>   parent(1/1)
```

Rational Field

```
>>>   type(ZZ)
```

<type 'sage.rings.integer_ring.IntegerRing_class'>

```
>>>   type(QQ)
```

<class 'sage.rings.rational_field.RationalField_with_category'>

```
>>>   ZZ.cardinality()
```

+Infinity

```
>>>   MyParent = QQ.cartesian_product(ZZ); MyParent
```

The cartesian product of (Rational Field, Integer Ring)

```
>>> MyParent.is_ring()
True
>>> MyParent.an_element()
(1/2, 1)
>>> QQ.category()
Category of quotient fields
>>>
```

Quelques parents

```
>>> Integers()
Integer Ring
>>> Rationals()
Rational Field
>>> R = IntegerModRing(10); R
Ring of integers modulo 10
>>> R(8)^2
4
>>> Reals()
Real Field with 53 bits of precision
>>> Complexes()
Complex Field with 53 bits of precision
>>> MatrixSpace(RDF, 2, 3)
Full MatrixSpace of 2 by 3 dense matrices over Real Double Field
>>> PolynomialRing(QQ, 'x')
Univariate Polynomial Ring in x over Rational Field
>>> PolynomialRing(QQ, 'x, y, z')
Multivariate Polynomial Ring in x, y, z over Rational Field
>>> MatrixSpace(PolynomialRing(ZZ, 'x'), 2)
Full MatrixSpace of 2 by 2 dense matrices over Univariate Polynomial Ring in x over Integer Ring
>>> _.random_element()
[ -x^2 -x]
[ -6*x^2 - 3*x - 3 -16*x^2 + 2*x + 4]
>>>
```

Conversions

```
>>> RDF
Real Double Field
>>> RDF(42)
42.0
>>> RDF(42).parent()
Real Double Field
>>> ZZ(1.0)
1
```

```
>>> ZZ(1.5)

    ---------------------------------------------------------------------------
    TypeError Traceback (most recent call last)
    <ipython-input-69-562488d24cf3> in <module>()
    ----> 1 ZZ(RealNumber('1.5'))

    /home/marc/co/sage/local/lib/python2.7/site-packages/sage/structure/parent.so in
    sage.structure.parent.Parent.__call__ (build/cythonized/sage/structure/parent.c:9603)()

    /home/marc/co/sage/local/lib/python2.7/site-packages/sage/structure/coerce_maps.so in
    sage.structure.coerce_maps.NamedConvertMap._call_
    (build/cythonized/sage/structure/coerce_maps.c:5577)()

    /home/marc/co/sage/local/lib/python2.7/site-packages/sage/rings/real_mpfr.so in
    sage.rings.real_mpfr.RealNumber._integer_ (build/cythonized/sage/rings/real_mpfr.c:15923)()

    TypeError: Attempt to coerce non-integral RealNumber to Integer

>>>
```

Coercitions

(= conversions canoniques automatiques)

```
>>> a = 42
    a, a.parent()

    (42, Integer Ring)

>>> b = a + 1/2
    b, b.parent()

    (85/2, Rational Field)

>>> c = b + 1/2
    (c, c.parent())

    (43, Rational Field)

>>> d = ZZ(c)
    (d, d.parent())

    (43, Integer Ring)

>>> M = MatrixSpace(ZZ, 3); M

    Full MatrixSpace of 3 by 3 dense matrices over Integer Ring

>>> obj = M.identity_matrix() + 1/2

>>> obj

    [3/2 0 0]
    [ 0 3/2 0]
    [ 0 0 3/2]

>>> obj.parent()

    Full MatrixSpace of 3 by 3 dense matrices over Rational Field

>>>
```

Changement de parent

```
>>> M = MatrixSpace(RDF, 2)

>>> M

    Full MatrixSpace of 2 by 2 dense matrices over Real Double Field

>>> mat = M.random_element(); mat

    [ -0.9291095239848672 0.7144542993819145]
    [-0.4285627437025185 0.6202965684624147]

>>> mat.inverse()

    [-2.2962684025138707 2.6448297735670647]
    [-1.5865043339815423 3.439459366492412]
```

```
>>> MatrixSpace(Reals(100), 2)(mat)
```

```
[-0.92910952398486723602388792642 0.71445429938191451491036332300]
[-0.42856627437025185400898408261 0.62029656846241465473212883808]
```

```
>>> mat.change_ring(Reals(100)).inverse()
```

```
[-2.2962684025138705309483785961 2.6448297735670647696478834303]
[-1.5865043339815423680868343285 3.4394593664924119799809269914]
```

```
>>> mat.change_ring(RealIntervalField(100)).inverse()
```

```
[-2.2962684025138705309483785962? 2.6448297735670647696478834304?]
[-1.5865043339815423680868343285? 3.4394593664924119799809269915?]
```

```
>>> mat.change_ring(QQ).inverse()
```

```
[-1396257666051660686644101151365609/608055079503375731997410887877620
108662309340487574027637228462848/4108480266914700891874397891065]
[ -482341009465798246976695292844493/304027539751687865998705443938 81
282619018721780166732907796570 2/821696053382940178374879578213]
```

```
>>>
```

Flottants, intervalles...

```
>>> RDF
```

```
Real Double Field
```

```
>>> RDF.precision()
```

```
53
```

```
>>> x = RDF(1/3)
```

```
>>> sin(x)
```

```
0.3271946967961522
```

```
>>> x * 2^1024
```

```
+infinity
```

```
>>> RR
```

```
Real Field with 53 bits of precision
```

```
>>> y = RR(1/3)
```

```
>>> y * 2^1024
```

```
5.99231044954105e307
```

```
>>> QQ(y)
```

```
1/3
```

```
>>> y.exact_rational()
```

```
6004799503160661/18014398509481984
```

```
>>> y.sign_mantissa_exponent()
```

```
(1, 6004799503160661, -54)
```

```
>>> sqrt(-x)
```

```
0.5773502691896257*I
```

```
>>> sqrt(-x).parent()
```

```
Complex Double Field
```

```
>>> sqrt(-y).parent()
```

```
Complex Field with 53 bits of precision
```

```
>>> Reals(prec=230, rnd='RNDU')
```

```
Real Field with 230 bits of precision and rounding RNDU
```

```
>>> Reals(prec=230, rnd='RNDZ')

    Real Field with 230 bits of precision and rounding RNDZ

>>> RealIntervalField(prec=100)

    Real Interval Field with 100 bits of precision

>>> ComplexIntervalField(prec=100)

    Complex Interval Field with 100 bits of precision

>>> RIF, CIF

    (Real Interval Field with 53 bits of precision,
    Complex Interval Field with 53 bits of precision)

>>> 0.42*1/3

    0.140000000000000

>>> 0.42000000000000000000000000000000*1/3

    0.14000000000000000000000000000000

>>> type(0.42)

    <type 'sage.rings.real_mpfr.RealLiteral'>

>>> RR(1/3) + RIF(1/3)

    0.6666666666666667?

>>> RLF

    Real Lazy Field

>>> mypi = RLF(pi)
    mypi

    3.141592653589794?

>>> mypi.n(200)

    3.1415926535897932384626433832795028841971693993751058209749

>>> CLF

    Complex Lazy Field

>>>
```