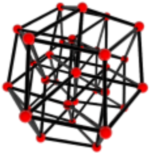
# SAGE: Open Source Mathematics

http://www.sagemath.org
William Stein
Math 581f

October 12, 2007



1. Part 1: Sage
2. Part 2: Number Fields

# Part 1: SAGE

**Abstract:** Explain how using Python and SAGE is very likely to **improve your efficiency and ability** to do mathematical research that involves computation. (This is an unabashed sales pitch – *I really want you to use Python and SAGE.* Also, using computation to aid research is a *major trend* in mathematics research right now.)

**Target Audience:** Mathematical researchers who **demand the best possible tools** for the job (even if they are expensive).

# The Python Programming Language

python™

**Python** is a powerful modern interpreted programming language.

- “Python is fast enough for our site and allows us to **produce maintainable features in record times**, with a minimum of developers,” said Cuong Do, Software Architect, **YouTube.com.**
- “Google has made no secret of the fact they use Python a lot for a number of internal projects. Even knowing that, once **I was an employee, I was amazed at how much Python code there actually is in the Google source code system**.”, said Guido van Rossum, **Google**, creator of Python.
- “Python plays a key role in our production pipeline. Without it a project the size of **Star Wars: Episode II** would have been very difficult to pull off. From crowd rendering to batch processing to compositing, **Python binds all things together**,” said Tommy Burnette, Senior Technical Director, **Industrial Light & Magic.**

# Python is...

python™

- Easy for you to **define your own data types** and methods on it. **bitstreams, ciphers, rings, whatever**
- Very clean language that results in **easy to *read* code**.
- **Easy to learn**:
  - **Free:** Dive into Python `http://www.diveintopython.org/`
  - **Free:** Python Tutorial `http://docs.python.org/tut/`
- A **huge number of libraries**: statistics, networking, databases, bioinformatic, physics, video games, 3d graphics, and serious mathematics (via SAGE)
- Very easy to **use any C/C++ libraries** from Python.
- Excellent support for **string manipulation and bit fiddling**.
- Cython – a **Python compiler** (`http://www.cython.org`).

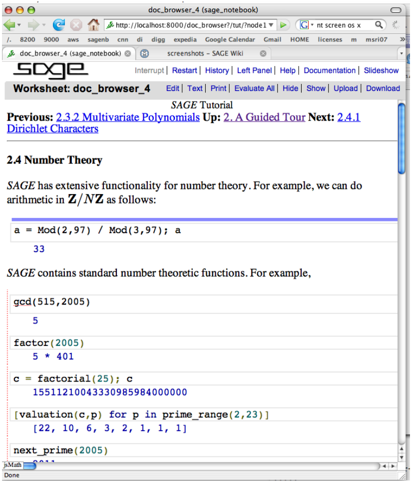
# SAGE Event Timeline

1. **1999–2005:** (**Berkeley, Harvard**) I wrote over 25,000 lines of **Magma** code. **I really like Magma compared to C++!**
2. But the languages of Magma, Mathematica, and Maple are **old-fashioned and painful** compared to Python.
3. And I **need to** be able to **see inside and change anything** in my software in order to be the best in the world at my research.
4. **Magma is frustrating** and is a **terrible longterm investment**.
5. **Feb 2005:** I released **SAGE-0.1** – a Python math library.
6. **Feb 2006: UCSD SAGE Days 1** – SAGE 1.0.
7. **October 2006: U Washington SAGE Days 2** workshop.
8. **March 2007: UCLA SAGE Days 3** workshop.
9. **June 2007: U Washington SAGE Days 4** workshop.
10. **Now:** SAGE-2.8.6; well over **100 contributors to SAGE**.
11. **October 2007: Clay Math Institute SAGE Days 5** workshop.
12. **November 2007: Heilbronn Institute SAGE Days 6**

# Welcome to SAGE!

```
$ sage
--------------------------------------------------------------
| SAGE Version 2.8.6
| Type notebook() for the GUI, and license() for information.
--------------------------------------------------------------
sage: 2 + 2
4
sage: notebook()
***************************************************
* Open your web browser to https://localhost:8000 *
***************************************************
....
```

# The SAGE Notebook




- Connect either to a program running on your computer, or a program running elsewhere.
- Create **embedded graphics**
- **Typeset** mathematical expressions
- Add and delete input
- Start and interrupt **multiple calculations** at once.
- The notebook also works with **Magma**, GAP, PARI, Singular, Macaulay2, Fortran, etc.!

# SAGE Makes Python Usable for Mathematics

SAGE provides **serious computing power** to make Python a truly usable tool for **your research**.
SAGE is **nearly 200,000 lines of new code** that ties together many libraries and programs and provides **much new functionality**:

- **Algebra** and **calculus**: Maxima, Sympy
- **Arbitrary precision arithmetic**: GMP, MPFR, MPFI, NTL, quaddouble, Givaro
- **Algebraic geometry**: Singular, Macaulay2
- **Arithmetic Geometry**: PARI, NTL, mwrank, ecm, FLINTQS
- **Exact linear algebra**: Linbox, IML
- **Graphics** (2d and 3d): MatPlotLib, Tachyon3d, VTK (optional)
- **Group theory**: GAP
- **MATLAB-like functionality – linear algebra, optimization, etc.**: GSL, Scipy, Numpy

Chances are, you can do it using SAGE.

# Use Most Mathematics Software from Within SAGE

SAGE makes it possible for you to **use most mathematics software together**.

- SAGE includes **interfaces to** Magma, Maple, Mathematica, MATLAB, and MuPAD ...
- and also the free programs Axiom, GAP, GP/PARI, Macaulay2, Maxima, Octave, and Singular.
- This makes it **easier to benefit from existing code** you or others have already written.

# Some Shortcomings of SAGE

1. There are currently probably **less than a thousand users** of SAGE (there are millions of Python users).
2. **Not robust enough** – sometimes interrupt doesn't interrupt, etc.
3. SAGE is **sometimes much slower** than Magma or Mathematica (and sometimes faster, to be fair).
4. SAGE is new – there are **too many bugs**.

However, if you think something is bad **you can fix it yourself**. Example, `number_of_partitions`...

# Example: Number of Partitions

```
sage: list(partitions(5))
[(1, 1, 1, 1, 1), (1, 1, 1, 2), (1, 2, 2), (1, 1, 3), (2
sage: number_of_partitions(5)
7
```

1. The beginning of the Mathematica tour has an assertion that: **"Mathematica computes the number of partitions of 1 billion in a few seconds – a frontier number theory calculation".**
2. **SAGE (and Magma!) would take years** to do that, so I posted on sage-devel; **72 posts** among **15 people** followed.
3. Now – thanks to Jon Bobber (U Mich grad student) SAGE is faster at this than any other program in the world on *my laptop*, and promises to be several times faster soon:

```
sage: time len(str(number_of_partitions(10^9)))
CPU times: user 67.21 s, sys: 0.34 s, total: 67.56 s
35219
```

Mathematica 6.0 takes 83s, and 6.1 takes 77s.

# Some Advantages of SAGE

1. SAGE is the **only** serious general purpose mathematics software that uses a **mainstream programing language** (Python).
2. SAGE is the **only** program that allows you to use Maple, Mathematica, Magma, etc., all **together**.
3. SAGE has **more functionality out of the box** than any other open source mathematics software.
4. SAGE has a **Huge**, **active**, and well rounded **developer community**: `sage-devel` mailing list has **180** subscribers, working very hard on everything from highly optimized arithmetic, to high school education, to computing modular forms. Usually about 30 people get patches accepting into SAGE every month.
5. SAGE **development is done in the open**. You can read about why all decision are made, have input into decisions, see a list of every change anybody has made, etc. This is *totally different than the situation with Magma and Mathematica.*

# Download SAGE for Windows, Mac OS X, and Linux

- **Download** SAGE for free at `http://sagemath.org`

- You can **compile SAGE** yourself from source, and **change anything** about SAGE.

# Discussion

1. **Can** SAGE do...?
2. **How** does SAGE do...?
3. Funding:
   - I want the quality of SAGE to be better than Magma, Maple, Matlab, and Mathematica, and this is **impossible** without significant funding. SAGE must be a truly professional tool.
   - SAGE is free open source software, so, like Python, Firefox, Linux, **SAGE needs to be used by major organizations who will pay salaries of developers**.
   - Some components of SAGE (e.g., Scipy) already have commercial support and developers.
   - Thoughts?
4. **Please use SAGE**.

# Part 2: Number Fields

# Elements of Number Fields

1. **All elements** represented as **absolute polys over $\mathbb{Q}$**, though arbitrary towers of relative number fields are supported.
2. (J Mohler) **Uses NTL $\mathbb{Z}[x]$** for all arithmetic (ZZ poly and denominator). Slightly slower than Magma, though not too bad. (timings use V2.13-10 Magma on Intel Core2 Linux)
3. (R Bradshaw) Special **optimized class for quadratic fields**, that is off today, and will be on very soon. Maybe fastest in the world.

# Basic arithmetic...

```
sage: K.<a> = NumberField(x^5 + 17*x^3 + 2*x^2 + 3*x -15
sage: b = (a+1/3)^10; b
-766181/243*a^4 - 12095555/729*a^3 - 2411338/729*a^2 + 1
sage: time for _ in xrange(10^5): c=b*b
CPU times: user 1.56 s, sys: 0.04 s, total: 1.60 s
Wall time: 1.61
```

# Basic arithmetic... (bigger degree)

```
sage: K.<a> = NumberField(x^100 + 17*x^3 + 2*x^2 + 3*x -
sage: b = (a+1/3)^100
sage: time for _ in xrange(10^3): c=b*b
CPU times: user 0.91 s, sys: 0.00 s, total: 0.91 s
Wall time: 0.98
```

# Sage basic arithmetic... (even bigger degree)

```
sage: K.<a> = NumberField(x^500 + 17*x^3 + 2*x^2 + 3*x -
sage: b = (a+1/3)^500
sage: time for _ in xrange(10^2): c=b*b
CPU times: user 2.17 s, sys: 0.01 s, total: 2.18 s
Wall time: 2.18
```

## FLINT will make Sage faster for general number fields arithmetic...

1. **Now in Sage, but only in a testing/development form.**
2. **See Bill Hart's talk from Sage Days 5.**

# Sage Quadratic field arithmetic...

Thanks to Robert Bradshaw, quadratic field arithmetic in Sage is (about to be!) very fast in Sage:

```
sage: K.<a> = QuadraticField(7)
sage: b = (2/3)*a + 5/8
sage: time for _ in xrange(10^5): c=b*b + b*b
CPU times: user 0.31 s, sys: 0.00 s, total: 0.31 s
```

```
MAGMA:
> K<a> := QuadraticField(7);
> b := (2/3)*a + 5/8;
> time for i in [1..10^5] do c := b*b+b*b; end for;
Time: 0.610
```

```
PARI:
? b = Mod((2/3)*a + 5/8, a^2 -7); gettime;
? for(i=0,10^5,c=b*b+b*b); gettime/1000.0
%6 = 0.6730000000000000000000000000000000000
```

# A relative number field arithmetic example...

```
sage: K.<a,b,c> = NumberField([x^2 + 1, x^3 - 2, x^2 - 5
# a^2 == -1, b^3 == 2, c^2 == 5
sage: d = (a+b+c)^5; d
(140*b^2 + (80*c + 10)*b + 40*c + 76)*a + (20*c + 2)*b^2
sage: time for _ in xrange(10^3): c=d*d + d*d
CPU times: user 1.42 s, sys: 0.04 s, total: 1.46 s
sage: d.polynomial()       # absolute poly rep
535290512/29215513833*x^11 + 186767562/9738504611*x^10 -
```

# Class Group Computation

1. Easy to set a global proof True and proof False flag for all number field functions (me and David Roe).
2. The **default is proof True**, **which can easily be thousands of times slower than proof=False!**
3. Sage uses **PARI** for its class group computations.
4. Class group computations in Sage are already usually **faster than in Magma (Bill Hart)**.

# Class Group Examples...

```
sage: K.<a> = NumberField(x^3 + 1838)
sage: C = K.class_group(); C
Class group of order 27 with structure C9 x C3 of Number Field
sage: C.gens()
[Fractional ideal class (50, a - 8) of Number Field in a with
 Fractional ideal class (26, a - 2) of Number Field in a with
sage: I = C.0; J = C.1
sage: I
Fractional ideal class (50, a - 8) of Number Field in a with d
sage: I^9
Trivial principal fractional ideal class of Number Field in a
sage: J^2
Fractional ideal class (6, a^2 - 2*a - 2) of Number Field in a
```

# Class Groups: Sage versus Magma without Proof

Timings from Bill Hart (see sage-devel):
**Proof = False**

```
deg, bits, iter   :   Pari    Magma

2,   10,   10000  :     27s     86s
2,   20,    2000  :     25s    142s
2,   30,     300  :     25s    115s
2,   40,     100  :  50-80s    348s
```

```
                                                  Pari   Magma
x^2 - 678650306441557*x + 232491039415161  :     5.81s   243s
x^2 + 400359911885097*x + 1023437292772615 :     8.33s   ....
x^2 + 788021445418312*x + 62108142321374   :   136.??s   ....
x^2 + 310104001090081*x + 526420096868844  :     2.18s   156s
x^2 + 29148692184930*x + 697845351766239   :     1.45s    63.5s
```

# Class Groups: Sage versus Magma without Proof (degree 3)

Timings from Bill Hart (see sage-devel):
**Proof = False**

```
deg, bits, iter   :   Pari    Magma
3,    5,   5000   :    15s      28s
3,   10,   1000   :    23s      26s
3,   15,    100   : 15-20s   14-39s
```

```
                                                  Pari      Magma
x^3 - 327878*x^2 - 1038886*x + 711300        :  1.12s     10.6s
x^3 + 244636*x^2 + 536860*x - 435475         :  0.545s     5.25s
x^3 - 840752*x^2 + 979860*x - 141846         :  2.38s     26.15s
x^3 - 994421*x^2 - 866767*x - 513979         :  3.53s     35.5s
x^3 - 649099*x^2 + 997454*x + 787504         :  0.332s     3.81s
x^3 + 33354817*x^2 - 17000985*x - 4985420    : 10.2s     109s
x^3 + 16766060*x^2 + 491009*x - 25868840     :  8.24s    111s
x^3 - 3069789*x^2 + 31777984*x - 24323311    :  7.93s    100s
x^3 + 11823123*x^2 + 20775154*x - 20239321   : 17.6s     192s
x^3 - 26450070*x^2 + 10700466*x - 27026226   : 38.7s      ....
```

# Class Groups: Sage versus Magma with Proof

Timings from Bill Hart (see sage-devel):
**Proof = True**

```
Degree, Bits, Iterations : Pari    Magma
2,      10,   5000       : 29s     72s
2,      15,   100        : 19-38s 9-24s

x^2 + 16537*x - 774810  :  0.088s 4.89s
x^2 - 88874*x - 377973  :  1.35s  7.64s
x^2 - 807645*x + 521195 : 46.0s  64.9s
x^2 + 298895*x + 178437 : 20.9s  12.5s
x^2 - 980711*x + 369932 : 92.2s  94.2s
```

# Relative Number Fields

Many nice convenience functions for moving between absolute and relative extensions, vector spaces, etc.

```
sage: K.<a,b,c> = NumberField([x^2 + 1, x^2 + 3, x^2 + 5])
sage: (a+b+c).matrix()
[b + c     1]
[   -1 b + c]
sage: L = K.base_field(); M = L.base_field()
sage: (a+b+c).norm(L)
2*c*b + -7
sage: (a+b+c).norm(M)
-11
sage: z = (a+b+c).norm(L); z.norm(M)
-11
sage: R.<x> = K[]
sage: f = (x^3 - (a+b)*x + c)*(x-2*a)*(x^2 - b); f
x^6 + ((-2)*a)*x^5 + ((-1)*a + (-2)*b)*x^4 + ...
sage: f.factor()
(x + (-2)*a) * (x^2 + (-1)*b) * (x^3 + ((-1)*a + (-1)*b)*x + c
```

# Embeddings

```
sage: K.<a,b,c> = NumberField([x^2 + 1, x^2 + 3, x^2 + 5])
sage: K.embeddings(L)
[Relative number field endomorphism of Number Field in a with
  Defn: a |--> a         b |--> b          c |--> c,
...   Relative number field endomorphism of Number Field in a
  Defn: a |--> (-1)*a    b |--> (-1)*b     c |--> c]
sage: f = K.embeddings(L)[-1];    f(a+b+c+2/3)
(-1)*a + (-1)*b + c + 2/3

sage: K.<a> = NumberField(x^3 - 2)
sage: L = K.galois_closure()
sage: K.embeddings(L)
sage: K.<a> = NumberField(x^3 - 2)
sage: L = K.galois_closure()
sage: K.embeddings(L)   # 3 morphisms output; L can be CC, CDF
[Ring morphism:
  From: Number Field in a with defining polynomial x^3 - 2
  To:   Number Field in a1 with defining polynomial x^6 + 40*x
  Defn: a |--> 1/84*a1^4 + 13/42*a1,  ...
```

# Orders

```
sage: K.<a> = NumberField(x^3 - 2)
sage: O2 = K.order(2*a); O2
Order with module basis 1, 2*a, 4*a^2 in Number Field in a wit
sage: O3 = K.order([12*a^2, 24]); O3
Order with module basis 1, 288*a, 12*a^2 in Number Field in a
sage: O3.index_in(O2)
432
sage: a in O2
False
sage: O2.intersection(O3)
Order with module basis 1, 288*a, 12*a^2 in Number Field in a

sage: time NumberField(x^19 + 23*x + 12, 'a').maximal_order()
CPU times: user 0.04 s, sys: 0.02 s, total: 0.05 s
Wall time: 0.06
Order with module basis 1, 1/6*a^18 + 1/6*a^17 + 1/6*a^16 + ..
```

# Thanks. Questions?