RIMS Kôkyûroku Bessatsu
B32 (2012), 125–144

数論研究者のための Sage
(Sage for number theorists)

By

木村巌 (Iwao Kimura)*

Abstract

Sage is an *open source* software for computer algebra and numerical computation. The aim of the Sage project is to create a viable free open source alternative to Magma, Maple, Mathematica and Matlab. In this article, we give a brief introduction to Sage for number theorists.

§1. Sage とは

Sage とは，*free* の数式処理・数値計算システムの名称である．Sage 開発プロジェクトの目標は，Magma, Maple, Mathematica, Matlab などのツールを置き換えることができる，フリーかつオープンソースなソフトウエアの開発である．開発プロジェクトを率いるのは，楕円曲線，アーベル多様体，保型形式の周辺を専門とする W. Stein（University of Washington）である．Sage には，数論に関する様々なアルゴリズムが特に豊富に実装されている．

§1.1. インストールする前に使ってみる

パソコンに Sage をインストールする前に，試しに使う方法がある．一つは，The Sage Notebook `http://www.sagenb.org/` を使うことである．ユーザ名とパスワードを設定するだけで，ウェブブラウザ越しに Sage を試すことができる．これは，Sage の開発プロジェクトが設置したクラスタマシン上で実行されている．本稿の著者が講演の際に使用した Sage ノートブックを，`http://www.sagenb.org/home/pub/2611/` で公開しているが，図 1 は，それをウェブブラウザで表示した様子である（表示閲覧は，ユーザ名の設定なしに可能である）．

Received April 10, 2011. Revised February 28, 2012.
2010 Mathematics Subject Classification(s): 11Y40
Key Words: Sage, Computational number theory
*富山大学理工学研究部（理学），〒 930-8555 富山市五福 3190
e-mail: `iwao@sci.u-toyama.ac.jp`

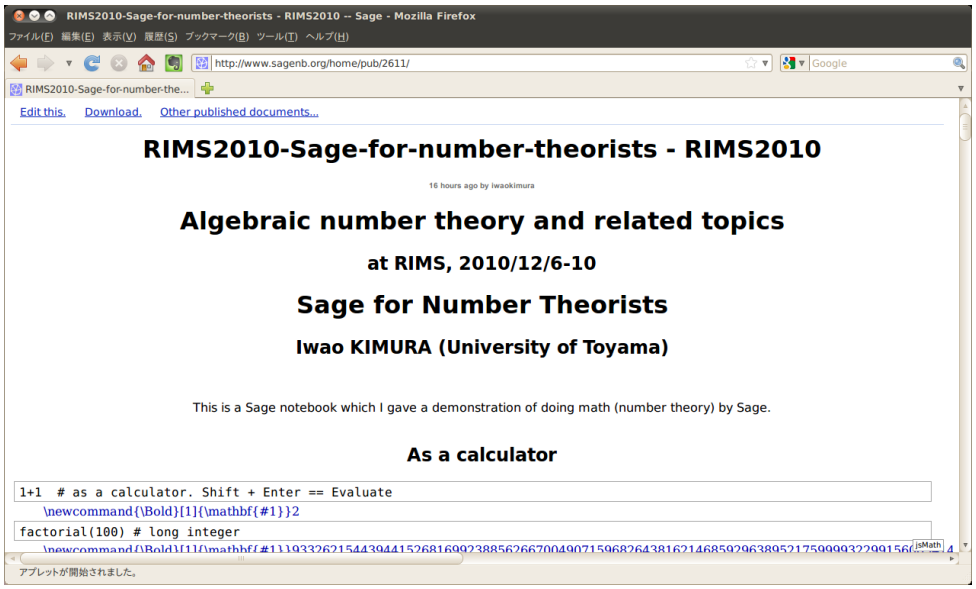

図 1. Sagenb のスクリーンショット

もう一つは，無償で配布されている，数学に特化した Linux の KNOPPIX/Math ［7］を使うことである． KNOPPIX/Math の DVD からパソコンを起動すると，ハードディスク内の Microsoft Windows とは無関係に Linux が起動する．Sage がインストールされているので，気軽に試すことができる．KNOPPIX/Math については，例えば，濱田［29］参照．

§ 1.2. インストール

Sage は，Microsoft Windows, Apple MacOS X, 各種のデスクトップ用 Linux で使うことができる．

パソコンに Sage をインストールするのは簡単である．大まかに言えば，ダウンロードして，展開するだけである．Sage の開発方針の一つに，”Battery included”（電池同梱[1]）がある．

特に，使用中の OS が，Apple MacOS X, もしくは Linux の主要なディストリビューション（Ubuntu など）ならば，前段の記述がそのまま当てはまる．OS が Microsoft Windows の場合は，実は Windows 上で Linux を起動できるようにする仮想化ソフト（例えば，VMWare player ［26］, VirtualBox ［13］）と，仮想化ソフト向けの Sage とを導入する必要がある．

Sage は，`http://www.sagemath.org/` からダウンロードできる．ブラウザから OS が判別され，適切なファイルがダウンロードされる．ファイルの容量がやや大きい（400MB 前後）ことに注意する．詳細な手順もそこにある通りである．

[1] この場合は，他に必要なものがなくすぐに使える，といった意味．

§1.3. CUI

Sageを起動するには，配布ファイルを展開したフォルダで，コマンドラインから，”./sage”とタイプする．昔ながらの，文字端末を用いたインターフェース（Character User Interface）である．カーソルキーによって行の編集やそれまで入力した履歴を遡ることができる．また，直前の計算結果は”_”（アンダースコア）に保持されている．

配布ファイルを解凍して，最初に Sage を呼び出すときだけ，やや時間がかかる（数十秒程度）．Sage を終了するには，`quit` と入力する．

§1.4. GUI

Sage は，他の商用の数式処理システムと同様の GUI（Graphical User Interface）を備えている．Sage ではこれをノートブックという．ただし，Sage 専用の GUI が新たに実装されたのではなく，いくつかの既存のコンポーネントを組み合わせることで，GUI を実現したという方が正しい．特に，GUI の表示には，ウェブブラウザ（例えば，Google Chrome, Mozilla Firefox, Internet Explorer, Apple Safari）を使う．数式部分は TEX で表示されるため，見栄えにも違和感がない．

Sage をコマンドラインから起動した後で，`notebook()` とするだけで，ウェブブラウザが起動し，Sage ノートブックが表示される．

リスト 1. Sage ノートブックの起動

```
iwao@octa:~$ ./sage-4.6/sage
┌────────────────────────────────────────────────────────────┐
| Sage Version 4.6, Release Date: 2010-10-30                 |
| Type notebook() for the GUI, and license() for information.|
└────────────────────────────────────────────────────────────┘
sage: notebook()
```

ノートブックは，実際に計算を行う「ワークシート」の集まりである．画面左端の「New Worksheet」をクリックすると，ワークシートが新規作成され，そのタイトルを入力するよう促される．

ワークシートでは，セルに入力して，すぐ下に表示される ‘evaluate’ をクリックするか，Shift+Enter を押し下げるかのいずれかで，計算が実行される．また，セルの上部に，マウスを置くと青紫に色が変わるスペースがあり，そこを Shift+マウス左クリックすると，TinyMCE ［2］という Javascript ベースのエディタが起動する．TinyMCE で編集しているときは，$で括った LATEX の数式は，適切に処理されて表示される．

実際には，Sage 同梱のウェブサーバが起動され，上述のウェブブラウザがそこにアクセスしている．つまり，実際に計算するソフトウェアとしての Sage（ウェブサーバ）と，計算結果を綺麗に表示するノートブック（ウェブブラウザ）とが完全に分離されている．これらが同一のコンピュータで稼働している必要も無い．

例えば，冒頭で紹介した `www.sagenb.org` がその例になっている．また，自分の研究室のデスクトップパソコンで Sage を走らせ，出張先で，持参したラップトップパソコン

上のウェブブラウザ経由で Sage での計算を行う，と言ったこともできる．詳細は環境に応じて変わるので，`notebook?` で表示されるヘルプを参照されたい．

GUI 版の Sage を終了するには，ワークシート右上の”Save & quit” のボタンを押した後，ノートブック右上の”Sign out” をクリックすればよい．

§1.5. Sage の特徴

Sage は，商用ソフトである Magma, Maple, Mathematica や Matlab といった数式処理系，数値計算ソフトに比肩できるものを，フリーソフトで構築しようというプロジェクトである．だが一方で，Sage は，全体が新規に書き下ろされたソフトウェアではない．

Sage プロジェクトのモットーの一つが,「車輪を再発明するのではなく，車を作ろう」である．既存のフリーソフトウェアプロジェクトの結果を躊躇無く取り込み，それらを Python というプログラミング言語で統合し共通のインターフェースで使えるようにした，一つのソフトウェアを構築する事を目指している．

例えば，Sage の数式処理部分は主に Ginac［1］, Singular［5］, Maxima［9］である．また，代数体の数論に関する部分は Pari-gp ［14］であり，楕円曲線に関する計算は mwrank ［4］であり……という具合である．

Sage のドキュメントには，研究で Sage を使った場合にはその旨を（参考文献の形で）述べてほしいとあるが，同時に，主に使ったサブシステムについても挙げてほしい，とある．例えば，Sage を用いて代数体の計算をしたなら，Sage のみならず，Pari-gp を挙げてほしい，ということである．どのサブシステムを使ったかは，リスト 18（135 頁）のようにして知ることができる．

Sage の開発は，いわゆるオープンディヴェロップメント，即ち，ソースコードをバージョン管理システムにより公開し，誰でも参加できる体制となっている．Sage の配布物に含まれるソフトウェアのライセンスは，全て GPL v2 である[2]．

それ以外の，オプショナルなパッケージの中には，修正 BSD, Apache License, MIT License など異なるライセンスを採用するものもある．しかし，全体として「フリーオープンソースライセンス[3]」となっている．

Sage が，様々なフリーの数学ソフトを統合することで成り立っていることの利点は，無償で入手できること以外にも多々ある．例えば，最新のアルゴリズムが実装された，高度に専門的なパッケージを，気軽に，ある程度共通なインターフェースで使うことができる点である．

さらに，それらを元にして，自分なりのアルゴリズムの実装などを行う際，また，Sage の計算結果が不審に見える際などに，ソースコードを参照できることも非常に大きな利点である．

一方で，より進んだ計算をしたい場合には，どうしても個々のサブシステムについての知識が必要になる．単に Sage 経由で呼び出すだけではなく，様々なパラメータを指定しないと，計算が失敗したりすることがある．

[2] GNU General Public License, Versin 2, `http://www.gnu.org/licenses/gpl-2.0.html`

[3] `http://www.opensource.org` の意味での．

いずれにしても，多数の熱心なユーザのいる Sage では，メーリングリストやウェブで，十分な情報が得られることが多い．5 節で，これらの情報源について触れる．

§2. Python

Sage は，プログラミング言語としては Python そのものである（わずかに変更がある）．

Python は，Guid van Rossum により開発が始められたスクリプト言語であり，特に欧米では広く使われている．Python もオープンソースのプロジェクト[4]である．

Linuix, Apple MacOS X, Microsoft Windows をはじめ，様々な環境で使うことができる．

Python の特徴としては，次のような点が挙げられる．インデントによるブロック構造（後述），動的な型付け（値には型があるが，変数にはオブジェクトへの参照が入る），自動ガベージコレクション（メモリ管理は Python インタープリタが暗黙に行う），オブジェクト指向のサポート，モジュールによる名前空間の分離，Unicode による多言語処理など，現代的な機能がサポートされている．

ユーザが多いことからドキュメントがよく整備されており，ライブラリも広範に渡って用意されている．なお，本稿執筆時点の Python の最新版は 2.7 系列と 3.1 系列であるが，バージョン 2 系列と 3 系列とでは差違が大きく，互換性がない．以下では，Sage で使われているバージョン 2 についてのみ説明する[5]．

§2.1. Python 速習

Python の特徴としてしばしば言及される事に，ブロック構造がインデント（字下げ）で表現される事，があるだろう．例えば C 言語では，ブロック構造は {, } で表される．普通は，C 言語でもインデント（タブによって字下げ）するが，これはプログラムを見やすくするためである．Python では，ブロック構造をインデントで表すことが文法として定められている．

以下で CUI（§1.3）を用いて，条件文，ループ，データ構造（リスト，辞書），関数など，ごく基本的な項目を概説する．Python への入門としては，Python のウェブページにある，Python Tutorial ［25］がよいと思う．

条件文（if 文）は次の様になる（`sage:`はプロンプトであり，インデントは自動的に行われるが，必要に応じて調節する．例えば次の例では，`else` を入力する前に行頭まで戻る必要がある．入力が完了していないことを示すために，`....:`が補われる．# から行末まではコメントであり，Sage は無視する）：

リスト 2. 字下げの例

[4]プロジェクトのウェブページは，`http://www.python.org` である．

[5]Sage で使われている Python は 2.6 系列である．Python のバージョン 2 系列は，2.7 系列で終了となり，開発の主力はバージョン 3 に移行する．Sage も，いずれは Python 3 系列へ移行するはずだが，時期などは明確になっていない．

```
sage: a = 1 # assign 1 to a variable a.
sage: if a==1:
....:     print """a␣is␣1"""
....: else:
....:     print """a␣is␣not␣1"""
a is 1
sage:
```

行編集の際に誤りなどで適切に字下げを行わないと，エラーになる．

リスト 3. 字下げが正しくない例

```
sage: if a==1:
....: print """a␣is␣1""" # missing tab causes an error.
------------------------------------------------------------
IndentationError: expected an indented block (<ipython console>,
    line 2)
```

データを一列に並べた，いわゆるリストは，Python でも基本的なデータ型である．カンマで区切って，ブラケットで括ることでリストが定義される．

リストの各要素を列挙し，それらについて何らかの処理を行うときには `for` 文を使う．配列の要素に直接アクセスする際は，`[]` を用いる（Python の配列は 0 番目から始まる）：

リスト 4. 配列と for 文

```
sage: a = ['one', 'two', 'three']
sage: for x in a:
....:     print x, len(x)
one 3
two 3
three 5
sage: print a[1]
two
```

辞書というデータ型もあり，これは二つのデータ，キーと値を ':' で対にしたものをカンマで区切って，カーリーブレース {,} で括ることで辞書が定義される．次のリストで，`print x,` の行末の"," は，改行を抑制する，という作用がある：

リスト 5. 辞書と for 文

```
sage: d={'one':1, 'two':2, 'three':3}
sage: d.keys() # the list of keys
['three', 'two', 'one']
sage: d.values() # the list of values
[3, 2, 1]
sage: for k in d.iterkeys():
....:     print d[k],
3 2 1
```

辞書のキーについてループするには，上のように d.iterkeys() に対してループすればよい．これはイテレータというものだが割愛する．

ある範囲の整数についてループを回したいときには，range() という関数を使う．range() は，指定しなれば 0 から始まる連続した整数の列をリストとして返す．

リスト 6. 10 回ループ

```
sage: for x in range(10):
          print x, # note that trailing ','
....:
0 1 2 3 4 5 6 7 8 9
```

始点やステップを指定することももちろんできる：

リスト 7. 1 から 3 飛びで

```
sage: for x in range(1,10,3):
          print x,
....:
1 4 7
```

また，一定のルールで生成される元たちを要素とするリストを作るときは，いわゆる「リスト内包表記（list comprehension)」が便利である．これは，集合の記法に似ており数学者には馴染みやすい．さらに，関数 sum() により，リストの総和を求めている．

リスト 8. リスト内包表記

```
sage: [x^2 for x in range(10)] # square of 0, 1, ..., 9.
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
sage: sum([x^2 for x in range(10)])
285
```

関数定義は次のようにする．double が関数名，x がパラメタである：

リスト 9. 関数定義の例：引数を 2 倍する関数

```
sage: def double(x):
....:     return 2*x
....:
sage: double(10)
20
```

以上，実行例はいずれも Sage での例だが，Python でもまったく同様である（ただし，Python では 2 乗の表記は x**2 である.）．

§3. Sage の使い方

次に，Sage ならではの機能を試していこう．

§3.1. 素朴な使い方

コマンドを打って，答えが返ってくる，という，電卓同様の使い方である．

リスト 10. 5 番目の Fermat 数の素因数分解

```
sage: factor(2^2^5+1) # prime factorization of the 5-th Fermat number
641 * 6700417
```

もう少し自明でない例として，有理数体 $\mathbf{Q}$（以下のリストで，QQ と表示されているもの）上の 1 変数多項式環を定義してみよう．

リスト 11. 多項式環

```
sage: R.<x> = PolynomialRing(QQ); R
Univariate Polynomial Ring in x over Rational Field
```

`PolynomialRing()` は，クラスの名前[6]であり同時に，そのクラスのインスタンスを生成する関数名でもある．タイプするには長いが，タブキーによる補完が効くので適宜使用する．また，`R.<x>=QQ[]` と略記することもできる．行末のセミコロンは，出力を抑制する働きがある．また，関数名の後に '?' をつければ，その関数のヘルプが表示される．さらに，'??' をつけると，その関数のソースコードが表示される．

引き続いて，この多項式環の元を定義し，因子分解してみよう：

```
sage: f = 2*x^2+3*x+1; f
2*x^2 + 3*x + 1
sage: factor? # ヘルプの出力略
sage: factor(f) # polynomial factorization
(2) * (x + 1/2) * (x + 1)
sage: f.factor() # this is the same as above.
```

最後の例は，多項式 `f` に，自らを因子分解せよ，というメッセージを送る，という形式である[7]．

同じ計算を，今度は 7 進体 $\mathbf{Q}_7$ に対して行ってみる：

リスト 12. 7 進体上の 1 変数多項式の計算

```
sage: Q7 = Qp(7); Q7 # the field of p-adic numbers, here p = 7
7-adic Field with capped relative precision 20
sage: S.<x>=Q7[]; S
Univariate Polynomial Ring in x over 7-adic Field with capped
    relative precision 20
sage: f7=S(f); f7 # conversion from QQ[x] to Q7[x]
(2 + O(7^20))*x^2 + (3 + O(7^20))*x + (1 + O(7^20))
sage: factor(f7)
```

[6] クラスの名前は，いわゆる CamelCase（大文字小文字の切り替えで単語を分かつ書き方）をする．

[7] クラス `PolynomialRing` のオブジェクト `f` に，メッセージ（もしくはオブジェクトメソッドとも言う）`factor` を送る，という事．オブジェクトメソッドは小文字で，単語を分かつにはアンダースコア '_' を用いる．

```
(2 + O(7^20)) * ((1 + O(7^20))*x + (1 + O(7^20))) * ((1 + O(7^20))*x
    + (4 + 3*7 + 3*7^2 + 3*7^3 + 3*7^4 + 3*7^5 + 3*7^6 + 3*7^7 +
    3*7^8 + 3*7^9 + 3*7^10 + 3*7^11 + 3*7^12 + 3*7^13 + 3*7^14 +
    3*7^15 + 3*7^16 + 3*7^17 + 3*7^18 + 3*7^19 + O(7^20)))
```

3 つ目の入力は，有理数係数の多項式を，$\mathbf{Q}_7$ 係数の多項式へ明示的に変換している（4 節を参照）．7 進数体の元は，7 の冪が 20 までの近似値として表されている（指定して変えることもできる）．第 3 項の定数項は 1/2 の 7 進展開である．

有限体の例を見てみよう．10^5 の次の最初の素数を p として，p 元体を F とし，その原始根 $b = 2$ を求め，b についての 3 (mod p) の離散対数を求めている．また，p^2 元体を定義するのも簡単である．：

リスト 13. 有限体の計算例

```
sage: p = next_prime(10^5); p # the prime next to 10^5
100003
sage: F = FiniteField(p); F # the finite field of p elements
Finite Field of size 100003
sage: b=F.multiplicative_generator(); b # a primitive root mod p
2
sage: F(3).log(b) # the discrete log of 3 w. r. t. b
86449
sage: FF.<a> = FiniteField(p^2); FF
Finite Field in a of size 100003^2
```

最後の例で，`a` は p 元体上の生成元を与える．

なお，初等数論，有限体や有限体上の楕円曲線，また公開鍵暗号への応用などを Sage を使いながら解説したテキストとして，Stein [18] を挙げる．

次に 2 次体の計算例を挙げる．次のリストで，`d` は 2 次体の生成元（引数に与えた整数の平方根）を与える．`kk` や `F` からそれらの生成元を得るには，`kk.gen()`, `F.gen()` のようにする．

リスト 14. 2 次体の計算例

```
sage: kk.<d>=QuadraticField(-974); kk
Number Field in d with defining polynomial x^2 + 974
sage: kk.class_number()
36
sage: C=kk.class_group(); C
Class group of order 36 with structure C12 x C3 of Number Field in
    d with defining polynomial x^2 + 974
sage: C.elementary_divisors()
[3, 12]
sage: F.<d>=QuadraticField(199); F # a real quadratic field
Number Field in d with defining polynomial x^2 - 199
sage: F.class_number() # its class number
1
```

```
sage: F.units() # its unit group (non-fundamental?)
[1153080099*d - 16266196520]
sage: e = F.units()[0]; e
1153080099*d - 16266196520
sage: -1/e # fundamental
1153080099*d + 16266196520
sage: gp.quadunit(4*199)
16266196520 + 1153080099*w
```

現在のところ，2 次体の単数を計算すると，必ずしも基本単数とは限らない結果が返るようである．最後の計算は，Pari-gp の `quadunit()` を直接呼び出している．

円分体の計算例．7 分体を定義し，類数，単数群，部分体の一覧を出力させる：

リスト 15. 円分体の計算例

```
sage: k=CyclotomicField(7); k
Cyclotomic Field of order 7 and degree 6
sage: print k.class_number(), k.units()
1 [zeta7^5 + zeta7, zeta7^4 + zeta7^3 + 1]
sage: k.subfields() # 結果略
```

更に，Galois 群や素数の分解，分岐理論などを確認してみよう：

リスト 16. 円分体の計算例（続)

```
sage: G=k.galois_group(); G
Galois group of Cyclotomic Field of order 7 and degree 6
sage: p2=k.prime_above(2); p2
Fractional ideal (-zeta7^5 - zeta7^3 - zeta7^2)
sage: Z=p2.decomposition_group(); Z
Subgroup [(), (1,3,4)(2,5,6), (1,4,3)(2,6,5)] of Galois group of
    Cyclotomic Field of order 7 and degree 6
sage: Z.fixed_field()
(Number Field in zeta70 with defining polynomial x^2 + x + 2, Ring
     morphism:
  From: Number Field in zeta70 with defining polynomial x^2 + x +
      2
  To:   Cyclotomic Field of order 7 and degree 6
  Defn: zeta70 |--> zeta7^4 + zeta7^2 + zeta7)
sage: s=p2.artin_symbol(); s
(1,3,4)(2,5,6)
sage: s(k.gen()) # k.gen() is the generator of k
zeta7^2
```

なお，Sage での計算は**全て数学的に厳密な結果を返すことをデフォルト**の挙動としている．例えば，類数や類群の計算の際に，一般リーマン仮説を仮定したり，更に強い経験的な仮説を仮定すると，計算時間を早めることができることが知られている．しかし，Sage では特に指定しなければ，現時点で証明されている事実だけに基づいて計算を行う．

仮説に基づいた計算を行いたい場合，引数として明示的に'proof=False' を指定する：

リスト 17. proof=False を指定.

```
sage: CyclotomicField(23).class_number(proof=False)
3
```

この指定がないと，23 分体の類数の計算は現実的な時間内には終了しない．計算を中断させるには，CUI ならば Control+C 連打，GUI ならば，ノートブックの上部にある，"Action" のセレクトボックスから "Interrupt" を選ぶ.

計算にどのサブシステムを使用したかを表示させるには，次のようにする：

リスト 18. 計算に使ったシステム

```
sage: from sage.misc.citation import get_systems
sage: get_systems('k.subfields()')
['PARI', 'MPFI', 'FLINT', 'MPFR', 'GMP', 'NTL']
```

リスト 15 の中の部分体を列挙する計算で，Pari-gp ［14］, MPFI ［8］, Flint ［11］, MPFR ［10］, GMP ［6］, NTL ［15］が使われていることが分かる.

有理数体上の楕円曲線の例を少し見ておこう：

リスト 19. 有理数体上の楕円曲線の計算例

```
sage: E = EllipticCurve([-82, 0]); E # if only 2 arg's a, b are
    given, then it means y^2 = x^3 + ax + b.
Elliptic Curve defined by y^2 = x^3 - 82*x over Rational Field
sage: E.discriminant().factor() # good red. outside 2, 41.
2^9 * 41^3
sage: E.gens() # generator of rational points.
[(-9 : 3 : 1), (-8 : 12 : 1), (-1 : 9 : 1)]
sage: E.has_cm() # this is a CM elliptic curve.
True
sage: L=E.lseries(); L
Complex L-series of the Elliptic Curve defined by y^2 = x^3 - 82*x
     over Rational Field
sage: L.taylor_series()
(1.19494180422039e-22)*z + (-4.86468940989224e-22)*z^2 +
    17.7891345770978*z^3 - 72.4207775497614*z^4 +
    161.461585497778*z^5 + O(z^6)
sage: E.analytic_rank()
3
```

最後の L 関数のテイラー展開では，1 次, 2 次の係数が非常に小さいことに注意されたい.

§ 3.2. 少し進んだ使い方

関数の定義，ファイル入出力を伴う使い方，オブジェクトの永続化，などを述べる.

円分体（素数分体）の相対類数を，解析的類数公式を用いて計算する例を見よう．まず，奇素数 p に対して，p 分体の相対類数を h_p^- と書くことにすると，これは解析的類数公式によって，次のように与えられるのだった：

$$h_p^- = wQ \prod_{\chi} \left(-\frac{B_{1,\chi}}{2} \right),$$

ここで，w は p 分体内の 1 のべき根の個数なので $2p$, Q は Hasse の単数指数でこの場合 1 であることが知られている．積は導手 p の奇な Dirichlet 指標 χ を渡り，$B_{1,\chi}$ は χ に付随する 1 次の一般化 Bernoulli 数であった（例えば，Washington ［27, Chap. 4, Th. 4.17］参照）．

Sage には，与えられた導手を持つ Dirichlet 指標の群を返す関数 `DirichletGroup()` がある．今の場合は，Dirichlet 指標の群が巡回群なので特に簡単である．Dirichlet 指標に対して，一般化 Bernoulli 数，Gauss 和，Jacobi 和などを返す関数が用意されている．

リスト 20. Dirichlet 指標の群

```
sage: DG=DirichletGroup(23);
sage: chi=DG.gen() # a generator
sage: chi
Dirichlet character modulo 23 of conductor 23 mapping 5 |-->
    zeta22
sage: chi.bernoulli(1) # the 1st Bernoulli number
-6/23*zeta22^9 + 14/23*zeta22^8 + 6/23*zeta22^7 - 2/23*zeta22^6 +
    12/23*zeta22^5 - 10/23*zeta22^4 - 8/23*zeta22^3 - 14/23*zeta22
    ^2 - 18/23*zeta22 - 16/23
sage: chi.gauss_sum() # 結果略
```

なお，Dirichlet 指標の計算機への実装については，Stein ［17, Chap. 4］を見よ．

p 分体の相対類数を求める関数としてまとめよう．奇素数 p が与えられたとき，一般化 Bernoulli 数 $-B_{1,\chi}/2$ のリストをリスト内包表記（リスト 8 参照）で作り，それらの積を取ればよい．(次のリストの `return` の行は，紙幅の都合で折り返している．)

リスト 21. p 分体の相対類数

```
sage: def hpminus_analytic(p):
....:     DG = DirichletGroup(p)
....:     chi = DG.gens()[0]
....:     return 2*p*prod([-((chi)^(2*k+1)).bernoulli(1)/2 for k
    in range(0,(p-1)/2)])
....:
sage: hpminus_analytic(23)
3
```

一般の n 分体の相対類数を計算する関数を書くのも簡単である．`hnminus_analytic()` という関数として定義したとしよう．関数の冒頭の `r"""`から`"""`まではコメントである．ヘルプメッセージとして，`hnminus_analytic?`としたときに表示される．

リスト 22. n 分体の相対類数．relativeclassno.sage として保存．

```
def hnminus_analytic(n):
    r"""
␣␣␣␣This␣function␣computes␣the␣relative␣class␣number␣of␣an
␣␣␣␣n-th␣cyclotomic␣field␣by␣the␣analytic␣class␣number␣formula.
␣␣␣␣"""
    w = CyclotomicField(n).zeta_order()
    if is_prime_power(n):
        Q = 1
    else:
        Q=2
    DG = DirichletGroup(n)
    return w*Q*prod(-(chi.bernoulli(1))/2 for chi in DG if chi.
        is_odd())
```

上記の関数を別のファイルとして作成し，Sage へ読み込むには次のようにする．上の関数定義の部分を，例えば relativeclassno.sage というファイルに保存する（リスト 22）．拡張子は何でもよい．エディタが言語に応じたモードを持っている場合には，Python モードになるようにしておくと便利である．

Sage に読み込むには，attach() という関数を使う．次のようにすると，Sage は当該ファイルを監視し，更新がある度に読み込み直す．一度読み込むだけならば，load() という関数を使う．

リスト 23. Sage への読み込み

```
sage: attach "~/Lang/Sage/relativeclassno.sage"
```

なお，虚アーベル体の相対類数を計算する為の関数他の Magma による実装例が，木田 [28] にある．

これらの関数を使って，数表を作ってみよう．リスト内包表記を使って，素数 p と h_p^- の組（タプル）を要素とするリストを作り，そこからさらに，相対類数を因数分解したリストを作る（リスト 24 の 2 行目で ZZ() があるのは，Sage は hpminus_analytic() の返り値を円分体の数と認識しているため，有理整数に変換しているのである．conversion については 4 節を参照.）：

リスト 24. 表の作成

```
sage: pandhpm=[(p, hpminus_analytic(p)) for p in primes(20,100)];
sage: pandhpmandfactor=[(t[0], t[1], factor(ZZ(t[1]))) for t in
    pandhpm]
```

計算結果は紙幅の為に省略する．

Sage を終了してしまうと，上記の数表も消えてしまう．次のようにして，Sage のデータベースに保存しておき，次に Sage を起動したときに呼び出すことができる：

```
sage: db_save(pandhpmandfactor,'pandhpmandfactor') # save
sage: db('pandhpmandfactor') # load
```

これも結果は省略する．

次に，Sage に組み込まれているサブシステムを，特にパラメータを指定して使用する例として，2012 が合同数か?という問題[8]を取り上げる．計算には，J. Cremona の mwrank［4］を呼び出すのだが，探索範囲を指定する必要があった．

mwrank のパラメータとして，`-b 15` を指定している．これは，4 次の等質空間の点を探索する際の高さの範囲を指定している（デフォルトの値よりも広げている）：

リスト 25. mwrank をパラメータを指定して使う例

```
sage: E2012=EllipticCurve([-2012^2,0]); E2012
Elliptic Curve defined by y^2 = x^3 - 4048144*x over Rational Field
sage: Emw=E2012.mwrank(options='-b 15'); Emw # 出力略
```

この計算により $y^2 = x^3 - 4048144x$ 上の整数点が見つかり，特に 2012 が合同数であることも分かる．

Sage を使っていると，バグと思しき事象に突き当たることもある．ある実 2 次体上の楕円曲線の有理点（無限遠点とは異なる）を探す計算例を見よう[9]．実 2 次体 $\mathbf{Q}(\sqrt{d})$ の基本単数 ε_d に対して，この 2 次体上の楕円曲線 $E_d : y^2 = x^3 + 1728\varepsilon_d$ を考える．

リスト 26. $\mathbf{Q}(\sqrt{41})$ 上のある楕円曲線の計算例

```
sage: Q41.<a>=QuadraticField(41);
sage: eps = UnitGroup(Q41).fundamental_units()[0];
sage: E = EllipticCurve(Q41, [0, 1728*eps]); E
Elliptic Curve defined by y^2 = x^3 + (8640*a+55296) over Number
    Field in a with defining polynomial x^2 - 41
sage: dscnt = E.simon_two_descent(verbose=1); dscnt # 出力略
sage: E(dscnt[2][0])
(-377788/93025*a - 1952448/93025 : -205379776/28372625*a -
    1032512096/28372625 : 1)
sage: E.rank()
2
sage: E.gens()
[(-377788/93025*a - 1952448/93025 : -205379776/28372625*a -
    1032512096/28372625 : 1)]
sage: E(Q41)
Abelian group of points on Elliptic Curve defined by y^2 = x^3 +
    (8640*a+55296) over Number Field in a with defining polynomial
     x^2 - 41
```

この計算は正常に終了している．問題は，$d = 43$ の時である．2010 年 12 月の時点では，$d = 43$ の場合は計算できず，これは，代数体上の楕円曲線の代数的降下を計算するため

[8] この計算例は中村博昭氏（岡山大）から筆者へ，口頭での質問であった．

[9] この計算例は，横山俊一氏（九州大）から筆者への質問がきっかけであった．

のプログラム ell.gp ［16］のバグが原因であった．ell.gp は Pari-gp でかかれた独立なプログラムであり，作者は D. Simon である．既に横山氏と Simon 氏とで情報交換があり，最新版では修正済みという．

Sage のバグと思われる挙動を発見した際には，Sage ユーザのメーリングリスト sage-support で質問するという手がある（5 節参照）．Sage のバージョン，OS, どのような計算をして，どのような結果が出て，どうあるべきだったか，などを簡潔に記述する．

§ 3.3. 数表の利用

Sage には，いくつかの数表が含まれている．また，オプションパッケージとして，追加で導入することもできる．オプションパッケージの確認と追加は以下のようにする（パソコンがインターネットに接続されている必要がある）：

リスト 27. オプションパッケージの確認と追加

```
sage: optional_packages() # 出力略
sage: install_package('database_cremona_ellcurve')
```

上では，J. Cremona の楕円曲線のデータベース［3］を，オプションパッケージとして追加している．

例えば，Cremona の表から導手 11 の楕円曲線を抜き出すには次のようにする．結果は，導手類の名前をキー，a 不変量，ランク，トーション群の位数のリストを値とする辞書である（辞書についてはリスト 5，130 頁を参照）：

リスト 28. Cremona の表の使い方

```
sage: c=CremonaDatabase(); c
Cremona database of elliptic curves
sage: E11=c.allcurves(11); E11
{'a1': [[0, -1, 1, -10, -20], 0, 5], 'a3': [[0, -1, 1, 0, 0], 0,
    5], 'a2': [[0, -1, 1, -7820, -263580], 0, 1]}
```

Cremona の表の他にも，J. Jones による分岐を制限した代数体（6 次以下）の表や，Odlyzko の Riemann zeta 関数の零点の表，Sloane の OEIS へのインターフェースなどが用意されている．

§ 4. Parent/Element, Category, Coercion and Conversion

Sage では，数学的な対象が，その構造も込めてソフトウェア上に実現されている．一つは，数学での「圏と関手」であり，もう一つは，Parent/Element というものである．

Sage は例えば，有理整数環が，単なる整数の集まりではなく，Euclid 整域の圏，単項イデアル整域の圏，可換整域の圏（射は可換環の間のそれを取る），それぞれの対象であり，また，加法に関しては加法群の圏，乗法に関してはモノイドの圏，それぞれの対象であることを知っている．また，これらの圏の間の関係も知っている．

リスト 29. Categories of ZZ

```
sage: ZZ.categories()
[Category of euclidean domains, Category of principal ideal
    domains, Category of gcd domains, Category of integral domains
    , Category of commutative rings, Category of domains, Category
     of rings, Category of rngs, Category of commutative additive
    groups, Category of semirings, Category of commutative
    additive monoids, Category of commutative additive semigroups,
     Category of additive magmas, Category of monoids, Category of
     semigroups, Category of magmas, Category of sets, Category of
     sets with partial maps, Category of objects]
sage: QQ.categories() # 出力略.
```

一方，数学的な概念を正確にコンピュータの上に実装すると，それを使うことが非常に煩雑になることがある．できるだけ人間による指示なしに，自動化できるところは自動化したい．

例えば，整数 1 と有理数 1/2 の和は，当然有理数体の元 3/2 として定まるべきである．一方 1 は有理整数環の元であり，1/2 は有理数体の元である．もし，1 + (1/2) の計算をする際に，ユーザが，1 を有理数に変換し，その上で 1/2 と足し算せよ，と指示しなければならないとしたら，煩雑でたまらないだろう．Sage で，どのように解決されているかを簡単に見ておく．

1 や 1/2 はそれぞれ，しかるべき圏に属している．ただし，1 の属する圏は有理整数環の元の成す圏（集合）であり，有理整数環その物ではないことに注意する：

リスト 30. Category of 1, 1/2

```
sage: 1.category()
Category of elements of Integer Ring
sage: 1.category()==ZZ # caution!
False
sage: (1/2).category()
Category of elements of Rational Field
sage: (1+1/2).category()
Category of elements of Rational Field
```

いずれにせよ，ユーザが介入せずとも適切に coerce（ある圏の対象から別の圏の対象へと「自然に」変換）され，正しい結果が得られている．

Sage では，1 と有理整数環との関係は，上記の圏論的な関係以外にもう一つある．それは，1 の Parent が有理整数環であること，1 は有理整数環の Element であること，である．Parent/Element の関係は，Magma に由来するものらしい（Stein［20］参照）．Sage の Category が，数学的な圏の実装であったのに対し，Parent/Element の関係は，圏の対象と，その元，との関係を表す．

リスト 31. Parent/Element の関係

```
sage: 1.parent()
Integer Ring
sage: (1/2).parent()
Rational Field
sage: (1+1/2).parent()
Rational Field
```

もう少し自明でない例，例えば，有理整数係数の，x を変数とする 1 変数多項式環の元 x と，1/2 の和を考える．和は有理数係数の x を変数とする 1 変数多項式環に定まるべきで，実際 Sage でもそうなる．

リスト 32. $x + (1/2)$ の例

```
sage: R.<x> = ZZ[];
sage: x.parent()
Univariate Polynomial Ring in x over Integer Ring
sage: (1/2).parent()
Rational Field
sage: (x+1/2).parent()
Univariate Polynomial Ring in x over Rational Field
```

この時に Sage の内部で行われる処理は，大体次のようになる．まず x と 1/2 の Parent 達 R と有理数体に対して，それがどのように構成されたものかを調べる：

リスト 33. Parent 達の構成を調べる

```
sage: R.construction()
(Poly[x], Integer Ring)
sage: QQ.construction()
(FractionField, Integer Ring)
```

共通の構成要素として，有理整数環 $\mathbf{Z}$ (Integer Ring) が見つかる．そこから，R ($= \mathbf{Z}[x]$) に対しては，有理数体上の 1 変数多項式環への coercion が得られ，有理数体についても，そこから有理数体上の 1 変数多項式環への coercion が得られる．この過程を実際に見ると，次のようになる：

リスト 34. coercion の発見

```
sage: cm=sage.structure.element.get_coercion_model()
sage: cm.discover_coercion(R, QQ)
(Conversion map:
From: Univariate Polynomial Ring in x over Integer Ring
To:   Univariate Polynomial Ring in x over Rational Field,
    Polynomial base injection morphism:
From: Rational Field
To:   Univariate Polynomial Ring in x over Rational Field)
```

$x + (1/2)$ は，$\mathbf{Z}[x]$ の商体，つまり有理関数体 $\mathbf{Q}(x)$ の元と思ってもよいが，$\mathbf{Q}[x]$ の元と見た方が無駄がない．この辺の判断は，Sage が常識を働かせてくれる．

以上，Coercion, Parent/Element の詳細については，Sage reference manual ［22］の”The Coercion Model” の項などを参照していただきたい．

ただし，Sage を使っていて，この節に述べたようなことが気になることは，それほどないはずである．Sage が可能な限りうまく処理してくれる．一方で，次のような例もあるので注意が必要である：

リスト 35. 10/2 の Parent は？

```
sage: (10/2).parent()
Rational Field
sage: 10/2==ZZ(10/2)
True
```

結果の 5 を整数と見るためには，明示的に整数に変換する必要がある．この様な明示的な変換を，conversion という（リスト 12, 132 頁も参照）．conversion は，数学的な自然さなどは措き，あれば便利，という雰囲気のものである．例えば，有限体の元を有理整数へ convert することができる．

§5. Sage の情報・結語

Sage に関する解説文書は多数ある．Sage 開発陣によるものが，`http://www.sagemath.org/help.html#SageStandardDoc` に集積されている．まず Sage tutorial ［23］を見るのがよいと思う．また，本稿の趣旨に近く，より進んだ話題を扱っているのが，Stein ［21］である．基本的なところから詳細に解説された文献として，Kosan ［12］ がある．この日本語訳が，横田博史氏により公開されている［24］．また，Sage 開発プロジェクトのリーダ W. Stein が，プロジェクトが軌道に乗るまでの経緯をつづった［19］も大変興味深い．

Sage の使い方，Sage の開発者らの情報交換の場，日本の Sage ユーザの情報交換の場として，それぞれ，

sage-support `http://groups.google.com/group/sage-support`

sage-devel `http://groups.google.com/group/sage-devel`

sage-japan `http://groups.google.com/group/sage-japan`

がある．いずれもメーリングリストを購読することなく，上記の URL から見ることもできる．また，asksage というウェブページ `http://ask.sagemath.org/questions/` でも，Sage の使い方に関する活発な情報交換が行われている．

本稿では，Magma, Maple, Mathematica や Matlab などの商用数式処理・数値計算システムを置き換えうる *free* なソフトウェア Sage を，特に数論の研究に使う，という観点から紹介した．拙文をきっかけに，国内からも Sage への貢献が現れれば望外の喜びである．

謝辞：2010 年度「代数的整数論とその周辺」主催者ならびにプログラム責任者の皆様方に御礼申し上げます．原稿にコメントをくださいました横山俊一さん（九大数理院）に感謝いたします．また，丁寧に読み，多くのコメントをくださいました査読者に御礼申し上げます．

参考文献

[1] *Ginac is not a CAS*, http://www.ginac.de/.
[2] Moxiecode Systems AB., *TinyMCE*, http://tinymce.moxiecode.com/.
[3] J. E. Cremona, *Elliptic Curve Data*, University of Warwick, http://www.warwick.ac.uk/~masgaj/ftp/data/.
[4] ———, *mwrank and related programs for elliptic curves over* **Q**, University of Warwick, http://www.warwick.ac.uk/~masgaj/mwrank/.
[5] G.-M.; Pfister G.; Schönemann H. Decker, W.; Greuel, Singular *3-1-2 — A computer algebra system for polynomial computations*, (2010), http://www.singular.uni-kl.de.
[6] The GMP developers, *GMP, the GNU Multiple Precision arithmetic library, edition 4.3.2*, FSF, Jan 2011, http://gmplib.org/.
[7] The KNOPPIX/Math developing team, *KNOPPIX/Math 2011 Japanese edition*, knoppix_v6.4.4-math-dvd-20110303-ja.iso, http://www.knoppix-math.org/, 2011.
[8] Fabrice Rouillier et. al., *MPFI, multiple precision interval packages*, INRIA, 2010, https://gforge.inria.fr/projects/mpfi.
[9] The Maxima Group, *Maxima, a Computer Algebra System. Version 5.18.1*, http://maxima.sourceforge.net/.
[10] Guillaume Hanrot, Vincent Lefévre, Patrick Pélissier, Philippe Théveny, and Paul Zimmermann, *MPFR, multiple precision floating-point reliable library, version 3.0.0.*, FSF, June 2010, http://www.mpfr.org/.
[11] William Hart, Fredrik Johanssony, and Sebastian Pancratzz, *FLINT version 2.1.0*, 9 March 2011, http://www.flintlib.org/.
[12] Ted Kosan, *SAGE for newbies*, Feb. 2008, http://sage.math.washington.edu/home/tkosan/newbies_book/sage_for_newbies_v1.23.pdf.
[13] Oracle, *VirtualBox*, Oracle, http://www.virtualbox.org.
[14] PARI Group, Bordeaux, *PARI/GP, Version 2.4.3*, 2008, available from http://pari.math.u-bordeaux.fr/.
[15] V. Shoup, *NTL, a library for doing number theory*, August 2009, http://www.shoup.net/ntl/.
[16] D. Simon, *ell.gp*, Universié Caen, March 2011, http://www.math.unicaen.fr/~simon/ell.gp.
[17] William Stein, *Modular forms, a computational approach*, Graduate Studies in Mathematics, vol. 79, American Mathematical Society, Providence, RI, 2007, With an appendix by Paul E. Gunnells. MR 2289048 (2008d:11037)
[18] ———, *Elementary number theory: primes, congruences, and secrets, a computational approach*, Undergraduate Texts in Mathematics, Springer, New York, 2009. MR 2464052 (2009i:11002)
[19] ———, *Mathematical software and me: A very personal recollection*, Dec 2009, http://wstein.org/mathsoftbio/history.pdf.

［20］ ———, *Brief history and motivation behind the Sage coercion model*, Tech. report, Nov. 2010, blog post on http://sagemath.blogspot.com/2010/11/brief-history-and-motivation-behind.htm.
［21］ ———, *Three lectures about explicit methods in number theory using Sage, Release 4.6.2*, March 2011, http://www.sagemath.org/doc/bordeaux_2008/.
［22］ Sage Development Team, *Sage, Reference Manual, Release 4.6.2*, March 2011, http://www.sagemath.org/doc/reference/.
［23］ The Sage Development Team, *Sage, Tutorial, Release 4.6.2*, March 2011, http://www.sagemath.org/doc/tutorial/.
［24］ 横田博史（翻訳） Ted Kosan, **はじめての** *SAGE*, 2 2008, http://www.bekkoame.ne.jp/~ponpoko/KNOPPIX/sage_for_newbies_ja.pdf.
［25］ Guido van Rossum and Fred L. Drake, JR., *Python tutorial release 2.7.1*, Python Software Foundation, March 29 2011, available from http://docs.python.org/tutorial/.
［26］ VMWare, Inc., *VMWare player*, VMWare, Inc., http://www.vmware.com/jp/products/player/overview.html.
［27］ Lawrence C. Washington, *Introduction to cyclotomic fields*, second ed., Springer-Verlag, New York, 1997. MR 97h:11130
［28］ 木田雅成, **数論研究者のための** *Magma* **入門**, 第 7 回北陸数論研究集会報告集 (平林幹人, 野村明人, 山下浩, 木村巌, and 菅野孝史, eds.), 2009, pp. 59–79.
［29］ 濱田龍義, **数学ソフトウェアあれこれ／オープンソースというひとつの流れ**, 数学セミナー **49** (2010), no. 9, 8–14.