A Tour Of Sage

Release 9.5

The Sage Development Team

CONTENTS

1	Sage as a Calculator	3
2	Power Computing with Sage	5
3	Accessing Algorithms in Sage	7

This is a tour of Sage that closely follows the tour of Mathematica that is at the beginning of the Mathematica Book.

CONTENTS 1

2 CONTENTS

CHAPTER

ONE

SAGE AS A CALCULATOR

The Sage command line has a sage: prompt; you do not have to add it. If you use the Sage notebook, then put everything after the sage: prompt in an input cell, and press shift-enter to compute the corresponding output.

```
sage: 3 + 5
8
```

The caret symbol means "raise to a power".

```
sage: 57.1 ^ 100
4.60904368661396e175
```

We compute the inverse of a 2×2 matrix in Sage.

```
sage: matrix([[1,2], [3,4]])^(-1)
[ -2   1]
[ 3/2 -1/2]
```

Here we integrate a simple function.

```
sage: x = var('x')  # create a symbolic variable
sage: integrate(sqrt(x)*sqrt(1+x), x)
1/4*((x + 1)^{3/2}/x^{3/2}) + sqrt(x + 1)/sqrt(x)/((x + 1)^{2/x^2} - 2*(x + 1)/x + 1) - 1/s*log(sqrt(x + 1)/sqrt(x) + 1) + 1/8*log(sqrt(x + 1)/sqrt(x) - 1)
```

This asks Sage to solve a quadratic equation. The symbol == represents equality in Sage.

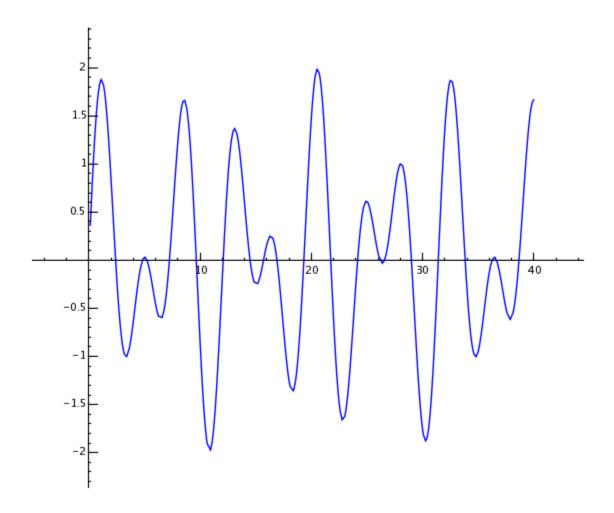
```
sage: a = var('a')
sage: S = solve(x^2 + x == a, x); S
[x == -1/2*sqrt(4*a + 1) - 1/2, x == 1/2*sqrt(4*a + 1) - 1/2]
```

The result is a list of equalities.

```
sage: S[0].rhs()
-1/2*sqrt(4*a + 1) - 1/2
```

Naturally, Sage can plot various useful functions.

```
sage: show(plot(sin(x) + sin(1.6*x), 0, 40))
```



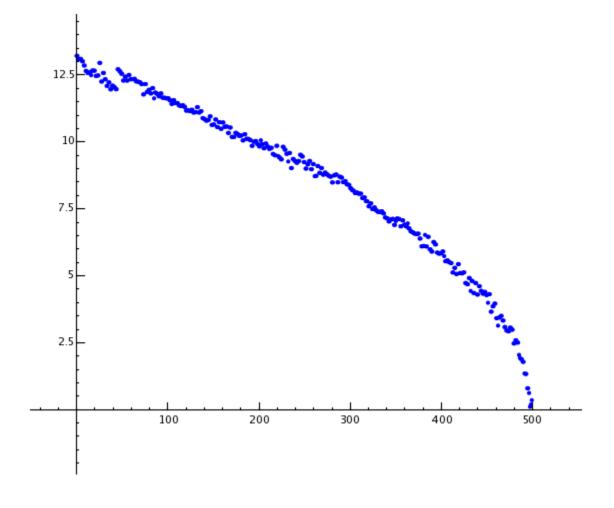
POWER COMPUTING WITH SAGE

First we create a 500×500 matrix of random numbers.

```
sage: m = random_matrix(RDF,500)
```

It takes Sage a few seconds to compute the eigenvalues of the matrix and plot them.

```
sage: e = m.eigenvalues() #about 2 seconds
sage: w = [(i, abs(e[i])) for i in range(len(e))]
sage: show(points(w))
```



Thanks to the GNU Multiprecision Library (GMP), Sage can handle very large numbers, even numbers with millions or billions of digits.

```
sage: factorial(100)
9332621544394415268169923885626670049071596826438162146859296389521759999322991560894146397615651828625
sage: n = factorial(1000000) #about 2.5 seconds
```

This computes at least 100 digits of π .

This asks Sage to factor a polynomial in two variables.

Sage takes just under 5 seconds to compute the numbers of ways to partition one hundred million as a sum of positive integers.

```
sage: z = Partitions(10^8).cardinality() #about 4.5 seconds
sage: str(z)[:40]
'1760517045946249141360373894679135204009'
```

\sim	ш	٨	D	ГΕ	R
L	п	А	Р.	ᇉ	ĸ

THREE

ACCESSING ALGORITHMS IN SAGE

Whenever you use Sage you are accessing one of the world's largest collections of open source computational algorithms.