
Sage 之旅

发行版本 10.8

Sage 开发团队

2025 年 12 月 23 日

Contents

这是一篇关于如何使用 Sage 计算器的简短介绍。

Sage 的命令行提示符为“sage:”。在实验以下示例时，你只需输入提示符后的部分。

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

```
>>> from sage.all import *
>>> Integer(3) + Integer(5)
8
```

如果你在 Jupyter notebook 中使用 Sage，也可以将提示符后的内容放入输入单元格，然后按 Shift-Enter 来获取相应的输出。

尖号 (^) 表示“乘方”。

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

```
>>> from sage.all import *
>>> RealNumber('57.1')**Integer(100)
4.60904368661396e175
```

在 Sage 中计算一个 2×2 矩阵的逆矩阵。

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

```
>>> from sage.all import *
>>> matrix([[Integer(1), Integer(2)], [Integer(3), Integer(4)]])**(-Integer(1))
[ -2   1]
[ 3/2 -1/2]
```

这里我们对一个简单函数进行积分。

```
sage: x = var('x') # 创建符号变量
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)
- 1/8*log(sqrt(x + 1)/sqrt(x) + 1) + 1/8*log(sqrt(x + 1)/sqrt(x) - 1)
```

```
>>> from sage.all import *
>>> x = var('x') # 创建符号变量
>>> integrate(sqrt(x) * sqrt(Integer(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)
- 1/8*log(sqrt(x + 1)/sqrt(x) + 1) + 1/8*log(sqrt(x + 1)/sqrt(x) - 1)
```

这里我们让 Sage 解一个二次方程。在 Sage 中，符号 == 表示相等。

```
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]
```

```
>>> from sage.all import *
>>> a = var('a')
>>> S = solve(x**Integer(2) + x == a, x); S
[x == -1/2*sqrt(4*a + 1) - 1/2, x == 1/2*sqrt(4*a + 1) - 1/2]
```

结果是一个等式列表。

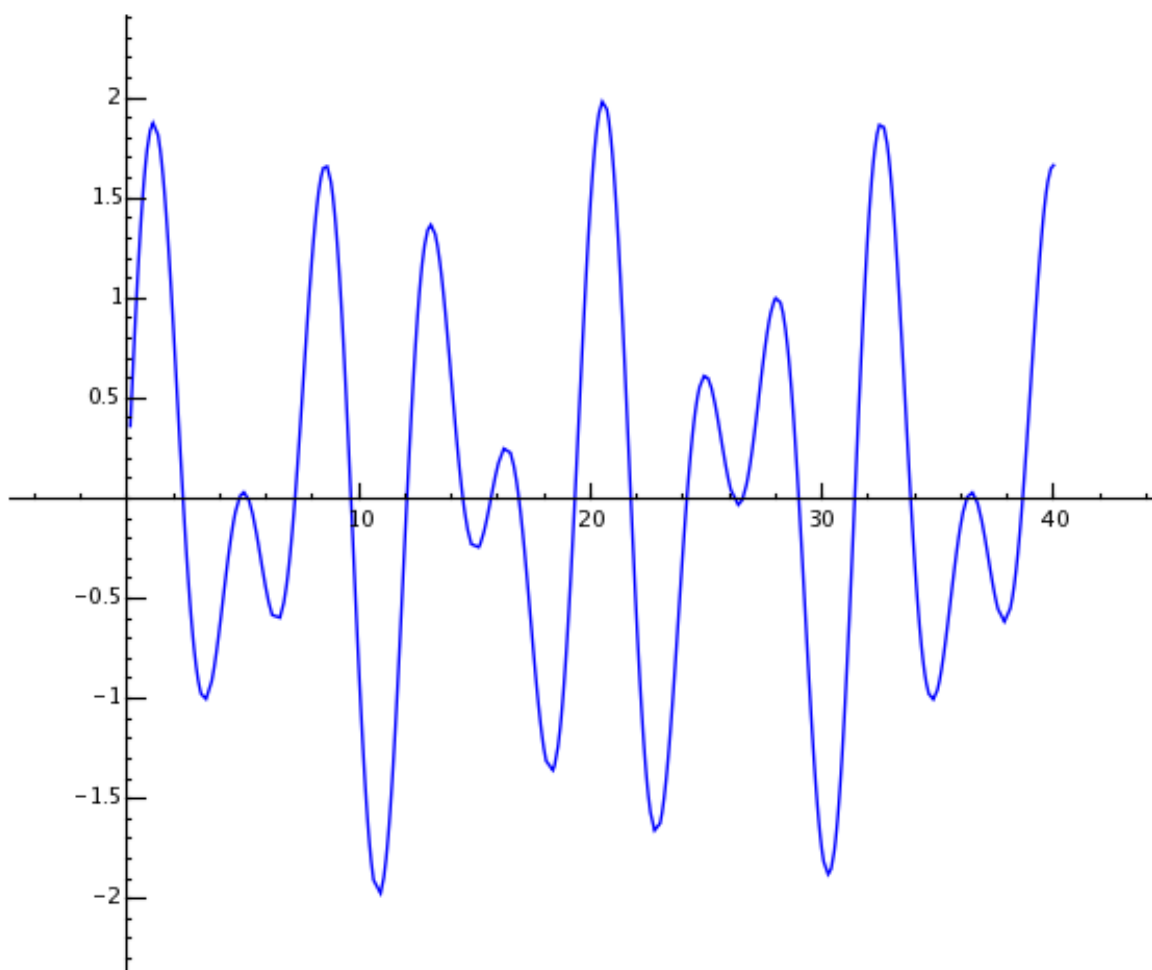
```
sage: S[0].rhs() # 方程的右侧
-1/2*sqrt(4*a + 1) - 1/2
```

```
>>> from sage.all import *
>>> S[Integer(0)].rhs() # 方程的右侧
-1/2*sqrt(4*a + 1) - 1/2
```

Sage 当然可以绘制各种常用函数。

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

```
>>> from sage.all import *
>>> show(plot(sin(x) + sin(RealNumber('1.6')*x), Integer(0), Integer(40)))
```



Sage 是一个非常强大的计算器。为了体验它的能力，首先我们创建一个 500×500 的随机数矩阵。

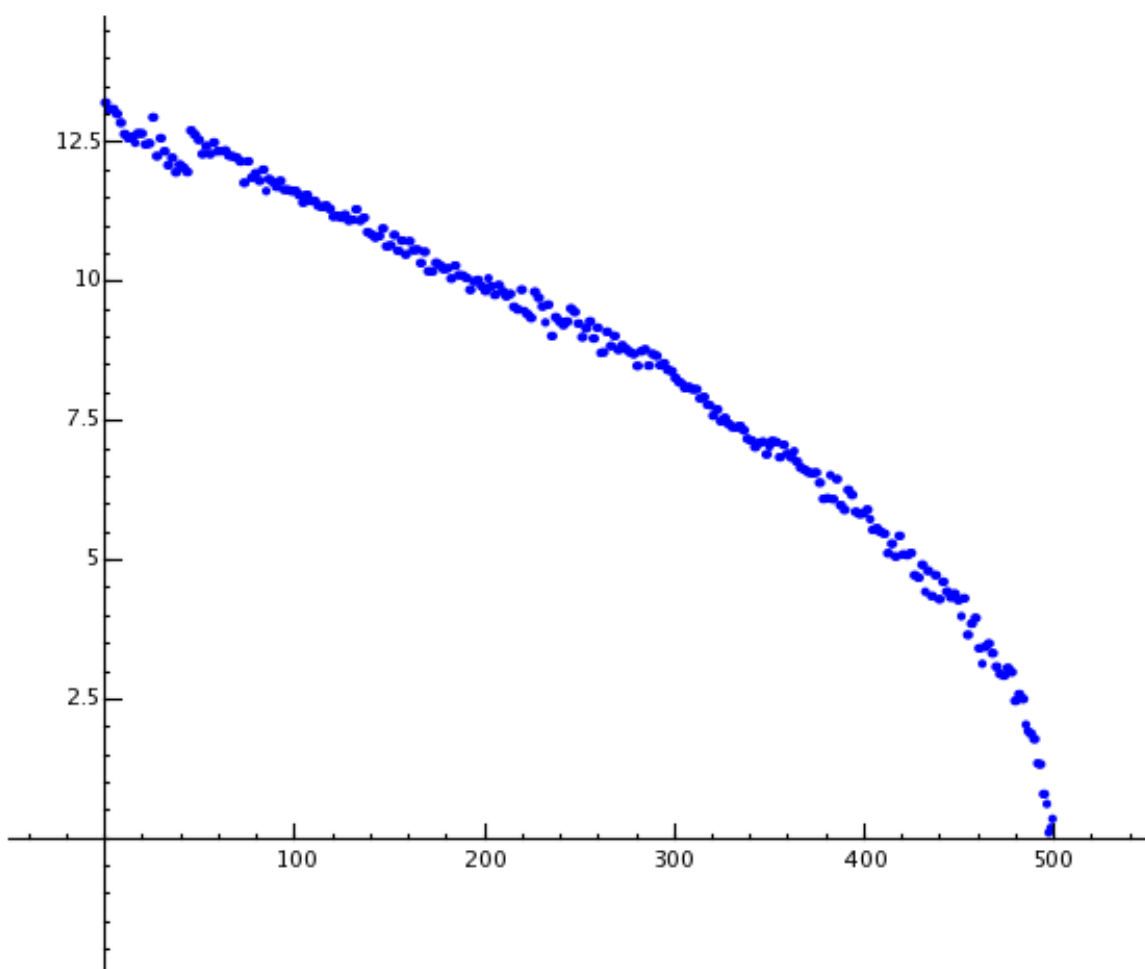
```
sage: m = random_matrix(RDF, 500)
```

```
>>> from sage.all import *
>>> m = random_matrix(RDF, Integer(500))
```

Sage 仅需一秒钟就能计算出矩阵的特征值并绘制它们。

```
sage: e = m.eigenvalues() # 大约 1 秒
sage: w = [(i, abs(e[i])) for i in range(len(e))]
sage: show(points(w))
```

```
>>> from sage.all import *
>>> e = m.eigenvalues() # 大约 1 秒
>>> w = [(i, abs(e[i])) for i in range(len(e))]
>>> show(points(w))
```



Sage 可以处理非常大的数字，甚至是数百万或数十亿位的数字。

```
sage: factorial(100)
933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565
↪182862536979208272237582511852109168640000000000000000000000
```

```
>>> from sage.all import *  
>>> factorial(Integer(100))  
933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565  
↪182862536979208272237582511852109168640000000000000000000000
```

```
sage: n = factorial(1000000) # 大约 1 秒
sage: len(n.digits())
5565709
```

```
>>> from sage.all import *
>>> n = factorial(Integer(1000000)) # 大约 1 秒
>>> len(n.digits())
5565709
```

计算 π 的前 100 位。

```
sage: N(pi, digits=100)
3.1415926535897932384626433832795028841971693993751058209749445923078164062862089986280348253421
↪17068
```

```
>>> from sage.all import *
>>> N(pi, digits=Integer(100))
3.1415926535897932384626433832795028841971693993751058209749445923078164062862089986280348253421
↪17068
```

让 Sage 对一个二元多项式进行因式分解。

```
sage: R.<x,y> = QQ[]
sage: F = factor(x^99 + y^99)
sage: F
(x + y) * (x^2 - x*y + y^2) * (x^6 - x^3*y^3 + y^6) *
(x^10 - x^9*y + x^8*y^2 - x^7*y^3 + x^6*y^4 - x^5*y^5 +
x^4*y^6 - x^3*y^7 + x^2*y^8 - x*y^9 + y^10) *
(x^20 + x^19*y - x^17*y^3 - x^16*y^4 + x^14*y^6 + x^13*y^7 -
x^11*y^9 - x^10*y^10 - x^9*y^11 + x^7*y^13 + x^6*y^14 -
x^4*y^16 - x^3*y^17 + x*y^19 + y^20) * (x^60 + x^57*y^3 -
x^51*y^9 - x^48*y^12 + x^42*y^18 + x^39*y^21 - x^33*y^27 -
x^30*y^30 - x^27*y^33 + x^21*y^39 + x^18*y^42 - x^12*y^48 -
x^9*y^51 + x^3*y^57 + y^60)
sage: F.expand()
x^99 + y^99
```

```
>>> from sage.all import *
>>> R = QQ['x, y']; (x, y,) = R._first_ngens(2)
>>> F = factor(x**Integer(99) + y**Integer(99))
>>> F
(x + y) * (x^2 - x*y + y^2) * (x^6 - x^3*y^3 + y^6) *
(x^10 - x^9*y + x^8*y^2 - x^7*y^3 + x^6*y^4 - x^5*y^5 +
x^4*y^6 - x^3*y^7 + x^2*y^8 - x*y^9 + y^10) *
(x^20 + x^19*y - x^17*y^3 - x^16*y^4 + x^14*y^6 + x^13*y^7 -
x^11*y^9 - x^10*y^10 - x^9*y^11 + x^7*y^13 + x^6*y^14 -
x^4*y^16 - x^3*y^17 + x*y^19 + y^20) * (x^60 + x^57*y^3 -
x^51*y^9 - x^48*y^12 + x^42*y^18 + x^39*y^21 - x^33*y^27 -
x^30*y^30 - x^27*y^33 + x^21*y^39 + x^18*y^42 - x^12*y^48
```

(续下页)

(接上页)

```
x^9*y^51 + x^3*y^57 + y^60)
>>> F.expand()
x^99 + y^99
```

Sage 可以在 1 秒内计算出将一亿分解为正整数之和的方式数量。

```
sage: z = Partitions(10^8).cardinality() # 大约 0.1 秒
sage: z
1760517045946249141360373894679135204009...
```

```
>>> from sage.all import *
>>> z = Partitions(Integer(10)**Integer(8)).cardinality() # 大约 0.1 秒
>>> z
1760517045946249141360373894679135204009...
```

Sage 是世界上最先进的开源数学软件。