

# Einführung in Sage - Einheit 1

Organisatorisches, Was ist Sage?, Streifzug durch Sage, Nützliches

Jochen Schulz

Georg-August Universität Göttingen



- Anmeldungen zu der Veranstaltung über StudIP  
<https://www.studip.uni-goettingen.de/>  
Einführung in Sage (Mathematische Anwendersysteme) (WS  
2009/2010)
- Alle Unterlagen (Aufgabenblätter, Vorlesungsfolien, Beispiele, Musterlösungen) können von der StudIP-Seite (Reiter Dateien) heruntergeladen werden

- Anmeldungen zu der Veranstaltung über StudIP  
<https://www.studip.uni-goettingen.de/>  
Einführung in Sage (Mathematische Anwendersysteme) (WS 2009/2010)
- Alle Unterlagen (Aufgabenblätter, Vorlesungsfolien, Beispiele, Musterlösungen) können von der StudIP-Seite (Reiter Dateien) heruntergeladen werden

## Dozent

Jochen Schulz

NAM, Zimmer 04 (Erdgeschoß)

Telefon: 39-4525 Email: [schulz@math.uni-goettingen.de](mailto:schulz@math.uni-goettingen.de)

XMPP: [schulz@jabber.num.math.uni-goettingen.de](mailto:schulz@jabber.num.math.uni-goettingen.de)

# Starten des Programms

**Vor.:** Account im CIP-Pool der Mathematischen Fakultät (MI und NAM):  
Registrierungs-Formular unter <https://ldap.math.uni-goettingen.de>  
(**Stud.It-Account** nötig!)

**Wiki** (<https://wiki.math.uni-goettingen.de/mediawiki>)

- Sage ist in Version 4.3 auf allen Rechnern der Mathematischen Fakultät installiert
- login direkt oder per **nxclient** auf [login.math.uni-goettingen.de](https://login.math.uni-goettingen.de),  
[s242.math.uni-goettingen.de](https://s242.math.uni-goettingen.de) und  
[sc1.math.uni-goettingen.de](https://sc1.math.uni-goettingen.de) bis [sc4.math.uni-goettingen.de](https://sc4.math.uni-goettingen.de)
- Starten von Sage: Im Terminal
  - `sagenotebook` (Notebook - Browser-basiert)
  - `sage` (Kommandozeilen-Version)

# Ablauf der Veranstaltung

- Blockveranstaltung vom 15.2.-26.2.2010
- Vorlesung: 9.15 Uhr bis 11.30 Uhr
- Nachmittags: 4 Übungsgruppen à je 1h 15min
  - 13:00-14:15 (Tutor: J. Schulz)
  - 14:15-15:30 (Tutor: C. Rügge)
  - 15:30-16:45 (Tutor: J. Schulz)
  - 16:45-18:00 (Tutor: C. Rügge)
- Scheinerwerb
  - Klausur am 1.3.2009; 10:00 - 12:00; Anmeldung über FlexNow!
  - Regelmäßige Teilnahme an den Übungen ist empfohlen

1. **Tag** Organisatorisches, Aufbau von Sage, Streifzug durch Sage

# Inhalt der Vorlesung

1. **Tag** Organisatorisches, Aufbau von Sage, Streifzug durch Sage
2. **Tag** Grundlagen Sage (grundlegende Datentypen, Ausdrücke), Lösen von Gleichungen, Symbolisches Rechnen

# Inhalt der Vorlesung

1. **Tag** Organisatorisches, Aufbau von Sage, Streifzug durch Sage
2. **Tag** Grundlagen Sage (grundlegende Datentypen, Ausdrücke), Lösen von Gleichungen, Symbolisches Rechnen
3. **Tag** Mengen, natürliche, rationale, reelle und komplexe Zahlen, Gleitkommazahlen



# Inhalt der Vorlesung

1. **Tag** Organisatorisches, Aufbau von Sage, Streifzug durch Sage
2. **Tag** Grundlagen Sage (grundlegende Datentypen, Ausdrücke), Lösen von Gleichungen, Symbolisches Rechnen
3. **Tag** Mengen, natürliche, rationale, reelle und komplexe Zahlen, Gleitkommazahlen
4. **Tag** Vektoren und Matrizen, Lineare Algebra in Sage, Programmieren I

# Inhalt der Vorlesung

1. **Tag** Organisatorisches, Aufbau von Sage, Streifzug durch Sage
2. **Tag** Grundlagen Sage (grundlegende Datentypen, Ausdrücke), Lösen von Gleichungen, Symbolisches Rechnen
3. **Tag** Mengen, natürliche, rationale, reelle und komplexe Zahlen, Gleitkommazahlen
4. **Tag** Vektoren und Matrizen, Lineare Algebra in Sage, Programmieren I
5. **Tag** Datencontainer in Sage, Lineare Abbildungen und Matrizen

# Inhalt der Vorlesung

1. **Tag** Organisatorisches, Aufbau von Sage, Streifzug durch Sage
2. **Tag** Grundlagen Sage (grundlegende Datentypen, Ausdrücke), Lösen von Gleichungen, Symbolisches Rechnen
3. **Tag** Mengen, natürliche, rationale, reelle und komplexe Zahlen, Gleitkommazahlen
4. **Tag** Vektoren und Matrizen, Lineare Algebra in Sage, Programmieren I
5. **Tag** Datencontainer in Sage, Lineare Abbildungen und Matrizen
6. **Tag** Folgen und Reihen

# Inhalt der Vorlesung

1. **Tag** Organisatorisches, Aufbau von Sage, Streifzug durch Sage
2. **Tag** Grundlagen Sage (grundlegende Datentypen, Ausdrücke), Lösen von Gleichungen, Symbolisches Rechnen
3. **Tag** Mengen, natürliche, rationale, reelle und komplexe Zahlen, Gleitkommazahlen
4. **Tag** Vektoren und Matrizen, Lineare Algebra in Sage, Programmieren I
5. **Tag** Datencontainer in Sage, Lineare Abbildungen und Matrizen
6. **Tag** Folgen und Reihen
7. **Tag** Reelle Funktionen, Grafiken

# Inhalt der Vorlesung

1. **Tag** Organisatorisches, Aufbau von Sage, Streifzug durch Sage
2. **Tag** Grundlagen Sage (grundlegende Datentypen, Ausdrücke), Lösen von Gleichungen, Symbolisches Rechnen
3. **Tag** Mengen, natürliche, rationale, reelle und komplexe Zahlen, Gleitkommazahlen
4. **Tag** Vektoren und Matrizen, Lineare Algebra in Sage, Programmieren I
5. **Tag** Datencontainer in Sage, Lineare Abbildungen und Matrizen
6. **Tag** Folgen und Reihen
7. **Tag** Reelle Funktionen, Grafiken
8. **Tag** Differenzial- und Integralrechnung

# Inhalt der Vorlesung

1. **Tag** Organisatorisches, Aufbau von Sage, Streifzug durch Sage
2. **Tag** Grundlagen Sage (grundlegende Datentypen, Ausdrücke), Lösen von Gleichungen, Symbolisches Rechnen
3. **Tag** Mengen, natürliche, rationale, reelle und komplexe Zahlen, Gleitkommazahlen
4. **Tag** Vektoren und Matrizen, Lineare Algebra in Sage, Programmieren I
5. **Tag** Datencontainer in Sage, Lineare Abbildungen und Matrizen
6. **Tag** Folgen und Reihen
7. **Tag** Reelle Funktionen, Grafiken
8. **Tag** Differenzial- und Integralrechnung
9. **Tag** Grundlagen der Programmierung, Zeichenketten (Strings)

# Inhalt der Vorlesung

1. **Tag** Organisatorisches, Aufbau von Sage, Streifzug durch Sage
2. **Tag** Grundlagen Sage (grundlegende Datentypen, Ausdrücke), Lösen von Gleichungen, Symbolisches Rechnen
3. **Tag** Mengen, natürliche, rationale, reelle und komplexe Zahlen, Gleitkommazahlen
4. **Tag** Vektoren und Matrizen, Lineare Algebra in Sage, Programmieren I
5. **Tag** Datencontainer in Sage, Lineare Abbildungen und Matrizen
6. **Tag** Folgen und Reihen
7. **Tag** Reelle Funktionen, Grafiken
8. **Tag** Differenzial- und Integralrechnung
9. **Tag** Grundlagen der Programmierung, Zeichenketten (Strings)
10. **Tag** Fragestunde

## 1 Was ist Sage?

## 2 Streifzug durch Sage

- Eine Kurvendiskussion
- Symbolisches Rechnen
- Etwas AGLA
- Etwas Programmieren
- Etwas Zahlentheorie

## 3 Nützliches und Hilfe



## 1 Was ist Sage?

## 2 Streifzug durch Sage

- Eine Kurvendiskussion
- Symbolisches Rechnen
- Etwas AGLA
- Etwas Programmieren
- Etwas Zahlentheorie

## 3 Nützliches und Hilfe

# Computeralgebra-Systeme

## Computeralgebra

beschäftigt sich mit **exakten** Berechnungen von mathematischen Objekten

## Mathematische Objekte

Natürliche Zahlen, reelle Zahlen, Polynome, Funktionen, Gruppen, Ringe,  
...

## Numerischen Berechnungen

Bei numerischen Rechnungen (z.B. Taschenrechner) benutzt man Zahlen in **Gleitpunktdarstellung**, also i.A. nur Näherungen an die gesuchte Lösung

# Computeralgebra $\neq$ Numerische Berechnung

## Beispiel

Mathematische Objekte	$\pi, \sqrt{2}$
Gleitpunktdarstellung (8 Stellen)	3.1415927, 1.4142136

## Allgemein

Sage	Schnittstelle für Mathematik-Software
LiveMath	Maple
Maxima	Free, GPL, von Sage benutzt
Mathematica	einer der Grossen
Maple	einer der Grossen
Matlab	Für große numerische Rechnungen (inkl. Mupad)
Octave	Für große numerische Rechnungen (GPL)
Magma	Spezielle mathematische Rechnungen (z.B. Algebra)
SymPy	Phython-Bibliotheken; als CAS-Verwendbar
SymbolicC++	Bibliotheken zur CA in C++

Überblick:[http://en.wikipedia.org/wiki/Comparison\\_of\\_computer\\_algebra\\_systems](http://en.wikipedia.org/wiki/Comparison_of_computer_algebra_systems)

- Ein Open-source (GPL) Mathematik Software System
- Verfügbar seit 24 Februar 2005
- Alternative zu den 4 M's: Magma, Maple, Mathematica, Matlab
- Basiert auf Python
- Objektorientiert
- Besitzt Frontends für viele externe Software
- Interface im Browser

von Joachim Neubüser (Gründer von GAP):

*You can read Sylow's Theorem and its proof [. . .] and then you can use Sylow's Theorem for the rest of your life free of charge, but for many computer algebra systems license fees have to be paid regularly [. . .]. You press buttons and you get answers in the same way as you get the bright pictures from your television set but you cannot control how they were made in either case.*

*With this situation two of the most basic rules of conduct in mathematics are violated: in mathematics information is passed on free of charge and everything is laid open for checking. Not applying these rules to computer algebra systems that are made for mathematical research [. . .] means moving in a most undesirable direction. Most important: can we expect somebody to believe a result of a program that he is not allowed to see?*

## Stärken

- Vereinigung von vielen anderen CAS und Libraries unter einer einheitlichen Oberfläche (Maxima, Pari, GAP, R, Magma, ..., wovon die freien grösstenteils bei Sage enthalten sind)

## Stärken

- Vereinigung von vielen anderen CAS und Libraries unter einer einheitlichen Oberfläche (Maxima, Pari, GAP, R, Magma, ..., wovon die freien grösstenteils bei Sage enthalten sind)
- Durch Python angebunden an eine mächtige Skriptsprache



## Stärken

- Vereinigung von vielen anderen CAS und Libraries unter einer einheitlichen Oberfläche (Maxima, Pari, GAP, R, Magma, ..., wovon die freien grösstenteils bei Sage enthalten sind)
- Durch Python angebunden an eine mächtige Skriptsprache
- umfangreiches Hilfesystem

## Stärken

- Vereinigung von vielen anderen CAS und Libraries unter einer einheitlichen Oberfläche (Maxima, Pari, GAP, R, Magma, ..., wovon die freien grösstenteils bei Sage enthalten sind)
- Durch Python angebunden an eine mächtige Skriptsprache
- umfangreiches Hilfesystem
- Viele freie (Unterrichts-)materialien im Internet

## Stärken

- Vereinigung von vielen anderen CAS und Libraries unter einer einheitlichen Oberfläche (Maxima, Pari, GAP, R, Magma, ..., wovon die freien grösstenteils bei Sage enthalten sind)
- Durch Python angebunden an eine mächtige Skriptsprache
- umfangreiches Hilfesystem
- Viele freie (Unterrichts-)materialien im Internet
- Source Code offen und gut dokumentiert

## Stärken

- Vereinigung von vielen anderen CAS und Libraries unter einer einheitlichen Oberfläche (Maxima, Pari, GAP, R, Magma, ..., wovon die freien grösstenteils bei Sage enthalten sind)
- Durch Python angebunden an eine mächtige Skriptsprache
- umfangreiches Hilfesystem
- Viele freie (Unterrichts-)materialien im Internet
- Source Code offen und gut dokumentiert

# Sage - Stärken und Schwächen

## Stärken

- Vereinigung von vielen anderen CAS und Libraries unter einer einheitlichen Oberfläche (Maxima, Pari, GAP, R, Magma, ..., wovon die freien grösstenteils bei Sage enthalten sind)
- Durch Python angebunden an eine mächtige Skriptsprache
- umfangreiches Hilfesystem
- Viele freie (Unterrichts-)materialien im Internet
- Source Code offen und gut dokumentiert

## Schwächen

- Befehlsumfang nicht so mächtig wie bei Maple, Mathematica oder Matlab

# Sage - Stärken und Schwächen

## Stärken

- Vereinigung von vielen anderen CAS und Libraries unter einer einheitlichen Oberfläche (Maxima, Pari, GAP, R, Magma, ..., wovon die freien grösstenteils bei Sage enthalten sind)
- Durch Python angebunden an eine mächtige Skriptsprache
- umfangreiches Hilfesystem
- Viele freie (Unterrichts-)materialien im Internet
- Source Code offen und gut dokumentiert

## Schwächen

- Befehlsumfang nicht so mächtig wie bei Maple, Mathematica oder Matlab
- es fehlt (noch) eine gute Entwicklungsumgebung (in Entwicklung: **Cantor**)

## 1 Was ist Sage?

## 2 Streifzug durch Sage

- Eine Kurvendiskussion
- Symbolisches Rechnen
- Etwas AGLA
- Etwas Programmieren
- Etwas Zahlentheorie

## 3 Nützliches und Hilfe

# Sage als Taschenrechner

Hier einige Beispiele:

```
3+4*10+12
```

55

```
sin(pi)
```

0

```
float(pi)
```

3.14159265358979

```
float(sqrt(2))
```

1.41421356237310



## 1 Was ist Sage?

## 2 Streifzug durch Sage

- Eine Kurvendiskussion
- Symbolisches Rechnen
- Etwas AGLA
- Etwas Programmieren
- Etwas Zahlentheorie

## 3 Nützliches und Hilfe

Betrachte die durch die reelle Zahl  $a$  parametrisierte Funktionenschar:

$$f: x \mapsto \frac{2x^2 - 20x + 42}{x - 1} + a, \quad a \in \mathbb{R}$$

- Eingabe der Funktion

```
var('a')  
f(x) = (2*x^2-20*x +42)/(x-1)+a
```

$$x \mapsto a + 2 \cdot (x^2 - 10x + 21) / (x - 1)$$

- Pol ?

```
f.limit(x=1, dir='minus')
```

$x \rightarrow -\infty$

```
f.limit(x=1, dir='plus')
```

$x \rightarrow +\infty$

- Umformen

```
f.full_simplify()
```

$x \rightarrow ((a - 20)x + 2x^2 - a + 42)/(x - 1)$

# Kurvendiskussion III

- Nullstellen

```
solve(f==0,x)
```

```
[x == -1/4*a - 1/4*sqrt(a^2 - 32*a + 64) + 5, x ==  
-1/4*a + 1/4*sqrt(a^2 - 32*a + 64) + 5]
```

- Berechnen der Ableitung

```
f.differentiate(x)
```

```
x |--> 4*(x - 5)/(x - 1) - 2*(x^2 - 10*x + 21)/(x -  
1)^2
```

# Kurvendiskussion IV

- Extremwerte

```
maxi = solve(f.differentiate(x)==0,x); maxi
```

```
[x == -2*sqrt(3) + 1, x == 2*sqrt(3) + 1]
```

- Lokale Minima und Maxima

```
float( ((f.diff(x)).diff(x))(maxi[0].rhs()) )
```

```
-1.1547005383792501
```

```
float( ((f.diff(x)).diff(x))(maxi[1].rhs()) )
```

```
1.1547005383792515
```

- Verhalten von  $f$  für große  $x$

```
f.limit(x=oo); f.limit(x=-oo)
```

```
x |--> +Infinity
```

```
x |--> -Infinity
```

- Definiere  $f_0, f_1, f_2$

```
f0 = f(x, a=0)
```

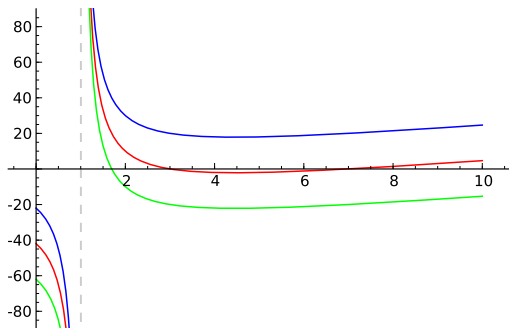
```
f1 = f(x, a=-20)
```

```
f2 = f(x, a=20); f0, f1, f2
```

```
(2*(x^2 - 10*x + 21)/(x - 1), 2*(x^2 - 10*x + 21)/(x  
- 1) - 20, 2*(x^2 - 10*x + 21)/(x - 1) + 20)
```

# Plot

```
p = plot(f0,detect_poles='show',xmin=0, xmax=10,color='red')
p += plot(f1,detect_poles='show',xmin=0, xmax=10,color='green')
p += plot(f2,detect_poles='show',xmin=0, xmax=10,color='blue'); p.show(ymin=-80, ymax=80)
```



# Zusammenfassung

- Deklarieren von Variablen mit `var()`, z.B. `var('a')`
- Definieren von Variablen mit `'='`, z.B. `a=3`
- Definieren von Funktionen mit `'='`, z.B. `f(x) = x^2 - 6*x`
- Symbolisches Rechnen
  - Grenzwertbestimmung: `f.limit(x=1, dir='<plus|minus>')`
  - Vereinfachen: `f.full_simplify()`
  - Bilden von Ableitungen `f.differentiate(x)`
- Lösen von Gleichungen: `solve( f(x)==0, x)`
- Berechnen numerischer Approximationen: `float(f(sqrt(3)+ 4))`
- Plotten einer Funktion: `plot(sin, (0,4))`



## 1 Was ist Sage?

## 2 Streifzug durch Sage

- Eine Kurvendiskussion
- Symbolisches Rechnen
- Etwas AGLA
- Etwas Programmieren
- Etwas Zahlentheorie

## 3 Nützliches und Hilfe

- Integrieren von  $\int_0^\infty x^4 e^{-x} dx$

```
integrate(x^4*exp(-x),x,0,oo)
```

24

- Stammfunktion von  $\frac{1+\sin(x)}{1+\cos(x)}$

```
f(x) = (1+sin(x))/(1+cos(x))  
g = f.integrate(x)  
g.full_simplify()
```

$x \mapsto -((\cos(x) + 1) \cdot \log(\cos(x) + 1) - \sin(x)) / (\cos(x) + 1)$

# Symbolisches Rechnen II

- Faktorisieren und Ausmultiplizieren

```
expand((x-1)*(x-2)*(x-3)*(x-4))
```

$$x^4 - 10x^3 + 35x^2 - 50x + 24$$

```
factor(_)
```

$$(x - 4)*(x - 3)*(x - 2)*(x - 1)$$

- Sortieren eines Ausdrucks bezüglich einer Unbekannten

```
var('b,a')  
g = x^2+2*x+b*x^2+sin(x)+a*x  
g.collect(x)
```

$$(b + 1)x^2 + (a + 2)x + \sin(x)$$

- Partialbruchzerlegung

```
g = x^2/(x^2-1)
g.partial_fraction()
```

$$1/2/(x-1) - 1/2/(x+1) + 1$$

- Vereinfachen von Ausdrücken  $\left(\frac{e^x-1}{e^{(1/2)x}+1}\right)$

```
g = (exp(x)-1)/(exp(x/2)+1)
g.simplify_radical()
```

$$e^{(1/2)x} - 1$$

## 1 Was ist Sage?

## 2 Streifzug durch Sage

- Eine Kurvendiskussion
- Symbolisches Rechnen
- Etwas AGLA
- Etwas Programmieren
- Etwas Zahlentheorie

## 3 Nützliches und Hilfe

Berechnen des Schnittpunkts der Ebene

$$E: \vec{x} = \begin{pmatrix} 2 \\ 1 \\ -1 \end{pmatrix} + l \begin{pmatrix} 1 \\ -1 \\ -1 \end{pmatrix} + m \begin{pmatrix} -3 \\ 1 \\ 4 \end{pmatrix}, \quad l, m \in \mathbb{R}$$

mit der Geraden

$$g: \vec{x} = \begin{pmatrix} 3 \\ 0 \\ 1 \end{pmatrix} + k \begin{pmatrix} 4 \\ -1 \\ 2 \end{pmatrix}, \quad k \in \mathbb{R}$$

# Grafische Darstellung

```
var('l,m'); E1 = 2+l-3*m; E2 = 1-l+m; E3 = -1-l+4*m
```

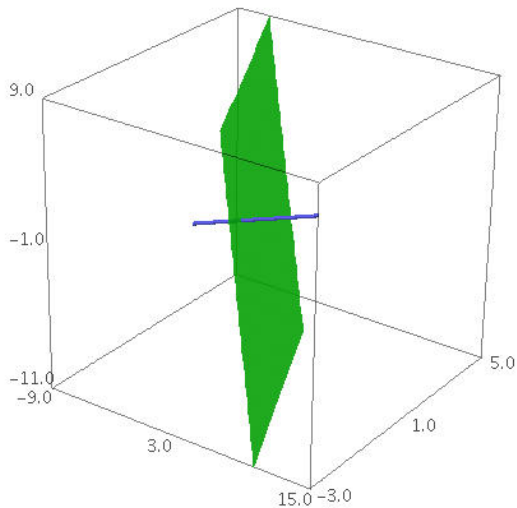
```
p = parametric_plot3d([E1,E2,E3],(l,-2,2),(m,-2,2), color  
    ='green', opacity=0.8)
```

```
var('k'); g1 = 3+4*k; g2 = -k; g3 = 1+2*k
```

```
p += parametric_plot3d( (g1,g2,g3), (k, -3, 3),thickness=  
    '3' )
```

```
p.show()
```

# Grafische Darstellung





Gleichsetzen ergibt:

$$\begin{pmatrix} 2 \\ 1 \\ -1 \end{pmatrix} + l \begin{pmatrix} 1 \\ -1 \\ -1 \end{pmatrix} + m \begin{pmatrix} -3 \\ 1 \\ 4 \end{pmatrix} = \begin{pmatrix} 3 \\ 0 \\ 1 \end{pmatrix} + k \begin{pmatrix} 4 \\ -1 \\ 2 \end{pmatrix}$$

oder

$$\underbrace{\begin{pmatrix} 1 & -3 & -4 \\ -1 & 1 & 1 \\ -1 & 4 & -2 \end{pmatrix}}_{=: A} \underbrace{\begin{pmatrix} l \\ m \\ k \end{pmatrix}}_{=: L} = \underbrace{\begin{pmatrix} 1 \\ -1 \\ 2 \end{pmatrix}}_{=: b}$$

oder  $AL = b$ .

# Definieren und Lösen des LGS

- Definieren der Matrix  $A$

```
A = matrix([[1,-3,-4],[-1,1,1],[-1,4,-2]]); A
```

```
[ 1 -3 -4]  
[-1  1  1]  
[-1  4 -2]
```

- Definieren des Vektors  $b$

```
b = vector([1,-1,2])
```

- Lösen von  $A L = b$

```
A.solve_right(b)
```

oder

```
A\b
```

ergibt

$(6/5, 3/5, -2/5)$

- Einsetzen in die Geradengleichung

```
x_s = matrix([g1,g2,g3]).subs(k=L[2]); x_s
```

$[7/5 \ 2/5 \ 1/5]$

- Matrizenoperationen

```
B = matrix([[1,0,0],[0,1,1],[1,1,1]])  
A*B; A-B; A+B
```

```
[-3 -7 -7] [ 0 -3 -4] [ 2 -3 -4]  
[ 0  2  2] [-1  0  0] [-1  2  2]  
[-3  2  2] [-2  3 -3] [ 0  5 -1]
```

- Berechnen der Inversen (mit Probe)

```
A(-1), A*A(-1)
```

```
[ -2/5 -22/15  1/15]  
[ -1/5  -2/5   1/5]  
[ -1/5  -1/15 -2/15]
```

```
[1 0 0]  
[0 1 0]  
[0 0 1]
```

## 1 Was ist Sage?

## 2 Streifzug durch Sage

- Eine Kurvendiskussion
- Symbolisches Rechnen
- Etwas AGLA
- Etwas Programmieren
- Etwas Zahlentheorie

## 3 Nützliches und Hilfe

- eine einzeilige Funktion kann man wie folgt definieren:

```
def <name>(<Argumente>) : return <Rueckgabe>
```

- Beispiel:

```
def fd(ex) : return diff(ex)  
fd(x^2)
```

2\*x

# Listen und Tuple

- Eine Liste ist in Sage (und Python) mit `[...]` gekennzeichnet
- Ein Tuple ist in Sage (und Python) mit `(...)` gekennzeichnet
- Beispiel:

```
liste = [21,22,24,23]
tuple = (liste[0], liste[2])
tuple, tuple[0]
```

`((21, 24), 21)`

```
liste.sort(); liste
```

`[21, 22, 23, 24]`

# Einfache Schleifen und Abfragen

- mit dem Konstrukt [`<expr(var)> for <var> in <range|liste>`] kann man einfache Schleifen konstruieren..

```
[m^2 for m in [1..5] ]
```

```
[1, 4, 9, 16, 25]
```

- .. und diese mit Abfragen kombinieren.

```
[m^2 for m in [1..5] if m%2==0]
```

```
[4, 16]
```



## 1 Was ist Sage?

## 2 Streifzug durch Sage

- Eine Kurvendiskussion
- Symbolisches Rechnen
- Etwas AGLA
- Etwas Programmieren
- Etwas Zahlentheorie

## 3 Nützliches und Hilfe

# Etwas Zahlentheorie I

Fermatsche Primzahlen:  $F_n = 2^{2^n} + 1$ . Finden Sie die kleinste Zahl  $F_n$ , die keine Primzahl ist!

```
def F(n): return 2^(2^n)+1  
[[F(m),is_prime(F(m))] for m in range(1,6)]
```

```
[[5, True], [17, True], [257, True], [65537, True],  
 [4294967297, False]]
```

```
divisors(int(F(5)))
```

```
[1, 641, 6700417, 4294967297]
```

# Etwas Zahlentheorie II

- Eine Liste der ersten Primzahlen bis 100

```
menge = range(1,101)
[m for m in menge if is_prime(m)]
oder filter(is_prime,menge)
```

[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43,  
47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]

- Mersenne-Primzahlen  $2^p - 1$ ,  $p$  Primzahl. Bestimmen der ersten Mersenne Primzahlen im Bereich  $\leq 200$ .

```
menge = range(1,201)
primes = [m for m in menge if is_prime(m)]
[2^m-1 for m in primes if is_prime(2^m-1)]
```

[3, 7, 31, 127, 8191, 131071, 524287, 2147483647,  
2305843009213693951,  
618970019642690137449562111,  
162259276829213363391578010288127,  
170141183460469231731687303715884105727]

# Etwas Zahlentheorie III

Wir geben für die natürlichen Zahlen  $\leq 1000$  an, wieviele Zahlen  $1, 2, 3, \dots$  Teiler haben.

```
menge = range(1,1001)
liste = [number_of_divisors(int(m)) for m in menge]
[(i,len([m for m in liste if m==i]))for i in range(1,51)]
```

```
[(1, 1), (2, 168), (3, 11), (4, 292), (5, 3), (6, 110),
 (7, 2), (8,
180), (9, 8), (10, 22), (11, 0), (12, 97), (13, 0), (14,
5), (15, 4),
...]
```

Teiler der Zahl 840:

```
divisors(840)
```

## 1 Was ist Sage?

## 2 Streifzug durch Sage

- Eine Kurvendiskussion
- Symbolisches Rechnen
- Etwas AGLA
- Etwas Programmieren
- Etwas Zahlentheorie

## 3 Nützliches und Hilfe

- Mehrere Befehle in einer Zeile durch ; trennen.
- Bei Eingaben, die über mehrere Zeilen gehen, kann ein Zeilenumbruch durch `<ENTER>` erreicht werden.
- Das Auswerten eines Blocks erfolgt mit `<SHIFT>+<ENTER>`.
- Ein neues Eingabefeld erhält man durch klicken auf den blauen, horizontalen Balken

- `_` referenziert die letzte Ausgabe.
- `[a..b]` oder `range(a,b+1)` erzeugt eine Liste von ganzen Zahlen von `a` bis `b`
- Löschen aller eigenen Variablen und Zurücksetzen auf den Anfangsstatus: `reset()`
- Das Feld aktivieren von **Typeset** lässt alle Ausgaben von  $\text{\LaTeX}$  rendern.
- Html- und/oder  $\text{\LaTeX}$ -Dokumentation: `<SHIFT>+<KLICK>` auf den blauen Balken
- Publish: Im Notebook kann durch klicken des **Publish**-Reiters das Notebook für alle offen gelegt werden.

- **Autocompletion** : mit der <TAB>-Taste erhält man alle möglichen Funktions- und/oder Variablen-Namen im gegebenen Kontext. Dies gilt insbesondere auch für Objektfunktionen (`object.function()`)
- **<command>?** : gibt ausführliche Hilfe zu `command` an.
- **help(<command>)** : öffnet ein Hilfefenster zu `command`.
- online Dokumentation:
  - Sage: <http://www.sagemath.org/doc/index.html>
  - Python: <http://docs.python.org/>