Einführung in Sage
Kurzreferenz

Überlebensregeln

- Mehrere Befehle in einer Zeile 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

Nützliches

- `_` refenziert die letzte Ausgabe (Warnung: unübersichtlich!).
- Löschen aller eigenen Variablen und Zurücksetzen auf den Anfangsstatus: `reset()`
- Aktivieren des Feldes *Typeset* lässt alle Ausgaben von LaTeX rendern.
- Dokumentation mit HTML und LaTeX-Formeln:`<SHIFT>+<KLICK>` auf den blauen Balken startet WYSIWYG-Editor.
- Publish: Im Notebook kann durch klicken des *Publish*-Reiters das Notebook für alle offen gelegt werden.

Hilfefunktionen

- 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.
- `<command>??` : gibt den source-code des `command` an.
- `help(<command>)` : öffnet ein Hilfefenster zu `command`.
- `search_doc('<begriff>')` : Sucht in der Hilfe nach <begriff>.
- Dokumentation:
  - Sage (lokal): `file:///usr/local/sage_4.7.2/devel/sage-main/doc/output/html/en/index.html`
  - Sage: `http://www.sagemath.org/doc/index.html`
  - Python: `http://docs.python.org/`

Datentypen Liste/Tuple- list(), tuple()

- Konstruktion

```
liste = [a,b,c,...]
liste = list(<sequence>)
```

Dictionaries- dictionaries

- Deklarieren eines Dictionaries:

```
d = {<Index1>:<Wert1>,<Index2>:<Wert2>,...}
```

- Zugriff auf Index:

```
d[<Index>]
```

map() und map_threaded()- map()
(Rekursive) Auswertung der Funktion auf das Objekt

```
map_threaded(<Funktion>,<Objekt>)
```

filter: filtert nach Wahrheitswert der übergebenen Funktion- filter()

```
filter(<funktion>,<menge oder Liste>)
```

Zahlen

- Körper/Gruppen:
  QQ (rationale Zahlen), ZZ (ganze Zahlen), RR (Reelle Zahlen)
- Wichtige Funktionen

`abs`	Absolutbetrag
`ceil`	Aufrunden
`floor`	Abrunden
`round`	Runden
`sqrt`	Wurzel
`digits`	Anzahl Stellen

- Annahmen- assume()

```
assume(<Annahme>
```

Matrix- matrix()

- Deklaration

```
matrix(<Koerper>,[<n>,<m>,[a11,...],[a21
    ,..],..])
```

Vektor- vector()

- Deklaration

```
vector([v1,v2,..])
```

Funktionen

- Mathematische Funktionen (Ausdrücke)

```
<f>(<x>,<y>,...) = <expr>
```

- einzeilige Deklaration- def

```
def <name> (<arg1>,<arg2>,..): return <
    Rueckgabewert>
```

- normale Deklaration

```
def <Name><(<arg1>,<arg2>,..)>:
    <Code-Block>
    return <Rueckgabewert>
```

Schleifen

- einzeilige for-Schleife- for

```
[<expr(x)> for <x> in <liste> if <bedingung
    >]
```

- for-Schleife

```
for <x> in <liste>:
    <Code-Block>
```

- while- Schleife- while()

```
while <expression> :
    <Code-block>
```

Abfragen- if

```
if <boolean expr>:
    <Code-Block>
```

Grafik- plot() / plot3d()

- 2D/3D Plot

```
plot(f2,(x,a,b),optionen,...)
plot3d(f3,(x,a,b),(y,c,d),optionen,...)
```

Differentiation

- Ableitungen- diff()

```
diff(<Ausdruck>,<var1>,<var2>,<var3>,...)
diff(<Ausdruck>,<var>,<anzahl>)
```

Taylorformel

- Taylorformel- taylor()

```
taylor(<funktion>,<var>,<stelle>,<grad>)
```

Integrale

- bestimmte/unbestimmte Integrale- integrate()

```
integrate(<funktion>,<var>[,<ug>,<og>])
```

Strings/Zeichenketten und Ausgabe- string

- Deklaration

```
string = '<Inhalt>'
```

- Typecast- str()

```
str(<vorher kein String>)
```

- print

```
print ("Text %<format> und %<format> \dots "
     % (x,y,\dots))
```

  wichtigsten Formate:
  - %i : integer
  - %f : float
  - %s : string