

Sage Tutorial

Im Laufe des Semesters wird es für einige der Schnittstellenaufgaben notwendig sein, eine Mathematiksoftware, ein sogenanntes Computeralgebrasystem (abgekürzt CAS, siehe auch <http://de.wikipedia.org/wiki/Computeralgebrasystem>), zu benutzen. Ziel dieser Aufgabe ist es, euch mit “Sage” (www.sagemath.org), einem solchen CAS, vertraut zu machen. Ihr könnt die kommenden Aufgaben mit einem CAS eurer Wahl bearbeiten, solange die Tutoren eure Aufgaben korrigieren können, ohne zusätzliche Software zu installieren. Nur die Aufgaben auf diesem Zettel können ausschließlich mit Sage bearbeitet werden. Wir empfehlen jedoch, auch für die kommenden Aufgaben Sage zu benutzen, da sie zur Bearbeitung mit Sage entworfen werden und wir Euch bei Fragen und Problemen mit Sage behilflich sein können.

Unsere Wahl fiel aus folgenden Gründen auf Sage:

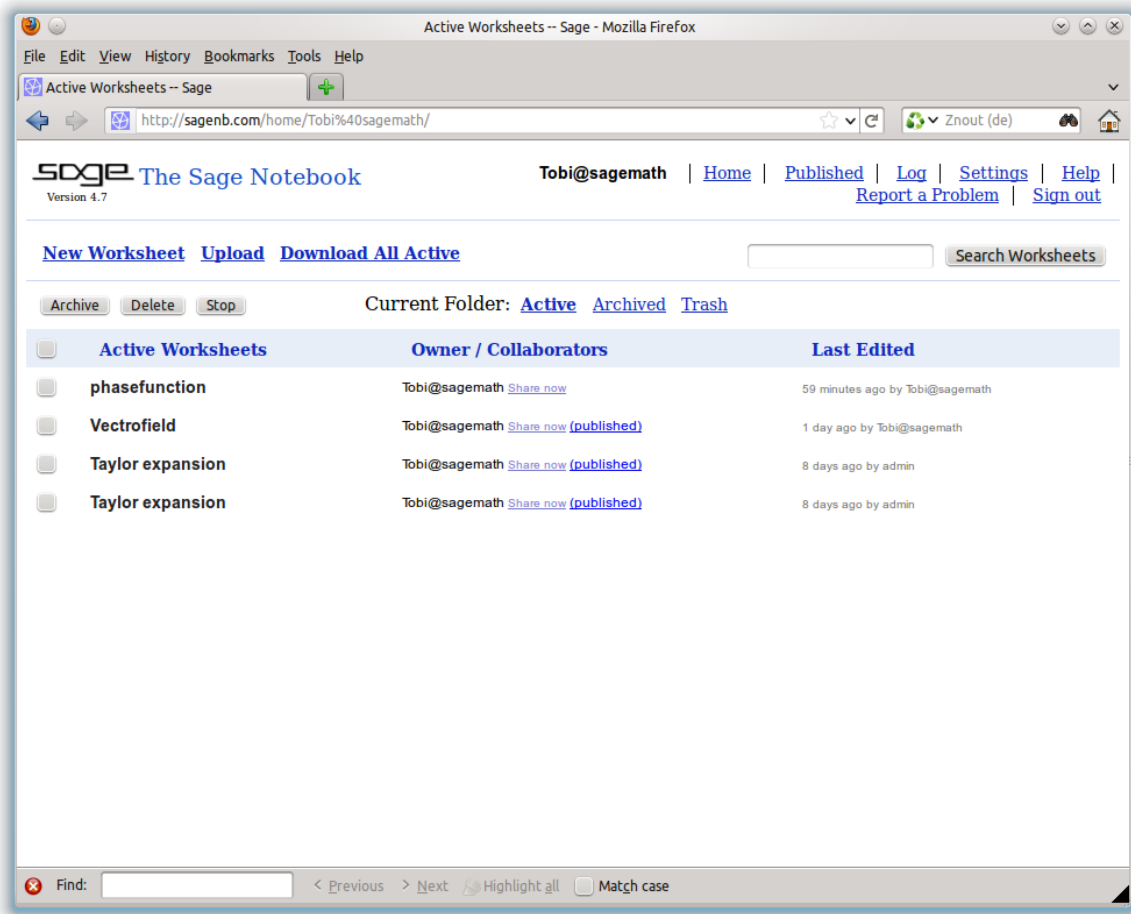
- Sage ist zwar noch eine sehr junge Software, besitzt jedoch schon einen großen Funktionsumfang und ist durchaus auch für professionelle Aufgaben in der Forschung geeignet. Es nutzt und bündelt die Fähigkeiten zahlreicher schon lang etablierter freier Software Pakete wie Maxima, Singular, GAP, numpy und scipy und viele andere.
- Sage ist freie Software, die Installation und Benutzung kostet damit im Gegensatz zu kommerziellen Produkten wie Mathematica oder Maple keine horrenden Lizenzgebühren. Das erst macht auch den Einsatz in der Schule möglich. Sage ist jedoch nicht nur kostenlos, sondern wirklich Allgemeingut und kann nicht verkauft oder irgendwann doch kostenpflichtig werden (dass solche Überlegungen für den Schulunterricht von Bedeutung sein können, zeigen die Leidenserfahrungen eines Lehrers mit MuPad: <http://haftendorn.uni-lueneburg.de/computer/cas/cas.htm>)
- Sage kann auf allen gängigen Betriebssystemen Linux, OS X, und Windows installiert werden. Es kann jedoch auch im vollen Funktionsumfang ohne jegliche Installation über sagenb.org genutzt werden. Einzige Voraussetzung ist ein geeigneter Webbrowser (empfohlen ist eine aktuelle firefox Version, es funktionieren aber wahrscheinlich die meisten gängigen Browser). Auch diese Eigenschaft macht Sage für die Nutzung in der Schule interessant.

Ziel des folgenden Tutorials ist es, Leuten, die noch nie mit einem CAS gearbeitet haben und keine Programmierkenntnisse besitzen, mit Sage vertraut zu machen. In das Tutorial eingebaut sind immer wieder kleine Aufgaben, die abgegeben werden sollen und für die es die Punkte gibt.

1 Erste Schritte

Um Sage online zu nutzen, muss man sich auf sagenb.com zuerst einen eigenen Account anlegen. Hierfür muss man nach einem Klick auf “Sign up for a new Sage Notebook account” jedoch nur noch einen Benutzernamen und ein Passwort auswählen. Nachdem

login erscheint eine Übersicht über alle Dateien (sogenannte “Worksheets”), die bisher mit diesem Account erstellt wurden. Eine neue Datei wird mit einem Klick auf “New Worksheet” erstellt.



Ein Sage Worksheet besteht aus sogenannten Zellen, in die eine oder mehrere Anweisungen eingegeben werden können. Zum Beispiel kann man Sage als ganz normalen Taschenrechner nutzen. Hierzu gibt man einfach den zu berechnenden Ausdruck in die Zelle ein und drückt gleichzeitig <shift> und <enter>.



Das Ergebnis erscheint dann unter der Zelle. Es ist auch möglich, mehrere Anweisungen in eine Zelle einzugeben, wie folgendes Beispiel zeigt:



Am Ende jeder Zeile wurde <enter> eingegeben, um in die nächste Zeile der Zelle zu gelangen und die nächste Anweisung einzugeben. Nach der letzten Zeile wurde mit <shift> <enter> der Befehl gegeben, alle Anweisungen in der Zelle nacheinander auszuführen. Wie man am obigen Beispiel erkennt, wird standardmäßig nur das Ergebnis der letzten Anweisung ausgegeben. Möchte man, dass

auch andere Ergebnisse ausgegeben werden, so schafft der Befehl “print” Abhilfe:

```
a=10
b=3
print a-b
a^b
```

```
7
1000
```

Selbstverständlich beherrscht Sage weit mehr als nur die Grundrechenarten. Über gewisse Funktionen kann man fast alle erdenklichen Rechenoperationen wie Differenzieren, Integrieren, Invertieren von Matrizen oder, wie folgendes Beispiel zeigt, Primfaktorzerlegungen ausführen:

```
factor(2063123)
```

```
23 * 271 * 331
```

Über den Zellen befinden sich zahlreiche Schaltflächen zum speichern, kopieren, laden usw. Die wichtigste Schaltfläche ist wohl “Save & quit”. Ein Klick hierauf speichert das aktuelle Worksheet und bringt einen zurück zur Übersicht über alle bisher behandelten Worksheets. Auf eine weitere hilfreiche Funktion für den Übungsbetrieb kann man über die Schaltfläche “Publish” zugreifen. Nach der Bestätigung des darauffolgenden Dialogs wird das aktuelle Worksheet veröffentlicht und man erhält eine URL über die das Worksheet aus dem Internet abrufbar ist. So können zum Beispiel die Lösungen von Übungsaufgaben dem Tutor ohne jegliches Drucken einfach durch die Weitergabe der URL übermittelt werden.

Aufgabe 1 (2 Punkte).

- Erstelle dir einen Account auf sagenb.org.
- Erzeuge ein neues Worksheet mit dem Namen “Primfaktorzerlegung” und berechne die Primfaktorzerlegung deiner Matrikelnummer (oder deines Geburtsdatums, oder einer sonstigen Langen Zahl, falls du nicht möchtest, dass deine Matrikelnummer online erscheint).
- Speichere das Worksheet und veröffentliche es.
- Notiere als Lösung auf dem Aufgabenblatt die URL des Worksheets.

2 Die Sage Tutorials - Plotting

Sage verfügt über ein sehr ausführliches Tutorial, in dem viele Funktionen von Sage erklärt werden. Dies ist entweder direkt aus dem Sage Notebook mittels Klick auf “Help→Tutorial” (das bedeutet klicke erst auf die Schaltfläche “Help” und auf der nächsten Seite auf die Schaltfläche “Tutorial”) möglich, oder über die Sage-Homepage sagemath.org. Dort befindet sich auch eine deutsche Version des Tutorials. Es ist jedoch empfehlenswert, in die deutsche Version nur rein zu schauen, wenn man in der englischen etwas nicht versteht. Wie in der ganzen Wissenschaftswelt ist auch die Sage Dokumentation und Hilfe fast ausschließlich auf Englisch vorhanden, sodass man sich am besten von Anfang an daran gewöhnt.

Beim Umgang mit Computerprogrammen ist es heutzutage eine unverzichtbare Schlüsselqualifikation, dass man sich mithilfe der im Internet verfügbaren Tutorials,

Dokumentationen oder dem durch das “Ergooglen” von Funktionen und Fehlermeldungen die benötigten Kenntnisse über das Programm aneignet. Die folgenden Aufgaben zielen darauf ab solche, Fähigkeiten zu erwerben und gleichzeitig weitere Funktionen von Sage kennen zu lernen.

Aufgabe 2 (4 Punkte). Lese das Kapitel “Plotting - Two-Dimensional Plots” durch und bearbeite im Anschluss folgende Aufgaben in jeweils einer Zelle:

- a) Plotte einen Kreis mit Radius 2 um den Punkt (1,1)
- b) Plotte einen grünen Kreis (Radius=1 Mittelpunkt=(0,0)) im “Aspect ratio”=1, mit und ohne Achsen.
- c) Plotte einen gefüllten roten Kreis und einen nicht gefüllten blauen Kreis (Radius und Mittelpunkte sollen frei gewählt werden) in das gleiche Koordinatensystem
- d) Plotte die Funktion $f(x) = x^3$ im Bereich von $-2 < x < 4$.
- e) Plotte die Funktionen $f(x) = e^{-\frac{1}{2}x} \cos(2\pi x)$ und $g(x) = e^{-\frac{1}{2}x}$ im Bereich $0 < x < 6$ in unterschiedlichen Farben in ein Koordinatensystem.

Bemerkung Die Funktion “save” scheint zumindest bei mir in der Sage online Version nicht zu funktionieren. Man kann die erzeugten Bilder jedoch ganze einfach mittels <Rechtsklick>→“Save Image as” auf dem eigenen PC speichern und weiterverwenden (für Hausarbeiten, Abschlussarbeiten...). Wenn man intensiver mit Sage arbeitet empfiehlt es sich aber sicher, Sage auch auf dem eigenen PC zu installieren.

3 Sage’s Reference Manual - Fortgeschritteneres Plotten

Das Sage Tutorial hat zum Ziel, eine Einführung in Sage zu liefern und den Leser mit einer bestimmten Auswahl an Funktionen bekannt zu machen. Speziellere Funktionen werden nicht erwähnt, diese müssen im Sage Reference Manual nachgeschlagen werden. Das Reference Manual findet man entweder auf sagemath.org (→Help/Documentation→Reference Manual) oder direkt aus dem Sage Notebook (→Help → Reference Manual).

Wir wollen uns mit Hilfe des Reference Manuals weitere Kenntnisse über das Plotten aneigenen und suchen nach der Möglichkeit die Achsen der Koordinatensysteme zu beschriften. Um die entsprechende Funktion zu finden öffnen wir das Kapitel “2D Plotting”. Auf den ersten Blick ist man vom Umfang des Kapitels wahrscheinlich erschlagen. Sage bietet beim Plotten so viele Funktionen, dass es nicht sinnvoll ist, das ganze Kapitel von vorne durchzulesen, bis man die richtige Funktion gefunden hat. Es ist daher wichtig, dass man lernt, in solchen Manuals schnell die gewünschte Funktion zu finden. Eine Hilfe hierbei kann es sein, mit der Suchoption des Internetbrowsers den Text gezielt nach bestimmten Wörtern zu durchsuchen. In unserem Fall suchen wir eine Funktion zum Anbringen von Beschriftungen (engl. labels) an den Achsen. Es empfiehlt sich also zum Beispiel den Text nach dem Begriff “label” zu durchsuchen (in Firefox geht das zum Beispiel mit der Tastenkombination <Strg>+<F>). Nach einigen nutzlosen Ergebnissen findet man die Funktion.

We set labels, since otherwise we won't see anything.

```
p.axes_labels(['$x$ axis', '$y$ axis'])
```

evaluate

In the plot below, notice that the labels are red:

```
p
```

axes_labels(*l=None*)

Set the axes labels.

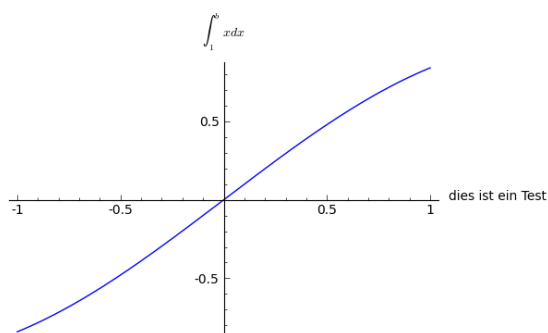
INPUT:

- *l* - (default: None) a list of two strings or None

OUTPUT: a 2-tuple of strings

Neben der Beschreibung der Funktion sind vor allem die Beispiele hilfreich. Um sich mit der Funktion vertraut zu machen, empfiehlt es sich die Beispiele einmal testweise in das Sage Notebook zu kopieren und damit ein bisschen herumzuxperimentieren und beispielsweise ein paar Parameter zu tauschen. Wer schon einmal mit Latex gearbeitet hat, könnte zum Beispiel auf die Idee kommen, dass zwischen zwei $\$$ -Zeichen Latex-Code für mathematische Formeln eingegeben werden kann. Auch wenn es an dieser Stelle im Text nicht erwähnt wird, zeigt ein kurzer Test, dass dies tatsächlich funktioniert:

```
p = plot(sin, (-1,1))
p.axes_labels(['dies ist ein Test', '$\int_1^{\infty} x dx$'])
show(p)
```



Die folgende Aufgabe soll die Nutzung des Reference Manuals üben. Auch wenn es natürlich ein Leichtes ist, die benötigten Befehle von anderen Leuten abzuschreiben und nicht selbst zu suchen, so ermutigen wir doch jeden, die Befehle selbst zu suchen. Dies ist das eigentliche Lernziel der Aufgabe.

Aufgabe 3 (4 Punkte).

- Suche die Funktionen zum Ändern der Label-Farbe und zum Hinzufügen einer Legende (Tipp engl. "legend"). Plote die beiden Funktionen aus Aufgabe 2 e) erneut und fügt eine Legende mit den Formeln für die Funktionen hinzu (Wer Latex beherrscht, darf das gerne mittels Latex machen, ansonsten reicht aber auch normaler Text). Beschrifte außerdem die beiden Achsen sinnvoll in einer Farbe deiner Wahl. Positioniere die Legende für den Plot unten rechts und füge einen Schatten hinzu.
- Finde heraus, ob es eine Funktion gibt, um das Maximum einer Funktion in einem vorgegebenen Intervall zu finden (Tipp: Wenn man nicht genau weiß, in welchem Kapitel des Reference Manuals man suchen soll, kann es auch hilfreich sein den Befehl über Google zu suchen). Teste den Befehl an einem Beispiel deiner Wahl.

- c) Finde heraus, wie man animierte Grafiken erstellt und erstelle eine Animation eines springenden Kreises.

4 Python – Schleifen und Listen

Jedes Computeralgebrasystem ist im Prinzip auch eine Programmiersprache, d.h. man kann in ihr Variablen und Arrays definieren und es existieren sogenannte Kontrollstrukturen wie if-Abfragen und for-Schleifen. Oft haben die CAS ihre eigene Sprache mit eigenem Syntax. Sage benutzt hingegen Python, eine Programmiersprache, die immer häufiger in der Forschung (z.B. Computational Physics) und der Softwareentwicklung eingesetzt wird. Wer sich also richtig in Sage einarbeitet, lernt gleichzeitig noch eine geläufige Programmiersprache und wer Python schon kennt, kann seine Kenntnisse in Sage direkt anwenden.

Dieses Kapitel ist in keinem Sinne eine Einführung in Python oder ein Einstieg in die Programmierung. Wir werden lediglich zwei Komponenten von Python kennen lernen, die zur Bearbeitung späterer Aufgaben hilfreich sein werden: Listen und for-Schleifen. Wer Python richtig lernen will, der sei auf eines der zahlreichen hervorragenden Python Tutorials im Internet verwiesen (siehe z.B. [1]).

4.1 Listen

Eine Liste ist ein Datentyp, in dem man verschiedene Objekte zusammenfassen kann. Listen werden in Python mit eckigen Klammern [] gekennzeichnet. Eine leere Liste lässt sich wie folgt leicht erzeugen:

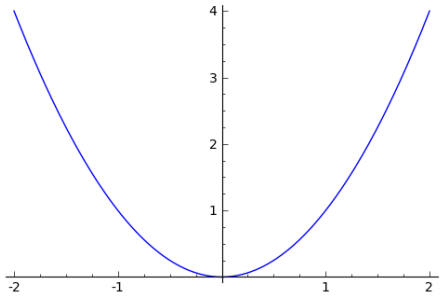
```
a=[]
print a
[]
```

Wie die `print` Ausgabe zeigt, enthält die so erzeugte Liste noch keine Einträge. Diese können mittels der Funktion `append()` ans Ende der Liste angehängt werden.

```
a.append(3)
a.append('Text')
print a
[3, 'Text']
```

Im Vergleich zu den Arrays anderer Programmiersprachen wie C++ sind die Listen in Python sehr flexibel. Die Gesamtanzahl der Elemente einer Liste muss zu Beginn nicht festgelegt werden und in einer Liste können ganz verschiedene Objekte enthalten sein. Auch Objekte, die deutlich komplizierter sind als die obigen Beispiele einer Zahl und eines Textes, können an eine Liste angehängt werden. Zum Beispiel die Grafik Objekte, die von der `plot()` Funktion zurückgegeben werden:

```
a.append(plot(x^2, (x, -2, 2)))
print a
[3, 'Text',
 ]
```



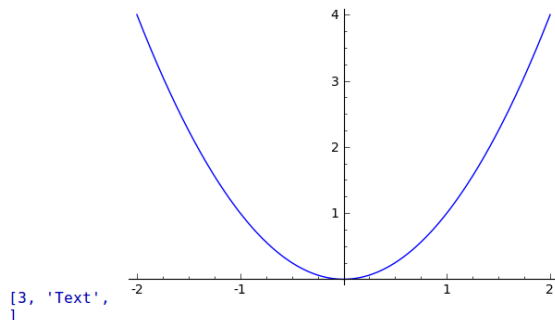
Auf die einzelnen Elemente einer Liste kann man mit den eckigen Klammern zugreifen. So gibt `a[2]` das Element an Position 2 in der Liste zurück. Zu beachten ist hierbei, dass, wie in der Informatik allgemein üblich, bei 0 angefangen wird zu zählen. Das erste Objekt in der Liste wird also mit `a[0]` angesprochen, das zweite mit `a[1]` usw.

```
print a[0]
print a[1]

3
Text
```

In Python können jedoch nicht nur leer Listen erzeugt werden, die dann nachträglich mit `append()` aufgefüllt werden müssen, die Listen können schon sofort mit Inhalt erzeugt werden. Auf diese Weise kann die obige Liste direkt in einer Zeile erzeugt werden.

```
b=[3, 'Text', plot(x^2, (x, -2, 2))]
print b
```



Neben dieser Möglichkeit konkrete Listen zu erstellen, gibt es auch noch die Möglichkeit automatisierte Listen zu erstellen, von denen die beiden folgenden für uns hilfreich sein werden:

```
c=range(10)
d=srange(-1,1,0.2)
print 'c='
print c
print 'd='
print d

c=
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
d=
[-1.0000000000000000, -0.8000000000000000, -0.6000000000000000,
-0.4000000000000000, -0.2000000000000000, -5.55111512312578e-17,
0.2000000000000000, 0.4000000000000000, 0.6000000000000000,
0.8000000000000000]
```

Die Funktion `range(n)` erzeugt also eine Liste mit n aufeinanderfolgenden ganzen Zahlen und mit der Funktion `srange` können Listen mit äquidistanten Zahlen zwischen einem angegebenen Start- und Endwert erzeugt werden.

Vielleicht ist Euch beim Lesen dieses Abschnittes aufgefallen, dass uns Listen schon in vorherigen Beispielen begegnet sind, auch wenn das an diesen Stellen verschwiegen wurde. Beispielsweise wurde der Funktion `axes_labels()` (Siehe Kapitel 3) eine Liste aus zwei Zeichenketten übergeben und die Funktion `animate()` aus Aufgabe 3 erstellt die Animation aus einer Liste von Grafik Objekten, wie wir im folgenden Kapitel noch genauer sehen werden.

Es gibt in Python noch eine Menge weiterer nützlicher Funktionen für Listen, die wir im Folgenden jedoch nicht unbedingt benötigen werden. Wer trotzdem mehr erfahren will, dem sein als Einstieg zum Beispiel der Abschnitt über Listen im Python Tutorial von Daniel Nögel [1] empfohlen (<http://www.freiesmagazin.de/freiesMagazin-2011-04-02>).

4.2 For-Schleifen

For-Schleifen stellen eine der wichtigsten sogenannten Kontrollstrukturen in der Programmierung dar, da sie es erlauben Befehle zu iterieren.

Die Funktion einer for-Schleife wird durch das folgende Beispiel deutlich:

```
for i in range(10):  
    print i  
    print 'Test'  
print 'Fertig'
```

```
0  
Test  
1  
Test  
2  
Test  
3  
Test  
4  
Test  
5  
Test  
6  
Test  
7  
Test  
8  
Test  
9  
Test  
Fertig
```

Was geschieht hier? Der erste Teil einer Schleife ist der sogenannte Kopf der Anweisung, in unserem Fall `for i in range(10):`. Die Funktion `range()` haben wir im vorherigen Kapitel schon kennen gelernt, sie liefert also eine Liste mit den Zahlen $[0, 1, \dots, 9]$. Die `for` Anweisung bewirkt nun, dass die Variable `i` diese Liste schrittweise durchläuft. Für jedes Element der Liste werden die Anweisungen, die nach dem Kopf eingerückten sind, ausgeführt. In unserem Fall also die Anweisungen `print i` und `print 'Test'`. Etwas gewöhnungsbedürftig in Python ist, dass die Einrückung der Anweisungen von Bedeutung ist. Auch dies wird am obigen Beispiel deutlich. Nur die beiden eingerückten Befehle werden in der `for`-Schleife iteriert. Die letzte, nicht eingerückte Anweisung `print 'Fertig'` wird nur einmal nach dem Durchlaufen der Schleife ausgeführt.

Auch eine `for`-Schleife ist uns beim Erstellen der Animation in Aufgabe 3 schon begegnet:

```
a = animate([circle((i,1+abs(3*sin(i))), 1) for i in srange(0,10,0.1)], xmin=-1,ymin=0,xmax=11,ymax=5, figsize=[12,5])  
show(a)
```

Mit den jetzigen Kenntnissen können wir den obigen Befehl besser verstehen, auch wenn sich die Syntax, die in diesem Befehl auftaucht, noch einmal von der obigen unterscheidet. `[circle(i,1+abs(3*sin(i))),1) for i in srange(0,10,0.1)]` erinnert aufgrund der eckigen Klammern zuerst einmal an eine Liste. In dieser Liste scheint dann eine `for`-Anweisung zu stehen, die die Liste füllt. Wir testen, ob diese Konstruktion auch außerhalb der `animate()` Funktion funktioniert.

```
print [i*i for i in range(10)]
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

In der Tat scheinen wir somit noch eine neue Art gefunden zu haben, schnell eine Liste zu erstellen. Das gleiche Resultat hätten wir mit unseren vorherigen Kenntnissen auch wie folgt erzielen können:

```
a=[]  
for i in range(10):  
    a.append(i*i)  
print a
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

Die erste Art diese Array zu erzeugen ist zwar wesentlich weniger Schreibaufwand, allerdings auch deutlich schwieriger nachzuvollziehen, da das Erzeugen des Arrays, die `for`-Schleife und die Ausgabe alles in einer Zeile zusammengefasst wird. Es ist am Anfang empfehlenswert auf solche Abkürzungen zu verzichten, um einen besseren Überblick über seine Anweisungen zu behalten.

Aufgabe 4 (4 Punkte).

- a) Berechne mittels einer `for`-Schleife die Fakultät Deiner Schuhgröße. Selbstverständlich gibt es in Sage auch direkt eine Funktion, die die Fakultät berechnet.

Finde diese heraus und teste Dein Ergebnis.

- b) Programmiere die Animation aus Aufgabe 3 noch einmal übersichtlicher, indem du zuerst mittels einer for-Schleife die Grafik Liste erzeugst und dieses dann an die `animate()` Funktion übergibst.
- c) Informiere Dich über die Funktion `list_plot`. Erzeuge eine Liste mit den Datenpunkten $\sin(2\pi \cdot 0.99 \cdot n)$ für $n = 1, 2, \dots, 150$ und plote die Liste. Erstelle einen weiteren Plot mit der Funktion $\sin(2 \cdot \pi \cdot t)$ und den Datenpunkten $(0.99n, \sin(2\pi \cdot 0.99n))$ mit $n = 1, 2, \dots, 150$.
- d) Nutze die “Share” Funktion des Sage Notebooks um Deine Lösung mit Deinem Tutor zu teilen.

Literatur

- [1] Daniel Nögel. Python-programmierung. *freies Magazin*, Python Sonderausgabe(04/2011), 2011.