1 Python Scripting für Physiker - Handout W3

1.1 Sichtbarkeitsbereiche von Variablen (Scoping Rules)

Ein wichtiges Konzept in Python (und jeder anderen Programmiersprache) sind Namesräume und Sichtbarkeitsbereiche von Variablen. Jede Variable in Python stellt eine Referenz auf ein Objekt im Speicher dar, welche über den Namen der Variable angesprochen werden kann. Damit der Interpreter weiss, welche Objekte und damit Variablennamen vorhanden sind, werden diese im sog. Namensraum abgelegt. Die Erzeugung des Namensraum erfolgt hierbei dynamisch, d.h. weisen wir einen Wert einer neuen Variable zu, wird der Name dieser Variable dem Namensraum hinzugefügt.
Hierbei ist zu beachten, dass Funktionen (genauso wie Klassen und Module) eigene Namensräume besitzen. Und jeder dieser Namensräume hat einen zugewiesenen Sichtbarkeitsbereich, der (vorerst) mit einer einfachen Regel beschrieben werden kann

- **Eine Funktion (oder Modul, Klasse) kann immer nur auf den eigenen Namensraum und den aller umschliessenden Funktionen (oder Module, Klassen) zugreifen**

```
Beispiel:
>>>A=2                        #Globaler Namensraum
>>>def func1():
>>>     B=3                   #Lokaler Namensraum von func1
>>>     print "func1: ",
>>>     print A,B             #Zugriff auf lokalen und umschliessenden Namensraum
>>>     def func2():
>>>             C=4           #Lokaler Namensraum von func2
>>>             print "func2: "
>>>             print A,B,C   #Zugriff auf alle umschliessenden Namensraeume
>>>     func2()
>>>func1()
>>>print A                     #Hier kann nur auf A zugegriffen werden, da die
                              #Namensraeume von func1 und func2 hier nicht sichtbar sind.
```

1.2 Grafische Darstellung

Mit der Zeit wurden zahlreiche verschiedene Pythonmodule für die grafische Darstellung entwickelt. An dieser Stelle werden wir uns mit dem Modul Gnuplot.py beschäftigen (Anm.: Im Teil Projektarbeit wird es auch noch eine kurze Einführung in das Modul matplotlib geben).
Auf http://wiki.python.org/moin/NumericAndScientific/Plotting gibt es für Interessierte eine umfassende Liste von verfügbaren Modulen.

1.3 Gnuplot

Gnuplot ist ein verbreitetes open-source Plottingprogramm. Mithilfe von Gnuplot können sehr schnell einfache Plots erzeugt werden. So können Daten z.B. schnell auf dem Bildschirm visualiert werden. Die Funktionalität von Gnuplot erlaubt es jedoch auch

umfangreichere formartierte Plots zu erstellen und z.B. direkt Postscript Dateien (also Vektorgrafik) zu erzeugen. Es folgt eine (sehr kurze) Einführung in Gnuplot.

```
...$gnuplot                          # Startet Gnuplot
gnuplot>?                            # Ruft die eingebaute Hilfe auf
gnuplot>? plot                       # Gibt die hilfe fuer den Befehl plot aus
gnuplot>plot sin(x)                  # Erzeugt Plotfenster und plottet Sinus von x
gnuplot>set xrange[0:2*pi]           # Der oben erzeugte Plot kann umformatiert werden
gnuplot>replot                       # Plot wird umformatiert
gnuplot>plot sin(x) with points      # Sinus von x wird mit Punkten dargestellt
```

1.4 Gnuplot.py Interface

Gnuplot.py ist ein Pythonmodul das es ermöglicht Gnuplot aus Python heraus zu steuern. An dieser Stelle gibt es ebenfalls einen kurzen Vorgriff auf die nächste Vorlesung in der Module eingeführt werden.

```
>>>from Gnuplot import Gnuplot      # Importiert aus dem Modul Gnuplot die Klasse Gnuplot
>>>gp=Gnuplot()                     # Erzeugt eine Gnuplot Instanz
>>>gp("f(x)=x**2")                  # Definiert f(x)
>>>gp("set xrange [0:10]")          # Setzt die xrange
>>>gp("plot f(x)")                  # Erzeugt Plotfenster und plottet f(x)
```

Es können auch Werte aus Python geplottet werden.

```
>>>from Gnuplot import Gnuplot
>>>a=[[1,1],[2,4],[3,9],[4,16]]
>>>gp=Gnuplot()
>>>gp.plot(a)
```

Um formatierten Output zu erzeugen empfiehlt es sich aber mit der Klasse Data zuarbeiten.

```
>>>from Gnuplot import Gnuplot, Data
>>>a=[[1,1],[2,4],[3,9],[4,16]]
>>>gpdata=Data(a)
# Mit der Methode set_option_colonsep(a,b) kann der Datensatz formatiert werden.
>>>gpdata.set_option_colonsep("with","lines linetpye 3 linewidth 3")
>>>gpdata.set_option_colonsep("with","l lt 3 lw 3")
>>>gpdata.set_option_colonsep("title","'y(x)=x^2'")
# Die Formatierung kann auch direkt bei der Initialisierung durchgefuehrt werden.
>>>gpdata=Data(a,with_="l lt 3 lw 3",title="y(x)=x^2")
# Beachte 'with_' (da 'with' Python Keyword ist)
>>>gp=Gnuplot()
>>>gpdata2=Data(a)
>>>gpdata2.set_option_colonsep("with","l lt 1 lw 3")
>>>gpdata2.set_option_colonsep("using","2:1")
>>>gpdata2.set_option_colonsep("title","'y(x)=sqrt(x)'")
>>>gp.replot(gpdata2)
# Data kann auch direkt mit separaten x,y Werten initialisiert werden
>>>x=[1,2,3,4]
>>>y=[1,4,9,16]
>>>gpdata3=Data(x,y)
...
```

Bisher wurde die Ausgabe auf dem Bildschirm geplottet. Mit der Methode hardcopy ist es möglich den Plot in eine Datei auszugeben.

```
>>>from Gnuplot import Gnuplot, Data
>>>a=[[1,1],[2,4],[3,9],[4,16]]
>>>gpdata=Data(a,with_="l lt 3 lw 3",title="y(x)=x^2")
gp.plot(gpdata)                          # gibt den Plot auf dem Bildschirm aus
gp.harcopy("xsquare.ps",color="True")    # erzeugt den Postscript Datei xsquare.ps
# Beachte Standardausgabe fuer hardcopy ist Postscript
gp.harcopy("xsquare.png",terminal='png') # erzeugt PNG Datei
```

1.4.1 Beispiel 2D Plot

Das folgende Beispiel veranschaulicht die Benutzung von Gnuplot und Gnuplot.py.

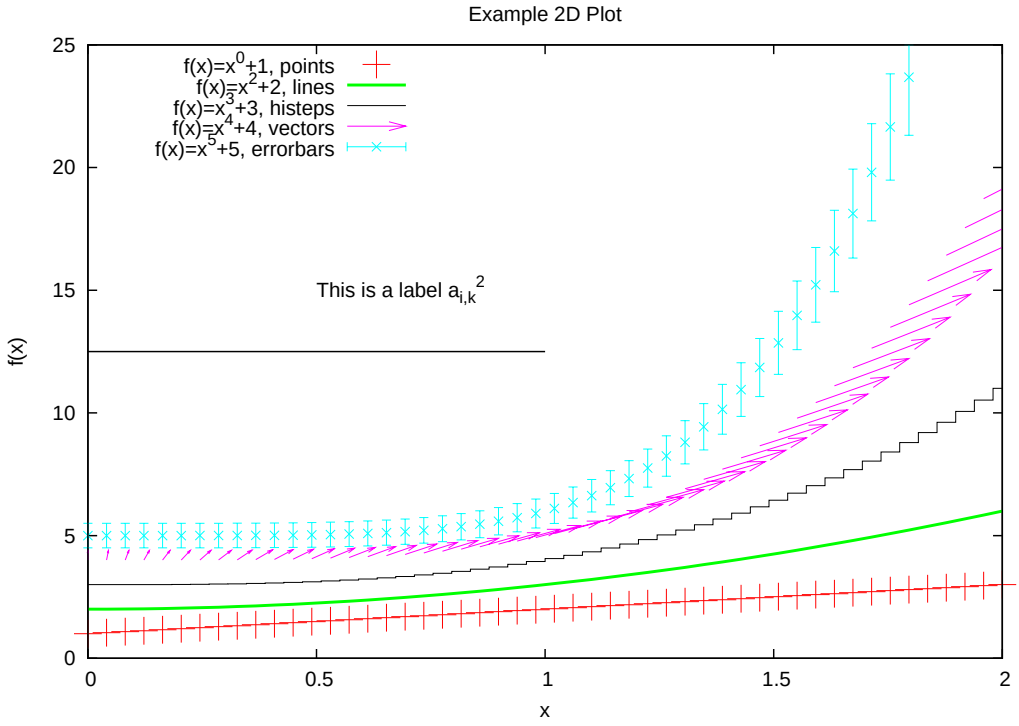

Abbildung 1: 2D Plot mit Gnuplot.py

CODE EXAMPLE:

```
 1 from numpy import linspace
 2 from Gnuplot import Gnuplot, Data
 3 x=linspace(0,2,50)
 4 a,b,c,d,e=x+1,x**2+2,x**3+3,x**4+4,x**5+5
 5 adata=Data(x,a,title="f(x)=x^{0}+1, points",with_="points pointsize 3")
 6 bdata=Data(x,b,title="f(x)=x^{2}+2, lines",with_="lines linewidth 4")
 7 cdata=Data(x,c,title="f(x)=x^{3}+3, histeps",with_="histeps linecolor 'black'")
 8 edata=Data(x,e,e*0.1,title="f(x)=x^{5}+5, errorbars",with_="errorbars")
 9 ddata=Data(x,d,x*0.1,d*0.1,title="f(x)=x^{4}+4, vectors",with_="vec")
10 gp=Gnuplot()
11 gp("set key left")
```

```
12 gp("set terminal x11 enhanced")
13 gp("set xlabel 'x'")
14 gp("set ylabel 'f(x)'")
15 gp("set xrange [0:2]")
16 gp("set yrange [0:25]")
17 gp("set title 'Example 2D Plot'")
18 gp("set arrow from graph %s,%s to graph %s, %s nohead"%(0.0,0.5,0.5,0.5))
19 gp("set label 'This is a label a_{i,k}^{2}' at %s,%s"%(0.5,15))
20 gp.plot(adata,bdata,cdata,ddata,edata)
21 gp.hardcopy("plot.ps",color=1,solid=1,enhanced=1)
22 raw_input()
```

- Zeile 3: Erzeugt 50 Werte zwischen 0 und 2 und speichert sie in *ndarray* x.
- Zeile 4: Erzeugt 5 arrays die $x+1$, x^2+2, ..., x^5+5 enthalten.
- Zeilen 5-9: 5 unterschiedlich formatierte Data Objekte werden erzeugt.
  - Zeile 8: Zum plotten von Fehlerbalken werden die Fehler als extra Werte übergeben $e*0.1$.
  - Zeile 9: Zum plotten von Vektoren werden die Startwerte x,d und die Inkremente $x*0.1$,$d*0.1$ übergeben.
- Zeile 10: Erzeugt eine Gnuplot Instanz.
- Zeile 11: Die Legende wird links platziert (wird aus den Titeln (title='abc') der Data Objekte erzeugt.
- Zeilen 13-16: Achsenbeschriftungen sowie Wertebereiche fuer die Achsen werden gesetzt.
- Zeile 17: Ein Titel fuer den Plot wird gesetzt.
- Zeile 18: Plottet einen Pfeil von $(0.0, 0.5)$ nach $(0.5, 0.5)$ ohne Spitze (keyword 'nohead'). Das keyword 'graph' vor den Koordinaten definiert das verwendete Koordinatensystem. 'graph' bedeutet, dass die linke untere Ecke (x_{min}, y_{min}) die Koordinaten $(0, 0)$ und rechte obere Ecke (x_{max}, y_{max}) die Koordinaten $(1, 1)$ hat.
- Zeile 19: Erzeugt ein Label. Da kein Koordinatensystem spezifiziert ist, wird das Standardsystem (X-, Y-Achse) verwendet. D.h. das Label wird an den Koordinaten $(x = 0.5,\ y = 15)$ platziert.
- Zeile 20: Plottet die 5 Gnuplot Data Objekte. Beachte es können beliebig viele Objekte übergeben werden.
- Zeile 21: Erzeut die Postscript-Datei 'plot.ps'

Nützliche Links:
Gnuplot Tutorial → http://t16web.lanl.gov/Kawano/gnuplot/index-e.html.
Beispiele → http://gnuplot.sourceforge.net/demo/index.html.
Sonderzeichen → http://ariadne.mse.uiuc.edu/Info/Gnuplot/ps_guide.pdf

2 Übung

Die Datei data3.dat enthält drei Datensätze normalverteilter Zufallszahlen. Jeder dieser Datensätze beginnt mit einem Header gefolgt von N zugehörigen Datenpunkten.

```
Datei: data.dat
### First Data Set
12.31
13.43
11.23
...
### Second Data Set
...
### Third Data Set
...
```

1. Erstellen Sie für jeden Datensatz eine Liste, welche die zugehörigen Datenpunkte enthält und stellen sie diese graphisch dar.
2. Erstellen sie eine Häufigkeitsverteilung der drei Datensätzen und stellen sie diese in einem separaten Plot dar (Eine Binbreite von 0.1 sollte hierfür ausreichen)
3. Berechnen Sie den Mittelwert und die Standardabweichung der Datensätze und plotten Sie die zugehörige Normalverteilung in den Plot aus Aufgabe (2).
4. (Advanced) Stellen sie $f(x, y) = \sin(x)\cos(y)$ graphisch so dar wie in Fig. 2 gezeigt.

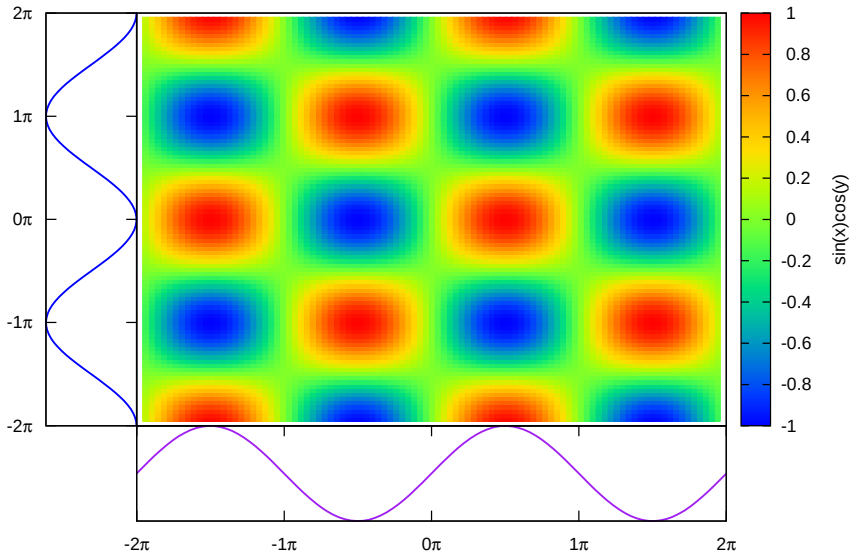

Abbildung 2: 3D Plot mit Gnuplot.py