2013

Seminar über IPython Notebook

Michael Jünger

Inhaltsverzeichnis

1. Einleitung

In vielen Bereichen des täglichen Lebens ist die Auswertung und Verarbeitung von Daten, welche uns in Form von Tabellen und Grafiken dargeboten werden nicht mehr wegzudenken. Sei es die Präsentation der Jahresbilanz im firmeninternen Meeting oder die Wahlergebnisse der bevorstehenden Bundestagswahl. Die Vorgehensweise ist dabei stets die gleiche. Daten werden gesammelt, durch Umfragen, Interviews oder Messungen und elektronisch beispielsweise in einer Datenbank gespeichert. Eine Software liest diese Daten aus dem Speicherort aus, führt gegebenenfalls weitere Berechnungen durch und stellt schlussendlich das Ergebnis grafisch in Form von Diagrammen dar. Hierfür existieren bereits zahlreiche Produkte auf dem Markt, wobei das bekannteste hierbei Excel aus der Office-Suite von Microsoft sein dürfte.

Ziel dieser Seminararbeit ist es, eine weitere, noch relativ neue Option zur Datenverarbeitung, deren Auswertung und Darstellung vorzustellen, das iPython Notebook. iPython ist ursprünglich ein interaktiver Interpreter der Programmiersprache Python, worin auch meine persönliche Motivation liegt mich mit dem iPython Notebook näher zu beschäftigen. Ich kenne Python als Programmiersprache aus Vorlesungen. Dort habe ich diese zum ersten Mal verwendet und lernte die Einfachheit, mit der es in Python möglich ist, schnell komplexe Programme zu erstellen zu schätzen. Unter Linux basierten Betriebssystemen ist Python noch immer die Programmiersprache meiner Wahl, auch da es in den meisten Distributionen bereits integriert ist. Das iPython Notebook stellt somit für mich eine weitere spannende und neue Möglichkeit dar, Python als Programmiersprache zu verwenden.

2. IPython und IPython Notebook

2.1. Geschichte

iPython ist ursprünglich ein interaktiver, shell-basierter Interpreter der Programmiersprache Python. Dieser bietet im Gegensatz zu dem in Python bereits integrierten Interpreter mehr Komfort und Funktionalität wie etwa Wortvervollständigung per Tabulatortaste. Das Ziel von iPython ist es, eine umfassende Entwicklungsumgebung für interaktives Programmieren zu schaffen, welche die Möglichkeit bietet, die Lauffähigkeit des Quellcodes zu verifizieren und auf darin enthaltene Fehler zu untersuchen.

Das iPython Notebook ist fester Bestandteil von iPython und seit der Version 0.12, welche am 18. Dezember 2011 erschien, integriert. Es baut auf dem iPython Interpreter auf und stellt dessen Funktionalität im Webbrowser zur Verfügung.

2.2. Installation/Konfiguration

IPython selbst ist sowohl für Linux als auch für OSX und Windows erhältlich und Voraussetzung zur Bereitstellung des Notebooks. Ich möchte an dieser Stelle jedoch lediglich kurz auf die Installation unter Linux eingehen. Die Installation erfolgt in der Regel über das Paketmanagementsystem der jeweiligen Distribution. Nach Abschluss dieser reicht der Aufruf von „`ipython notebook`“ im Terminal, woraufhin der Notebook-Server gestartet und der Standardbrowser an der entsprechenden URL geöffnet wird. Die Daten des Notebooks werden dabei im jeweiligen Benutzerverzeichnis gespeichert. Weiter ist es möglich, die Einstellungen eines Notebooks über eine Konfigurationsdatei vorzunehmen. Zur grafischen Darstellung von Daten lässt sich per Parameter oder Konfigurationsdatei das Modul Pylab in das Notebook integrieren.

Manche Situationen machen es erforderlich, einen Server dauerhaft laufen zu lassen. In diesen Fällen empfiehlt es sich ein Startskript unter `/etc/init.d` anzulegen (Nur bei Systemen, welche noch nicht den neuen System-Daemon im Einsatz haben). Dies hat zum einen den Vorteil, dass der Notebookserver beim Hochfahren des Systems mit gestartet wird. Zum anderen lässt sich der Server auf diese Weise per Befehl „`service DIENSTNAME START|STOP|RESTART`“ manuell starten, stoppen oder neu starten, wenn dies erforderlich ist. Meines Wissens existiert hierfür kein offizielles Startskript, welches diese Aufgabe übernimmt. Die folgende Abbildung zeigt daher einen möglichen Weg, dies zu realisieren.

Startskript für den iPython Notebook Server:

```
#!/bin/bash
### BEGIN INIT INFO
# Provides: ipynbook
# Required-Start:
# Required-Stop:
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: start ipython notebook server
# Description: iPython notebook start script.
### END INIT INFO
#
#chkconfig: 2345 7 93
# description: iPython notebook start script
#
# /etc/init.d/ipynbook
#

case $1 in
    start)
        cd /home/ipynbook/.ipython
        sudo -u ipynbook ipython notebook --profile=nbserver &
        ;;
    stop)
        killall ipython
        ;;
    restart)
        killall ipython
        cd /home/ipynbook/.ipython
        sudo -u ipynbook ipython notebook --profile=nbserver &
        ;;
    esac
```

Dies stellt jedoch noch keine optimale Lösung dar. Es ist zu beachten, dass bei einem Stop oder Neustart des Dienstes der Befehl „killall ipython“ aufgerufen wird, welcher alle iPython Instanzen beendet, also auch jene die nicht direkt zu dem gestarteten Notebookserver gehören. Die Befehle sollten also nur dann ausgeführt werden, wenn sichergestellt ist das keine anderen Instanzen von iPython auf dem System gestartet wurden.

2.3. Komponenten

2.3.1. Python

Python ist, wie eingangs schon erwähnt, als Programmiersprache die zentrale Komponente des iPython Notebooks. „Python wurde Anfang der 90er Jahre von Guido van Rossum am Zentrum für Wissen und Informatik (Centrum Wiskunde & Informatica) in Amsterdam als

Nachfolger der Lernsprache ABC entwickelt. Das Ziel von Python ist es möglichst übersichtlich zu sein. Die Anzahl der Schlüsselwörter wurde hierfür auf sehr wenige beschränkt und die Syntax reduziert. “ (Wikipedia, 2013)

Wie folgende Grafik verdeutlicht:

Gegenüberstellung von Hello World in Java (links) und Python (rechts)

```
/*
 * To change this template, choose Tools | Templates
 * and open the template in the editor.
 */
package helloworld;

/**
 *
 * @author michael
 */
public class HelloWorld {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        System.out.println("Hello World");
    }
}
```

```
print "Hello World"
```

Darüber hinaus schreibt Python kein Programmierparadigma vor. Während in Java Objektorientierte Strukturen wie Namensräume, sogenannte „packages“ und Klassen zwingend erforderlich sind, kann in Python der Programmcode ohne dieses Grundgerüst von mindestens einer Klasse und der Main-Methode ausgeführt werden. Die Verwendung von Klassen und die Verteilung des Programmcodes auf mehrere Dateien werden hierbei dennoch unterstützt. Somit bietet Python zum einen die Möglichkeit prozeduraler, skriptbasierter Programmabläufe und zum anderen die Umsetzung komplexer Programmstrukturen unter Zuhilfenahme objektorientierten Methoden.

2.3.2. Matplotlib

Bei der Matplotlib handelt es sich um eine Bibliothek zur Darstellung zweidimensionaler Daten, ähnlich Matlab. Entwickelt wurde das Modul von John D. Hunter, welcher am 28. August 2012 verstarb.

Die Matplotlib unterstützt eine Vielzahl von Darstellungsformen wie Balkendiagramme, Kuchendiagramme, Funktionsgrafen sowie mathematische Formeln, um nur wenige Beispiele zu nennen. Über Erweiterungsmodule lässt sich dies noch dazu beliebig erweitern. So existiert beispielsweise eine weitere Bibliothek namens mplot3d zur Darstellung dreidimensionaler Daten, welche jedoch bereits in matplotlib integriert ist.

Darüber hinaus lassen sich auch Vorgänge animieren um etwa die Entwicklung eines Vorganges im Zeitverlauf darzustellen. In Kombination mit der Einfachheit von Python lassen sich so mit minimalem Programmcode Prozesse simulieren, wie folgendes Beispiel zeigen soll.

C:\Users\michael\Downloads\slider_demo.py Mittwoch, 24. April 2013 14:08

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.widgets import Slider, Button, RadioButtons

ax = plt.subplot(111)
plt.subplots_adjust(left=0.25, bottom=0.25)
t = np.arange(0.0, 1.0, 0.001)
a0 = 5
f0 = 3
s = a0*np.sin(2*np.pi*f0*t)
l, = plt.plot(t,s, lw=2, color='red')
plt.axis([0, 1, -10, 10])

axcolor = 'lightgoldenrodyellow'
axfreq = plt.axes([0.25, 0.1, 0.65, 0.03], axisbg=axcolor)
axamp  = plt.axes([0.25, 0.15, 0.65, 0.03], axisbg=axcolor)

sfreq = Slider(axfreq, 'Freq', 0.1, 30.0, valinit=f0)
samp = Slider(axamp, 'Amp', 0.1, 10.0, valinit=a0)

def update(val):
    amp = samp.val
    freq = sfreq.val
    l.set_ydata(amp*np.sin(2*np.pi*freq*t))
    plt.draw()
sfreq.on_changed(update)
samp.on_changed(update)

resetax = plt.axes([0.8, 0.025, 0.1, 0.04])
button = Button(resetax, 'Reset', color=axcolor, hovercolor='0.975')
def reset(event):
    sfreq.reset()
    samp.reset()
button.on_clicked(reset)

rax = plt.axes([0.025, 0.5, 0.15, 0.15], axisbg=axcolor)
radio = RadioButtons(rax, ('red', 'blue', 'green'), active=0)
def colorfunc(label):
    l.set_color(label)
    plt.draw()
radio.on_clicked(colorfunc)

plt.show()
```

(Matplotlib, 2013)

2.3.3. Tkinter

„Tkinter ist ein Wrapper des TK für Python und war das erste seiner Art zur Erstellung grafischer Benutzeroberflächen, weshalb es bereits in Python integriert ist.“ (Wikipedia, 2013). „Das TK (Toolkit) ist ein freies und plattformübergreifendes Toolkit zur Erstellung grafischer Benutzeroberflächen. TK wurde für die TCL[1] (Tool Common Language) entwickelt.“ (Wikipedia, 2013) Neben Tkinter existieren heute zahlreiche weitere Toolkits wie wxWidgets, PyQt und PyGTK, um nur die wichtigsten zu nennen.

2.3.4. Numpy

„Numpy bietet für Python die Verwendung von Arrays und Matrizen sowie grundlegende Operationen auf diese an. So sind bereits Algorithmen zum Sortieren, manipulieren, statistische Basisoperationen, Fourier Transformation und viele weitere enthalten“ (Pylab, 2013). In Numpy sind demnach bereits viele Operationen integriert, welche für die Verarbeitung von numerischen Daten benötigt werden. Numpy eignet sich somit hervorragend für die Arbeit mit matplotlib.

2.3.5. Pylab

Das Modul Pylab vereint die beschriebenen Komponenten in einem gemeinsamen Namensraum und ermöglicht dadurch den einfachen und gemeinsamen Zugriff auf die zahlreichen Funktionen der verschiedenen Komponenten.

3. Anwendungsgebiete

Die Anwendungsmöglichkeiten des iPython Notebooks sind verschieden. So sind damit beispielsweise Schulungen denkbar, um die Programmiersprache Python selbst und den Umgang mit Pylab und den darin enthaltenen Komponenten erlernen. Weitaus interessanter und vielversprechender ist jedoch die Anwendung in jenen Bereichen, in denen es um die Auswertung und Darstellung von Daten geht. Genau für diesen Zweck bieten iPython und Pylab die entsprechenden Werkzeuge komplexer mathematischer Berechnungen und der Darstellung der Ergebnisse. Das iPython Notebook kann in der Gestalt als Weiterentwicklung gesehen werden. Eine Anpassung an den aktuellen Trend, Daten immer und überall zur Verfügung zu stellen, unabhängig von der verwendeten Plattform des Betrachters.

[1] Eine Open Source Skriptsprache

3.1. Tabellenkalkulation – Im Vergleich mit Microsoft Excel

Wie in Excel von Microsoft oder dem Programm Calc der Open Office Suite lassen sich auch im iPython Notebook Daten aus einem Speicherort auslesen und grafisch in Form von Diagrammen verschiedenster Arten darstellen.

Dies mag im Vergleich zu einem Programm mit komfortabler Benutzeroberfläche zunächst umständlich erscheinen. Schließlich bieten solche Produkte einen einfachen und übersichtlichen Zugriff auf alle relevanten Funktionen, leisten Unterstützung und verlangen dem Anwender nicht zwingend Programmierkenntnisse ab. Ein Gewinn an Komfort jedoch geht auch einher mit dem Verlust an Flexibilität. Dies soll nicht bedeuten das Programme wie Excel nicht in der Lage sind dem erfahrenen Anwender erweiterte Möglichkeiten zu bieten. Auch mit solchen Programmen ist es möglich komplexe Vorgänge zu berechnen und diese grafisch darzustellen. Eine Benutzeroberfläche aber wird idealerweise so gestaltet, dass diese den unerfahrenen Anwender nicht überfordert. Das bedeutet, dass auf den ersten Blick auf die Software zunächst einmal nur die notwendigsten Funktionen erkennbar sind und der Zugriff auf erweiterte Optionen nicht sofort ersichtlich wird. Sowohl für den Einsteiger als auch den erfahrenen Anwender ist es demnach notwendig, sich zusätzlich mit der Benutzeroberfläche vertraut zu machen.

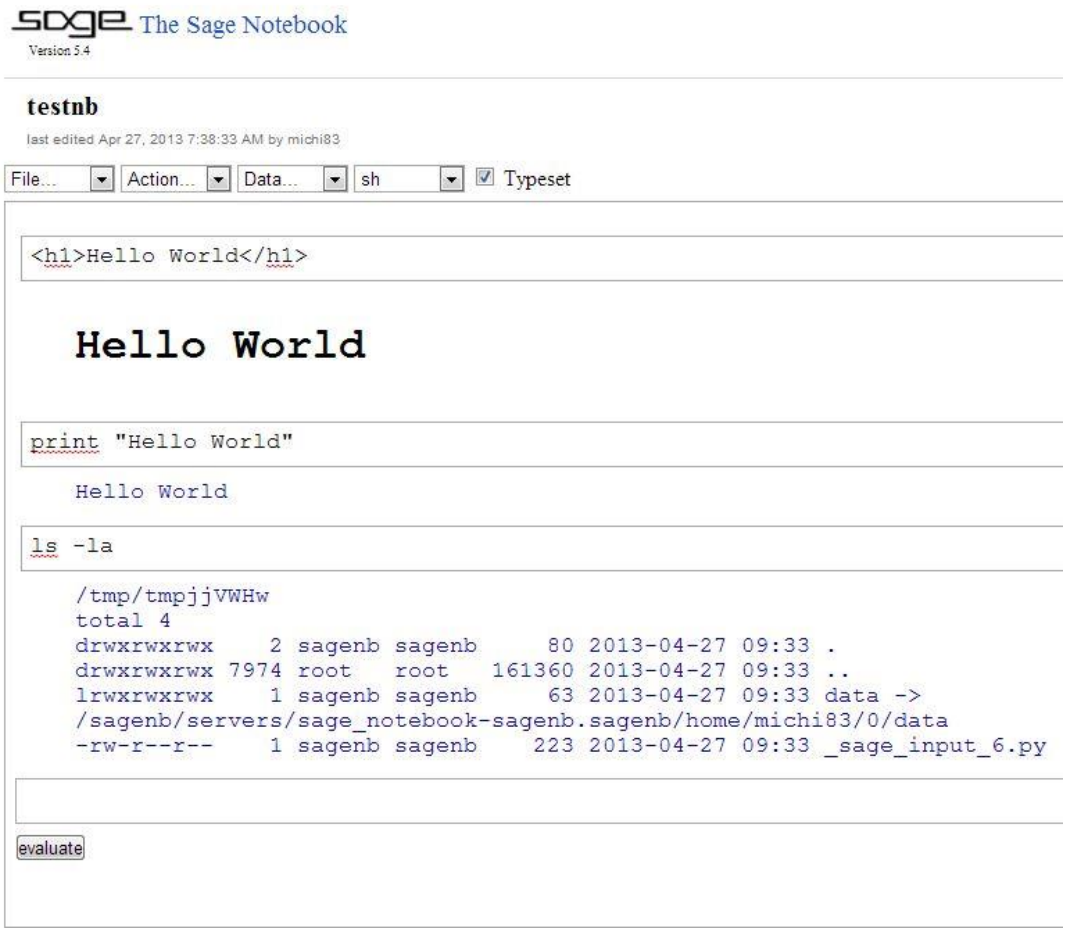
Im iPython Notebook hingegen erfolgt jede Berechnung und jede Darstellung rein auf Codebasis. Dies ist sicherlich weniger komfortabel als ein simpler Klick auf einen vordefinierten Button in einer Anwendung, bietet aber zu jederzeit die vollständige Kontrolle darüber was mit den Daten genau passieren soll. Und das nicht versteckt in mehr oder weniger einfach zu erreichenden Eingabemasken, sondern für jedermann ersichtlich. Somit bietet das iPython Notebook nicht nur maximale Flexibilität in der Verarbeitung von Daten sondern auch Transparenz über die zugrundeliegenden mathematischen Formeln. Letzteres kommt demnach der Philosophie der Quelloffenheit nach.

Darüber hinaus besteht auch die Möglichkeit, ein Notebook über einen öffentlich zugänglichen Server bereitzustellen. Die Daten liegen somit also zentral an einem Ort, auf den der Zugriff sogar über das Internet erfolgen kann. Der Zugriff auf diese Daten erfolgt über einen Webbrowser, die Interpretation des Quellcodes und schlussendlich das Rendern der berechneten Daten zu Diagrammen erfolgt lokal auf dem Server. Dies macht zum einen die Synchronisation eines Dokuments überflüssig, etwa wenn die Arbeit auf verschiedenen Rechnern erfolgt, für die nicht zu jederzeit ein gemeinsamer Speicherort zur Verfügung steht. Zum anderen entfällt die Notwendigkeit, dieselbe Anwendung auf jedem PC zu installieren, was bei einem kommerziellen Produkt wie Microsoft Excel nicht zuletzt auch aus wirtschaftlichen Aspekten von Vorteil ist.

4. Alternativen

4.1. Sage-Notebook

Ganz neu ist die Idee hinter dem iPython Notebook eine Alternative zu Maple oder Matlab zu bieten nicht. Ebenfalls in Python entwickelt entsteht seit dem Jahr 2005 das Sage Notebook, welches ebenfalls dieses Ziel verfolgt. Wie das iPython Notebook bietet die Lösung von Sage Unterstützung für mehrere Programmiersprachen an. So ist es beispielsweise möglich auch HTML- oder Matlab-Code Interpretieren zu lassen. Die Auswahl der unterstützten Programmiersprachen ist bei Sage jedoch vielfältiger. Eine Auflistung der Sprachen die iPython unterstützt kann mit dem Befehl **„`%lsmacig`“** angezeigt werden.



4.2. Fazit

Das Sage Notebook stellt ebenfalls eine gute Möglichkeit zur Auswertung und Visualisierung von Daten dar. Darüber hinaus spricht es eine breitere Masse an Nutzern an, da es nicht ausschließlich auf die Verwendung von Python beschränkt ist.

Für den einmaligen oder nur kurzfristigen Gebrauch stellt Sage sogar einen Notebookserver bereit. Der Zugriff auf diesen kann über externe Authentifizierungsstellen wie Google oder OpenID erfolgen. Wer keinen solchen Zugang besitzt kann sich bei Sage auch direkt registrieren und damit einen Account zur Nutzung eines Notebooks erlangen.

Die Benutzung stellt Sage hierbei kostenlos zur Verfügung. Für komplexe, rechenintensive Aufgaben sollte dies jedoch nicht das Mittel der Wahl sein, da die verfügbaren Ressourcen stark begrenzt sein dürften.

5. Probleme und Einschränkungen - Im Vergleich zu iPython in einer Shell

Da es sich beim iPython Notebook um ein relativ neues Projekt handelt, sind noch nicht alle Probleme behoben. Bei meinem eigenen Notebook-Server konnte ich nun schon mehrmals beobachten, dass die Ausgabe nicht immer im Webbrowser erscheint. Selbst bei einem simplen Befehl wie `print „Hello World“` wird das Ergebnis dann nicht mehr angezeigt. Ein Lösungsansatz aus dem Internet besagte, dass Sophos Antivirus die Anzeige der Ergebnisse blockiere. Abhilfe solle ein Eintrag des Domainnamens in der Konfiguration von Sophos im Authorization-Manager unter der Rubrik Websites bringen. Ich befolgte also der beschriebenen Anweisung, konnte daraufhin aber nur bedingt eine Verbesserung beobachten. Zwar wurde mir im Anschluss die Ausgabe angezeigt, das Problem tritt jedoch zeitweise noch immer auf.

Unabhängig dieser technischen Einschränkung musste ich mich bei der Verwendung des iPython Notebooks noch in dessen Verwendung umstellen. So ist beispielsweise die Eingabe von Daten über die Standardeingabe, wie ich es aus einer Shell-Umgebung kenne, im Notebook nicht möglich. Da aber die Möglichkeit geboten wird Webseiten in das Notebook einzubinden wäre die Eingabe von Daten an dieser Stelle per Webschnittstelle möglich. Die Daten würden dabei über ein Web-Formular in eine Datenbank geschrieben. Das Notebook kann diese dann von dort lesen und weiter verarbeiten.

Weitere Probleme oder Einschränkungen konnte ich bei meiner Arbeit mit dem Notebook bisher nicht feststellen. Da es sich außerdem um ein relativ neues Projekt handelt, ist ohnehin klar dass noch der ein oder andere Fehler zu beheben sein wird. Aufgrund der ansonsten aber sehr gut funktionierenden Arbeitsumgebung ist dies nicht den Entwicklern vorzuwerfen. Es ist vielmehr Teil eines gängigen Entwicklungsprozesses, den jedes Softwareprodukt durchläuft.

6. Sicherheitsaspekte

Wie in allen Bereichen der Informationstechnologie spielt Sicherheit auch bei der Verwendung des iPython Notebooks eine zentrale Rolle. Nicht zuletzt deshalb, da hier die Möglichkeit geboten wird Programmcode auf einem System auszuführen, welcher das System in schädlicher Weise verändern kann. Ist der Notebookserver noch dazu aus dem Internet erreichbar, steigt dieses Sicherheitsrisiko signifikant an.

Da dies bei meinem Notebookserver der Fall ist und ich diesen öffentlich zugänglich eingerichtet habe, bestanden meine ersten Maßnahmen zur Absicherung darin, mich mit den gebotenen Sicherheitskonzepten von iPython vertraut zu machen. An erster Stelle bot sich an den Notebookserver mit einem Passwort zu versehen. Um Zugriff auf den Server zu erlangen muss dieses erst eingegeben werden. Konfiguriert wird das Passwort in der zugehörigen Konfigurationsdatei des Notebooks, wobei es dort nicht im Klartext eingetragen wird, sondern verschlüsselt in Form eines Hashwertes. Eine entsprechende Methodik zum Erstellen des Passwort-Hashes wird in der Konfigurationsdatei beschrieben.

```
78 # Hashed password to use for web authentication.
79 #
80 # To generate, type in a python/IPython shell:
81 #
82 #   from IPython.lib import passwd; passwd()
83 #
84 # The string should be of the form type:salt:hashed-password.
85 c.NotebookApp.password = u'sha1:0a6340f970e0:aeb3a3adfea04f324499fe92b870ffd8546d0f8c'
86
```

Für eine sichere Verbindung über das https-Protokoll müssen in der Konfigurationsdatei jeweils noch ein Zertifikat sowie ein privater Schlüssel eingetragen werden. Ansonsten würde das Passwort unverschlüsselt übertragen, wodurch es für einen möglichen Angreifer ein leichtes wäre, dieses abzufangen.

Konfiguration für das Zertifikat:

```
63 # The full path to an SSL/TLS certificate file.
64 c.NotebookApp.certfile = u'/etc/ssl/ipynbook/ipn.crt'
65
```

Konfiguration für den privaten Schlüssel:

```
90 # The full path to a private key file for usage with SSL/TLS.
91 c.NotebookApp.keyfile = u'/etc/ssl/ipynbook/ipn.key'
```

Wie in beiden Abbildungen zu sehen wurde am Pfad „/etc/ssl/“ das Verzeichnis ipynbook angelegt, in welchem sich beide Dateien befinden. Dies ist notwendig da über Python der

Zugriff auf den privaten Schlüssel nicht als ssl-cert Benutzer erfolgt sondern als der Benutzer, welcher den Notebookserver gestartet hat. In meinem konkreten Beispiel ist dies der Benutzer ipynbook. Nur die Benutzer root und ipynbook können auf das Verzeichnis /etc/ssl/ipynbook/ zugreifen und die darin enthaltenen Dateien lesen. Selbst der Schreibzugriff ist dem Benutzer ipynbook untersagt.

Als zusätzliche Sicherheitsmaßnahme wurde, wie eben schon angedeutet, ein Benutzer ipynbook angelegt. Dieser hat den Zweck, dass der Notebookserver nicht als Benutzer root mit allen Rechten gestartet wird. So ist es per sudo-Befehl möglich den Server als eben dieser Benutzer auszuführen. Weiter hat der Benutzer ipynbook kein Recht, sich in einer Shell einzuloggen, was durch einen entsprechenden Eintrag in der Datei /etc/passwd in der Form /bin/false erreicht werden kann. Üblicherweise steht anstelle von false entweder sh oder bash, da diese in den meisten Distributionen als standard-Shell vordefiniert sind.

Auszug aus der Datei /etc/passwd:

```
31 ipynbook:x:1002:1004::/home/ipynbook:/bin/false
```

Dies ist zwingend erforderlich, da der Benutzer ipynbook kein Passwort besitzt, welches andernfalls zuerst eingegeben werden müsste. Dies ist jedoch bei Benutzern, welche speziell für Services eingerichtet wurden, damit diese in jenem Benutzerkontext ihren Dienst verrichten können gar nicht möglich.

Die geschilderte Konfiguration des Notebooks entstammt aus der offiziellen Dokumentation von iPython aus den Punkten „Security“ und „Running a public notebook server“ (IPython, 2013)

7. Demonstration

- Auswertung von zwei- und dreidimensionalen Daten
- Code aus externen Quellen einbinden

8. Literaturverzeichnis

IPython. (24. 04 2013). *IPython*. Von http://ipython.org/ipython-doc/dev/interactive/htmlnotebook.html abgerufen

Matplotlib. (24. 04 2013). *matplotlib.org*. Von http://matplotlib.org/mpl_examples/widgets/slider_demo.py abgerufen

Pylab. (27. 04 2013). *Pylab.org*. Von http://docs.scipy.org/doc/numpy/user/whatisnumpy.html abgerufen

Wikipedia. (23. 04 2013). *Wikipedia*. Von http://de.wikipedia.org/wiki/Python_(Programmiersprache) abgerufen

Wikipedia. (24. 04 2013). *Wikipedia*. Von http://de.wikipedia.org/wiki/Tkinter abgerufen

Wikipedia. (24. 04 2013). *Wikipedia*. Von http://de.wikipedia.org/wiki/Tk_(Toolkit) abgerufen