

Einführung in Numpy und Matplotlib

Vorlesung 2, Signale, Systeme und Sensoren

Prof. Dr. M. O. Franz

HTWG Konstanz, Fakultät für Informatik

Übersicht

1 Python

2 Wissenschaftliches Rechnen in Python

Übersicht

1 Python

2 Wissenschaftliches Rechnen in Python

Warum Python?

- Für die Signalverarbeitung wird eine weit verbreitete Skriptsprache benötigt, die einen einfachen Zugriff auf die meisten in Fortran oder C geschriebenen, miteinander nicht kompatiblen numerischen Programmbibliotheken liefert.
- Open Source, keine Lizenzkosten, läuft auf allen gängigen Plattformen
- Einfacher, intuitiver Code, Indizierung und Speichermodell folgt den aus anderen Programmiersprachen (z.B C, Java) bekannten Mustern (Index beginnt bei 0, row major, ...), gut lesbar, schnell erlernbar (*Rapid Prototyping*)
- Modulare und objektorientierte Programmierung, einfache Verwaltung und Installation von Paketen, gute Integration von Dokumentation und Unittests.
- Nachteile: starke Versionsabhängigkeit (v.a. Version 2.x vs. 3.x), als Interpretersprache ist Python deutlich langsamer als kompilierte Sprachen.

Dynamische Typisierung in Python

Im Gegensatz zu C und Java werden Variablen und Funktionen keine Typen zugewiesen. Stattdessen wird der Datentyp an jedes zur Laufzeit erzeugte Objekt als “Tag” bzw. Typreferenz angehängt, z.B.

```
>>>  
>>> x = 6  
>>> x = 'Hallo '  
>>>
```

x hat keinen Typ, sondern nur das Objekt, auf das x zeigt. Die Sprache wird durch die dynamische Typisierung sehr flexibel, verliert aber die bei der Compilierung durchgeführte statischen Typprüfung und damit an Laufzeitstabilität, Geschwindigkeit und Optimierungsmöglichkeiten.

Übersichtlicher Code durch Einrücken

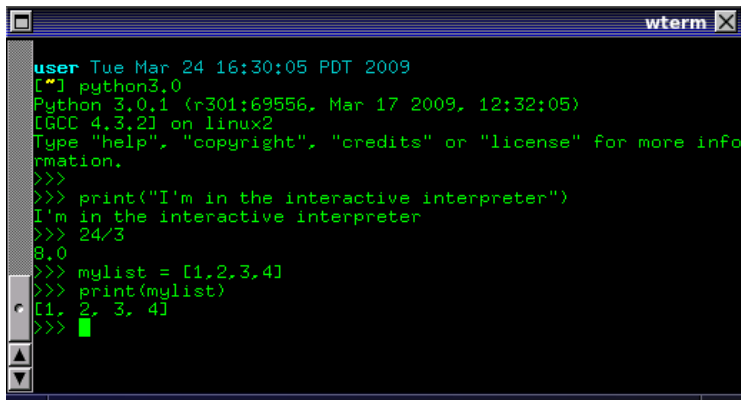
Wesentliches Ziel von Python ist ein einfach lesbarer Code. Blöcke werden durch Einrücken markiert (entweder Tab oder mindestens 4 Leerschritte):

```
1
2 for item in range(10):
3     print('I')
4     print('am')
5     print('a')
6     if item % 2 == 0:
7         print('funny')
8         print('and')
9         print('silly')
10    else:
11        print('dull')
12        print('and')
13        print('serious')
14    print('block')
15    print('used')
16    print('as')
17    print('example.')
```

Tip: im Editor jedes Tab automatisch durch 4 Spaces (Standard) ersetzen lassen.

Python-Interpreter

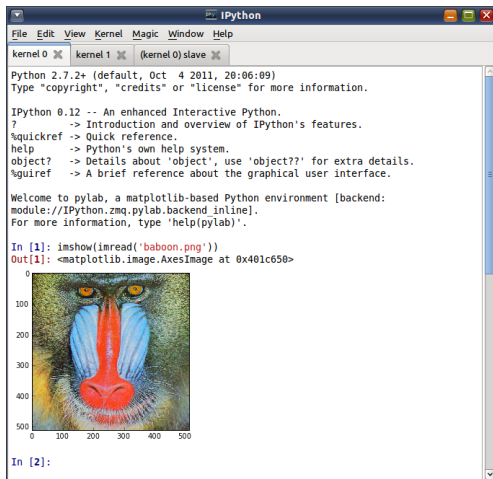
Python-Code kann interaktiv in den Interpreter eingegeben werden, der dann zeilenweise evaluiert wird. Es eignet sich dadurch gut zum Experimentieren und zur interaktiven Analyse. Reservierter Speicherplatz wird automatisch wieder freigegeben (*Garbage Collector*).

A screenshot of a terminal window titled 'wterm'. The window shows the Python 3.0.1 interactive interpreter. The prompt is 'user' and the date/time is 'Tue Mar 24 16:30:05 PDT 2009'. The user has entered 'python3.0' and the interpreter has started, displaying 'Python 3.0.1 (r301:69556, Mar 17 2009, 12:32:05) [GCC 4.3.2] on linux2'. The user has entered 'Type "help", "copyright", "credits" or "license" for more information.' and the prompt is '>'. The user has entered 'print("I'm in the interactive interpreter")' and the output is 'I'm in the interactive interpreter'. The user has entered '24/3' and the output is '8.0'. The user has entered 'mylist = [1,2,3,4]' and the prompt is '>'. The user has entered 'print(mylist)' and the output is '[1, 2, 3, 4]'. The prompt is '>'.

```
user Tue Mar 24 16:30:05 PDT 2009
[~] python3.0
Python 3.0.1 (r301:69556, Mar 17 2009, 12:32:05)
[GCC 4.3.2] on linux2
Type "help", "copyright", "credits" or "license" for more info
rmation.
>>>
>>> print("I'm in the interactive interpreter")
I'm in the interactive interpreter
>>> 24/3
8.0
>>> mylist = [1,2,3,4]
>>> print(mylist)
[1, 2, 3, 4]
>>> █
```

IPython

- Interaktive graphische Konsole für Python, Arbeitspferd bei der Analyse von Daten
- Tab completion, command history mit Pfeiltasten
- In-line-Editierung und komfortables Cut-and-Paste von Code
- Objektinspektor, automatische Extraktion von Docstrings, einfache Interaktion mit dem Betriebssystem



```
IPython
File Edit View Kernel Magic Window Help
kernel 0 x kernel 1 x (kernel 0) slave x

Python 2.7.2+ (default, Oct 4 2011, 20:06:09)
Type "copyright", "credits" or "license" for more information.

IPython 0.12 -- An enhanced Interactive Python.
? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.
%gui? -> A brief reference about the graphical user interface.

Welcome to pylab, a matplotlib-based Python environment [backend:
module://IPython.zmq.pylab.backend_inline].
For more information, type 'help(pylab)'.

In [1]: imshow(imread('baboon.png'))
Out[1]: <matplotlib.image.AxesImage at 0x401c650>

0
100
200
300
400
500
0 100 200 300 400 500

In [2]:
```


Python-Skripte

Längere Befehlsketten werden als Skripte in einfachen Textfiles mit der Endung `.py` abgespeichert. Die ersten beiden Zeilen des Files müssen folgende Form haben:

```
#!/usr/bin/env python
# -*- coding: UTF-8 -*-
```

- Kommentare werden mit `#` eingeleitet.
- Zeile 1: Pfad zum Python-Interpreter
- Zeile 2: Zeichencodierung, Standard ASCII, hier UTF-8 für Umlaute etc.
- Aufruf des Skripts durch

```
$ python myprogram.py
```

oder `x`-Flag setzen.

Spyder

The screenshot displays the Spyder Python IDE interface. The main editor window shows IPython code and its output. The code includes help documentation for IPython, followed by two execution blocks: `In [1]: x = linspace(-10, 10)` and `In [2]: plot(x, x**3)`. The output for the second block shows a `matplotlib.lines.Line2D` object. Below the code, a plot of $y = x^3$ is displayed, with the x-axis ranging from -10 to 10 and the y-axis from -1000 to 1000. The plot shows a blue curve passing through the origin. The bottom panel contains a console window with messages from the IPyKernel, including instructions on how to connect another client and the current execution time of 00:09:06. On the right side, the Variable explorer shows three variables: `e` (float, 2.7182818284590451), `pi` (float, 3.1415926535897931), and `x` (float64 array, `array([-10. , -9.5918367..., -8.367346 ...])`). Below the variable explorer, the Object inspector shows the `linspace` function from `numpy.core.function_base`, with its parameters and return value detailed.

File Edit Search Source Run Tools View ?

IPython (IPK1)

IPython 0.12.dev -- An enhanced Interactive Python.
? -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help -> Python's own help system.
object? -> Details about 'object', use 'object??' for extra details.
%gui? -> A brief reference about the graphical user interface.

In [1]: `x = linspace(-10, 10)`

In [2]: `plot(x, x**3)`

Out[2]: `[<matplotlib.lines.Line2D at 0x3033030>]`

1000
500
0
-500
-1000
-10 -5 0 5 10

In [3]:

IPython (IPK1) Editor - D:\Python\python\IPython\frontend\qt\kernelmanager.py

Console

Python 1 IPyKernel 1 00:09:06

[IPKernelApp] To connect another client to this kernel, use:
[IPKernelApp] --existing --shell=62459 --iopub=64449 --stdin=58721 --hb=2481

Internal console Console

Permissions: RW End-of-lines: CRLF Encoding: UTF-8-GUESSED Line: 169 Column: 1

Variable explorer

Name	Type	Size	Value
e	float	1	2.7182818284590451
pi	float	1	3.1415926535897931
x	float64	(50,)	array([-10. , -9.5918367..., -8.367346 ...])

Variable explorer Outline

Object inspector

Source Console Object `linspace` Options

`linspace(start, stop, num=50, endpoint=True, retstep=False)`
Function of `numpy.core.function_base` module

Return evenly spaced numbers over a specified interval.

Returns num evenly spaced samples, calculated over the interval `[start, stop]`.

The endpoint of the interval can optionally be excluded.

Parameters

start: scalar
The starting value of the sequence.

stop: scalar
The end value of the sequence, unless endpoint is set to False. In that case, the sequence consists of all but the last of `num + 1` evenly spaced samples, so that stop is excluded. Note that the step size changes when endpoint is False.

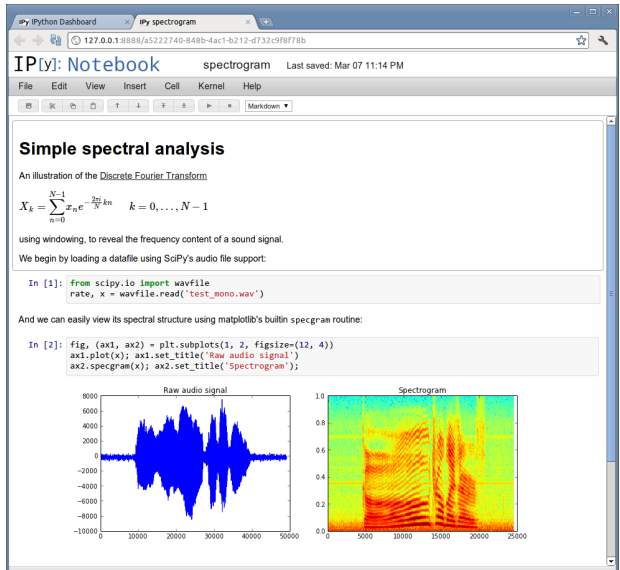
num: int, optional
Number of samples to generate. Default is 50.

endpoint: bool, optional
If True, stop is the last sample. Otherwise, it is not included. Default is True.

retstep: bool, optional
If True, return (samples, step), where step is the spacing between samples.

IPython notebooks

Code-Schnipsel
in kleinen
Ipython-Shells
können jederzeit
interaktiv neu
ausgeführt oder
abgeändert
werden.



Übersicht

1 Python

2 Wissenschaftliches Rechnen in Python

Der Scientific Python Software Stack

- Basis: **Numpy** - numerical Python: basiert auf den gleichen numerischen C/Fortran-Bibliotheken wie Matlab (LAPACK, BLAS, ATLAS). Sehr schnelle Operationen mit Vektoren, Matrizen, hochdimensionalen Arrays.
- **Scipy** - scientific Python: basierend auf Numpy, numerische Integration, Differentialgleichung, FFT, Signal- und Bildverarbeitung, Interpolation, Optimierung, Statistik, lineare Algebra, ...
- **Matplotlib** - mathematical plotting library: 2D- und 3D-Graphiken und Diagramme (ähnlich wie Matlab, Mathematika)
- **IPython** - interaktive Shell zur explorativen Entwicklung, Basis für die Matlab-artige IDE *Spyder* und für IPython-Notebooks.
- Weitere Pakete: Pandas, Sympy, ...

Numpy

- Python als Interpreter-Sprache läuft auf einer *virtuellen Maschine*, die eine Abstraktionsebene zwischen dem Code und der Hardware zur Verfügung stellt. Der Code wird dadurch portabel, läuft aber deutlich langsamer als kompilierter Code (z.B. Fortran, C, C++).
- Der Python-Interpreter kann aber kompilierte Bibliotheken aufrufen. Rechenintensive Algorithmen der Numerik und Signalverarbeitung werden daher nicht in Python implementiert, sondern in vorkompilierten Fortran- oder C-Bibliotheken.
- Numpy dient zur hochperformanten, vektorisierten Verarbeitung mehrdimensionaler Arrays und dient als Grundlage der Signalverarbeitung in Python.

Matplotlib

- Matplotlib ist ein Python-Paket zur Erzeugung von zwei- und dreidimensionalen wissenschaftlichen Diagrammen und Grafiken.
- Die erzeugten Grafiken haben Druckqualität und können direkt in Fachartikeln, Publikationen oder Berichten verwendet werden.
- Alle Aspekte der Plots können durch Anweisungen im Programm kontrolliert werden. Dadurch bleibt die Reproduzierbarkeit gewährleistet, wenn z.B. neue Daten dargestellt werden sollen.
- Mehr Informationen unter <http://matplotlib.org/>,
<http://matplotlib.org/gallery.html>,
<http://www.loria.fr/~rougier/teaching/matplotlib>,
<http://scipy-lectures.github.io/matplotlib/matplotlib.html>

Alles weitere...

- Installieren Sie Python auf Ihrem Rechner (Anleitung in Moodle)
- Programmieren Sie die Tutorien b. - e. in Moodle nach.
- Deadline: Testat No. 1, hier werden u.a. Ihre bis dahin erworbenen Python-Kenntnisse abgefragt.