Python - Open-Source-Werkzeuge für Wissenschaftler und Ingenieure

Chemnitzer Linux-Tage 2012, 18. März 2012

Autor: Dr.-Ing. Mike Müller
E-Mail: mmueller@python-academy.de

Übersicht

- besondere Situation von Wissenschaftlern und Ingenieuren
- Python
- wichtige Bibliotheken
- Entwicklung der Community

Wissenschaftler und Ingenieure

- Programmieren als Teil der Problemlösung
- traditionell C und FORTRAN
- kaum formale Ausbildung im Programmieren
- einfach zu erlernen, wirkungsvolle Werkzeuge nötig

Python

- entstanden 1989/90
- Guido van Rossum
- jetzt bei Version 2.7/3.2
- oft als Script-Sprache bezeichnet
- eignet sich für viele Aufgaben unterschiedlichster Größe

Merkmale

- sehr schnelle Entwicklung
- gut lesbare Syntax
- kompakter Sprachumfang
- Weg der geringsten Überraschung
- konsistent

Interaktiver Prompt

- schnelles Ausprobieren
- sofortige Reaktion

```
>>> 1 + 1
2
```

- Demo

Datenstrukturen

- Listen
- Dictionarys
- Sets
- Generatoren

Unterstützt mehrere Paradigmen

- prozedural
- objekt-orientiert
- funktional

Python für Wissenschaftler und Ingenieure

- passt sehr gut
- einfach zu erlernen
- viele Bibliotheken
- Einbinden von C und FORTRAN relativ einfach

NumPy

- meist genutzte Bibliothek für wissenschaftliche Anwendungen
- für die Arbeit mit numerischen Arrays
- orientiert sich an MATLAB und Array-Syntax von FORTRAN90
- 23 Datentypen
- effizient, da in C und FORTRAN implementiert

Schnelle Array-Erstellung

```
>>> import numpy
>>> numpy.arange(25).reshape(5, 5)
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24]])
```

Operationen mit ganzen Arrays

```
>>> a
array([[0, 1, 2],
       [3, 4, 5],
       [6, 7, 8]])
>>> b
array([[10, 11, 12],
       [13, 14, 15],
       [16, 17, 18]])
>>> a + b
array([[10, 12, 14],
       [16, 18, 20],
       [22, 24, 26]])
```

NumPy - Slicing

```
>>> c
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24]])
>>> c[0]
array([0, 1, 2, 3, 4])
```

NumPy - Slicing

```
>>> c[:, 0]
array([ 0,  5, 10, 15, 20])
>>> c[1:-1, 1:-1]
array([[ 6,  7,  8],
       [11, 12, 13],
       [16, 17, 18]])
```

NumPy - Ohne Schleifen

```
>>> c[c>5]
array([ 6,  7,  8,  9, 10, 11, 12, 13, 14, 15,
       16, 17, 18, 19, 20, 21, 22, 23, 24])
>>> numpy.where(c > 5, c, 0)
array([[ 0,  0,  0,  0,  0],
       [ 0,  6,  7,  8,  9],
       [10, 11, 12, 13, 14],
       [15, 16, 17, 18, 19],
       [20, 21, 22, 23, 24]])
```

NumPy - Universal Functions

```
>>> numpy.sin(c)
array([[ 0.        ,  0.84147098,  0.90929743,  0.14112001, -0.7568025 ],
       [-0.95892427, -0.2794155 ,  0.6569866 ,  0.98935825,  0.41211849],
       [-0.54402111, -0.99999021, -0.53657292,  0.42016704,  0.99060736],
       [ 0.65028784, -0.28790332, -0.96139749, -0.75098725,  0.14987721],
       [ 0.91294525,  0.83665564, -0.00885131, -0.8462204 , -0.90557836]])
```

SciPy

- Zusammenstellung vieler Erweiterungsmodule für unterschiedliche Zwecke
- Statistik
- Interpolation
- lineare Algebra
- Optimierung
- Signalverarbeitung
- sehr groß und wächst weiter

IPython

- verbesserte interaktive Umgebung
- Hilfe
- Nummerierung [In] und [Out]
- Vervollständigung
- Shell-Kommandos

IPython - Nummerierung und Hilfe

```
IPython 0.10 -- An enhanced Interactive Python.
?         -> Introduction and overview of IPython's features.
%quickref -> Quick reference.
help      -> Python's own help system.
object?   -> Details about 'object'. ?object also works, ?? prints more.

In [1]: a = 4

In [2]: b = 5

In [3]: a + b
Out[3]: 9

In [4]: _3
Out[4]: 9

In [5]: import sys

In [6]: ?sys
Type:           module
Base Class:     <type 'module'>
String Form:    <module 'sys' (built-in)>
Namespace:      Interactive
Docstring:
    This module provides access to some objects used or maintained by the
interpreter and to functions that interact strongly with the interpreter.

Dynamic objects:

argv -- command line arguments; argv[0] is the script pathname if known
path -- module search path; path[0] is the script directory, else ''
modules -- dictionary of loaded modules

displayhook -- called to show results in an interactive session
excepthook -- called to handle any uncaught exception other than SystemExit
  To customize printing in an interactive session or to install a custom
  top-level exception handler, assign other functions to replace these.

exitfunc -- if sys.exitfunc exists, this routine is called when Python exits
```

IPython - Vorschläge

```
In [8]: sys.a
sys.api_version sys.argv

In [8]: sys.v
sys.version      sys.version_info

In [8]: sys.p
sys.path                sys.path_importer_cache sys.prefix
sys.path_hooks          sys.platform            sys.py3kwarning

In [8]: ls
```

Matplotlib

- inspiriert von MATLAB
- schnell, interaktive 2-D-Grafiken
- OO-API vorhanden
- Speicherung in verschiedenen Vektor- und Pixelformaten
- Animationen möglich

Pylab - Funktionalität von MATLAB

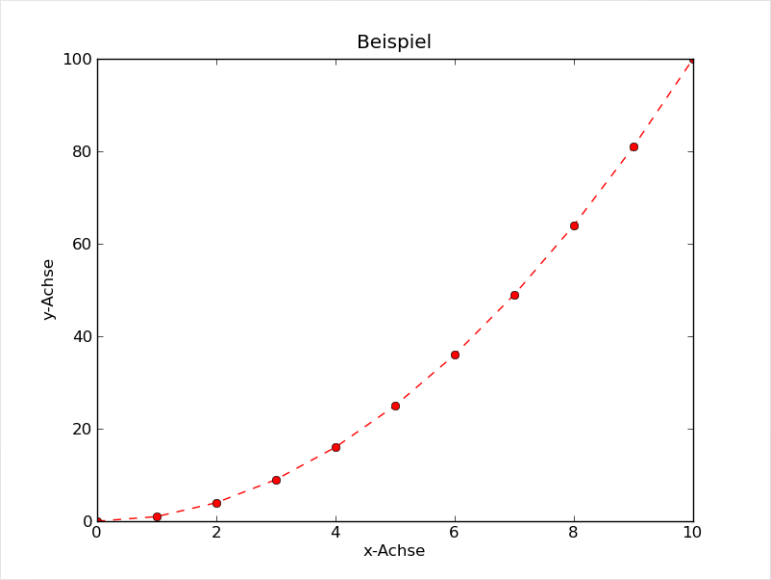
Aufruf von IPython im pylab-Modus:

```
ipython -pylab
```

Matplotlib - Beispiel

```
In [1]: x = arange(11)
In [2]: y = x * x
In [3]: plot(x, y, 'r--o')
Out[3]: [<matplotlib.lines.Line2D object at 0x0B9FE8B0>]
In [4]: title('Beispiel')
Out[4]: <matplotlib.text.Text object at 0x0B9EA150>
In [5]: xlabel('x-Achse')
Out[5]: <matplotlib.text.Text object at 0x0B9D6E90>
In [6]: ylabel('y-Achse')
Out[6]: <matplotlib.text.Text object at 0x0B9E2810>
```

Matplotlib - Beispiel



SymPy

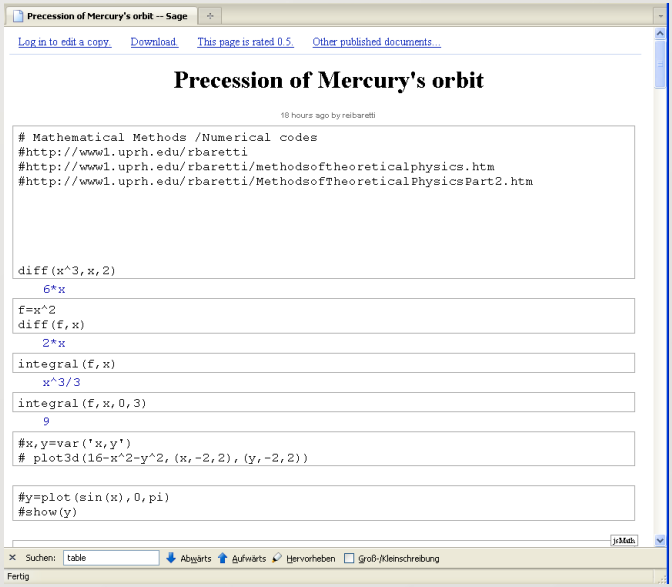
- symbolische Mathematik
- Ziel: Computer-Algebra-System

```
>>> from sympy import Symbol, cos
>>> x = Symbol("x")
>>> (1/cos(x)).series(x, 0, 10)
1 + x**2/2 + 5*x**4/24 + 61*x**6/720 + 277*x**8/8064 + O(x**10)
```

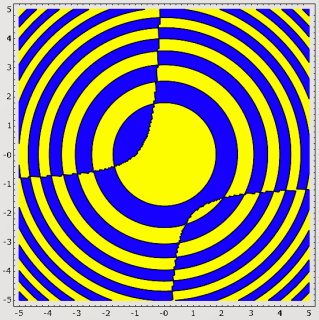
Sage

- als Ersatz von Magma, Maple, Mathematica und MATLAB
- Browser als GUI
- Python als Nutzerschnittstelle
- Verbindung von unterschiedlichen Bibliotheken in verschiedenen Sprachen

Sage



Sage

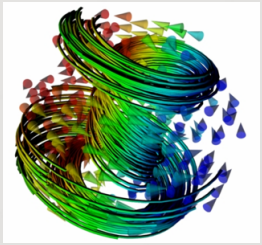


Sage:

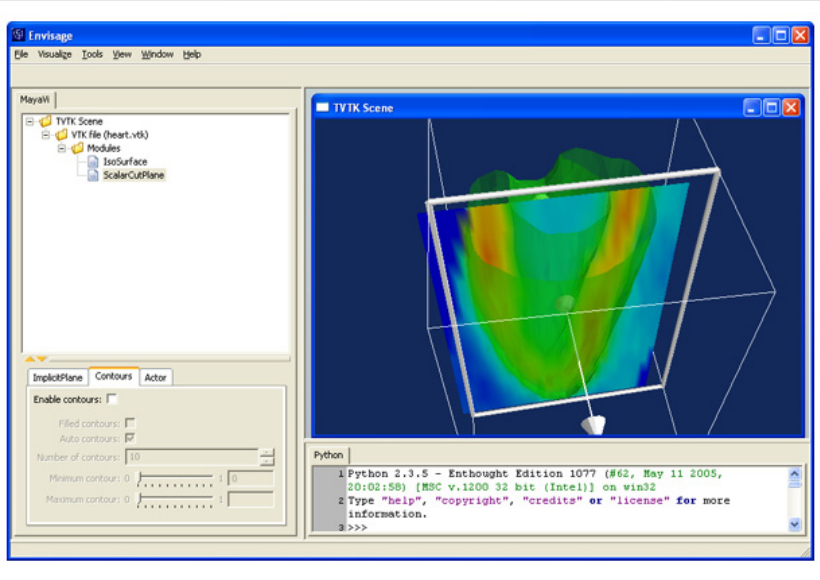
```
sage: var('x y')
sage: region_plot(sin(x^2 + y^2)/(1+y+x*y) > 0, (-5,5), (-5,5), ...
      incol='#ffff7f', outcol='#7f7fff', bordercol='black', ...
      plot_points=300).show(aspect_ratio=1)
```

MayaVi

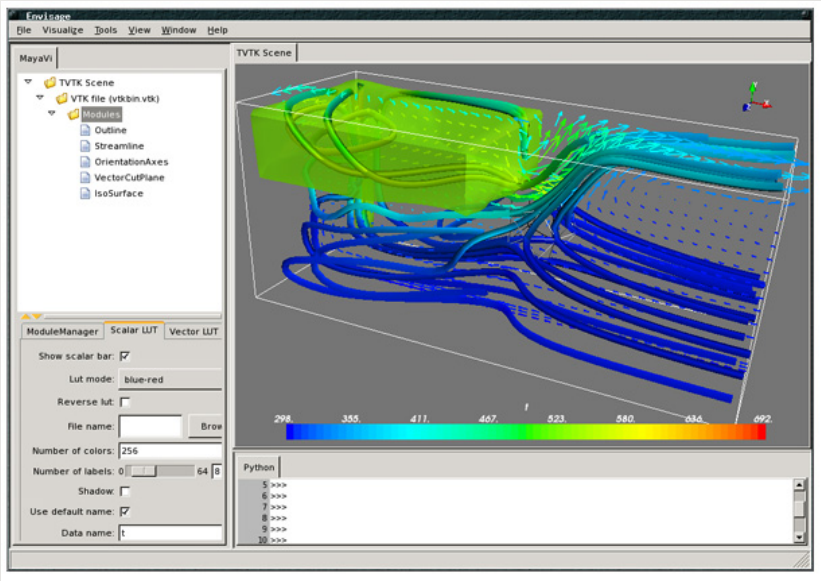
- 3D-Darstellungen
- nutzt VTK (C++-Bibliothek für 3D-Visualisierung)
- GUI-Anwendung
- mit Python-Skripten steuerbar
- Bibliothek



MayaVi - Beispiel



MayaVi - Beispiel



Daten speichern

- Text
- Datenbanken
- Excel
- netCDF
- HDF5
- ...

Verbinden mit anderen Sprachen

- C-API --> per Hand
- Cython == Python mit C-artigen Bestandteilen
  - statische Typisierung
  - direkter Ruf von C-Funktionen
  - unterstützt C++
- SWIG, SIP
- f2py

Cluster und Paralleles Rechnen

- multiprocessing
- MPI: mpi4py
- Pyro
- Cloud-Anbieter

GPGPUs

- PyCUDA
- PyOpenCL
- Copperhead

Distributionen

- Bündelung vieler Bibliotheken für verschiedene Plattformen
- aufeinander abgestimmt
- Probleme mit Abhängigkeiten und Kompilierung gelöst
- C und FORTRAN-Compiler
- Nicht-Python-Software
- Python(xy) 
- Enthought Python Distribution 

Community

- seit Mitte der 1990er Jahre Bibliotheken, Mailing-Listen etc.
- starke Zunahme der Anzahl der Nutzer in den letzten Jahren
- viele neue Bibliotheken in den letzten Jahren
- Konferenzen speziell für Python für wissenschaftliche Anwendungen

SciPy-Konferenz

- Konferenz in den USA
- bisher 10 mal
- 2011 ca. 200 Teilnehmer

EuroSciPy-Konferenz

- europäische Variante
- bisher viermal, zweimal Leipzig, zweimal Paris
- 2011 ca. 200 Teilnehmer
- 2012 in Brüssel

Ausblick

- weitere Zunahme der Verbreitung erwartet
- NumPy + Matplotlib == MATLAB-Ersatz
- Forschungsinstitutionen wie DLR nutzen Python intensiv
- andere Forschungsinstitute steigen auf Python um
- Nutzung für die Modellierung von Finanzprodukten in den USA per Verordnung gefordert

Fragen?