Python basierte Werkzeuge für wissenschaftliches Arbeiten

Carsten Knoll

Technische Universität Dresden

Institut für Regelungs- und Steuerungstheorie

28. April 2010

Gliederung

1. Einführung
2. Die Programmiersprache Python
3. Visualisierung
4. Wissenschaftliche Pakete
5. Python-basierte Werkzeuge
6. Fazit und Ausblick
7. Links

Gliederung

1. Einführung
2. Die Programmiersprache Python
3. Visualisierung
4. Wissenschaftliche Pakete
5. Python-basierte Werkzeuge
6. Fazit und Ausblick
7. Links

Motivation

- Warum Vortrag über Programmiersprache?
- Fehlender universitärer Anspruch, Zeitverschwendung, Propaganda?

- Weil sie es Wert ist!

Motivation

- Warum Vortrag über Programmiersprache?
- Fehlender universitärer Anspruch, Zeitverschwendung, Propaganda?

⚠ Weite Teile des Vortrages hochgradig subjektiv

- Weil sie es Wert ist!

Kurzbeschreibung

en.wikipedia:

Python is a **general-purpose**, **high-level** programming language. Its design philosophy emphasizes programmer **productivity** and code **readability**. Python's core syntax and semantics are minimalist, while the standard library is large and comprehensive. ...

- relativ jung, sehr dynamisch
- Schwerpunkte: Produktivität, Lesbarkeit und Erlernbarkeit
- Quelloffen und frei

Gliederung

1. Einführung
2. Die Programmiersprache Python
3. Visualisierung
4. Wissenschaftliche Pakete
5. Python-basierte Werkzeuge
6. Fazit und Ausblick
7. Links

Merkmale I

- Intuitive und oft selbsterklärende Syntax
  „Python liest sich fast wie Pseudo-code“
- Einrückungen haben semantische Bedeutung
  $\rightarrow$ Lesbarkeit erzwungen.
- Docstrings (Vom Programm aus zugängliche Kurzdokumentation des Programmteils)
- Ziel: Programmieren auf hohem Niveau
  - Mächtige eingebaute Datentypen (`list`, `dict`, `set`, ...)
  - „Batterien inklusive“
    $\rightarrow$ Sehr umfangreiche Standard-Bibliothek
    (> 280 Module: ... `anydb`, ... `zipfile`, ...)

Merkmale I

- Intuitive und oft selbsterklärende Syntax
  „Python liest sich fast wie Pseudo-code“
- Einrückungen haben semantische Bedeutung
  $\rightarrow$ Lesbarkeit erzwungen.
- Docstrings (Vom Programm aus zugängliche Kurzdokumentation des Programmteils)
- Ziel: Programmieren auf hohem Niveau
  - Mächtige eingebaute Datentypen (`list`, `dict`, `set`, ...)
  - „Batterien inklusive“
    $\rightarrow$ Sehr umfangreiche Standard-Bibliothek
    (> 280 Module: ... `anydb`, ... `zipfile`, ...)

Merkmale II

- Konzept: Alles ist ein Objekt (auch: Funktionen, Klassen, importierte Module)
- Trotzdem: Nutzung von *verschiedenen* Paradigmen (prozedural, objektorientiert, funktional)
- Modularisierung (Module, Pakete, Namensräume)
  $\rightsquigarrow$ Wiederverwendung
- Gute Möglichkeiten zur Fehlervermeidung und -suche: Exceptions, Tracebacks, integrierter Debugger, Log-Modul
- Möglichkeit von interaktiven Sitzungen $\rightarrow$ **IPython** (auch eingebettet)

Merkmale III

- Dynamische Typisierung (=`MATLAB`, $\neq$ `Java, C`)
  - Weniger Unterstützung durch die IDE
  - Manche Fehler fallen erst zur Laufzeit auf
  + Freiheit und Flexibilität
- Interpretierter Bytecode (=`Java, Ruby`)
  - Ausführungsgeschwindigkeit
  + Kein langer Kompilierungsvorgang

IPython

- Verbesserte interaktive Shell (wie `MATLAB, Scilab`)
- Sehr nützlich:
  - History, Autovervollständigung (TAB)
  - Sehr aufschlussreiche Fehlerausgabe
  - `?`- und `??` -Kommando
- Kann in eigenes Programm eingebettet werden

Bsp: Funktionen von Funktionen

- Bsp. aus der Funktionalanalysis VL:

  Skalierungsoperator

$$\sigma_\alpha : H_0 \to H_0,$$
$$\Phi \mapsto (x \mapsto \Phi(\alpha x))$$

```
def sigma(alpha, Phi):
    def newfnc(x):
        return Phi(alpha*x)
    return newfnc
```

- andere Anwendungen: Totzeitoperator, „Cachen“, Dynamische Programmierung, ...

Warum Python in Wissenschaft nutzen?

- Python-Interpreter: Freie Software
  - → Sehr gute Vorausetzung für Forschung und Lehre (Nachvollziebar- und Verfügbarkeit, Kosten)
  - → $\exists$ sehr viele gute *freie* Erweiterungen
- Erlernbarnbarkeit
- Produktivität
- Universalität
- Plattformunabhängigkeit
- Verbreitung (Erfahrungsaustausch, Speziallösungen)

Warum Python in Wissenschaft nutzen?

- Python-Interpreter: Freie Software
  - → Sehr gute Vorausetzung für Forschung und Lehre (Nachvollziebar- und Verfügbarkeit, Kosten)
  - → $\exists$ sehr viele gute *freie* Erweiterungen
- Erlernbarnbarkeit
- Produktivität
- Universalität
- Plattformunabhängigkeit
- Verbreitung (Erfahrungsaustausch, Speziallösungen)

Wie Python nutzen? (Eigene Erfahrungen)

- Numerisch Rechnen, Simulieren, Visualisieren (an Stelle `MATLAB`)
- Symbolisch Rechnen (an Stelle `Maple`)
- Hilfsarbeiten (z.Bsp. automatische Eingangstest-Erstellung)
- Interaktion mit u. Steuerung von Prozessen (an Stelle `LabView`)

Wie Python nutzen? (Eigene Erfahrungen)

- Numerisch Rechnen, Simulieren, Visualisieren (an Stelle `MATLAB`)
- Symbolisch Rechnen (an Stelle `Maple`)
- Hilfsarbeiten (z.Bsp. automatische Eingangstest-Erstellung)
- Interaktion mit u. Steuerung von Prozessen (an Stelle `LabView`)

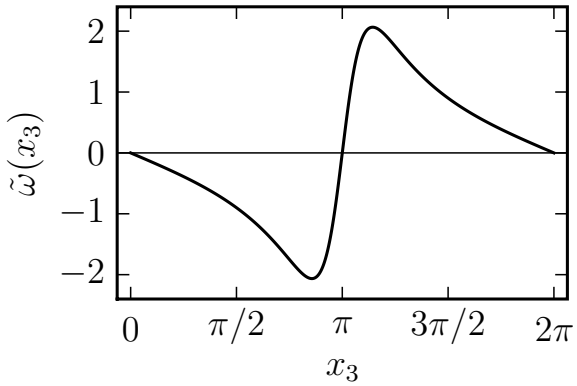
  (WHK-Tätigkeit bei Fraunhofer IIS und iapp)

  - Hardware-Ansteuerung (RS232, Parallelport, GPIB, TCP/IP...)
  - Grafische-Nutzer-Interaktion
  - (Weiche Echtzeit mit RTOS)

Gliederung

1. Einführung
2. Die Programmiersprache Python
3. Visualisierung
4. Wissenschaftliche Pakete
5. Python-basierte Werkzeuge
6. Fazit und Ausblick
7. Links

matplotlib



- Schwerpunkt 2D
- Inspiriert von MATLABs plot-Funktion
- Gute LaTeXUnterstützung
- Schöne Grafiken, bis ins Detail anpassbar
- Kantenglättung, Transparenz
- Innovative Integration von Zoomen und Verschieben
- Export von Bildern (pdf, png, jpg, svg)
- $\exists$ Sehr viele Beispiele mit Quellcode

pygame

- Fokus: Geschwindigkeit
- $\longrightarrow$ Visualisierung parallel zur Simulation
- Nutzerinteraktion (mit Maus, Tastatur) sehr einfach

vtk

- C++ Bibliothek zur 3D-Visualisierung
- Sehr gute Python Schnittstelle

Gliederung

1. Einführung
2. Die Programmiersprache Python
3. Visualisierung
4. Wissenschaftliche Pakete
5. Python-basierte Werkzeuge
6. Fazit und Ausblick
7. Links

numpy

- Herzstück: n-dimensionale Array-Klasse
- Funktionalität:
  - Standard-Operationen
  - Lineare Algebra (Eigenwerte, Inverse, LGS lösen)
  - FFT
  - ausgeprägte Zufalssgeneratoren (viele Verteilungen)
- Geschwindigkeits-kritische Teile in C(++) oder Fortran implementiert
  BLAS, LAPACK
- Unterschied zu `MATLAB`: Standard-Datentyp ist Array statt Vektor
  $\hookrightarrow$ andere Multiplikationsregeln („Broadcasting“)

scipy

- Sammlung von Paketen auf Basis von `numpy`
  - Optimierung
  - Statistik
  - Signalverarbeitung
  - DGL-Integratoren (ODE-Solver)
  - Interpolation, Splines, „Daten-Fitten“
  - spezielle Funktionen (Besselfktn., ellipt. Integrale, ...)
  - Schnittstellen zum Daten-In/-Export (aus txt, Bildern, Datenbanken)
- `ipython` + `scipy` + `numpy` + `matplotlib` $\approx$ MATLAB_{0815}

sympy

- Computer-Algebra-Paket in python
- Bietet:
  - Grundlegende Rechnenoperationen
  - Termmanipulation (Zusammenfassen und Ausmultiplizieren)
  - Symbolisches Differenzieren und Integrieren
  - Grenzwertberechnung
  - Matritzen-Rechnung (symbolische Determinante, Inverse, ...)
  - Lösen von Gleichungen und Gleichungssystemen
  - Pretty-Printing und LaTeX-Ausgabe
  - ...
- Vergleich mit `Maple`, `Mathematica`, `Mupad`, ... ?
- Sehr großer Vorteil: symbolische Berechnungen nahtlos in Programmablauf integrierbar

weitere Pakete

- `ScientificPython` (Quaternionen, Automatisches Differenzieren, ...)
- `biopython` (Molekularbiologie)
- `parallelpython` (Rechnen auf mehreren Kernen/Rechnern)
- ...
- $\exists$ sehr viele weitere Pakete, Module, Code-Schnipsel zu teilweise sehr speziellen Problemen
- außerdem Schnittstellen zu `octave`, `R`, `Matlab`, ...

- $\hookrightarrow$ Regelung / Steuerung?

  bisher nur einfaches LTI-Modul (Rechnen mit ÜF, Sprungantwort, Bode-Diagramm)

Gliederung

1 Einführung

2 Die Programmiersprache Python

3 Visualisierung

4 Wissenschaftliche Pakete

5 Python-basierte Werkzeuge

6 Fazit und Ausblick

7 Links

sage

- A Computer **S**ystem for **A**lgebra and **G**eometry **E**xperimentation
- Vereinigung von verschiedenen freien Mathe-Werkzeugen:
  - `Maxima` (allgem. symb. Rechnen)
  - `numpy/scipy`
  - `R` (Statistik)
  - `GAP` (Gruppen-Theorie)
  - `Singular` (polynomiale Algebra)
  - `NetworkX` (Graphen)
  - ...
- Ziel: Einheitliche Schnittstelle: Python
- Interessantes Feature: „Notebook“-Webserver

- ca. 1 GB-Alles-Inklusive-Paket
- Empfehlung für Studenten?

sage

- A Computer **S**ystem for **A**lgebra and **G**eometry **E**xperimentation
- Vereinigung von verschiedenen freien Mathe-Werkzeugen:
  - `Maxima` (allgem. symb. Rechnen)
  - `numpy/scipy`
  - `R` (Statistik)
  - `GAP` (Gruppen-Theorie)
  - `Singular` (polynomiale Algebra)
  - `NetworkX` (Graphen)
  - ...
- Ziel: Einheitliche Schnittstelle: Python
- Interessantes Feature: „Notebook“-Webserver

- ca. 1 GB-Alles-Inklusive-Paket
- Empfehlung für Studenten?

PyMBS

- **Py**thon **M**ulti **B**ody **S**ystems
- Automatisiertes Aufstellen von Bewegungsgleichungen von Mehrkörpersystemen
- Code-Generierung: `Modelica, MATLAB, C, Fortran`
- Textuelle Modellbeschreibung + Visualisierung
- Entwickler: 

  (drei TUD-Doktoranden (Professur für Baumaschienen und Fördertechnik) und anfangs ein Hobbyprogrammierer)

mercurial

- Python-basiertes *verteiltes Versionskontrollsystem*
- Dateibasiert $\rightarrow$ kein Server erforderlich
- Versionsgeschichte z. Bsp. von:
  - Programm-Quellcode
  - Latex-Code
  - Binärdaten
- Unterschiede zwischen Versionen visualisieren (graphisch)
- $\rightarrow$ Hilft bei Sicherheit und Ordnung
- Plattformunabhängig, $\exists$ gute GUIs

inkscape

- Freies Vektorgrafikprogramm (Vorbild: `CorelDraw ?`)
- Sehr kompfortabel zu nutzen
- Blockdiagramme, Schema-Zeichnungen, etc.
- Besser als `xfig` (kann mehr, man ist schneller)
- Python-Plugins (z.Bsp. für LaTeX-Unterstützung)

wikidpad

- „Desktop-Wiki“
- Programm zur strukturierten Ablage von Informationen
  - Interne + externe Verknüpfungen, Suchfunktion
  - Bilder aus Zwischenablage einfügen (z.Bsp Screenshots)
  - Hervorhebung
  - HTML-Export
- geeignet für:
  - Persönliche Notizen
  - Stoffsammlungen,
  - Geordnete Linksammlungen
  - Code Schnipsel
- Python-Kern u. -Plugins (z.Bsp. für LaTeX-Unterstützung)

wikidpad

- „Desktop-Wiki“
- Programm zur strukturierten Ablage von Informationen
  - Interne + externe Verknüpfungen, Suchfunktion
  - Bilder aus Zwischenablage einfügen (z.Bsp Screenshots)
  - Hervorhebung
  - HTML-Export
- geeignet für:
  - Persönliche Notizen
  - Stoffsammlungen,
  - Geordnete Linksammlungen
  - Code Schnipsel
- Python-Kern u. -Plugins (z.Bsp. für LaTeX-Unterstützung)

„IDE for your thoughts“

Gliederung

1. Einführung
2. Die Programmiersprache Python
3. Visualisierung
4. Wissenschaftliche Pakete
5. Python-basierte Werkzeuge
6. Fazit und Ausblick
7. Links

persönliches Fazit

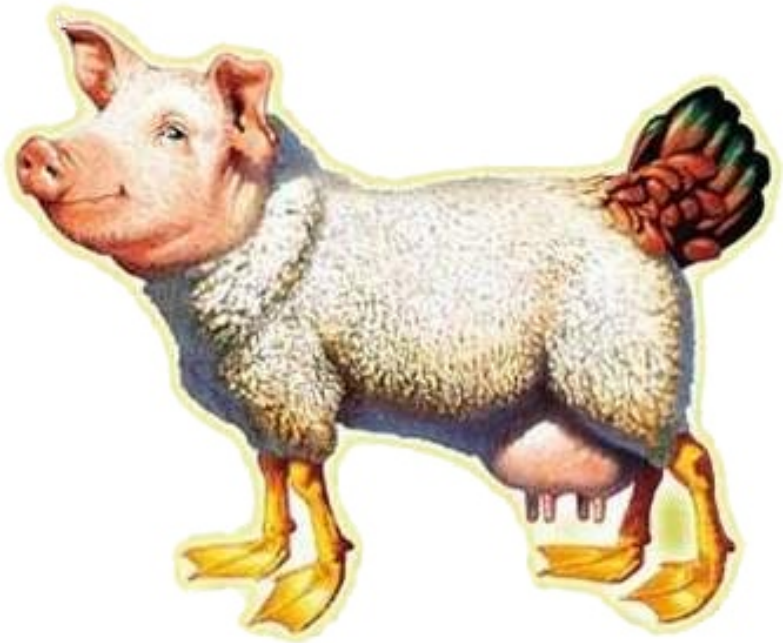
python:

- Vereinigt Universalität und Problemorientierung
- Viele gute Pakete für wissenschaftliches Arbeiten
- Durchdachte Sprache $\rightarrow$ durchdachte Programme

persönliches Fazit

python:

- Vereinigt Universalität und Problemorientierung
- Viele gute Pakete für wissenschaftliches Arbeiten
- Durchdachte Sprache $\rightarrow$ durchdachte Programme
- Eierlegende Wollmilchsau



Zukunfsmusik

- Sehr offen für Fragen und Ideen in dem Zsh.

? Sammlung von themenbezogenen Code-Schnipseln, Skripten

? Angebot für Studenten: Dokus, Übungsaufg. und Konsultationen
  + Für Studenten: erweitern Werkzeugkasten
  + Für uns: Kontakt zu interessierten/motivierten Studenten

Gliederung

1. Einführung
2. Die Programmiersprache Python
3. Visualisierung
4. Wissenschaftliche Pakete
5. Python-basierte Werkzeuge
6. Fazit und Ausblick
7. Links

Links

- Generell meist erfolgreich: Suche nach Projektname (Bsp. sympy) oder mit Konstruktion:

  python <engl. Fachbegriff>

  Beispiel: *python lu decomposition*

- Auswahl interessanter Seiten:
  - `http://www.thomas-guettler.de/vortraege/python/einfuehrung.html`
  - `http://www.pythonxy.com/` (Komplett Distribution für Windows)
  - `http://matplotlib.sourceforge.net/gallery.html`
  - `http://www.scipy.org/NumPy_for_Matlab_Users`
  - `http://www.sagemath.org/`