



TECHNISCHE
UNIVERSITÄT
DRESDEN

Fakultät Elektrotechnik und Informationstechnik Institut für Regelungs- und Steuerungstheorie

ANWENDUNGSORIENTIERTER PYTHONKURS FÜR INGENIEURE

Organisatorisches und Python-Grundlagen

Carsten Knoll, Sebastian Voigt, Paul Schwanse

Dresden, 14.10. 2013



DRESDEN
concept
Erarbeitet aus
Mikroelektronik
und Kultur

Beteiligte Personen

- Dipl.-Ing. Carsten Knoll
 - Institut für Regelungs- und Steuerungstheorie
- Dipl.-Ing. Christoph Statz
 - Institut für Nachrichtentechnik
- Dipl.-Ing. Sebastian Voigt
 - Institut für Verarbeitungsmaschinen und mobile Arbeitsmaschinen
- Unterstützt durch:
 - Dr.-Ing. Ines Gubsch, Andreas Kunze, Dominik Pataky, Victoria Vinis
- Vielen Dank an
 - Sandra Döring (Medienzentrum), Volker Fischer (ZIH), ...

Hintergrund

- Python-Nutzung einzelner Mitarbeiter → Kontakt, Austausch
 - Feststellung: Mangel an Lehrveranstaltungen
- ⇒ Eigeninitiative: Organisation eines Kurses (ehrenamtlich)
- Bisherige Durchgänge: SoSe 2011, SoSe 2012, SoSe 2013
 - Aktuell: Finanzielle Förderung durch das Hochschuldidaktische Zentrum Sachsens (HDS)
(SHK-Mittel)
-
- Neu: „Blended Learning“-Angebot: Online- und Präsenz- Phasen im Wechsel
 - Online: Wissens-Vermittlung (Screencasts), Verständnisfragen-Quiz, Forum (OPAL)
 - Präsenz: Komplexe-Übung (Anwesenheit mehrerer Tutoren)

Warum Python? (1)

Python als Programmiersprache

- Klare, lesbare Syntax (wenig „Ballast“)
- Dynamische Typisierung (wie MATLAB)
- Objektorientiert, prozedural, funktional programmierbar
- Nützliche eingebaute Datentypen (`list`, `tuple`, `dict`, `set`)
- Einfache Modularisierung
- Gute Fehlerverwaltung (Exceptions)
- Umfangreiche Standardbibliothek
- Einbindung von externem Code (C, C++, Fortran)

⇒

- Leicht zu lernen
- Problemorientiert (mächtig und flexibel)
- Motivationspotenzial ↗, Frustrationspotenzial ↘

Außerdem: Plattformunabhängig, Open Source, große und aktive Community

Warum Python? (2)

Python als Werkzeug für Ingenieure:

- Symbolisches Rechnen (Ableiten, Integrieren, Gl. lösen, ...)
- Numerisches Rechnen (lin. Algebra, DGLn, Optimierung, ...)
- Visualisieren (2D, 3D, in Publikationsqualität)
- Grafische Benutzerschnittstelle (GUI)
- Kommunikation mit externen Geräten (RS232, GPIB, ...)
- Parallelisierung

} Kursinhalt

Didaktisches Konzept

„Programmieren lernt man durch *aktives* Lesen und Schreiben von Code“

Online-Phasen

- Überblicksvortrag als Screencast (Grundkonzepte, Befehle, Fallen)
„Fingerübungen“ → selber mittippen
- Empfehlung: Videos herunterladen und in geeignetem Media-Player (z.B. mplayer, VLC) anschauen. (Zurückspringen, Geschwindigkeitsanpassung, ...)

Präsenzveranstaltungen

- Komplexen Übungsaufgabe,
 - Untergliederung in schaffbare Teilaufgaben
 - Vorgabe von Quellcode-Fragmenten, Ausführliche Lösung im Netz
- paarweise Programmierung
 - Ermöglicht und erfordert Kooperation
- *Aktive* Betreuung (live + Forum)

Beleg (Überschaubare Programmieraufgabe)

- Bedingung für Leistungsschein
- Binäre Bewertung nach einer kurzen Diskussion über das Programm
- **eigene Themenvorschläge erwünscht**

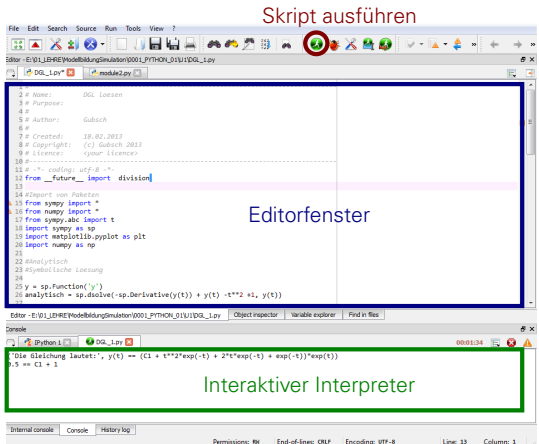
OPAL

- Lehrmaterial
- Kommunikationsmöglichkeiten
- Weiterführende Informationen (Links, ...)
- Evaluation (**jede Veranstaltung** (Online und Präsenz))

winpython

- Unter Linux: Installation von Python und zugehörigen Modulen über Versionsverwaltung bzw. PIP
- Empfehlung für Windows: <https://code.google.com/p/winpython/> enthält alle für uns wichtigen Pakete + IDE („Spyder“)
- **Wichtig:** Im Kurs wird Python 2.7 verwendet. (Nicht 3.x)

Spyder Entwicklungsumgebung (IDE)



Code-Beispiel

```
import math
print 'Hello World'
a = 10
b = 20.5
c = a+b+3**2
print math.sqrt(c)

while True:
    x = raw_input('Eingabe?') # gibt immer einen String
                               # zurueck
    if x == 'q':
        break # Schleife beenden
    print 'Ausgabe:', x
```

- Einrückungen haben syntaktische Bedeutung!
- DeFacto Standard: 4 Leerzeichen (in IDE: Blockweise mit <TAB>(→) und <SHIFT+TAB> (←))
- 01_Zusatz_Uebersicht_Python_Typen_Syntax.pdf

IPython

- *Interaktives* Arbeiten sehr nützlich
- besonders beim „Entdecken“ neuer Module, Features, ...
- IPython = „Interactive Python“ = verbesserte Python-Shell
 - History
 - Auto-Vervollständigung (<TAB>)
 - einfache Hilfe (<?> und <??>)
 - Farben, „magische“ Kommandos (z.B. `%paste`), ...
- lässt sich auch in eigene Programme einbetten:

```
from IPython import embed
...
x = 'abcdefg'
embed()
...
```

Zusammenfassung + Ausblick

Worum ging es?

- Organisatorische Infos
- Umgang mit `spyder`, Python-Grundlagen
- Übungsaufgabe

Wie gehts weiter?

- Evaluation (OPAL)
- Gedanken zu Belegthemen
- Revision des Stoffs (vor der nächsten Veranstaltung)