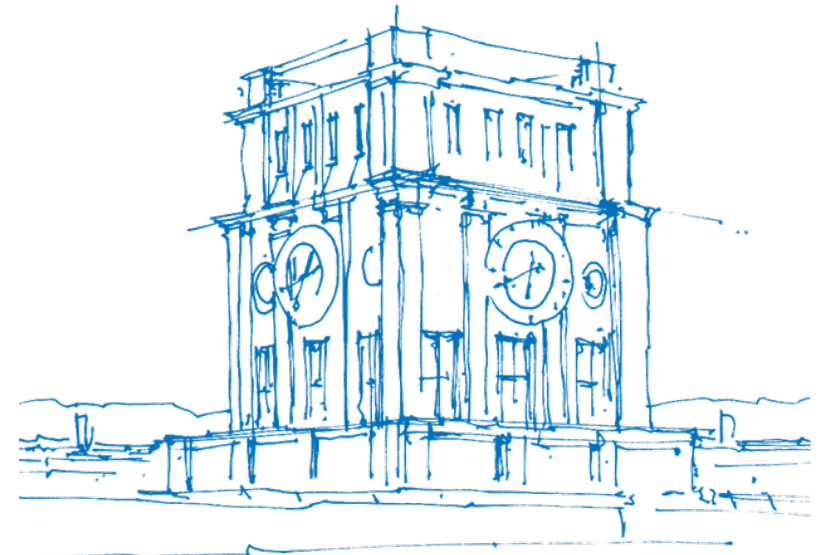


Einführung in die wissenschaftliche Programmierung

IN8008

Tobias Neckel
Scientific Computing in Computer Science,
Technische Universität München

Wintersemester 2016/2017



TUM Uhrenturm

Teil I

Organisatorisches und erste Schritte

Inhalte und Ziele

Einführung in die

- Keine Programmiererfahrung nötig
- Umgang mit PC/Betriebssystem sollte bekannt sein
- Mischung aus einfachen und schwierigen Teilen
- Kratzt an vielen Stellen nur an der Oberfläche

wissenschaftliche

- Mathematisches und physikalisches Grundverständnis
- Beispiele aus dem wissenschaftlichen Bereich
- Hauptziel bleibt Verständnis der Programmierkonzepte

Programmierung

- Hauptsächlich Implementierung
- Aber auch der Entwurf von Programmen
- Übliche Programmierkonstrukte und wissenschaftliche Bibliotheken
- Qualität

Organisatorisches

Raum und Zeit

- Modul IN8008
- Vorlesung Mo 10:15 - 11:45, Physik PH 2501 (HS1)
- Tutorübungen (vorauss.):
 - 1x** Di 14:15 - 15:45 Uhr
 - 1x** Mi 08:30 - 10:00 Uhr
 - 2x** Mi 12:15 - 13:45 Uhr
 - 1x** Fr 08:30 - 10:00 Uhr
 - 2x** Fr 12:30 - 14:00 Uhr
 - 2x** Fr 14:15 - 15:45 Uhr

Informationen

- Webseite: www5.in.tum.de → Teaching → Einführung in die wissenschaftliche Programmierung
- FRAGEN!
- Kein Vorlesungsskript; Folien-Printversion ⇒ Notizen machen

Organisatorisches II

Personen

- Alfredo Parra Hinojosa (hinojosa@in.tum.de): Übungsleitung
- Alexander, Johannes, Paul (EN) und Thomas: Tutoren
- Tobias Neckel (neckel@in.tum.de): Vorlesung

Nötige Software

- Python 2.5.x, 2.6.x oder 2.7.x (anaconda);
ACHTUNG: Python 3.x nicht abwärtskompatibel!
- Bibliotheken: numpy, scipy, matplotlib
- Beliebiger Editor (emacs, vi, gedit, Notepad++...)
- Im CIP-Pool sollte all das vorhanden sein
- **Hands-On:** In Übungen und in Vorlesung
⇒ gerne Notebook mitbringen
- Erste Übung (KW 42): Python-Install-Party :-)

Organisatorisches III

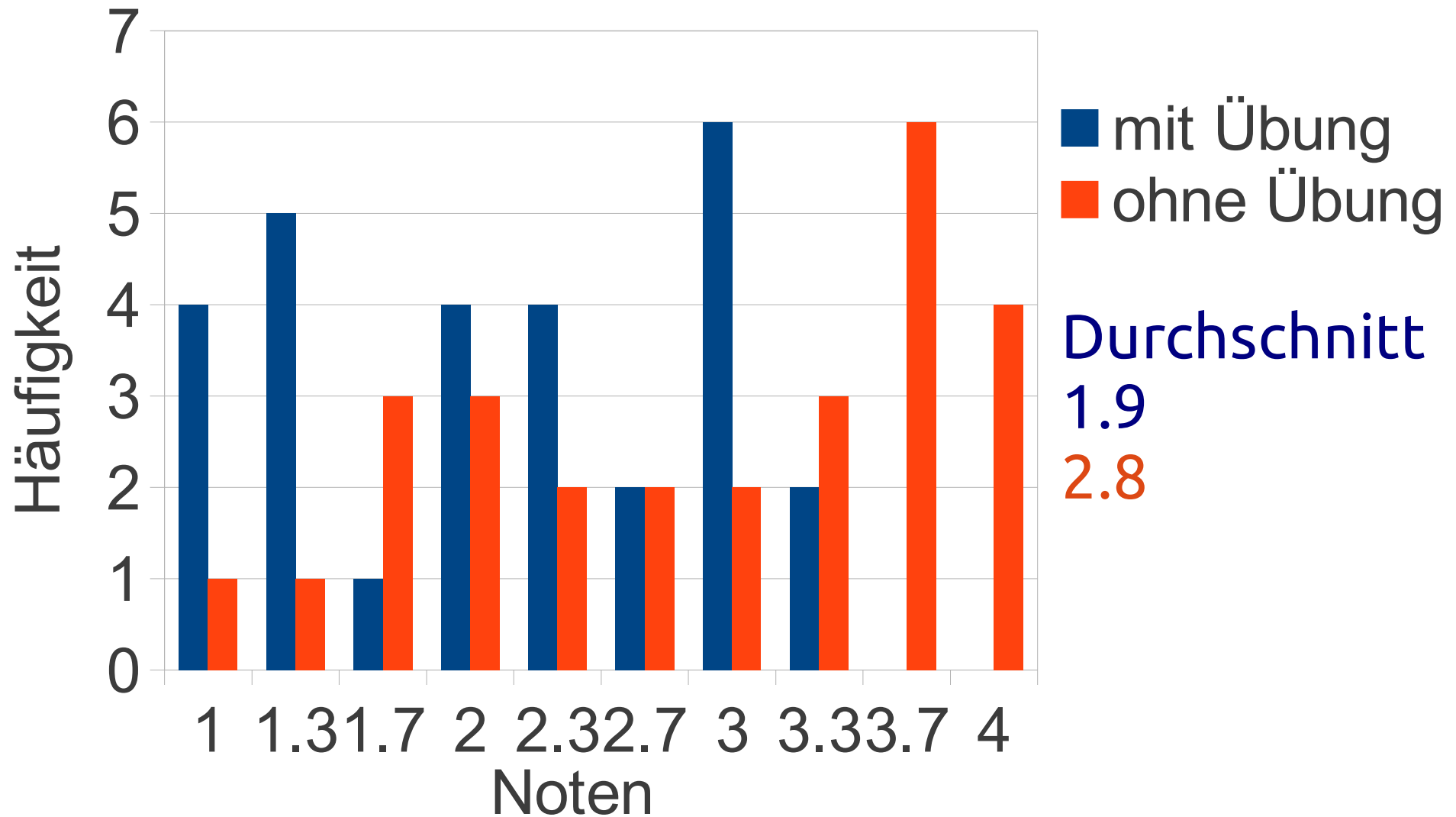
ToDo

- TUMOnline: Anmelden für Veranstaltung
- TUMOnline: Anmelden für Tutorübung
- TUMOnline: Anmelden für Klausur

Bonus

- Möglichkeit 4 Punkte in der Klausur vorab zu erarbeiten
- Voraussetzung: aktive Anwesenheit in 9 unterschiedlichen Übungen
- Im Fall von Krankheit
 - Falls Anwesenheit in 9 Übungen nicht möglich, dann ärztliches Attest nötig

Organisatorisches IV



Typen von Programmiersprachen

Deklarative Programmierung

- Funktional (Lisp, Scheme, make, ...)
- Logisch (Prolog)
- Sonstige (Sql, ...)

Imperative Programmierung

- Prozedural (C, Fortran, (Python), Pascal, ...)
- Modular (Bash, ...)
- Objektorientiert (C++, Java, Python, ...)

Skripting vs. Compilieren

Skriptsprachen

- Bash, Python, Perl, postscript, matlab/octave, ...
- Interaktiver Modus
- Teilweise wenig Optimierungen
- Einfach zu verwenden
- Keine Binärdateien (daher für kommerzielle Software meist ungeeignet)

Kompilierte Sprachen

- C, C++, Fortran, ...
- Effizient für sehr rechenaufwändige Aufgaben
- Sourcecode muss in Binärcode übersetzt werden

Sonstige

- Java

Warum Python? Warum Python in der Physik?

- Viele Physiker verwenden es (MPI etc.).
- Pre- und Postprocessing sehr elegant!
- Scripting!

Worum geht es in Python?

Python ist ...

- High-level (Wirklich high-level!)
- Komplette objektorientiert
- Interpretiert
- Skalierend
- Erweiterbar
- Portierbar (Windows, Linux, Mac OS X, Amiga, HPC, Cluster, Web Server, Palm/Handhelds, Mobiltelefone, ...)
- Vielseitig
- Einfach zu lernen, zu verstehen, zu verwenden und zu warten
- Kostenlos, open-source

Wenn ich nicht weiter weiß?

Literatur

- David M. Beasley: Python - Essential Reference
- Hans Petter Langtangen: A Primer on Scientific Programming with Python
- Vorlesungsfolien im Web (Druckversion 4 × 2)
- <https://docs.python.org/2.7/>
- www, google

Wenn ich nicht weiter weiß?

Literatur

- David M. Beasley: Python - Essential Reference
- Hans Petter Langtangen: A Primer on Scientific Programming with Python
- Vorlesungsfolien im Web (Druckversion 4 × 2)
- <https://docs.python.org/2.7/>
- www, google

Interaktive Hilfe

- `help()` für interaktive Hilfe
- `help('modules')` Liste der verfügbaren Module
- `help(object | 'command')` Hilfe zu Objekt oder Befehl

Worum geht es in Python?

Kann verwendet werden für:

- OS Skripting (In vielen Fällen besser als Bash)
- Internet Programmierung
- Wissenschaftliches Rechnen (selbst komplexe Simulationen)
- Parallelisierung
- GUI (TKinter)
- Visualisierung
- Rapid Prototyping
- ... und vieles mehr

Worum geht es in Python?

Ursprünge

aus wikipedia:

*About the origin of Python, Van Rossum wrote in 1996:[26]
Over six years ago, in December 1989, I was looking for a “hobby” programming project that would keep me occupied during the week around Christmas. My office ... would be closed, but I had a home computer, and not much else on my hands. I decided to write an interpreter for the new scripting language I had been thinking about lately: a descendant of ABC that would appeal to Unix/C hackers. I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of Monty Python’s Flying Circus).*

Worum geht es in Python?

Weitere Features (nicht in C++!)

- Dynamische Typisierung
- Automatische garbage collection
- Verschiedene Programmierparadigmen (Prozedural, OO, funktional, Aspekt-orientiert, ...)

Worum geht es in Python?

Weitere Features (nicht in C++!)

- Dynamische Typisierung
- Automatische garbage collection
- Verschiedene Programmierparadigmen (Prozedural, OO, funktional, Aspekt-orientiert, ...)

Generelle Programmierregeln

- Kommentare !
- Problem \Rightarrow Algorithmus \Rightarrow Programm
- Modulare Programmierung
- Generisch wo möglich, spezifisch wo nötig

Überblick

- Teil I: Erste Schritte
- Teil II: Datentypen
- Teil III: Kontrollstrukturen
- Teil IV: Funktionen und Module
- Teil V: IO und Datentypen
- Teil VI: Objektorientierte Programmierung
- Teil VII: Reguläre Ausdrücke
- Teil VIII: Exceptions
- Teil IX: Grafik
- Teil X: Partikelsysteme
- Teil XI: Datenstrukturen
- Teil XII: Wissenschaftliches Rechnen
- Teil XIII: Wärmeleitung
- Teil XIV: Lösung von Linearen Gleichungssystemen
- Teil XV: Mehrgitterverfahren

Python ausführen: python...

- Python interpreter starten (python oder ipython)
- Python-prompt: >>> Befehl eintippen und mit <Enter> bestätigen

```
$ python
Python 2.6.5 (r265:79063, Oct 1 2012, 22:04:36)
[GCC 4.4.3] on linux2
Type "help", "copyright", "credits" or "license" for
more information.
>>> help
Type help() for interactive help, or help(object) for
help about object.
>>> help()

Welcome to Python 2.7! This is the online help
utility. [...]
```

“Hello World!”

Python – lesbarer Code

- In Python

```
>>> print "Hello World!"  
Hello World!
```

“Hello World!”

Python – lesbarer Code

- In Python

```
>>> print "Hello World!"  
Hello World!
```

- Andere Beispiele: Java

```
public class Hello{  
    public static void main(String argv[]){  
        System.out.println("Hello World!");  
    }  
}  
  
$ javac Hello.java  
$ java Hello  
Hello World!
```

Variablen, Zuweisungen und Ausdrücke

```
>>> 3 + 4**2/8 + 1
>>> (6*3.5)*2 - 5
>>> 10e-4
>>> a = 17
>>> a
>>> a - 1.5
>>> a = a - 1
>>> b = "Hello_ World!"
>>> print b
>>> print a, b, (4 + 5**0.5)
```

- Zuweisung: Variablenname = Ausdruck
- Nicht zu verwechseln mit mathematischem =
- Ausgabe des Ergebnisses nur interaktiv
- `print` gibt eine String-Repräsentation von Objekten aus
- Leerzeichen im Ausdruck egal (Aber nicht VOR dem Ausdruck)

Variablennamen

- a-z, A-Z, 0-9, _
- Erstes Zeichen nicht 0-9
- _ ist erlaubt, hat aber spezielle Bedeutung
- Variablennamen sollten aussagekräftig sein
- Bei mathematischen Ausdrücken entsprechend den mathematischen Variablen
- Ansonsten deskriptiv
- Einige Worte sind reserviert:
`and, as, assert, break, class, continue, def, del, elif, else, except, False, finally, for, from, global, if, import, in, is, lambda, None, nonlocal, not, or, pass, raise, return, True, try, with, while, yield`
- Variablen in Python sind dynamisch typisiert

3.5 Wege zu Python

1. Python interpreter starten mittels `python`
2. IPython nutzen: `ipython`
3. Python auf einer Datei ausführen
- 3.5.** Python über ein ausführbares Skript nutzen

2. Python ausführen: ipython

Vorteile

- Erweiterte interaktive Shell
- Zusätzliche Shell Syntax
 - “magische” Funktionen, beginnend mit “%”, z.B. `%psearch` um Funktionen im Namensraum zu suchen
 - Zugriff auf Bash-Befehle: `%cd`, `%%bash`
 - Python-Skripte ausführen mit `run filename.py`
 - Logging, Makros, ...
 - Pretty printing (nested structures), umschalten `%pprint`
- Tab-Vervollständigung
- History
 - Vorherige Befehlsblöcke wiederholen
- Automatische Einrückung
- Mitgeliefert mit SciPy, default Konsole in `spyder`

3. Python ausführen: Datei übergeben

- Datei `square_me.py`:

```
print 6**2
```

- Ausführen mit

```
$ python square_me.py  
36
```

oder in `ipython`

```
[ ] run square_me.py  
36
```

3.5 Python ausführen: ausführbares Skript

- Gleicher Programmcode wie zuvor
- Dem Betriebssystem mitteilen, dass python verwendet werden soll:

```
#!/usr/bin/python  
print 6**2
```

- Datei ausführbar machen und ausführen

```
$ chmod u+x square_me.py  
$ ./square_me.py  
36
```