

Programmierung in Python

Einheit 0: Grundlagen der Programmierung

Prof. Dr. Martin Hepp

<https://mfhepp.github.io/pip/>

Lernziel dieser Einheit

- Was ist eine Programmiersprache und wie kann sie dazu verwendet werden, ein betriebliches Problem an einen Computer zu delegieren?
- Welche Ansätze gibt es, um ein Programm aus einer Programmiersprache in eine für Computer verständliche Form zu bringen?
- Welche Arten von Programmiersprachen gibt es und wodurch unterscheiden sie sich?
- Was versteht man unter Objektorientierung?
- Wie werden einfache Programme in der Sprache Python entwickelt?

Gliederung dieser Einheit

0.1 Grundlagen

0.2 Python

0.3 Installation der Programmierumgebung

0.1 Grundlagen

Definition Programmiersprache

„eine formale Sprache, mit der eine auf einer Hardware ablauffähige Software entwickelt werden kann.“

[MBKP2005, S. 25]

Beispiel BASIC:

```
10 PRINT "Hello World!"  
20 FOR I=1 TO 10  
30 PRINT I  
40 NEXT
```

Siehe auch: <https://de.wikipedia.org/wiki/Programmierparadigma>

Imperative Programmierung

Programm beschreibt, **wie ein Problem zu lösen ist**

1. Tue dies
2. Tue jenes
3. Wiederhole die folgende Anweisung fünf Mal:
 - Tue das
4. Wenn Bedingung erfüllt,
 - Tue dies
5. Falls nicht,
 - Tue jenes

Vgl. [[MBKP2012](#), S. 20]

Deklarative Programmierung

Programm beschreibt, WAS zu tun ist

Beispiele

Sortiere die Liste aller Studenten.

Finde alle Kunden, die mehr als 10,000 Euro Umsatz getätigt haben.

Vgl [[MBKP2012](#), S. 20f.]

Quelltext

Text eines Programms in Programmiersprache

1. Für Menschen verständlich
2. Für Computer nicht direkt ausführbar

```
import os

text = "Programming in Python is a mighty skill."
os.system('say %s' %text)
```

Maschinencode

Repräsentation von Programmen als Zahlenfolge, die ein Computer ausführen kann

```
$ hexdump -C -n128 firefox
00000000  ca fe ba be 00 00 00 02  01 00 00 07 80 00 00 03  |.....
00000010  00 00 10 00 00 00 62 90  00 00 00 0c 00 00 00 07  |.....b.....
00000020  00 00 00 03 00 00 80 00  00 00 5d 80 00 00 00 0c  |.....].....
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 00  |.....
.
.
```

Siehe auch: <https://de.wikipedia.org/wiki/Maschinensprache>

Techniken, um Quelltext in Maschinencode zu übersetzen

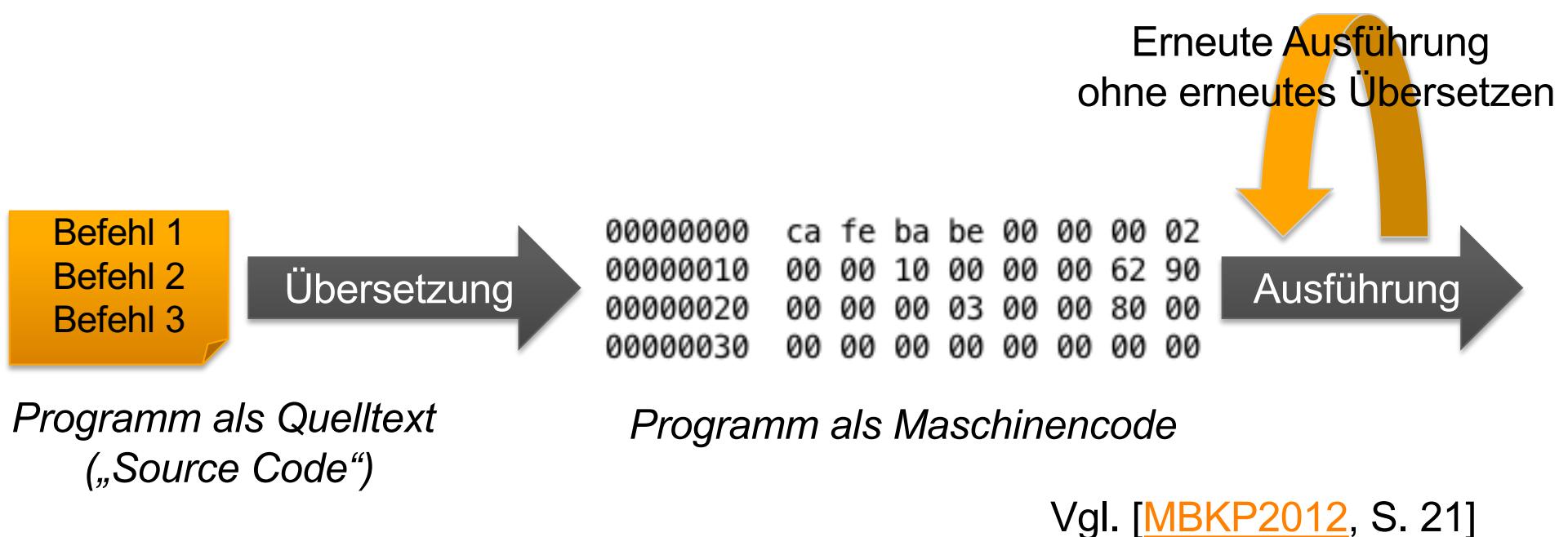
1. Compiler
2. Interpreter
3. Kombinierter Ansatz: Bytecode und Virtual Machines

Vgl. [[MBKP2012](#), S. 21]

Compiler

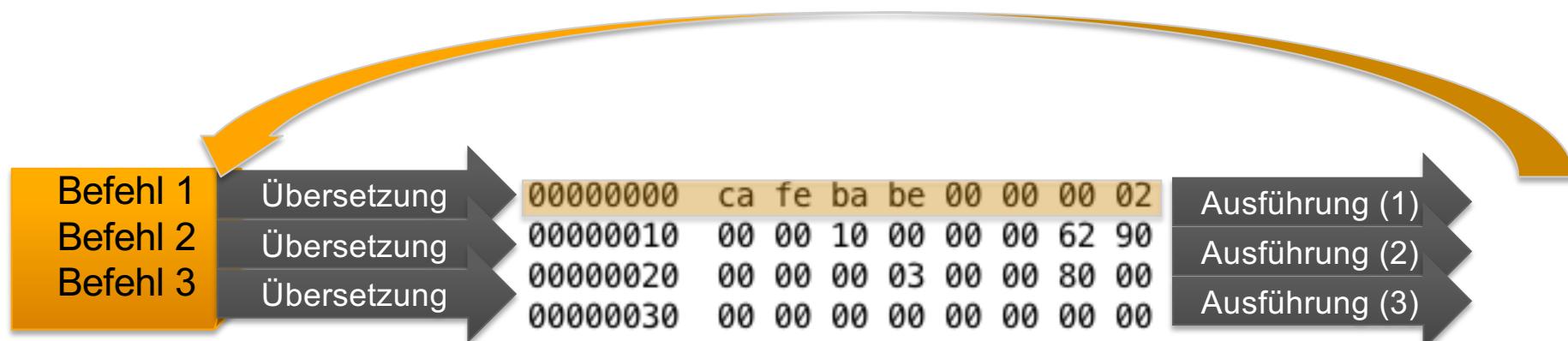
Übersetzung des **gesamten** Quelltextes zur Entwurfszeit

- Testen langsamer
 - Gesamtes Programm muss übersetzt werden, bis es gestartet werden kann
- Ablauf schneller
 - Ablauf wird nicht durch Übersetzung gebremst



Übersetzen des Quelltextes **Schritt-für-Schritt** zur Laufzeit

- Testen schneller
 - Programmablauf kann starten, sobald der erste Programmteil übersetzt wurde
- Ablauf langsamer
 - Programmablauf wird immer wieder durch Übersetzen unterbrochen
- Portabilität



Programm als Quelltext
„Source Code“

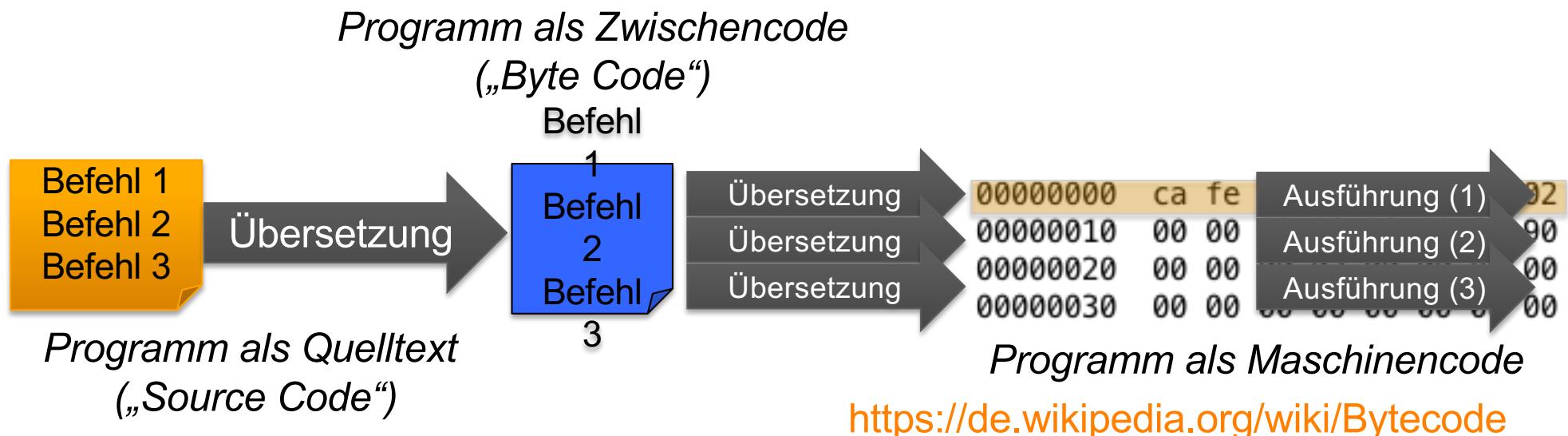
Programm als Maschinencode

Vgl. [[MBKP2012](#), S. 21]

Kombinierter Ansatz: Bytecode und Virtual Machines

Compiler erzeugt Zwischencode, Interpreter führt aus

- Beispiel: Java
 - Vorteile
 - „Write once, run anywhere“
 - Guter Kompromiss aus Geschwindigkeit und Portabilität
 - Ökonomie
 - Quelltext (bedingt) schützbar



Syntax von Programmiersprachen

Konventionen für Programmtext

1. Regeln, wie Zeichen und Wörter in einem Programm verwendet und kombiniert werden

- Beispiele:
 - Zeilenummern oder nicht
 - = oder := für Zuweisungsoperator

2. Schlüsselwörter / Reservierte Wörter

3. Kommentare

Programmcode muss die Syntax immer präzise einhalten!

Schlüsselwörter (Keyword)

Reservierte Wörter in einer Programmiersprache

Befehle

print

if ... else

Datentypen

char

string

int

float

Merke: Reservierte Wörter dürfen nicht als Namen für eigene Elemente wie Variablen und Funktionen verwendet werden.

Variablen

Platzhalter für Werte

Variablen sind Platzhalter für Werte, auf die im Programmablauf über einen **Namen** zugegriffen werden kann.

`Benzinpreis = 1.37`

`Tankvolumen = 42`

`Kosten_pro_Tankfuellung = Benzinpreis * Tankvolumen`

Variablen können im Programmablauf mit neuen Werten gefüllt werden:

`Benzinpreis = Benzinpreis + 0.10`

Konstanten

Namen für Werte, die im Programmablauf nicht verändert werden

`AUTHOR_NAME = "Martin Hepp"`

`CRAWLING_SPEED = 8`

In Python gibt es keine Konstanten im eigentlichen Sinne, sondern nur die Konvention, dass Variablennamen in Großbuchstaben als Konstanten zu verwenden sind.

Datentypen

Speicherungsformat und Verhalten von Werten

Der Datentyp eines Wertes legt fest:

1. In welchem Format der Wert im Speicher des Computers abgelegt wird.

Beispiel: „12“ kann als Zahl „zwölf“ (Binär: 00001100)

oder als Folge der Zeichen „1“ und „2“ gespeichert werden.

2. Welche Operationen zulässig sind und wie diese genau wirken.

Beispiel: Wenn man die Zahl 12 mit der Zahl 2 multipliziert, erhält man den Wert 24. Wenn man die Zeichenfolge „12“ verdoppelt, erhält man „1212“.

Hinweis: Auch Ausdrücke, Funktionen und Objekte haben einen Typ; das kann hier jedoch zunächst ignoriert werden.

Vgl. [https://de.wikipedia.org/wiki/Typisierung_\(Informatik\)](https://de.wikipedia.org/wiki/Typisierung_(Informatik))

Typisierung von Variablen

Zwei Ansätze

1. Static Typing

- Festlegung und Prüfung des Datentyps beim Entwurf
- **Vorteil:** Der Compiler oder Interpreter kann Fehler erkennen, wenn ein unpassender Wert zugewiesen wird.

2. Dynamic Typing

- Festlegung und Prüfung des Datentyps bei der Ausführung
- **Vorteil:** Derselbe Programmteil kann für unterschiedliche Arten von Werten verwendet werden.

Vgl. https://de.wikipedia.org/wiki/Dynamische_Typisierung

- Python nutzt Dynamic Typing.
- Seit Version 3.5 gibt es einen Mechanismus, der sich „Type Hints“ nennt und die Vorteile beider Ansätze kombiniert. Für weitere Informationen, siehe hier: <https://docs.python.org/3/library/typing.html>

Kommentare

Erläuterungen für Menschen, die der Computer ignorieren soll

- Manchmal ist es für das Verständnis eines Programms hilfreich, wenn kurze Erklärungen für Menschen in den Quelltext eingefügt werden.
- Diese müssen markiert werden, damit der Interpreter oder Compiler erkennt, dass er sie ignorieren soll.
- Besser ist es, verständlich zu programmieren!

Beispiel: Sprechende Variablennamen

zinssatz = 0.05 vs. **z** = 0.05

Prozeduren und Unterprogramme

Prozedurale Programmierung

Aufspalten eines Programms in Unterprogramme
("Subroutines")

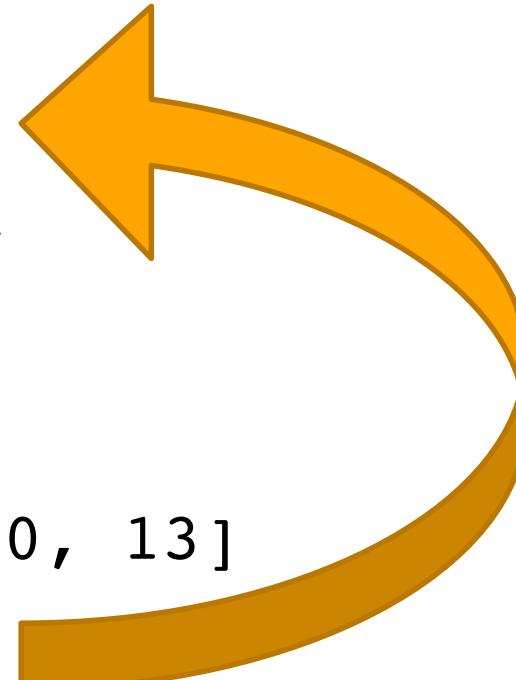
Funktion sortiere (liste):

Befehle dieser Funktion

Hauptprogramm:

a = [1, 2, 6, 3, 10, 100, 13]

ergebnis = sortiere (a)



0.2 Python

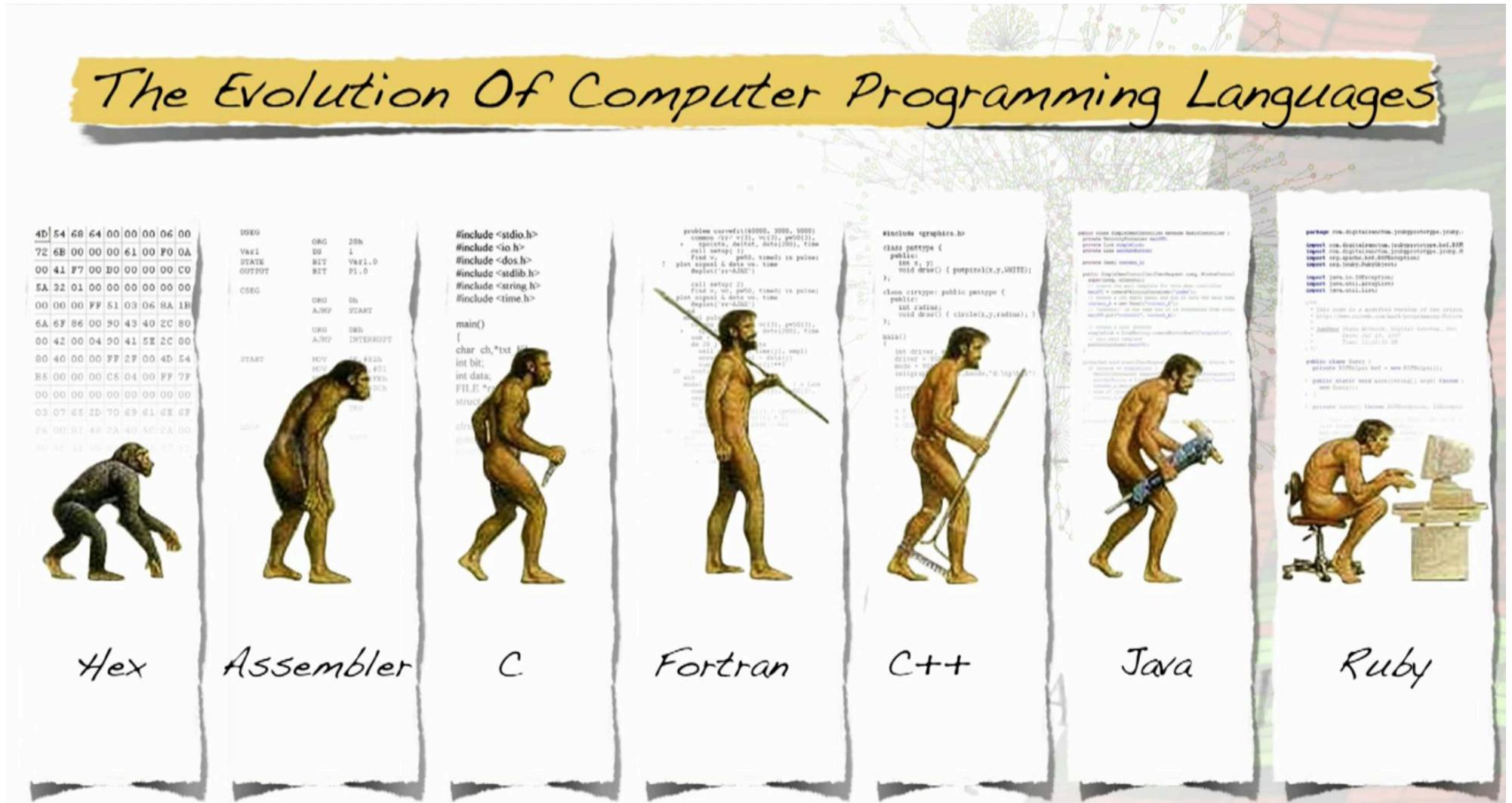
<https://www.python.org/>

The screenshot shows the Python.org homepage with a dark blue header. The Python logo is on the left, followed by the word "python" in white. To the right are buttons for "Donate", a search bar with a magnifying glass icon, and "GO". Below the header is a navigation menu with tabs: "About", "Downloads", "Documentation", "Community", "Success Stories", "News", and "Events". A large central area features a code snippet in a terminal window:

```
# Python 3: Fibonacci series up to n
>>> def fib(n):
>>>     a, b = 0, 1
>>>     while a < n:
>>>         print(a, end=' ')
>>>         a, b = b, a+b
>>>     print()
>>> fib(1000)
0 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610
987
```

To the right of the code, a yellow button with a right-pointing arrow is positioned above a section titled "Functions Defined". This section contains text about defining functions in Python 3, mentioning mandatory and optional arguments, keyword arguments, and arbitrary argument lists, with a link to "More about defining functions in Python 3". At the bottom right of this section are five small numbered buttons: 1, 2, 3, 4, and 5. Below this section is a quote: "Python is a programming language that lets you work quickly and integrate systems more effectively. [» Learn More](#)".

The Evolution of Programming Languages



Bildnachweis: dullhunk/Flickr, <http://www.flickr.com/photos/dullhunk/4833512699/>, CC BY

Assemblersprache

Programmiersprache für Maschinencode

1. Mnemonics für Befehle

- MOV – Move
- ADD – Add
- SUB – Subtract
- INC – Increment

2. Symbolische Namen für Adressen

3. Motivation

Siehe auch: <https://de.wikipedia.org/wiki/Assemblersprache>

Wichtige Programmiersprachen

1. FORTRAN – FORmula TRANslator (1954)
2. LISP- List Processor (1959)
3. COBOL – Common Business Oriented Language (1959)
4. ALGOL – ALGorithmic Language (1958/1960)
5. BASIC – Beginner's All-Purpose Symbolic Instruction Code (1964)
6. C (1972)
7. Pascal (1971)
8. C++ (1983)
9. Python (1980s; 1989)
10. Java (1995)
11. Javascript (ca. 1995)
12. PHP (1995)

Programming in Python

- Leicht zu lernen
- Gut lesbar
- Mächtig
- Schnell
- Für viele Plattformen verfügbar



NASA uses Python...



... so does Rackspace, Industrial Light and Magic, AstraZeneca, Honeywell, and many others.

<https://www.python.org/>

YouTube wurde fast vollständig in Python entwickelt

The screenshot shows a YouTube search results page for "programming python". The results are as follows:

- Python 1: Installation and beginner's tutorial.** 8:25, 2 years ago, 46,182 views, murfville
- Python Programming Tutorial - 1 - Installing Python** 3:19, 6 months ago, 21,713 views, thenewboston
- Python Programming Tutorial - 2 - Numbers and Math** 5:40, 6 months ago, 15,960 views, thenewboston
- First 5 Minutes Programming with Python** 2:44, 2 years ago, 27,281 views, ianATshowmedo

[Python-Dev] [Python-checkins] MSI being downloaded 10x morethan all other files??!

Guido van Rossum guido@python.org
Tue Dec 12 17:37:50 CET 2006

- Previous message: [Python-Dev] [Python-checkins] MSI being downloaded 10x morethan all other files??!
- Next message: [Python-Dev] [Python-checkins] MSI being downloaded 10x morethan all other files??!
- Messages sorted by: [date] [thread] [subject] [author]

And I just found out (after everyone else probably i-) that YouTube is almost entirely written in Python. (And now I can rub shoulders with the developers since they're all Googlers now... i-)
 0 12/12/06, Kurt B. Kaiser <kbb@shore.net> wrote:
 > "Fredrik Lundh" <fredrik@pythonware.com> writes:
 >>> The Rails buzz seems to be jumping to Python lately.
 >>
 > fwiw, the people I see pick up Python haven't even heard of Ruby or
 > Rails (not every one is doing web 2.0 stuff, after all).
 >
 > Yes, separate but related groups and issues.
 >
 > MIT's adopting Python in their introductory programs will drive
 > other schools in our direction, I think.
 >
 > --
 > KBB
 >
 > Python-Dev mailing list
 > Python-Dev@python.org
 > <http://mail.python.org/mailman/listinfo/python-dev>
 > Unsubscribe: <http://mail.python.org/mailman/options/python-dev/guido@python.org>

--Guido van Rossum (home page: <http://www.python.org/~guido/>)

<http://mail.python.org/pipermail/python-dev/2006-December/070323.html>

Python has been an important part of Google since the beginning, and remains so as the system grows and evolved. Today dozens of Google engineers use Python, and we're looking for more people with skills in this language.

-- Peter Norvig, Director of Research at Google

Beispiel: Steuerung des Web Browsers

```
# open five random Wikipedia pages
import webbrowser

URI =
'http://en.wikipedia.org/wiki/Special:Rando
m'

for i in range(5):
    webbrowser.open_new_tab(URI)
```

Ergebnis

Zusatzmaterial

Linköping – Wikipedia, the free encyclopedia

http://en.wikipedia.org/wiki/Linköping

Linköping – Wikipedia, the free encyclopedia Battle of Immae – Wikipedia, ... Torleiv Maseng – Wikipedia, ... Glen Echo – Wikipedia, the free encyclopedia Howard B. Lee – Wikipedia, the free encyclopedia

Make a donation to Wikipedia and give the gift of knowledge!

article discussion edit this page history Try Beta Log in / create account

Linköping

From Wikipedia, the free encyclopedia

Coordinates: 58°24'N 15°37'E

Linköping

Linköping [lɪnø:pɪŋ] is a city in southern Sweden, with 97,428 inhabitants in 2005.^[1] It is the seat of Linköping Municipality with 140,367 inhabitants (2007) and the capital of Östergötland County. Linköping is also the episcopal see of the Diocese of Linköping (Church of Sweden) and is well known for its cathedral.

Linköping is the center of an old cultural region and celebrated its 700th anniversary in 1987. Nowadays Linköping is known for its university and its high-technology industry. Dominating the city's skyline from afar is the steeple of the cathedral.

The city is situated south of lake Roxen (which is part of the historically important water paths Motala ström and the Göta Canal) where the main road from Stockholm to Helsingborg crosses the river Stångån (and Kinda kanal).

This road was part of the Eriksgata route that the newly elected king had to travel according to medieval Swedish Law. In the 20th century road system, it was first called Riksettan (national highway no 1). It is currently called E4 and has been redirected to pass outside the city on the north side. Further contributing to Linköping's excellent communications is its situation on the main southern railway line connecting Stockholm with Malmö and Danish capital Copenhagen. There is also a minor airport, Linköping SAAB Airport.

Contents [hide]

- 1 History
- 2 Culture
- 3 Sport
- 4 Industry
- 5 Sister cities
- 6 See also
- 7 References

Central square in Linköping

Motto: Where Ideas Become Reality (Där idéer blir verklighet)



Done

Python: Wichtige syntaktische Konventionen

- Keine Zeilenummern
- Schlüsselwörter
 - `print, if, else, for, in,`
- Zuweisungsoperator =
- Groß-/Kleinschreibung muss beachtet werden:

```
a = 10
```

```
print A
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
NameError: name 'A' is not defined
```

Wichtige Änderung seit HT 2017

Wechsel auf Python 3.x

Achtung: Ab dem HT 2017 verwenden wir die Sprachversion **3.x** von Python. In den früheren Trimestern wurde **2.x** verwendet!

Es gibt einige wesentliche Änderungen zwischen Python 2.x und 3.x. Ein Python 2.x-Programm läuft nicht ohne Weiteres in Python 3.x und umgekehrt.

Die wesentlichen Unterschiede werden auf den folgenden Folien jeweils hervorgehoben und erläutert.

Für einen Überblick, siehe,

- <https://docs.python.org/3.0/whatsnew/3.0.html>
- https://sebastianraschka.com/Articles/2014_python_2_3_key_diff.html
- http://python-future.org/compatible_idioms.html#essential-syntax-differences

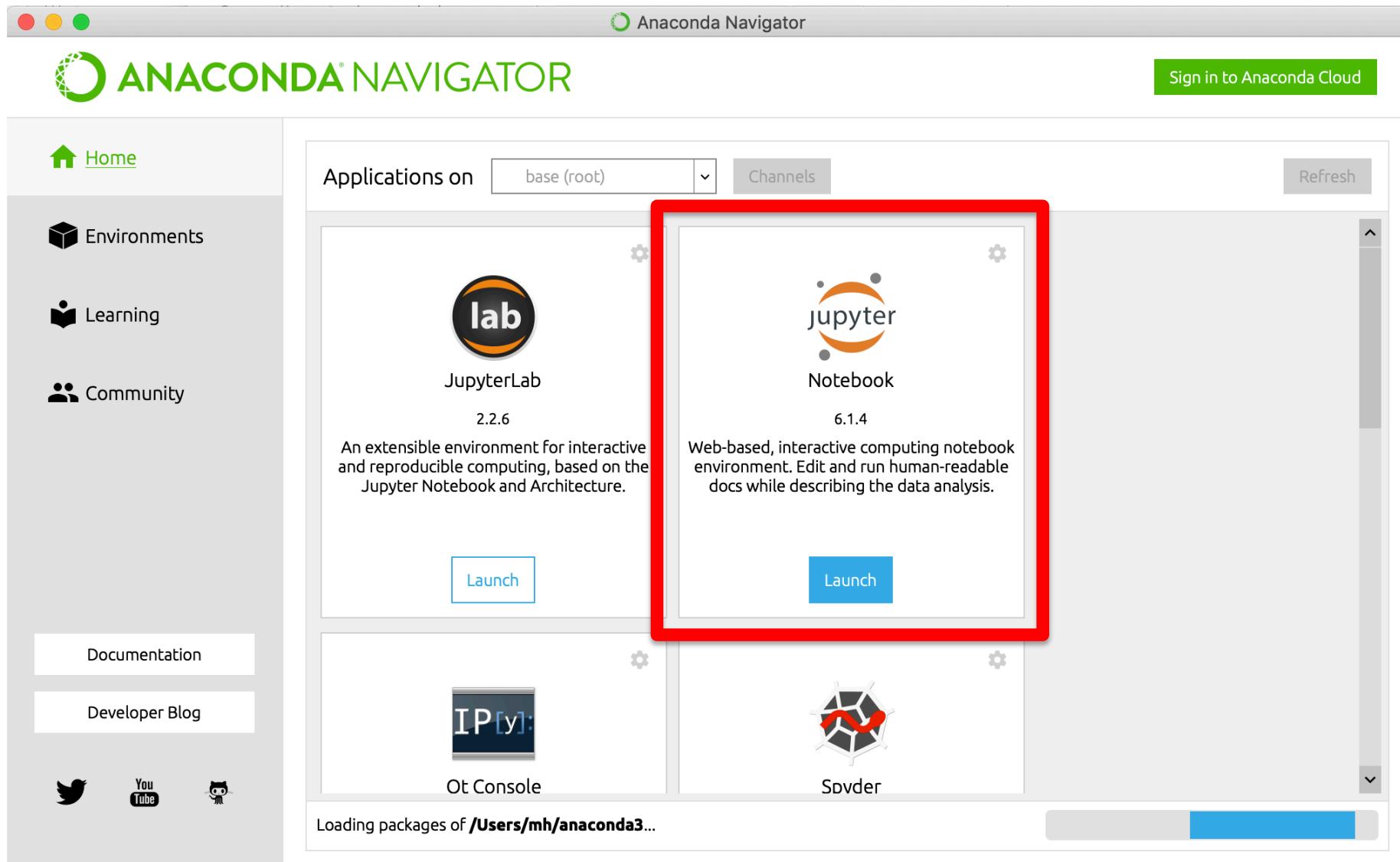
0.3 Installation von Anaconda / Jupyter Notebooks

<https://www.anaconda.com/download/>

A screenshot of the Anaconda download page. At the top, there's a navigation bar with the Anaconda logo, a search bar, and links for Products, Pricing, Solutions, Resources, Blog, and Company. A 'Get Started' button is also present. Below the navigation, the text 'Individual Edition' and 'Your data science toolkit' is displayed. A paragraph describes the Individual Edition as the easiest way to perform Python/R data science and machine learning on a single machine, developed for solo practitioners. It mentions over 20 million users worldwide and thousands of open-source packages and libraries. A 'Download' button is located below this text. A large red arrow points downwards from the top of the slide towards the 'Anaconda Installers' section. This section contains links for Windows, MacOS, and Linux, each with their respective operating system icons. Under Windows, there are links for Python 3.8: 64-Bit Graphical Installer (466 MB) and 32-Bit Graphical Installer (397 MB). Under MacOS, there are links for Python 3.8: 64-Bit Graphical Installer (462 MB) and 64-Bit Command Line Installer (454 MB). Under Linux, there are links for Python 3.8: 64-Bit (x86) Installer (550 MB) and 64-Bit (Power8 and Power9) Installer (290 MB).

Platform	Python Version	Installer Type	File Size
Windows	Python 3.8	64-Bit Graphical Installer	466 MB
		32-Bit Graphical Installer	397 MB
MacOS	Python 3.8	64-Bit Graphical Installer	462 MB
		64-Bit Command Line Installer	454 MB
Linux	Python 3.8	64-Bit (x86) Installer	550 MB
		64-Bit (Power8 and Power9) Installer	290 MB

Anaconda Navigator



Jupyter Notebooks

The screenshot shows a web-based Jupyter Notebook interface. At the top, there is a browser-like header with tabs for "Seminar Room 11", "Teaching/PIP - W", "Digistore24", "Thank you for yo", "Downloads - An", and "pip-lecture link/". The URL in the address bar is "localhost:8888/tree/pip-lecture%20link". Below the header, the Jupyter logo is visible, along with a "Logout" button. A navigation bar includes "Files", "Running", "Clusters", and "Conda" buttons. A message "Select items to perform actions on them." is displayed above a file browser. The browser lists files and folders under the path "/ pip-lecture link". The files listed are:

File/Folder	Last Modified
..	seconds ago
exercises	5 months ago
notebooks	4 months ago
slides	5 months ago
Geo Experiments.ipynb	5 months ago
Linear Algebra.ipynb	5 months ago
Untitled.ipynb	21 days ago
LICENSE	5 months ago
README.md	5 months ago

Jupyter Notebooks

The screenshot shows a Jupyter Notebook interface on a web browser. The URL in the address bar is `localhost:8888/tree/pip-lecture%20link`. The interface includes a sidebar with tabs for `Files`, `Running`, `Clusters`, and `Code`. The `Files` tab is active, displaying a list of files and folders under the path `/ pip-lecture link`. A red box highlights a modal dialog titled "Notebook:" containing a list of available environments: Python [conda env:py27], Python [conda env:py36], Python [conda root], and Python [default]. Below this, another red box highlights a list of other options: Text File, Folder, and Terminal. To the right of the modal, a list of notebooks is shown, also with a red box highlighting the top section. The list includes:

Name	Last Modified
	seconds ago
	5 months ago
	4 months ago
	5 months ago
	5 months ago
	21 days ago
	5 months ago
	5 months ago

Unser erstes Notebook

The screenshot shows a Jupyter Notebook interface. At the top, there is a browser-like header with navigation icons (back, forward, search, etc.), a URL bar showing 'localhost:8888/notebooks/pip-lecture%20link/Untitled...', and user profile information. Below the header, the title bar displays 'jupyter Untitled1' and a Python logo icon with a 'Logout' button. The main menu bar includes 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Help', 'Trusted' (which is highlighted), and a language selector set to 'Python [default]'. A toolbar below the menu contains various icons for file operations like saving, opening, and deleting. The central workspace shows a code cell starting with 'In []:' followed by the Python code: `print('Nerd is the new Cool')`. The code is written in green, and the string is in red.

Philosophie: Literate Programming

Idee: Programme so schreiben, als würde man sie einem Menschen erklären.

„Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on n explaining to human beings what we want a computer to do.“

```
1 for person in ['joe', 'judy']:  
2     print('Hello', person)
```

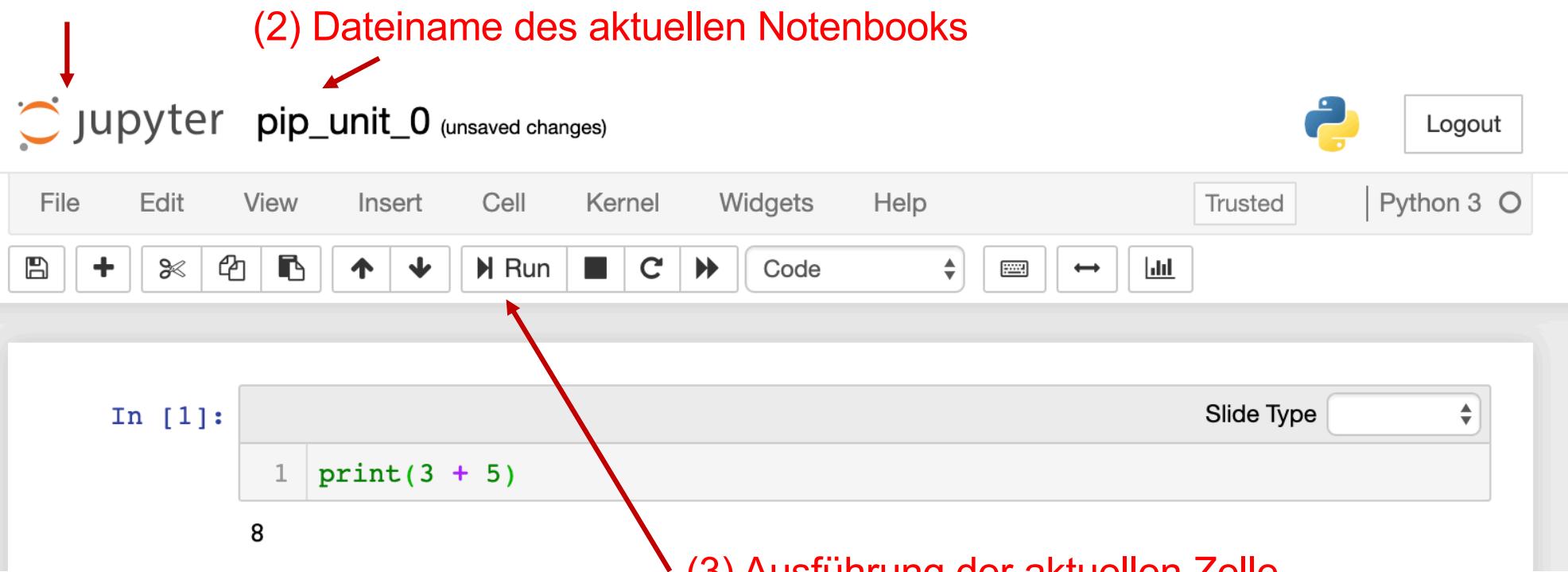
```
Hello joe  
Hello judy
```

Donald E. Knuth

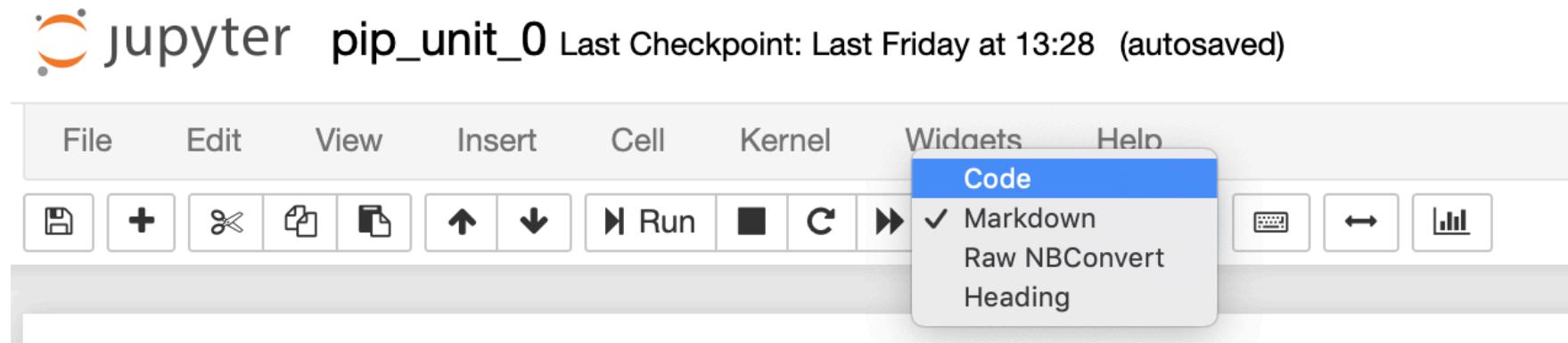
<http://www.literateprogramming.com/knuthweb.pdf>

Bedienung

(1) Link zu allen Notebooks



Zellen und Zelleninhalte



- Code:** Programmcode (hier: Python)
- Markdown:** Text zur Erläuterung
- Heading:** Überschriften (auch mit Markdown möglich)
- Raw NBConvert:** Sonderformat für spezielle Zwecke (hier nicht behandelt)

Nützliche Tastenkombinationen

1. Neue Zelle unterhalb
 - Esc + B („below“)
2. Neue Zelle oberhalb
 - Esc + A (“above“)
3. Zelle aufteilen an aktueller Position (Split)
 - Ctrl + Shift + -
4. Aktuelle Zelle ausführen
 - Shift + Enter
5. Zellentyp in „Code“ ändern
 - Esc + Y
 - change the cell type to *Code*
6. Zellentyp in „Markdown“ ändern
 - Esc + M

<https://towardsdatascience.com/jupyter-notebook-shortcuts-bf0101a98330>

Markdown

Einfache Auszeichnungssprache, um Text zu formatieren

```
1 # Überschrift 1
2 ## Überschrift 2
3 ### Überschrift 3
4
5 *kursiv*
6
7 **fett**
8
9 ***fett und kursiv***
10
11 Listen:
12 1. Auf die Nummern
13 1. kommt es
14 1. nicht an
15
16 Listen
17 - Bullet 1
18 - Bullet 2
```

Überschrift 1

Überschrift 2

Überschrift 3

kursiv

fett

fett und kursiv

Listen:

1. Auf die Nummern
2. kommt es
3. nicht an

Listen

- Bullet 1
- Bullet 2

<https://de.wikipedia.org/wiki/Markdown>

<https://help.github.com/en/github/writing-on-github/basic-writing-and-formatting-syntax>

<https://daringfireball.net/projects/markdown/>

Bearbeitung einer Zelle abbrechen

Z.B. bei Programmfehler / Endlosschleife

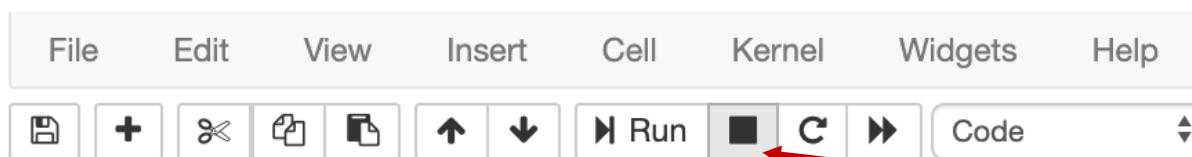
Sternchen zeigt an,
dass die Ausführung der
Zelle noch läuft.

In [*]:

```
1 # Diese Schleife läuft ewig
2 while True:
3     pass
4
```



Gefüllter Kreis zeigt an,
dass der Python-Interpreter
noch beschäftigt ist

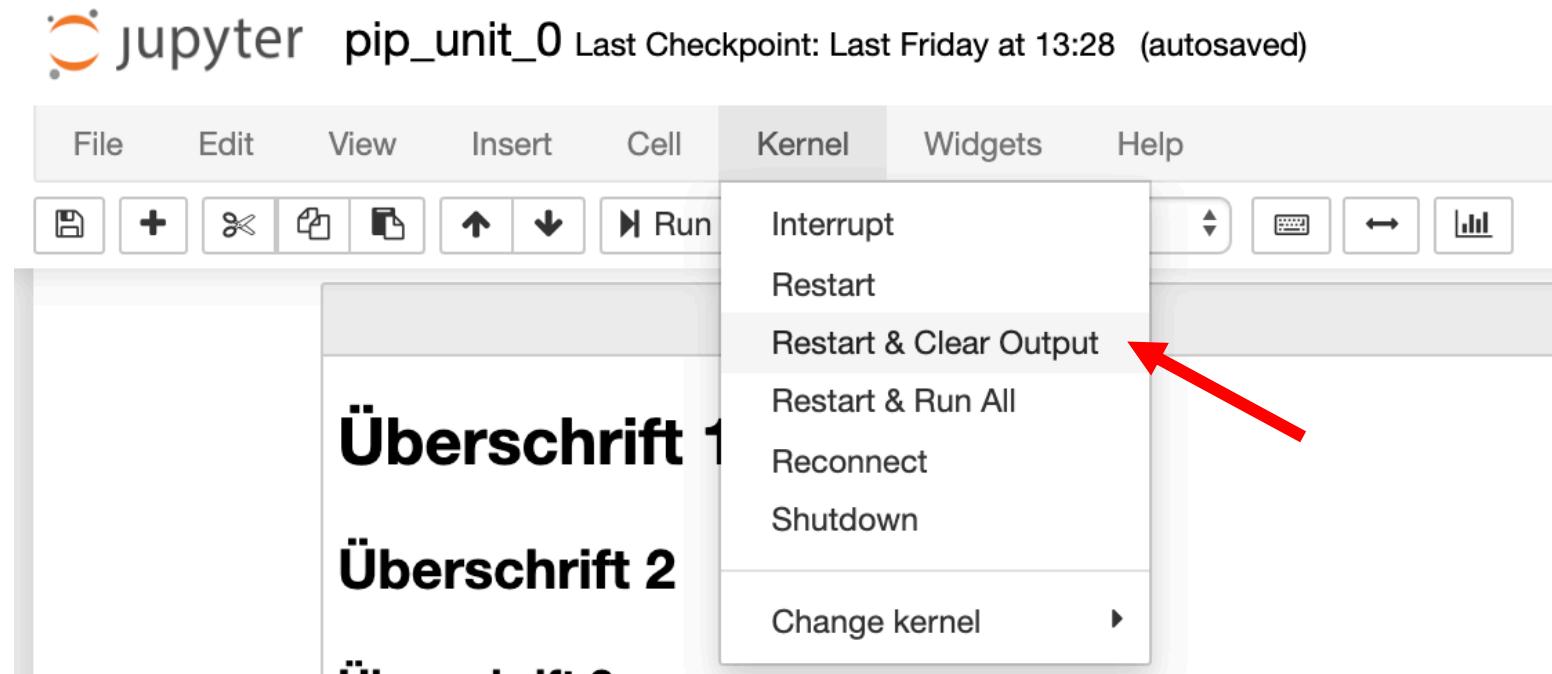


```
KeyboardInterrupt
<ipython-input-3-9f0840b03a7e> in <module>
    1 # Diese Schleife läuft ewig
    2 while True:
--> 3     pass

KeyboardInterrupt:
```

Hier klicken

Wenn nichts mehr geht: Kernel Restart

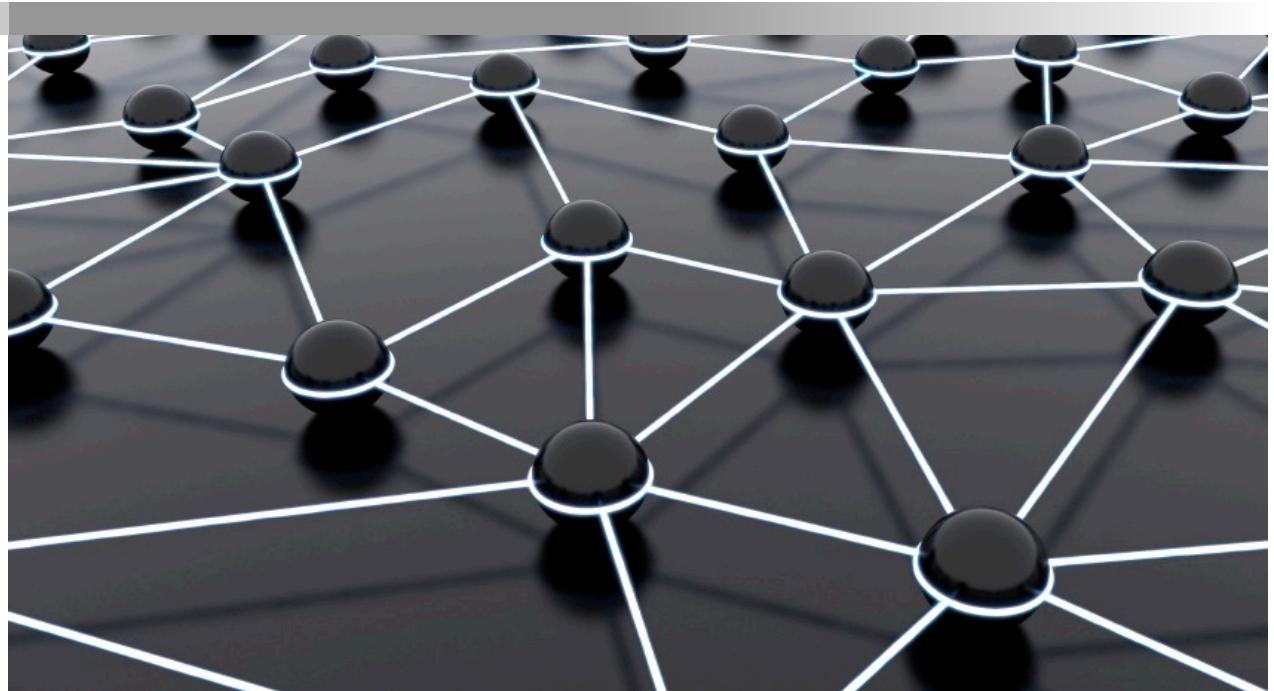
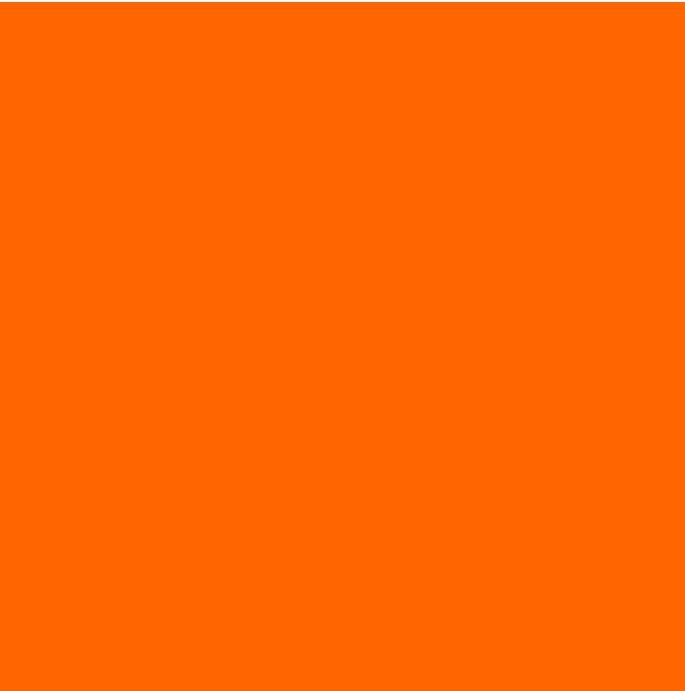


Aufgaben

1. Wiederholen Sie die Vorlesungsfolien.
2. Installieren Sie die Programmierumgebung Anaconda.
3. Erzeugen Sie ein leeres Jupyter-Notebook und speichern Sie es unter dem Namen „mein_notebook.ipynb“.
4. Fügen Sie eine Zeile Programmcode hinzu und führen Sie die Zelle aus.
5. Fügen Sie eine Zeile Markdown-Text hinzu und führen Sie die Zelle aus.
6. Speichern Sie den letzten Stand Ihres Notebooks und suchen Sie die Datei (Endung *.ipynb) in Ihrem Dateisystem.

Literatur

1. [Gru15]: Joel Grus: *Data Science from Scratch. First Principles with Python*, O'Reilly Media, 2015.
 - Programmierbeispiele zu diesem Buch: <https://github.com/joelgrus/data-science-from-scratch>
2. [Har18]: Scott Harvey: *Data Science from Scratch: Comprehensive guide with essential principles of Data Science*, CreateSpace Independent Publishing Platform, 2018.
3. [MBKP2012]: **Mertens/Bodendorf/König/Picot/Schumann/Hess:** *Grundzüge der Wirtschaftsinformatik*, 11. Auflage, Springer 2012. Dieses Buch ist aus dem Uninetz als PDF / eBook verfügbar:
<https://link.springer.com/book/10.1007/978-3-642-30515-3>



Vielen Dank.

Prof. Dr. Martin Hepp

<https://mfhepp.github.io/pip/>

Professur für Web Science und Digitalisierung