



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 |.....|
..
```

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

Techniken, um Quelltext in Maschinencode zu übersetzen

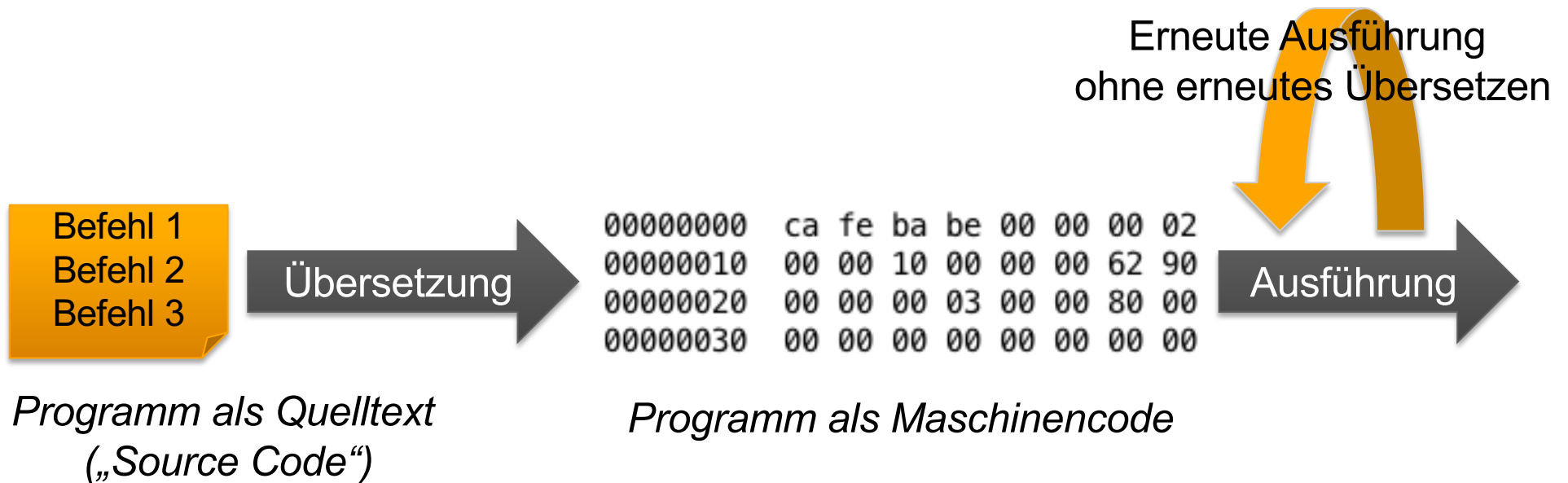
1. Compiler
2. Interpreter
3. Kombiniertes Ansatz: Bytecode und Virtual Machines

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

Compiler

Übersetzung des **gesamten** Quelltextes zur Entwurfszeit

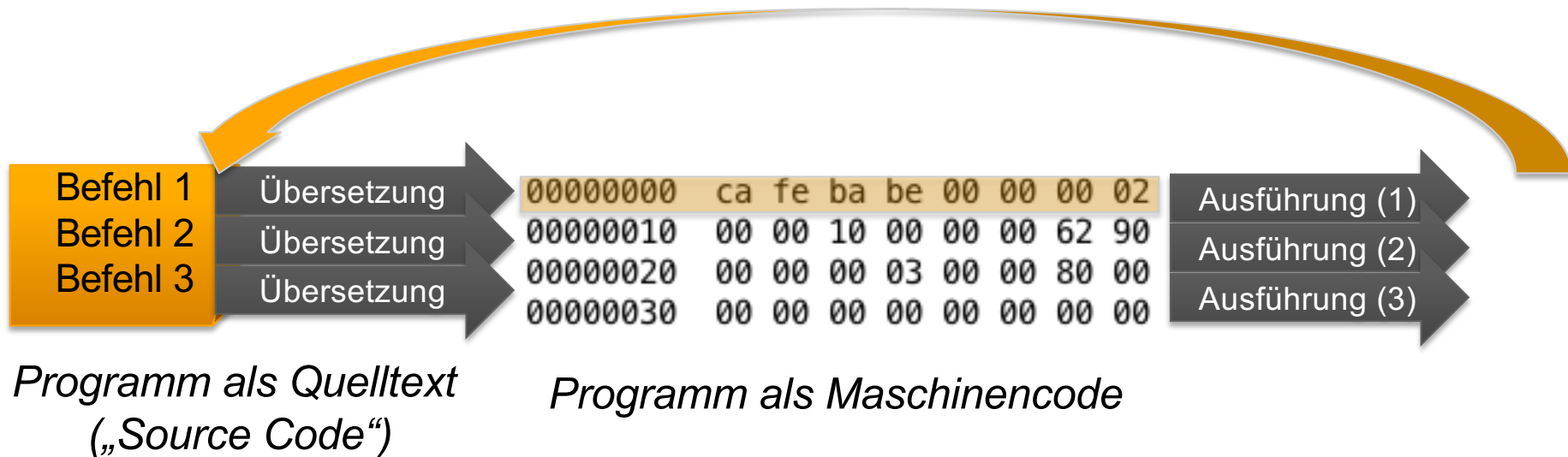
- Testen langsamer
 - Gesamtes Programm muss übersetzt werden, bis es gestartet werden kann
- Ablauf schneller
 - Ablauf wird nicht durch Übersetzung gebremst



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

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

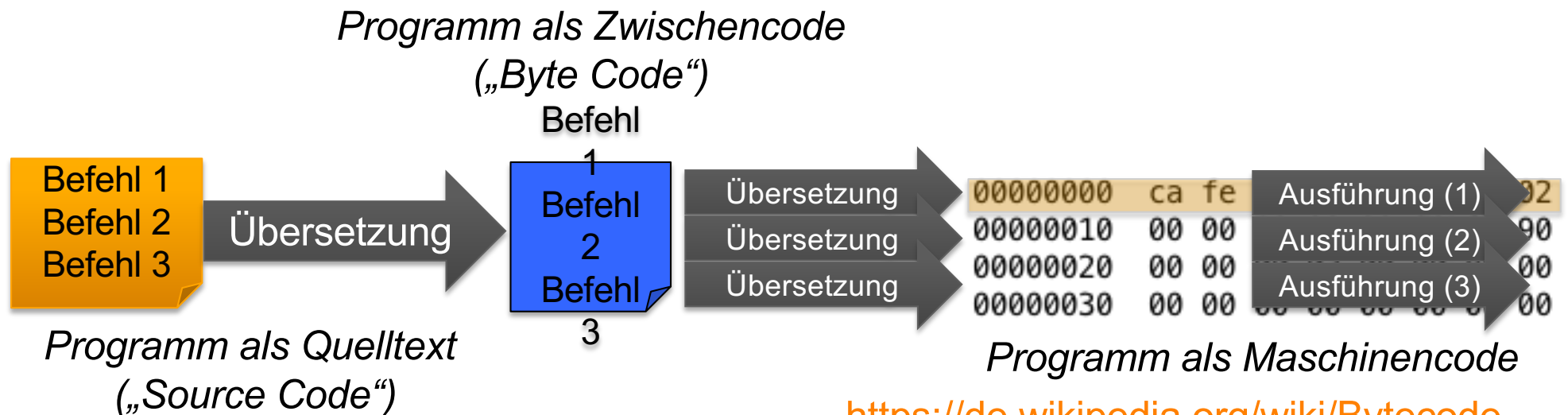


Vgl. [[MBKP2012](#), S. 21]

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



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

Syntax von Programmiersprachen

Konventionen für Programmtext

1. Regeln, wie Zeichen und Wörter in einem Programm verwendet und kombiniert werden
 - Beispiele:
 - Zeilennummern 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

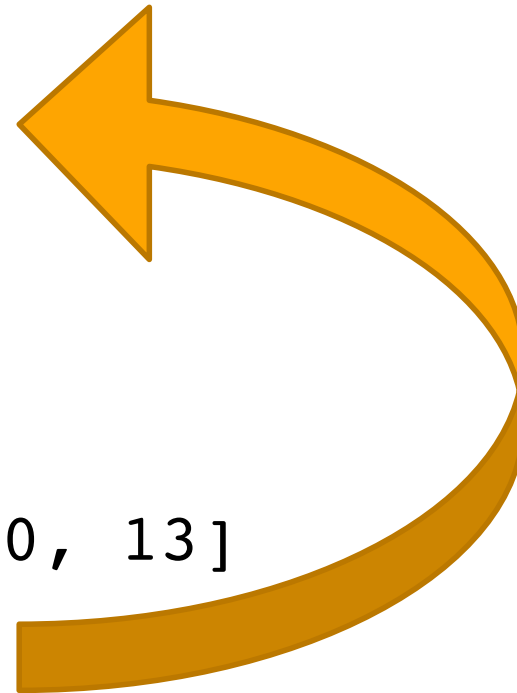
Prozedurale Programmierung

Aufspalten eines Programms in Unterprogramme („Subroutines“)

Funktion `sortiere (liste)`:
Befehle dieser Funktion

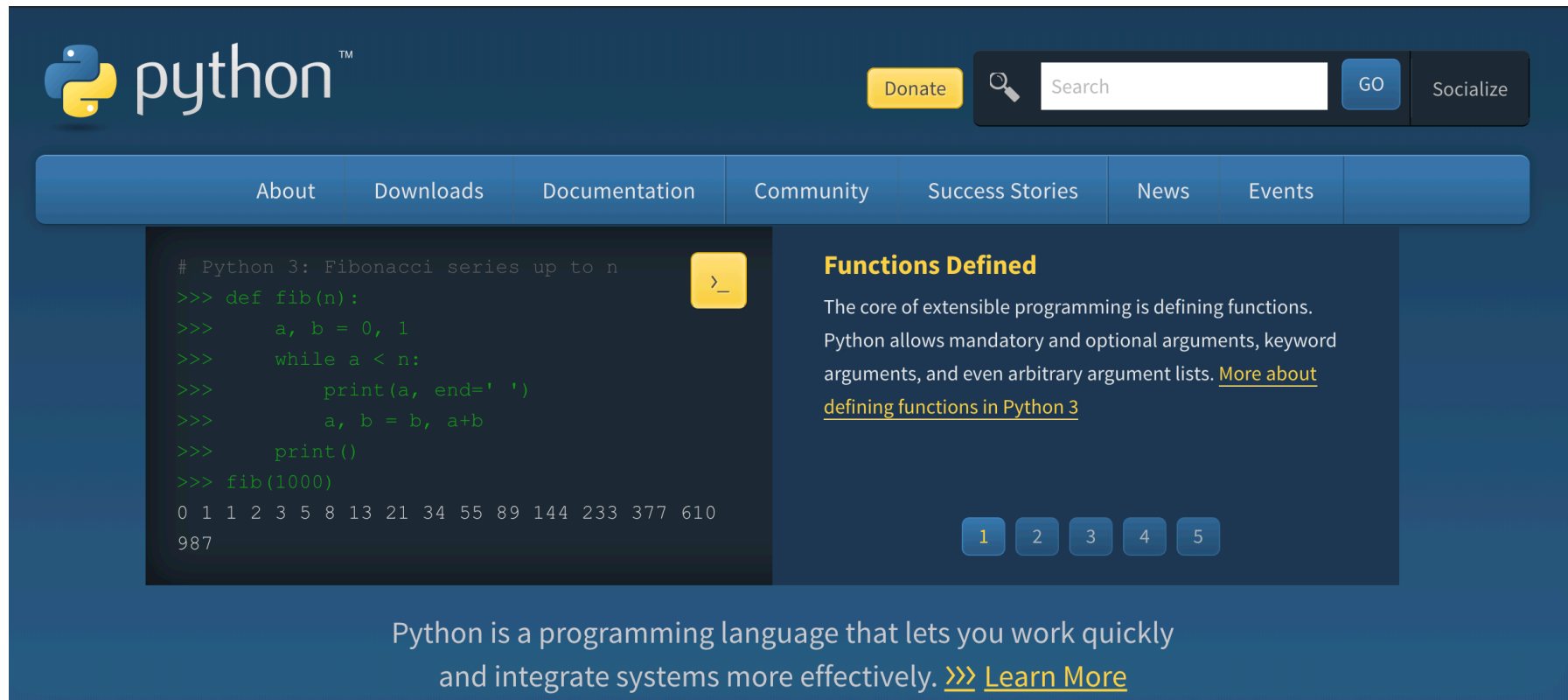
Hauptprogramm:

```
a = [1, 2, 6, 3, 10, 100, 13]  
ergebnis = sortiere (a)
```



0.2 Python

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



The screenshot shows the Python.org website interface. At the top left is the Python logo. To its right are a yellow 'Donate' button, a search bar with a magnifying glass icon and a 'GO' button, and a 'Socialize' button. Below these is a navigation menu with buttons for 'About', 'Downloads', 'Documentation', 'Community', 'Success Stories', 'News', and 'Events'. The main content area is split into two columns. The left column contains a code editor with a yellow 'Run' button (a yellow square with a black cursor) and the following Python code:

```
# 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
```

The right column features a section titled 'Functions Defined' in yellow. The text below reads: 'The core of extensible programming is defining functions. Python allows mandatory and optional arguments, keyword arguments, and even arbitrary argument lists. [More about defining functions in Python 3](#)'. At the bottom of this section are five blue buttons labeled '1', '2', '3', '4', and '5'. Below the code editor and article snippet, a dark blue banner contains the text: 'Python is a programming language that lets you work quickly and integrate systems more effectively. >>> [Learn More](#)'.

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



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

YouTube wurde fast vollständig in Python entwickelt

The screenshot shows a YouTube search results page for the query "programming python". The search bar at the top contains the text "programming python" and a "Search" button. Below the search bar, the page displays "Broadcast Yourself™" and navigation links for "Home", "Videos", "Channels", and "Programmes". The search results are listed as "“programming python” results 1 - 20 of about 625". There are tabs for "All", "Channels", and "Playlists", and sorting options for "Sort by: Relevance" and "Uploaded: Anytime".

The search results list four videos:

- Python 1: Installation and beginner's tutorial.** Here's a very basic tutorial to get you introduced to the Python programming language. Table of Contents * Download Python * Intro to the command ...
 8:25 ★★★★★ 2 years ago 46,182 views murvillage
- Python Programming Tutorial - 1 - Installing Python** Part 2 - www.youtube.com Tutorial on how to install python. One of many tutorials to come.
 3:19 ★★★★★ 6 months ago 21,713 views thenewboston
 part 1 - part 2 - part 3 - part 4 - part 5 ... more
- Python Programming Tutorial - 2 - Numbers and Math** Part 3 - www.youtube.com Fun tutorial with numbers! Yay!
 5:40 ★★★★★ 6 months ago 15,960 views thenewboston
 part 1 - part 2 - part 3 - part 4 - part 5 ... more
- First 5 Minutes Programming with Python** How do you get started with Python Programming in just 5 Minutes? Here I show you, using Python 2.5 and the bundled IDLE editor, how to start ...
 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 :-)) that YouTube is almost entirely written in Python. (And now I can rub shoulders with the developers since they're all Googlers now... :-))

```
On 12/12/06, Kurt B. Kaiser <kbk@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.
>
--
> KBK
>
> Python-Dev mailing list
> Python-Dev at python.org
> http://mail.python.org/mailman/listinfo/python-dev
> Unsubscribe: http://mail.python.org/mailman/options/python-dev/guido40python.org
```

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

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

Google & Python

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:Random'
```

```
for i in range(5):
```

```
    webbrowser.open_new_tab(URI)
```

The screenshot shows a web browser window displaying the Wikipedia article for Linköping. The browser's address bar shows the URL <http://en.wikipedia.org/wiki/Linköping>. The page title is "Linköping - Wikipedia, the free encyclopedia".

The article content includes the following text:

Linköping [[ɪ̃nːœ̂ːpɪŋ]] 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](#).

The article includes a "Contents" table of contents with the following items:

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

On the right side of the article, there is a photo of the central square in Linköping, captioned "Central square in Linköping". Below the photo is the motto: "Motto: Where Ideas Become Reality (Där idéer blir verklighet)". A map of Sweden is also visible at the bottom right of the article content area.

The left sidebar contains navigation links such as "Main page", "Contents", "Featured content", "Current events", and "Random article". It also includes a search box and a "toolbox" with links like "What links here", "Related changes", "Upload file", and "Special pages".

Python: Wichtige syntaktische Konventionen

- Keine Zeilennummern
- 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/>



Individual Edition

Your data science toolkit

With over 20 million users worldwide, the open-source Individual Edition (Distribution) is the easiest way to perform Python/R data science and machine learning on a single machine. Developed for solo practitioners, it is the toolkit that equips you to work with thousands of open-source packages and libraries.

[Download](#)



Anaconda Installers

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)

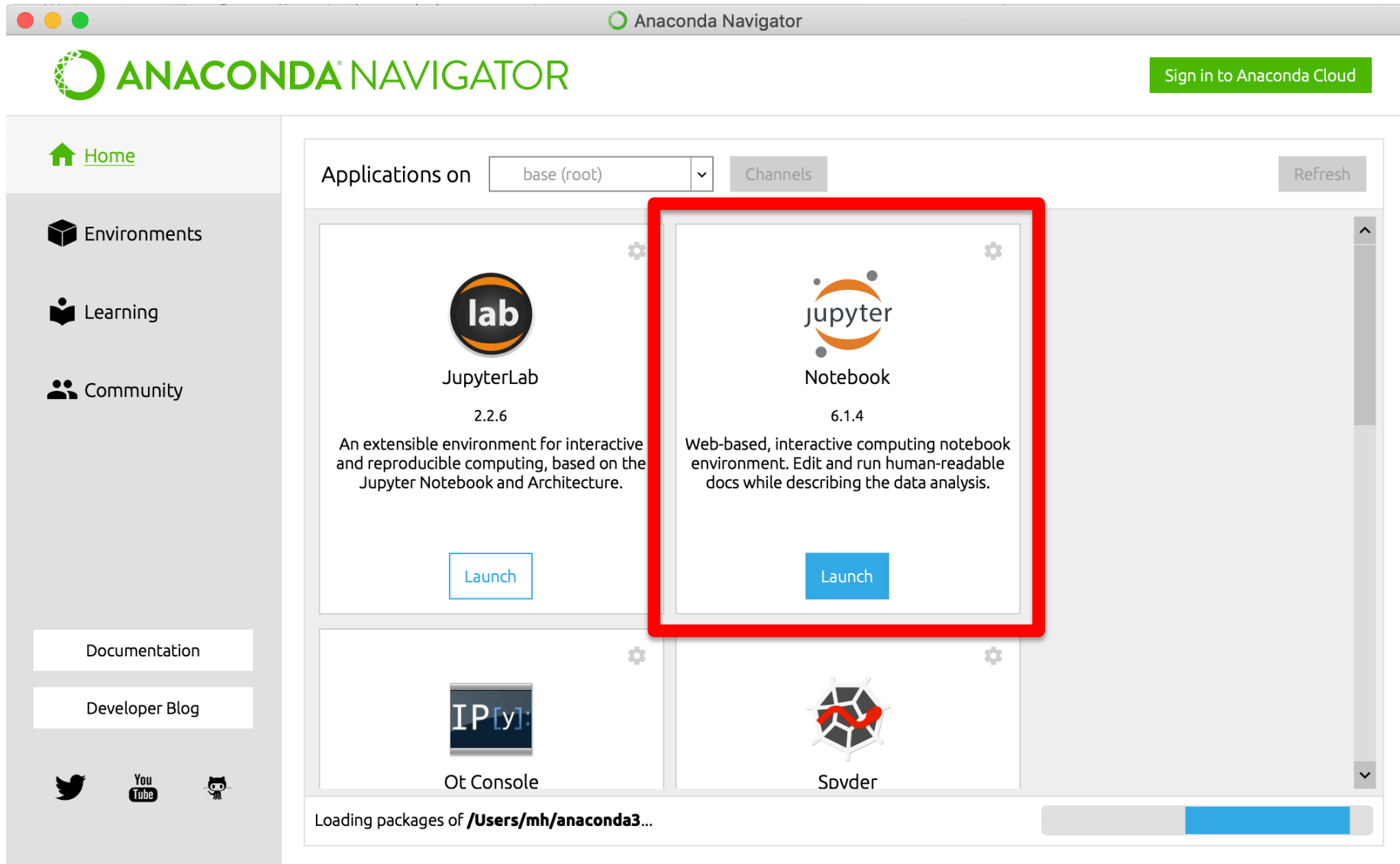
Linux

Python 3.8

64-Bit (x86) Installer (550 MB)

64-Bit (Power8 and Power9) Installer (290 MB)

Anaconda Navigator



Jupyter Notebooks

localhost:8888/tree/pip-lecture%20link

jupyter Logout

Files Running Clusters Conda

Select items to perform actions on them. Upload New ↻

<input type="checkbox"/>	Name ↑	Last Modified ↑
<input type="checkbox"/>	..	seconds ago
<input type="checkbox"/>	exercises	5 months ago
<input type="checkbox"/>	notebooks	4 months ago
<input type="checkbox"/>	slides	5 months ago
<input type="checkbox"/>	Geo Experiments.ipynb	5 months ago
<input type="checkbox"/>	Linear Algebra.ipynb	5 months ago
<input type="checkbox"/>	Untitled.ipynb	21 days ago
<input type="checkbox"/>	LICENSE	5 months ago
<input type="checkbox"/>	README.md	5 months ago

Jupyter Notebooks

The screenshot shows a web browser window with the URL `localhost:8888/tree/pip-lecture%20link`. The Jupyter interface includes a 'Logout' button in the top right. Below the browser window, there are two red rectangular boxes highlighting specific parts of the interface:

- The larger box on the left highlights the 'New' dropdown menu, which is open and shows the following options:
 - Notebook:
 - Python [conda env:py27]
 - Python [conda env:py36]
 - Python [conda root]
 - Python [default]
 - Other:
 - Text File
 - Folder
 - Terminal
- The smaller box on the right highlights the 'Upload', 'New', and refresh icons in the file browser toolbar.

The file browser on the left shows a directory structure with folders like `exercises`, `notebooks`, and `slides`, and files like `Geo Experiments.ipynb`, `Linear Algebra.ipynb`, `Untitled.ipynb`, `LICENSE`, and `README.md`. The right side of the interface shows a table with columns for 'Name' and 'Last Modified'.

Unser erstes Notebook

The image shows a Jupyter Notebook interface in a web browser. The browser's address bar displays the URL `localhost:8888/notebooks/pip-lecture%20link/Untitled...`. The Jupyter logo and the text "jupyter Untitled1" are visible in the top left. On the top right, there is a Python logo and a "Logout" button. Below the browser window is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Help, Trusted, and Python [default]. A toolbar below the menu contains icons for saving, adding, deleting, copying, pasting, undo, redo, and running code. The main area of the notebook shows a code cell with the following Python code:

```
In [ ]: print('Nerd is the new Cool')
```

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 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>

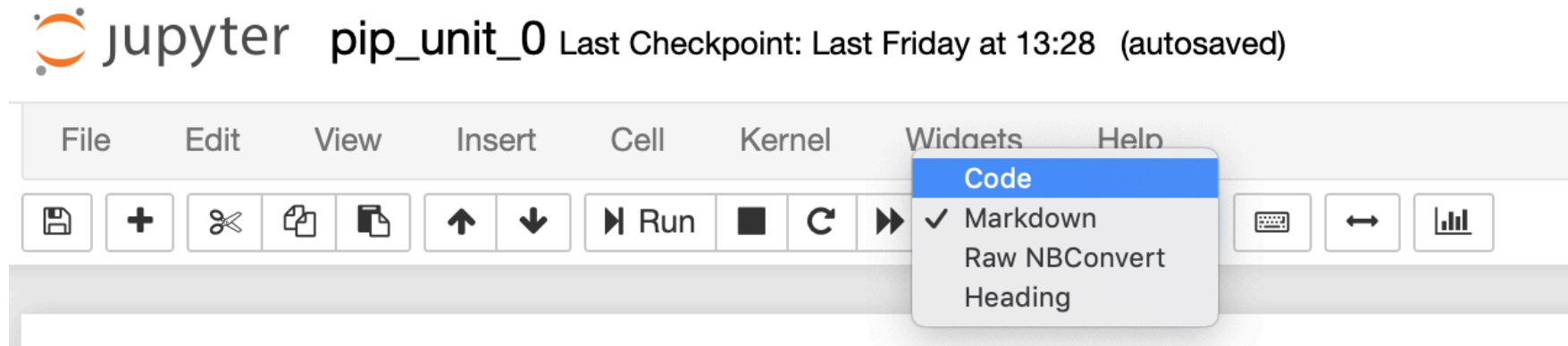
Bedienung

(1) Link zu allen Notebooks

(2) Dateiname des aktuellen Notebooks

The screenshot shows the JupyterLab interface. At the top left, the Jupyter logo is followed by the text 'jupyter pip_unit_0 (unsaved changes)'. A red arrow points from the text '(1) Link zu allen Notebooks' to the Jupyter logo. Another red arrow points from the text '(2) Dateiname des aktuellen Notebooks' to the notebook name 'pip_unit_0'. Below this is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. To the right of the menu bar are 'Trusted' and 'Python 3' indicators. Below the menu bar is a toolbar with icons for save, add, cut, copy, paste, up, down, run, stop, refresh, and next. A red arrow points from the text '(3) Ausführung der aktuellen Zelle' to the 'Run' button in the toolbar. The main area shows a code cell with the prompt 'In [1]:' and the code 'print(3 + 5)'. The output of the cell is '8'. A 'Slide Type' dropdown menu is visible in the top right of the code cell area.

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
```

Trusted | Python 3 ●

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

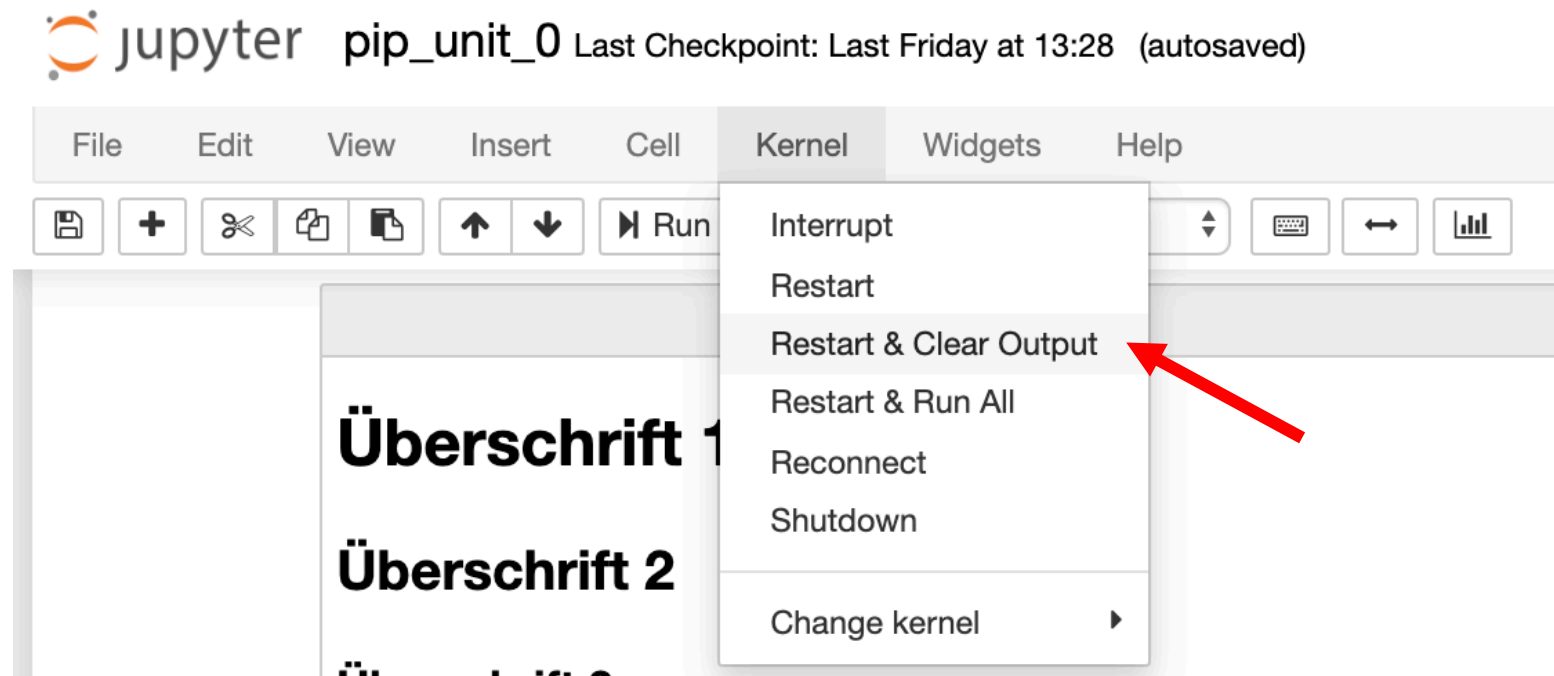


```
-----
KeyboardInterrupt                                Traceback (most recent call last)
<ipython-input-3-9f0840b03a7e> in <module>
      1 # Diese Schleife läuft ewig
      2 while True:
----> 3     pass

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