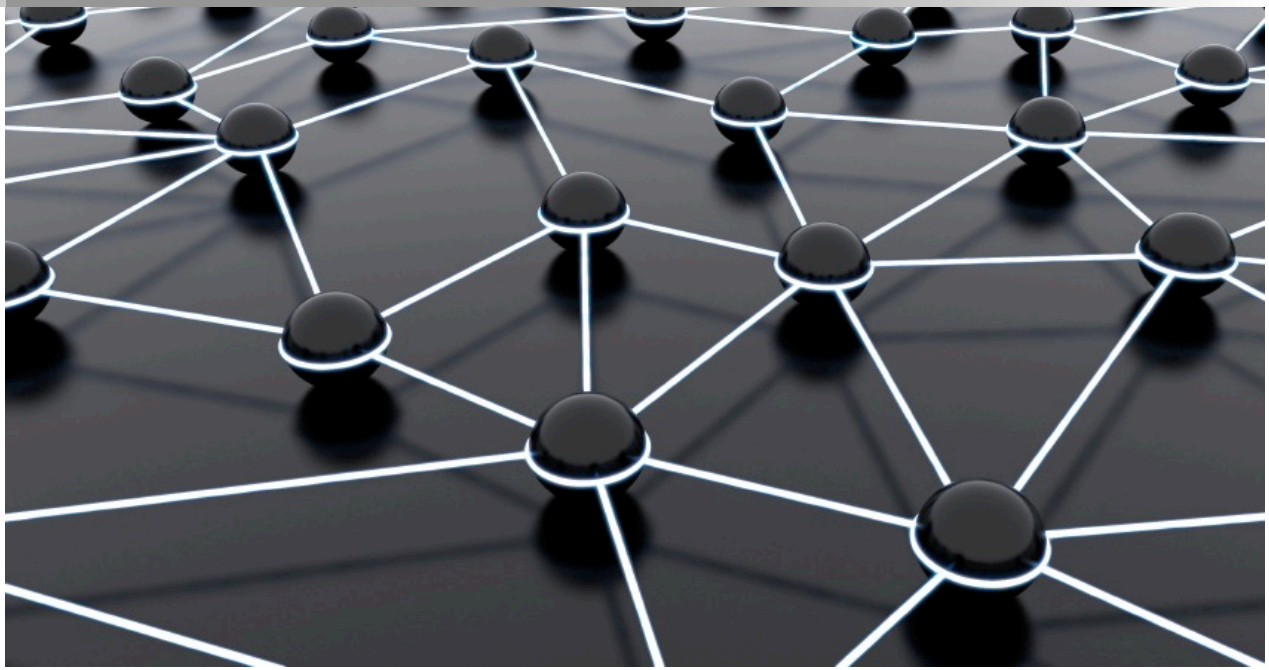


Programmierung in Python

Einheit 0: Grundlagen der Programmierung

Prof. Dr. Martin Hepp

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

Professur für Web Science und Digitalisierung

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/Maschinensprache

Techniken, um Quelltext in Maschinencode zu übersetzen

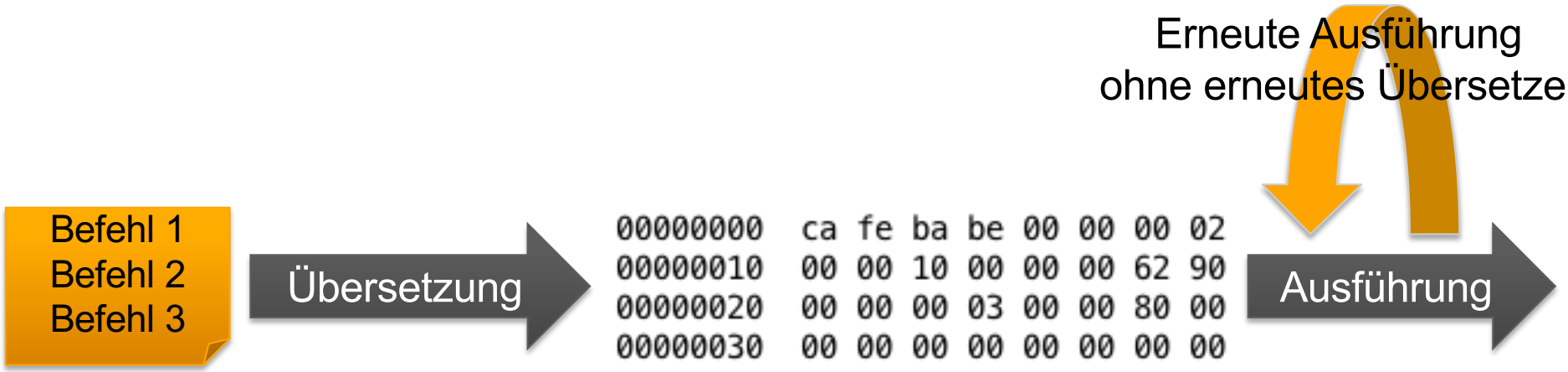
1. Compiler
2. Interpreter
3. Kombinierter Ansatz: Bytecode und Virtual Machines

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



Programm als Quelltext („Source Code“)

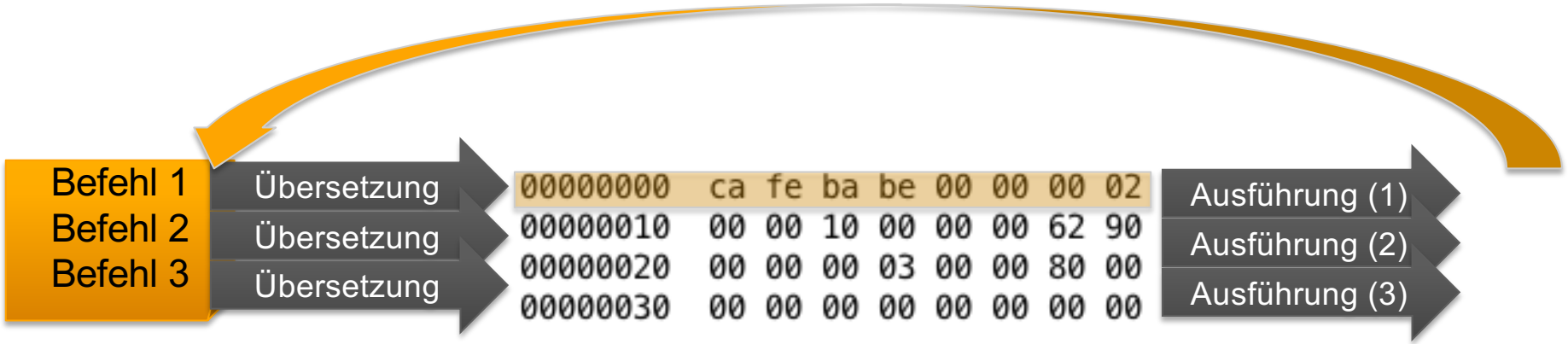
Programm als Maschinencode

Vgl. [MBKP2012, S. 21]

Interpreter

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“)

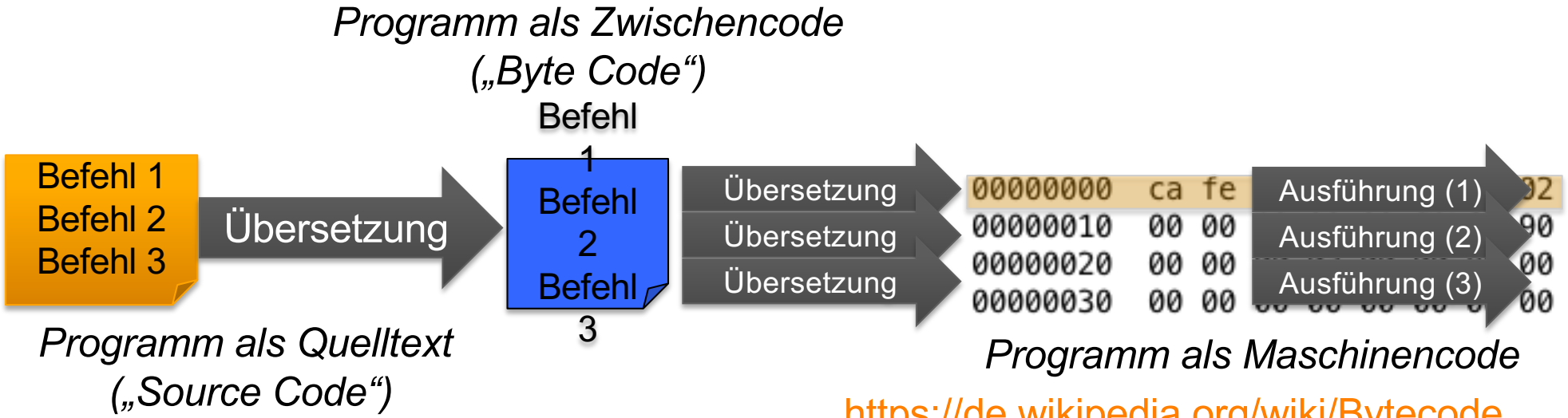
Programm als Maschinencode

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



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)

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 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

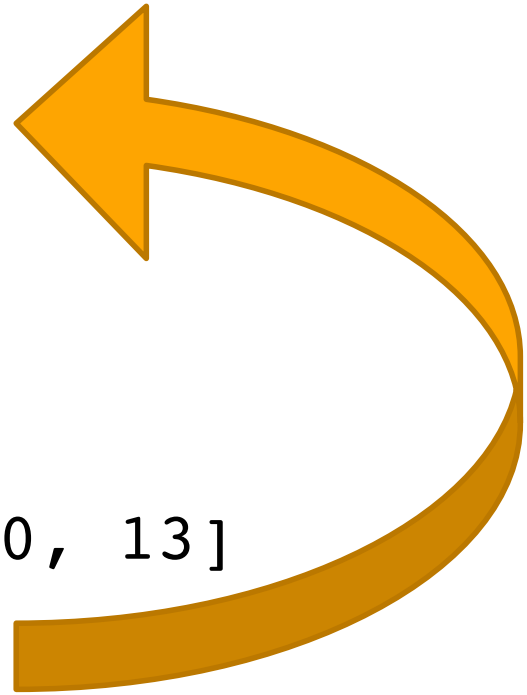
Prozedurale Programmierung

Aufspalten eines Programms in Unterprogramme („Subroutines“)

```
Funktion sortiere (liste):
    Befehle dieser Funktion
```

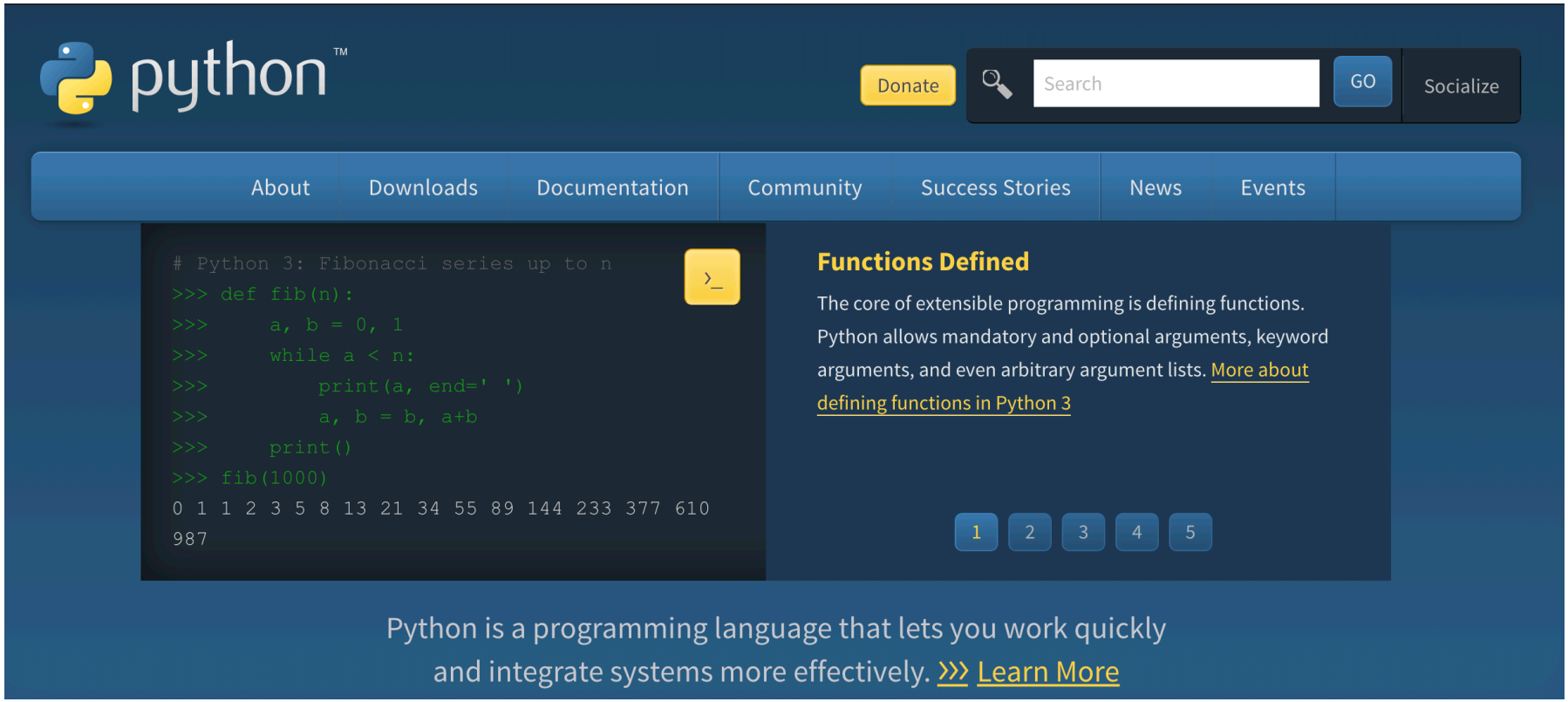
Hauptprogram:

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



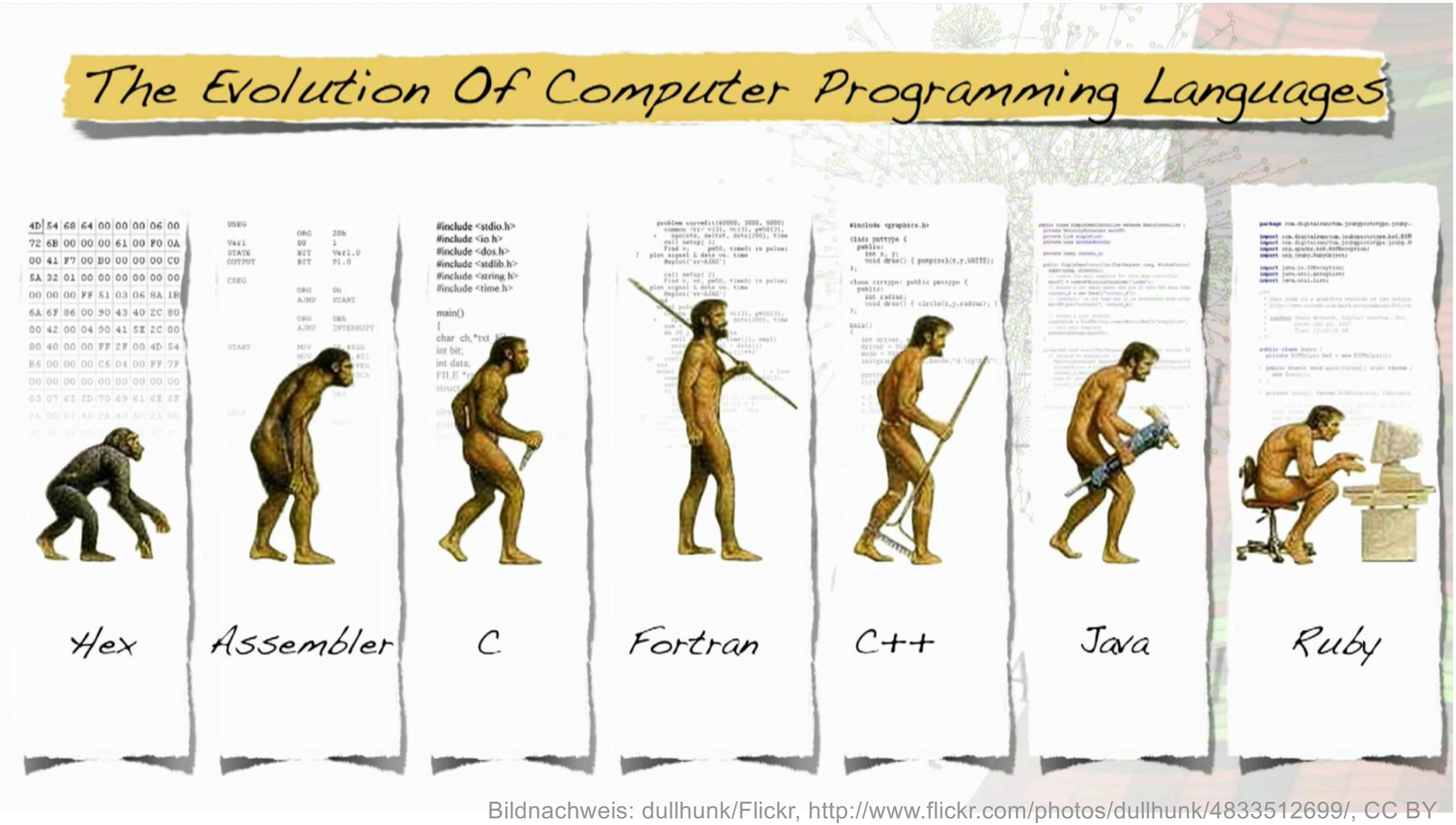
0.2 Python

https://www.python.org/



Zusatzmaterial

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

Zusatzmaterial

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)

Zusatzmaterial

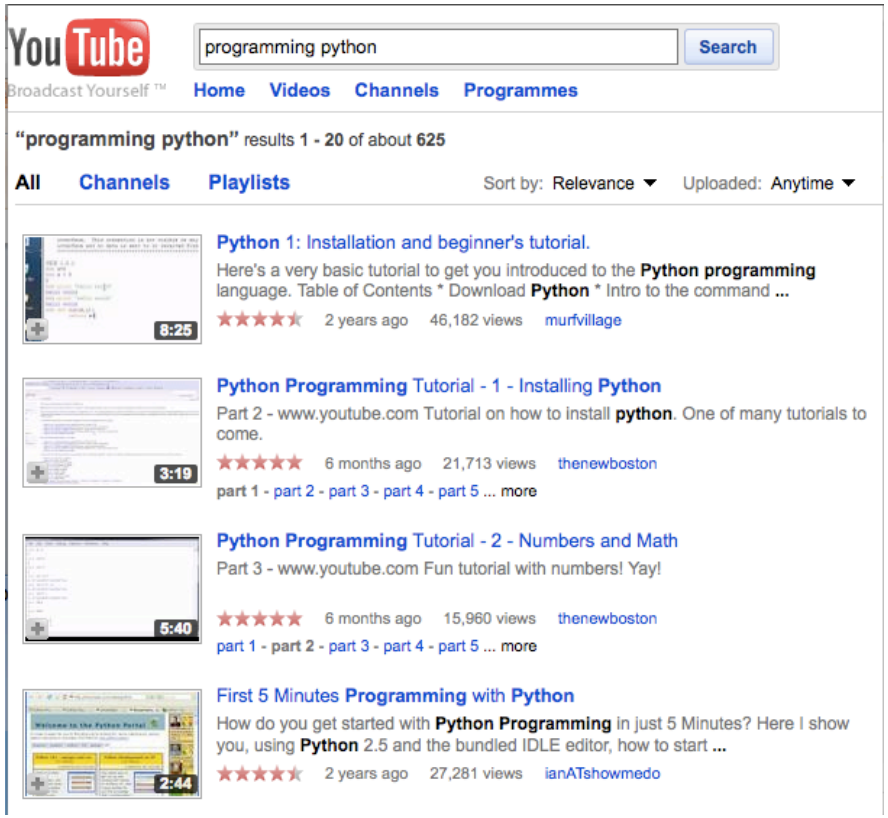
Programming in Python

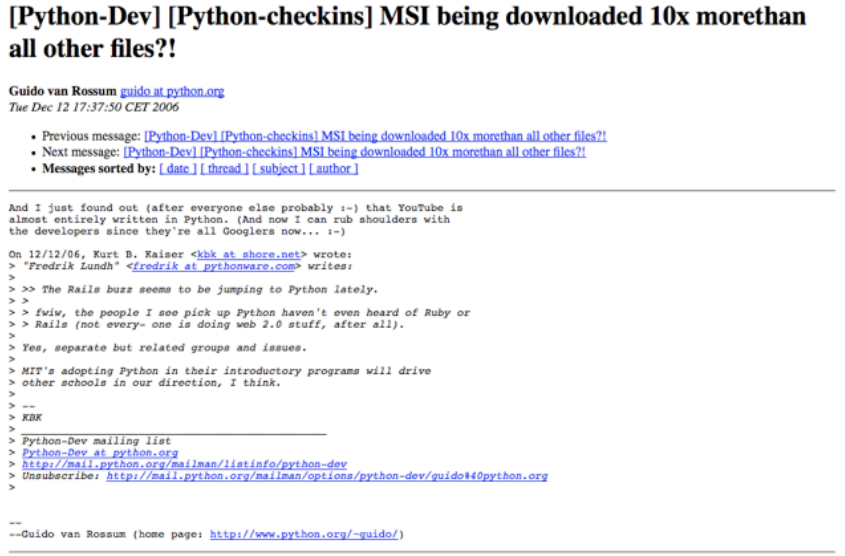
- Leicht zu lernen
- Gut lesbar
- Mächtig
- Schnell
- Für viele Plattformen verfügbar

https://www.python.org/

Zusatzmaterial

YouTube wurde fast vollständig in Python entwickelt





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

Guido van Rossum guido at 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 at shore.net> wrote:
> *"Fredrik Lundh" <fredrik at 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/guido%40python.org*
>

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

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

Zusatzmaterial

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

Zusatzmaterial

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

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/

Individual Edition

Your data science toolkit

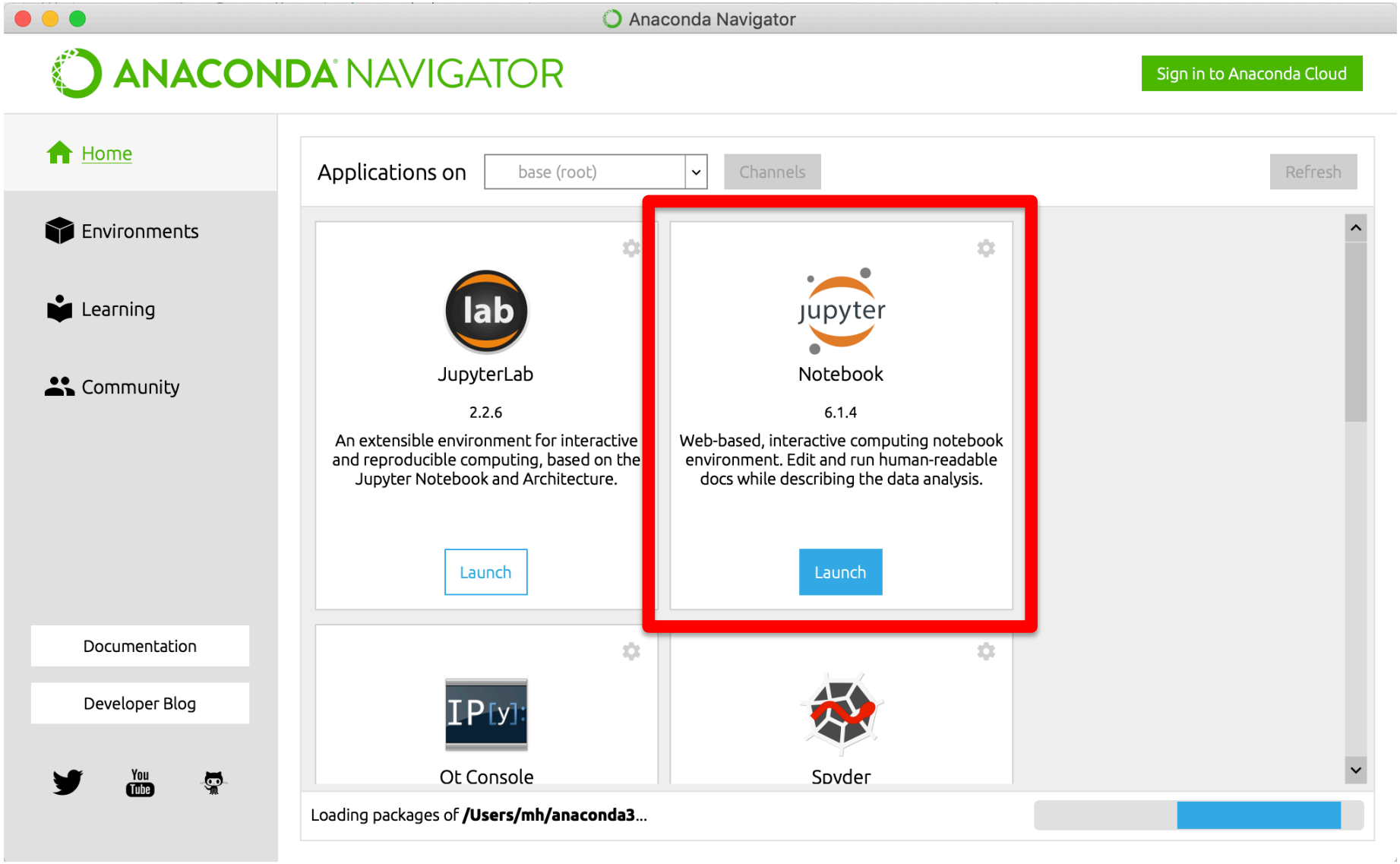
With over 20 million users worldwide, the open-source Individual Edition (Distribution) is the easiest way to perform Python/R data science and machine learning on a single machine. Developed for solo practitioners, it is the toolkit that equips you to work with thousands of open-source packages and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

Download

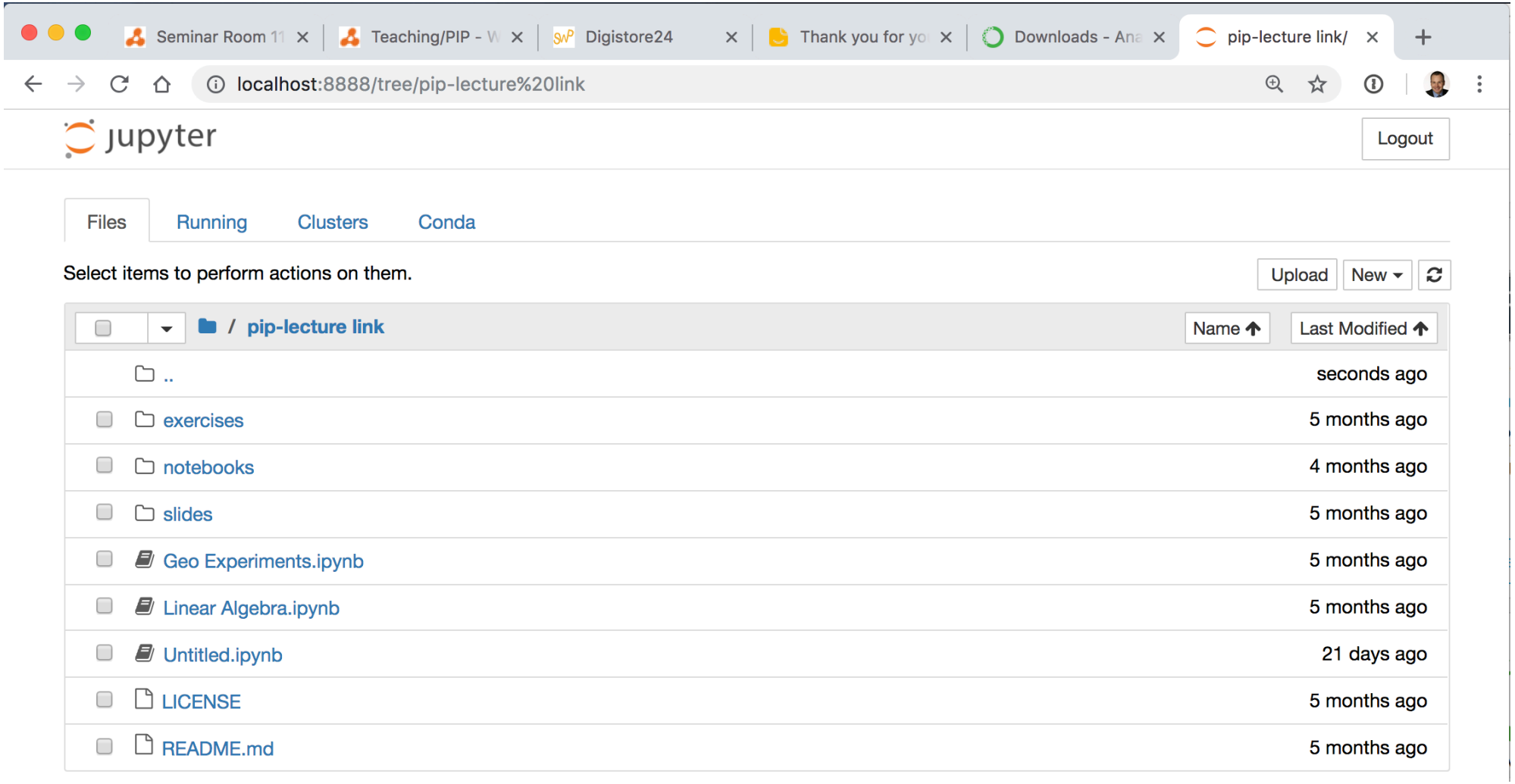
Anaconda Installers

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

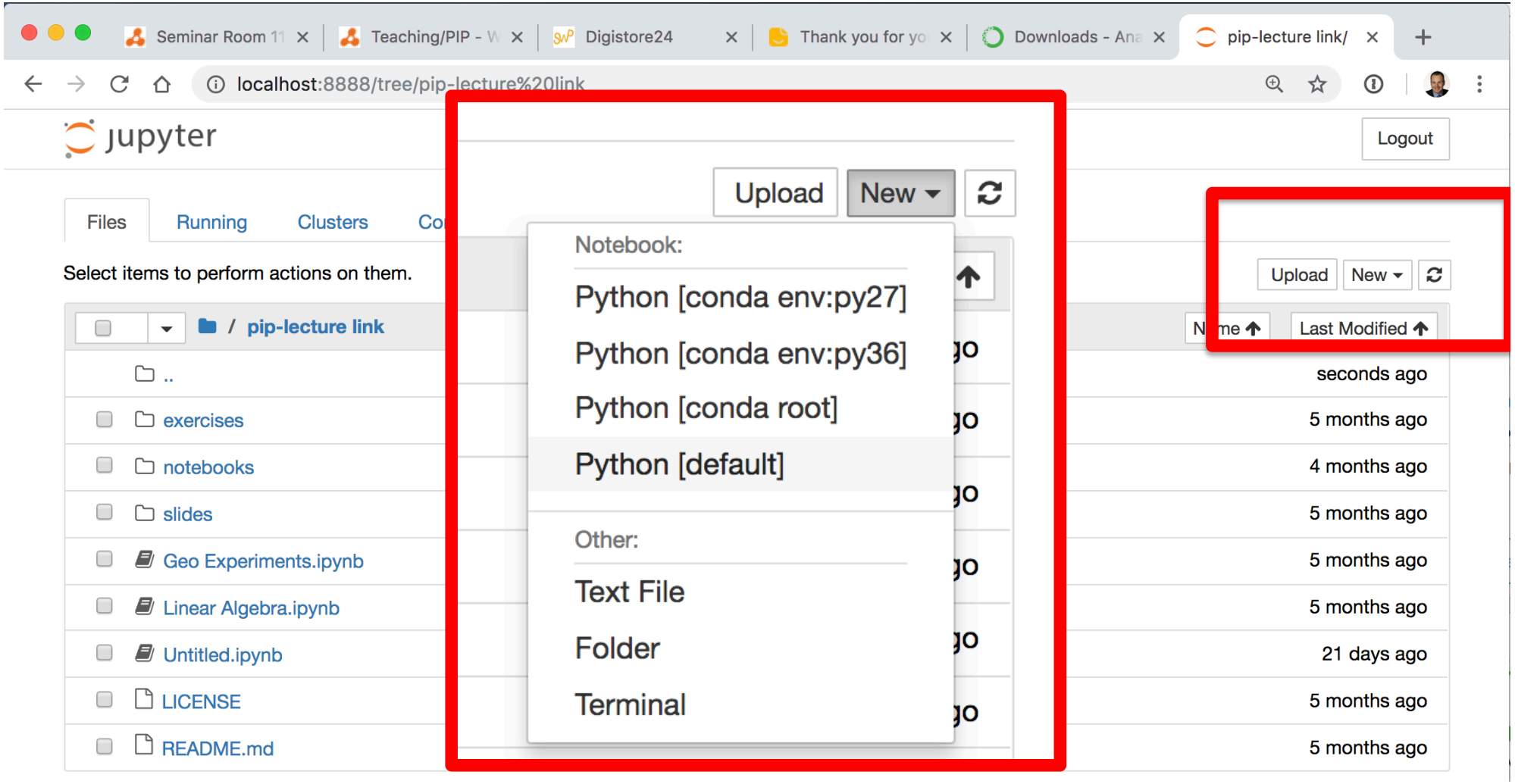
Anaconda Navigator



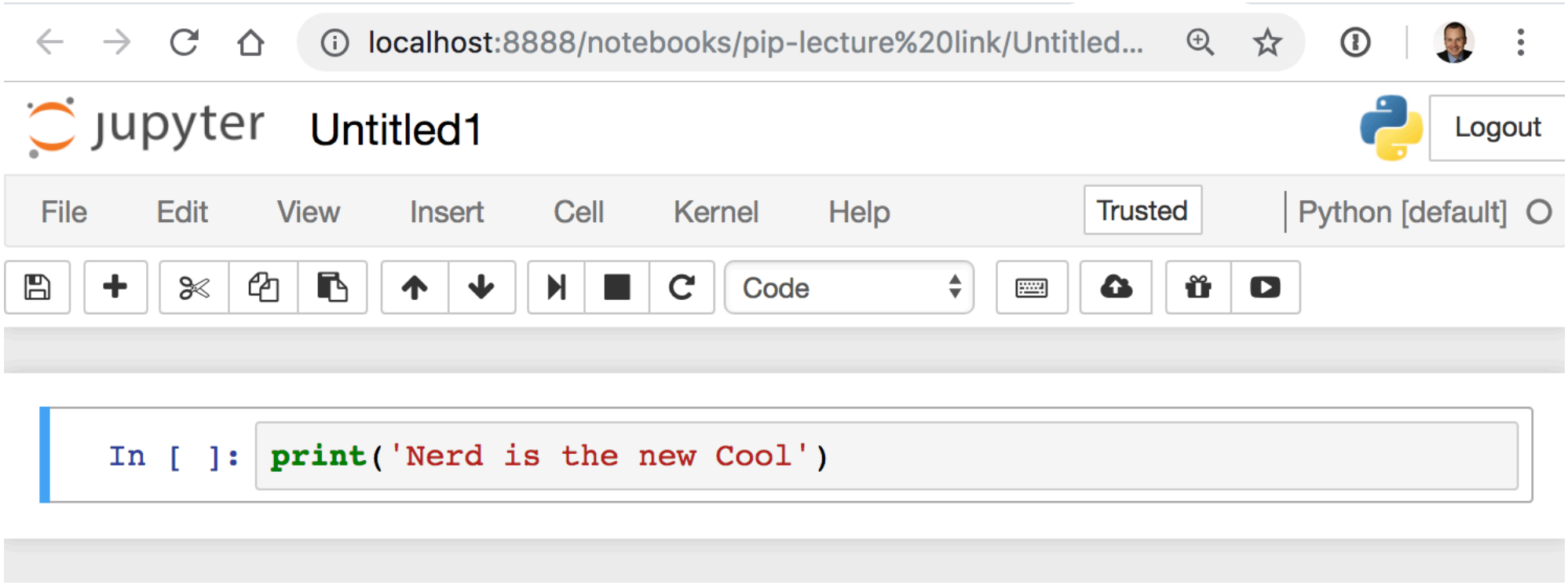
Jupyter Notebooks



Jupyter Notebooks



Unser erstes Notebook



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.“

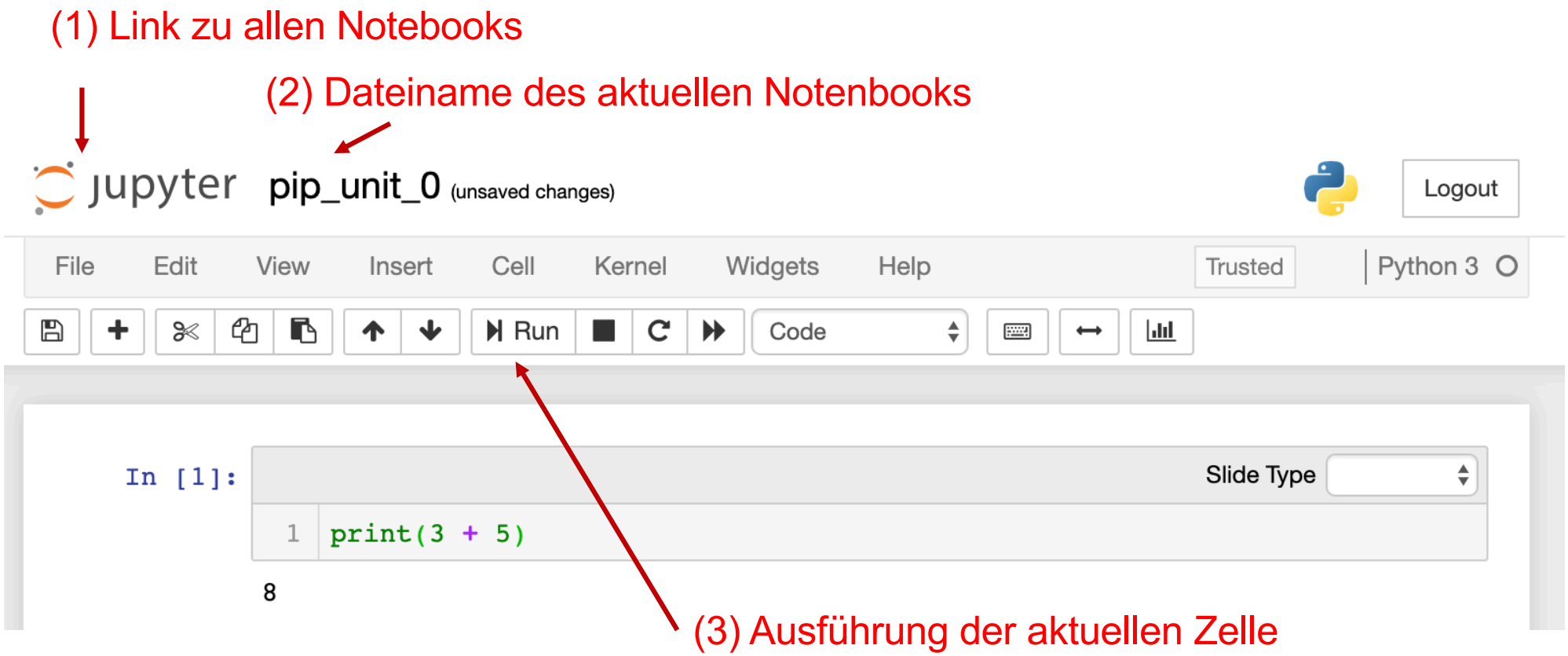
```
1 for person in ['joe', 'judy']:
2     print('Hello', person)
```

```
Hello joe
Hello judy
```

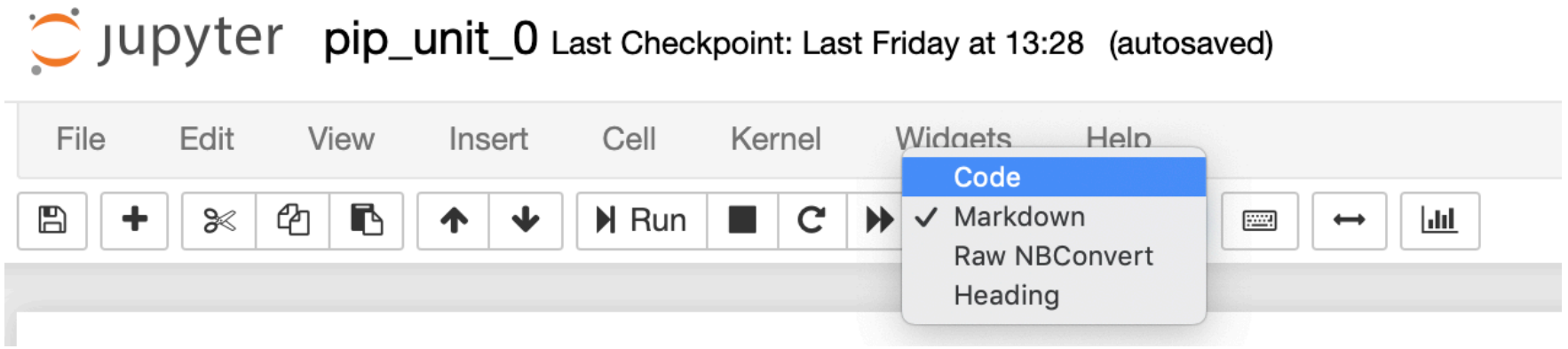
Donald E. Knuth

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

Bedienung


(1) Link zu allen Notebooks
(2) Dateiname des aktuellen Notenbooks
jupyter
pip_unit_0 (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
In [1]:
Slide Type
1 print(3 + 5)
8
(3) Ausführung der aktuellen Zelle

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/jypyter-notebook-shortcuts-bf0101a98330

Markdown

Einfache Auszeichnungssprache, um Text zu formatieren

```
# Überschrift 1
## Überschrift 2
### Überschrift 3

*kursiv*

**fett**

***fett und kursiv***

Listen:
1. Auf die Nummern
1. kommt es
1. nicht an

Listen
- Bullet 1
- 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

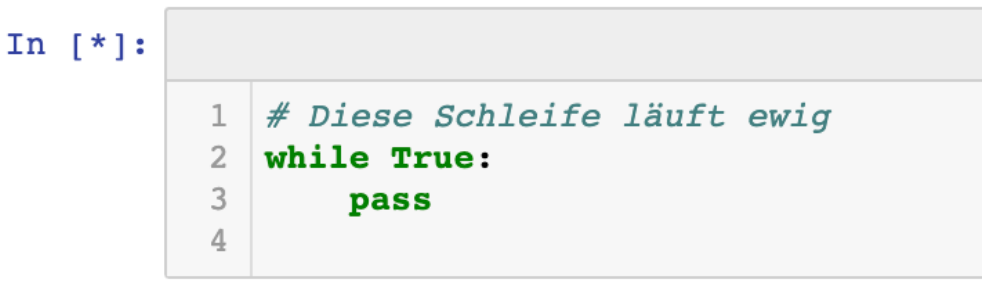
- Bullet 1
- Bullet 2

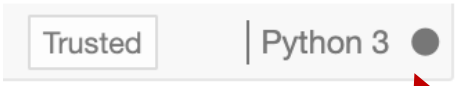
https://de.wikipedia.org/wiki/Markdown
https://help.github.com/en/github/writing-on-github/basic-writing-and-formatting-syntax
https://daringfireball.net/projects/markdown/

Bearbeitung einer Zelle abbrechen

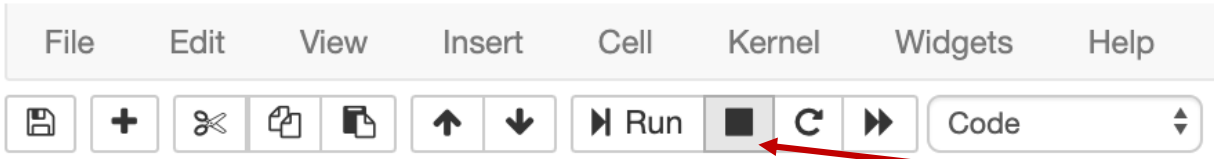
Z.B. bei Programmfehler / Endlosschleife

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





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

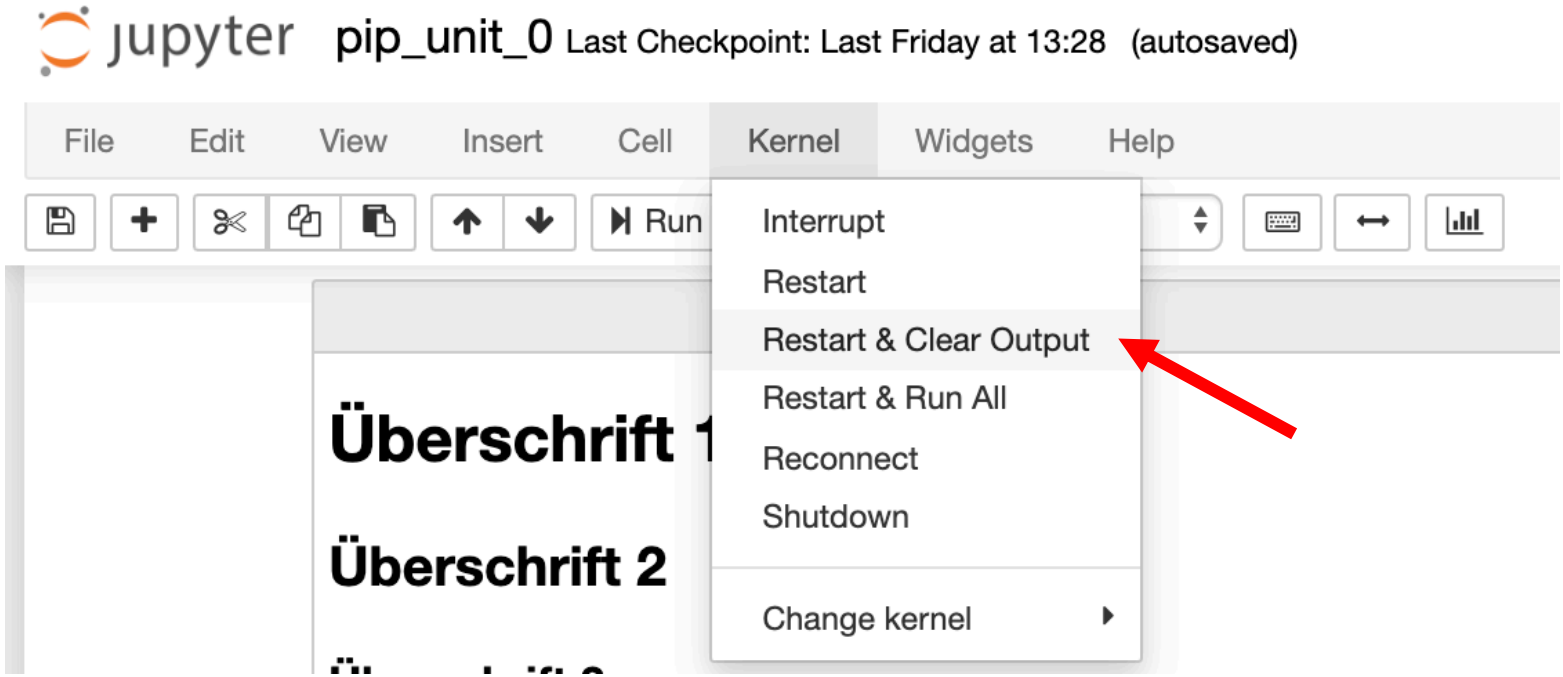


Hier klicken

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

KeyboardInterrupt:
```

Wenn nichts mehr geht: Kernel Restart

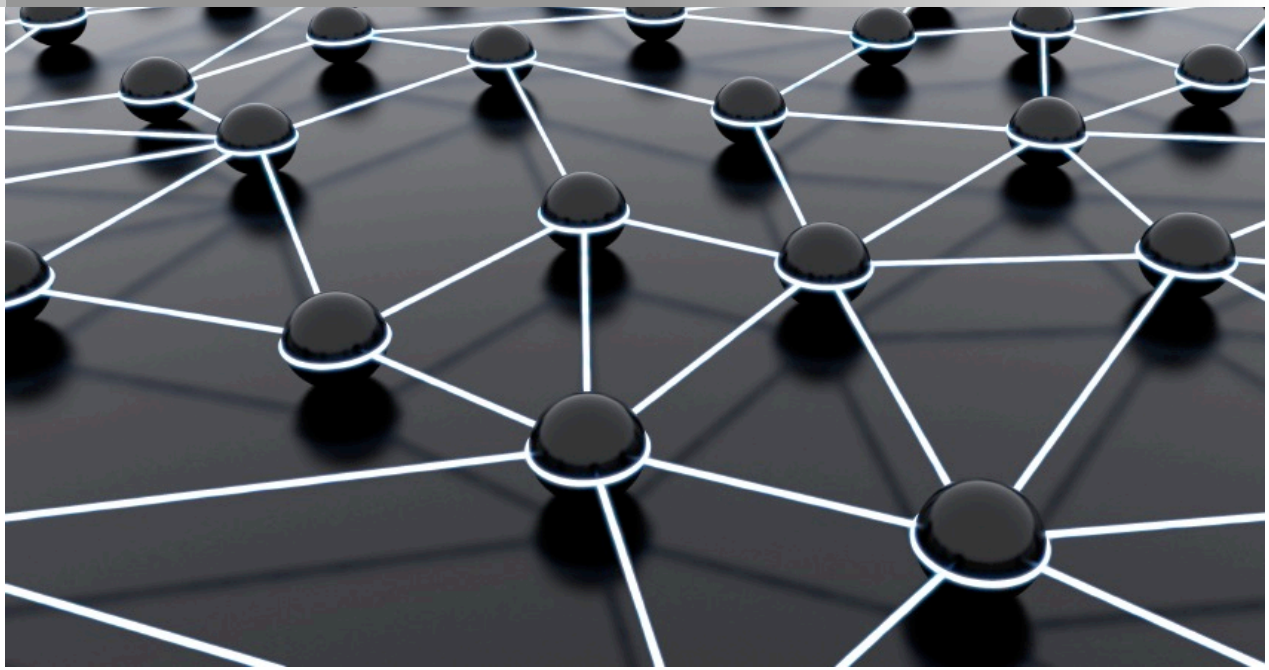


Aufgaben

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