

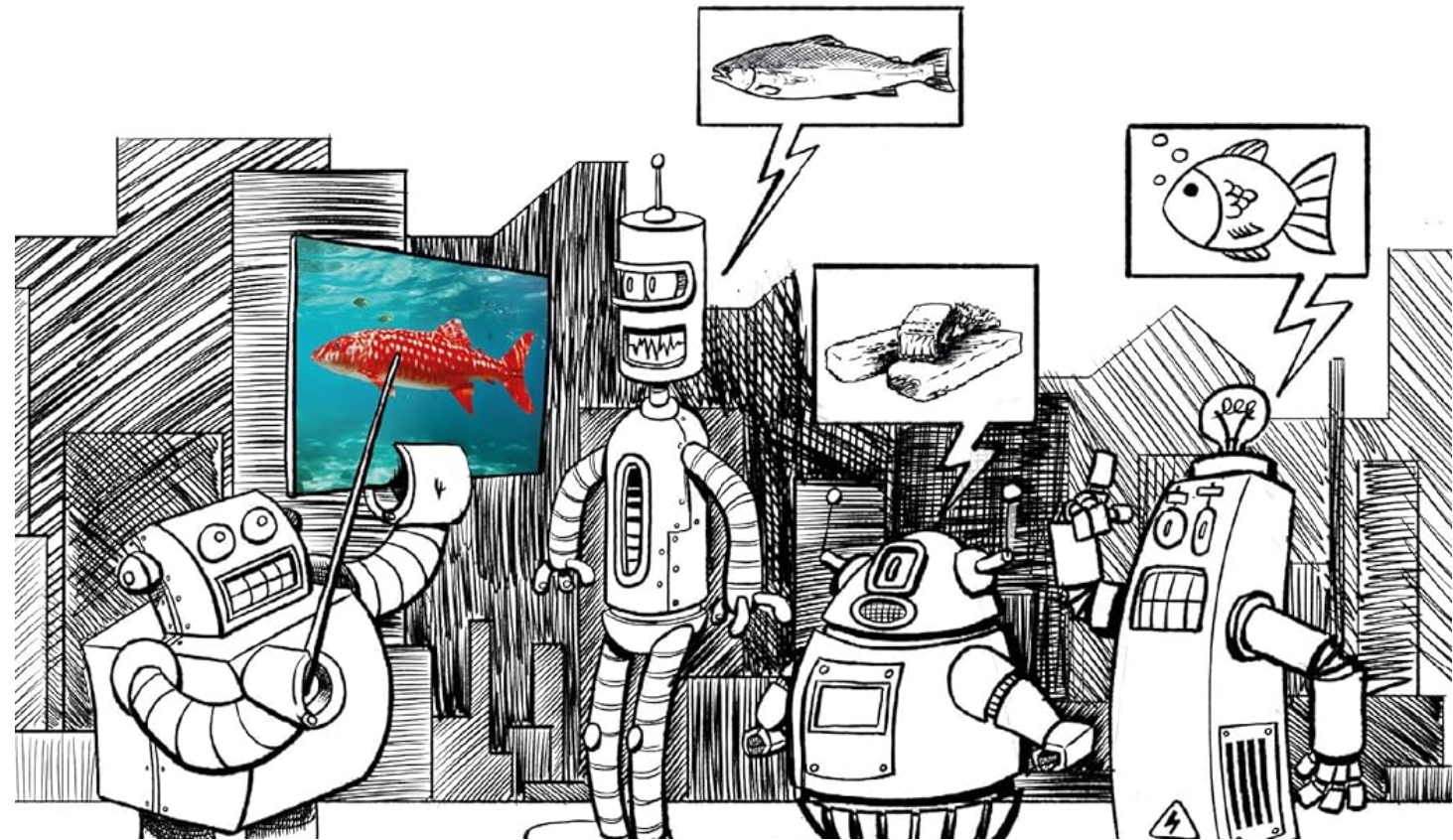
# Künstliche Intelligenz kapieren und programmieren

## Teil 1: Denkende Maschinen

Michael Weigend  
Universität Münster



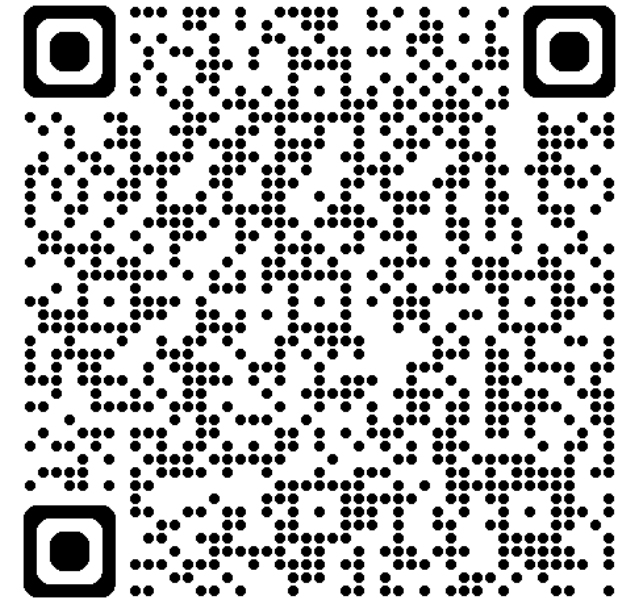
mw@creative-informatics.de  
www.creative-informatics.de  
2024



Materialien bei GitHub:  
<https://github.com/mweigend/ki-workshop>

# Damit keine Idee verloren geht ...

Google-Docs-Dokument zu diesem  
Workshop für unsere Arbeitsergebnisse



[https://docs.google.com/document/d/140INsIEWA\\_AwUwtpVMCZqtH7\\_eOkU8rmvfrgWqE5M3E/edit?usp=sharing](https://docs.google.com/document/d/140INsIEWA_AwUwtpVMCZqtH7_eOkU8rmvfrgWqE5M3E/edit?usp=sharing)

# Wer sind wir?

# Tag 1

Zeit	Thema	Inhalte
9.00	Denkende Maschinen	Pädagogische Konzepte, Einstieg in Python, Chatbots und Assistenzsysteme
11.15	KI als Spielgegner	Modellieren mit Listen, Nim-Spiel mit KI als Gegner
12.30	<i>Mittagspause</i>	
13.30	Klassifizieren	Entscheidungsbaum, k-Means-Clustering
14.45	Lernen	Lernfähiger Währungsrechner, Wegsuche, Fußgänger erkennen
16.00	<i>Ende</i>	

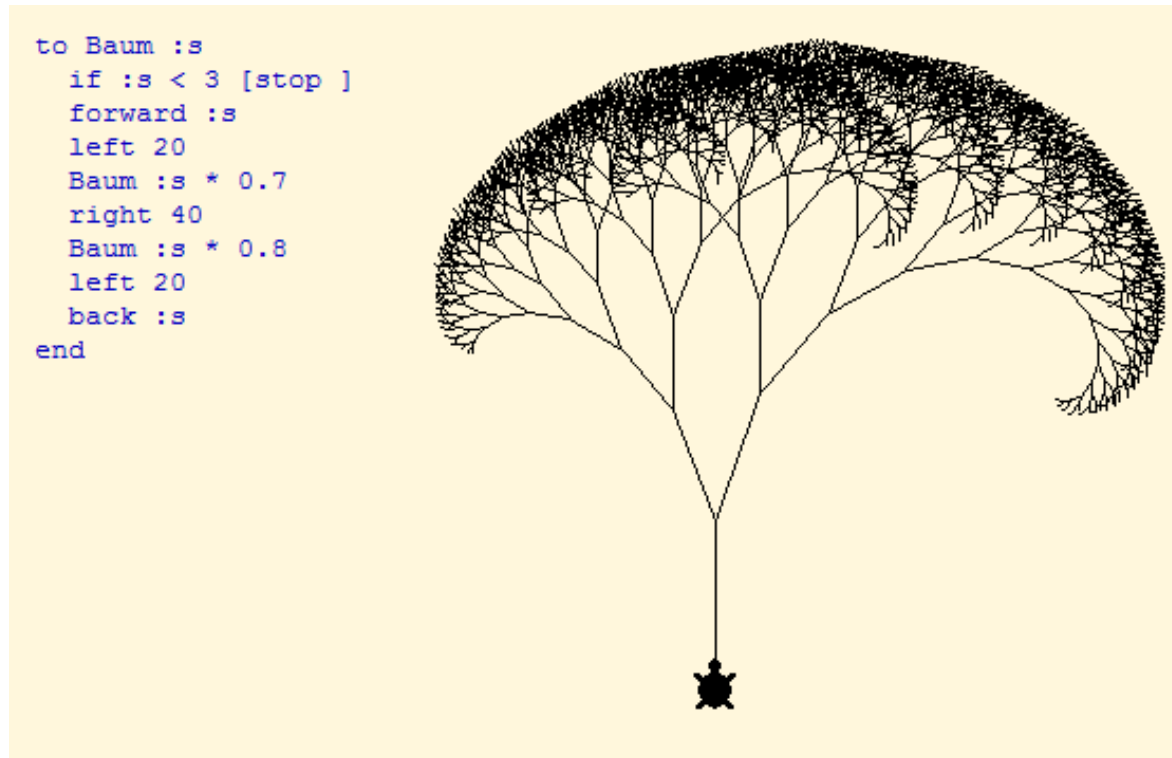
# Tag 2

Zeit	Thema	Inhalte
9.00	Perzeptron	Neuron, Aktivierungsfunktion, Daten visualisieren mit Matplotlib, Rosenblatt-Perzeptron für logische Operationen
11.00	Aus Fehlern lernen	Error-Backpropagation, einfaches künstliches neuronales Netz (KNN) mit verborgenen Knoten
12.30	<i>Mittagspause</i>	
13.30	Ziffern erkennen	NumPy, KNN mit Array-Operationen, das Ziffern erkennen kann
15.00	Anwendung von KI	Verkehrsschilder erkennen, Gesichter erfassen, Experimente mit OpenCV, Schlussrunde
16.00	<i>Ende</i>	

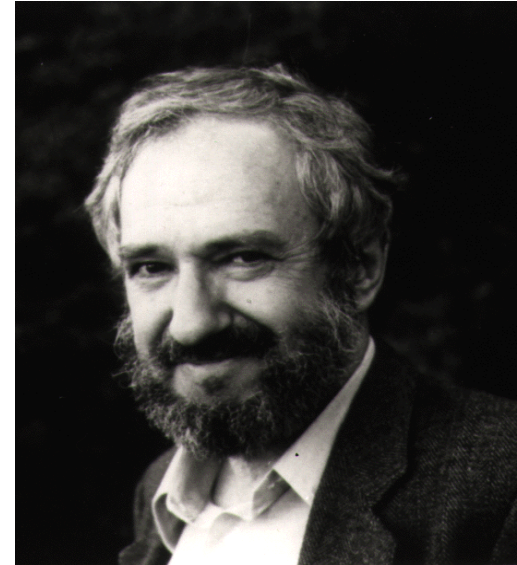
# 1.1 Pädagogische Konzepte

# 1.1.1 Konstruktivismus

Idee: Etwas Interessantes konstruieren und neues Wissen entdecken.



1967: LOGO mit Turtle-Grafik



Seymour Papert (MIT, Cambridge USA)

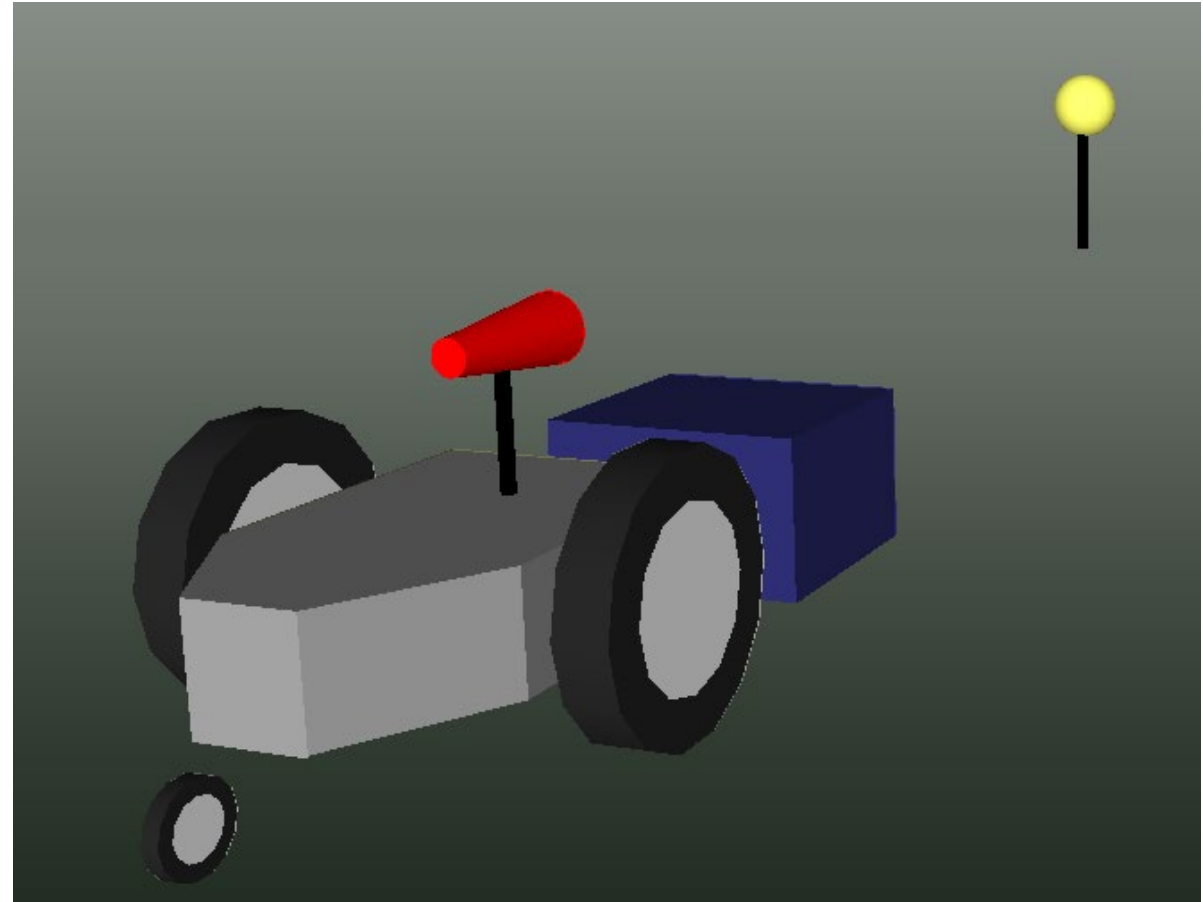
Foto: *mit.media.edu*

# Powerful Ideas

Problem: Lichtsuchender Roboter  
wird durch flaches Hindernis gestoppt.

Lösung: Nichtdeterminismus

Mit Wahrscheinlichkeit  $p$   
[Gehe  $x$  Schritte nach rechts]

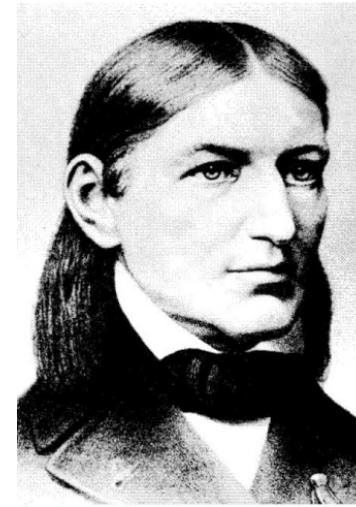




# Wurzeln



Jean Piaget (1896-1980):  
Konstruktivismus  
Lernen = subjektgesteuerte Rekonstruktion



Friedrich Fröbel (1782-1852)  
Kindergarten, Spielgaben



# Mitchel Resnick

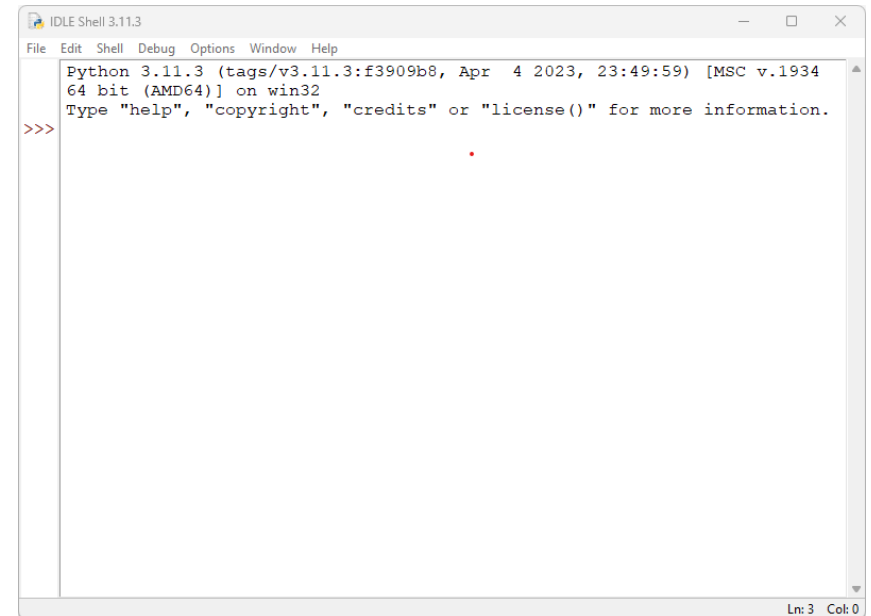
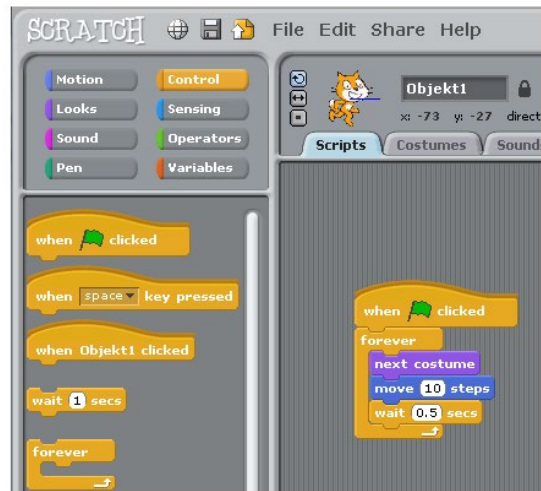
Professor am MIT und Leiter des Lifelong Kindergarten (MIT Medialab)



Konstruktionismus mit Betonung der Förderung von Kreativität.


# Moderne digitale Werkzeuge




- Niedrige Eintrittsschwelle (“low threshold high ceiling”)
- Schnelle Enzwicklungszyklen

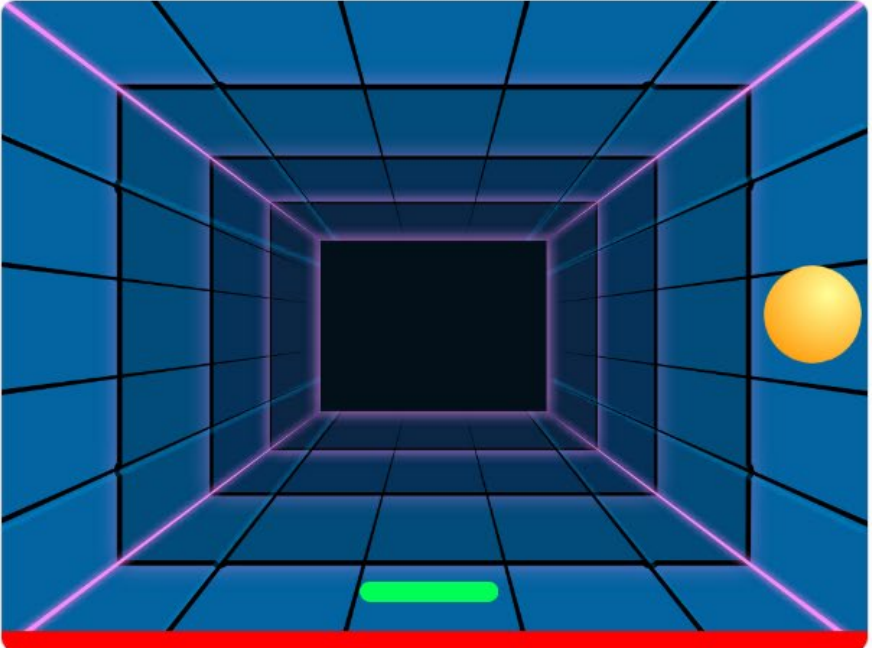


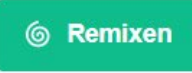




# Starterprojekt

 **Pong Starter**  
by [Scratchteam](#)



 **Remixen**  **Schau hinein**

 Danke an [natalie](#) für das Originalprojekt [Pong Starter](#).





**Anleitung**

Move the mouse to control the paddle.  
REMIX TIPS:

- \* Change what the ball looks like.
- \* Change the score if the ball touches the paddle.
- \* Add music that plays when the green flag is clicked.

**Anmerkungen und Danksagungen**

Credit to [@natalie](#)!

 17854  12647  29060  2538669

© 13. Mai 2013

- Kleines Programm, das schon funktioniert
- Start einer iterativen Entwicklung

# Ein funktionierendes Computerprogramm ist ein reales beobachtbares Phänomen

Angeleiteter Versuch  
„Kartoffelbatterie“

Wie  
funktioniert  
das?

Klappt das auch  
mit einem Apfel  
statt einer  
Kartoffel?

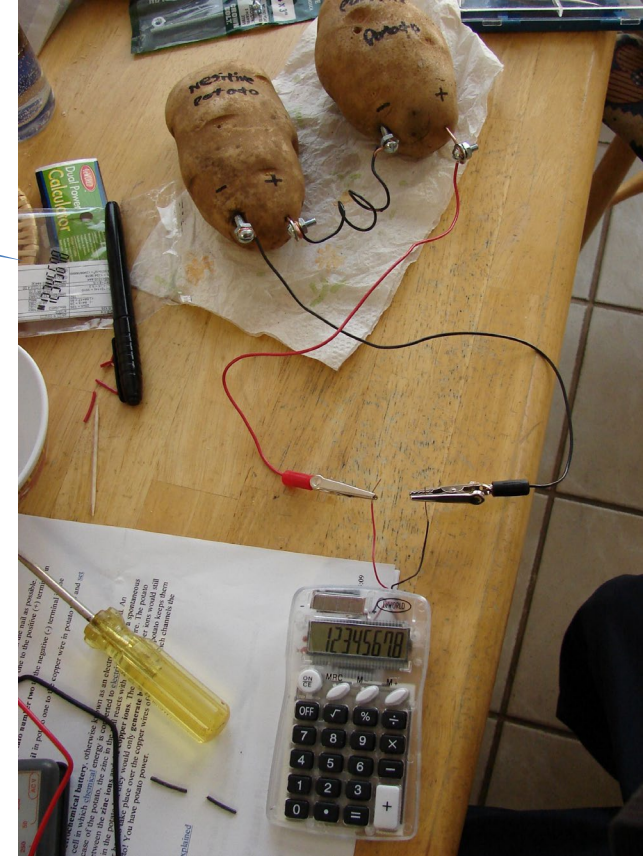


Foto: 2010 cc-by Loadmaster (David R. Tribble) in Wikimedia Commons

# 1.1.2 Fundamentale Ideen der Informatik

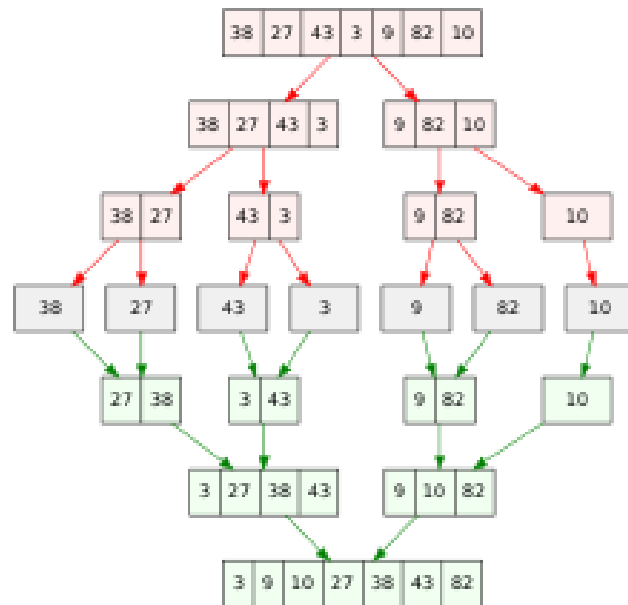
Eine fundamentale Idee bezgl. eines Gegenstandsbereichs (Wissenschaft, Teilgebiet) ist ein Denk-, Handlungs-, Beschreibungs- oder Erklärungsschema, das

1. in verschiedenen Gebieten des Bereichs vielfältig anwendbar oder erkennbar ist (Horizontalkriterium),
2. auf jedem intellektuellen Niveau aufgezeigt und vermittelt werden kann (Vertikalkriterium),
3. in der historischen Entwicklung des Bereichs deutlich wahrnehmbar ist und längerfristig relevant bleibt (Zeitkriterium),
4. einen Bezug zu Sprache und Denken des Alltags und der Lebenswelt besitzt (Sinnkriterium).

Andreas Schwill 1993 <http://informatikdidaktik.de/Forschung/Schriften/ZDM.pdf>

# Beispiel: Divide and Conquer

- a) Zerteile ein Problem
- b) Löse die Teilprobleme
- c) Kombiniere die Teillösungen zu einer Gesamtlösung



Mergesort

# Horizontalkriterium

Divide and Conquer



Sortieralgorithmen

Suchalgorithmen

Graphenalgorithmen

Bildverarbeitung

Textsuche und  
-verarbeitung

Machine Learning

Kryptographie



# Vertikalkriterium

Schule,  
Hochschule

Quicksort

Kindergarten,  
Primarschule



Kunstwerk von Kindern in der  
Nationalbibliothek Amsterdam

Divide and Conquer

# Zeitkriterium



Euklid: Die Elemente  
3. Jahrhundert vor Christus

Divide and Conquer

# Sinnkriterium

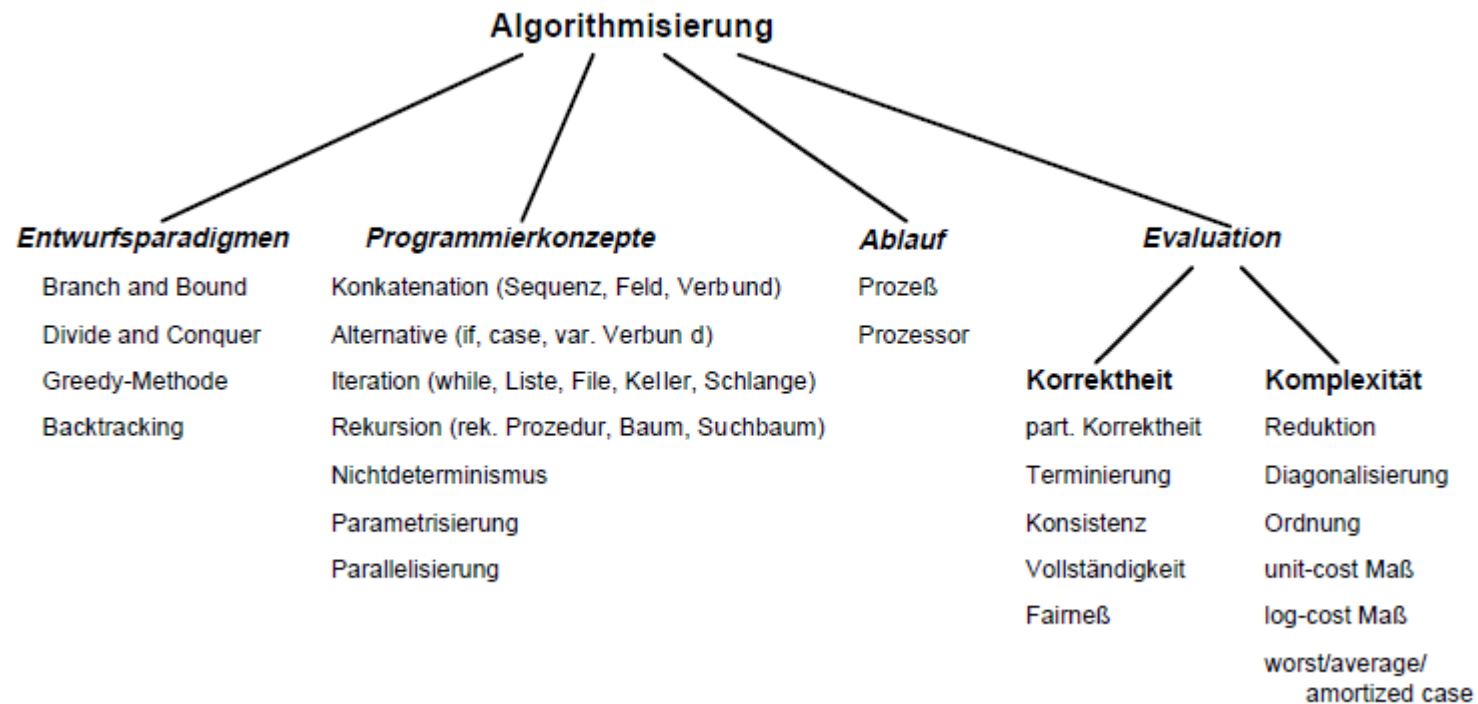


Divide and Conquer

# Übung 1.1 (5 min)

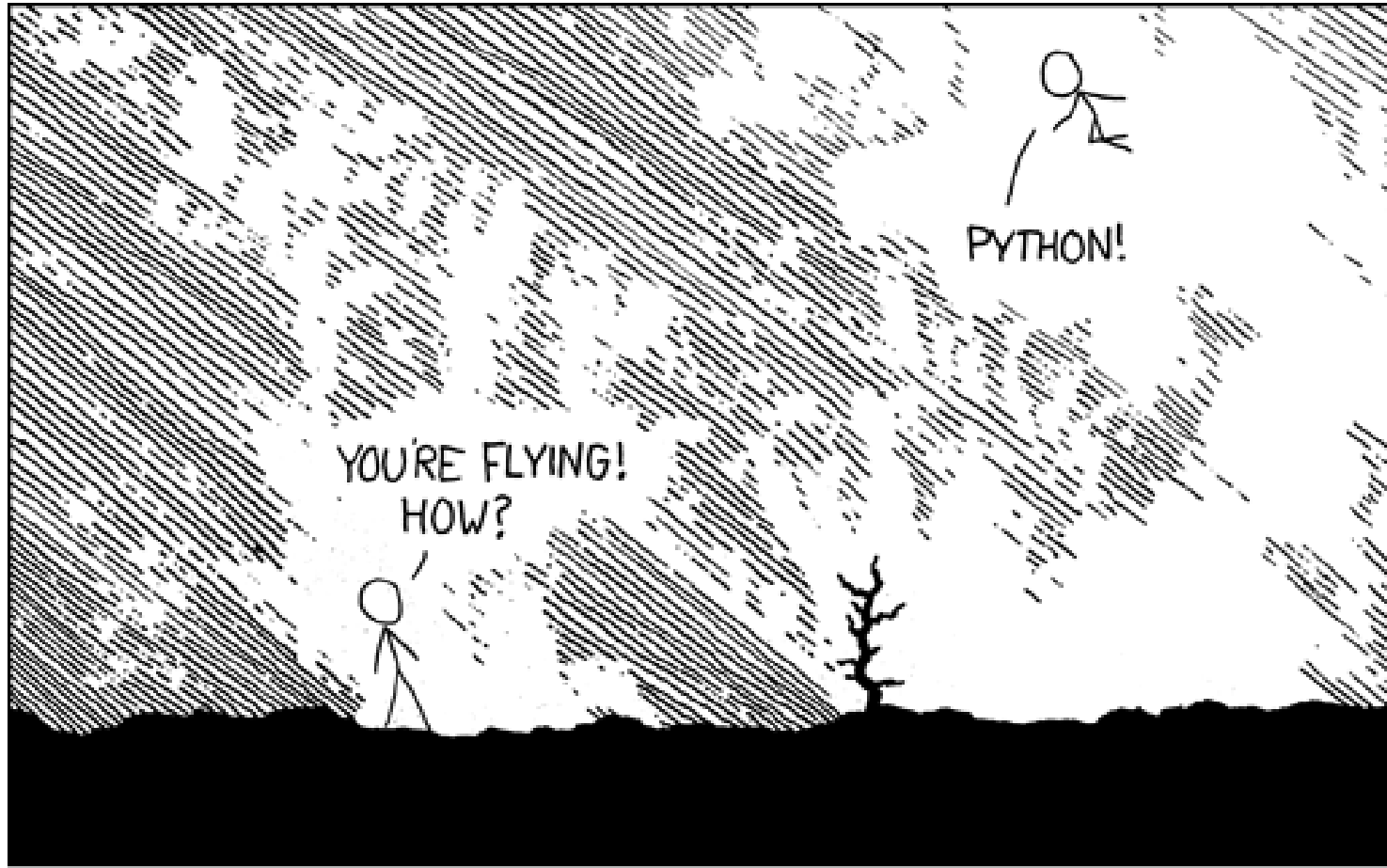
- Beschreiben Sie Ihrem Sitznachbarn eine fundamentale Idee, die Sie im Unterricht umgesetzt haben oder umsetzen möchten. Prüfen Sie inwiefern Vertikal-, Horizontal-, Zeit- und Sinnkriterium erfüllt sind.
- Beschreiben Sie Ihrem Nachbarn ein überraschendes Lernerlebnis im Sinne des Konstruktivismus von Seymour Papert, das Sie selbst bei einem Projekt gehabt haben. (Es muss kein Programmierprojekt sein.)

# Masteridee Algorithmisierung



*Schwill 1993*

# 1.2 Einstieg in Python



Schon mal mit  
Python  
programmiert?  
NumPy?  
Matplotlib?  
Tensorflow?  
OpenCV?



# Guido van Rossum (NL/USA)



Erfinder von  
Python

# Wo wird Python angewandt?



- Web Development [Google](#), [BSCW](#), [Instagram](#), [Dropbox](#)
- Sicherheitssensible Anwendungen [Versicherungen](#), [Banken](#)
- Medienindustrie [Disney VR Studio](#), [Industrial Light & Magic](#)
- Wissenschaft [American Space Telescope Institute](#), [Deutsche Gesellschaft für Luft- und Raumfahrt](#)
- Robotik [Universal Robots](#)
- Künstliche Intelligenz

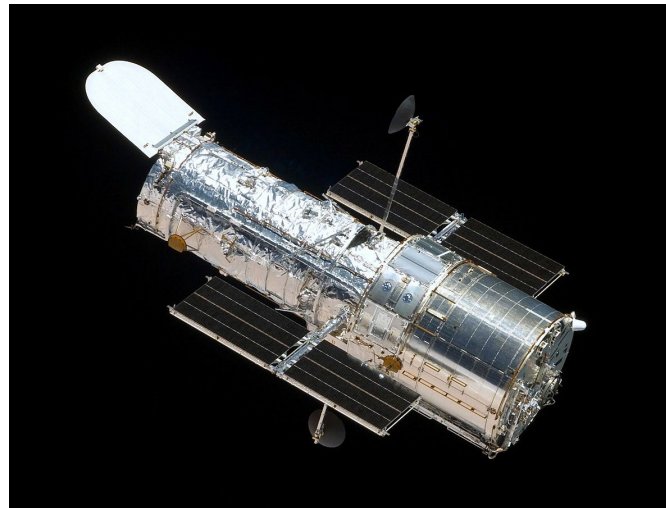


Foto: Josh Baxt 2009 cc by-sa



# Vorteile von Python

- Einfach
- Konsistent
- Kurze gut lesbare Programmtexte
- Plattformunabhängig
- Open Source
- Mehrere Programmierparadigmen
- Viele freie Module für spezielle Zwecke (PyPI)

# Programm

- Algorithmus in formaler Sprache
- Kann vom Computer ausgeführt werden
- Folge von Anweisungen

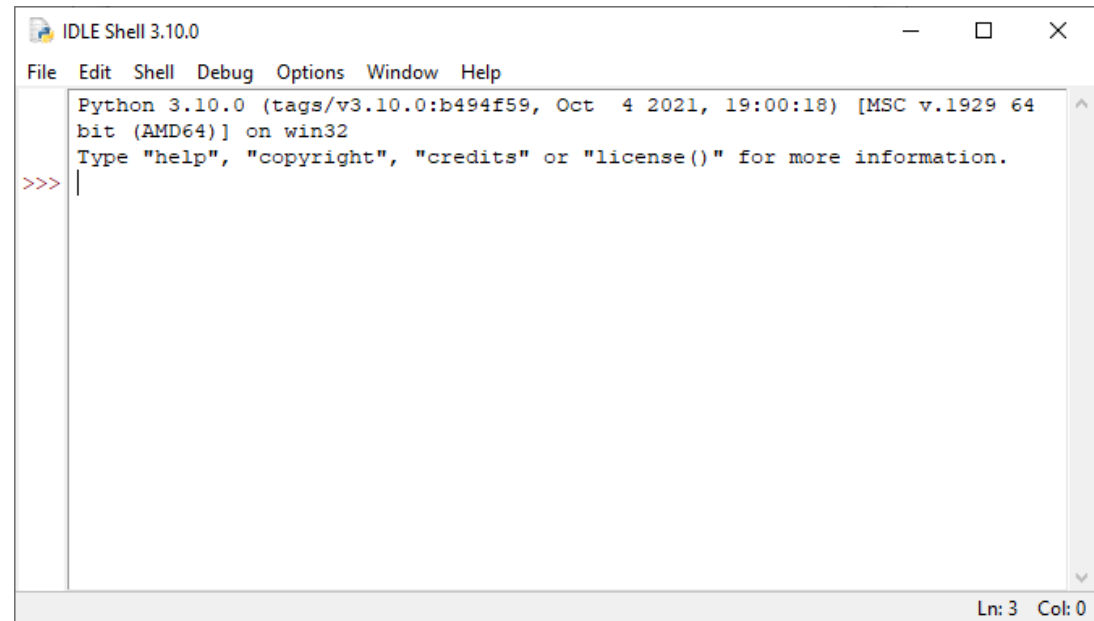
```
age = input("Your age: ")
if 14 < age < 60:
    print("Welcome! ")
    print("Entrance fee: 10 Euro.")
else:
    print ("Entrance fee: 5 Euro. ")
```

# IDLE

## Integrated Learning and Development Environment



Eric Idle in Monty Python's „Life of Brian“



# Python Shell – interaktiver Modus

Arithmetische Ausdrücke  
Logische Ausdrücke  
Funktionsaufrufe  
Import des Moduls math

## REPL

Read Evaluate Print Loop

# Python Shell – interaktiver Modus

```
>>> 2 + 3
5
>>> 2+3
5
>>> 0 > 2
False
>>> 1 != 2
True
>>> 2 == 1 + 1
True
>>> round(1.23)
1
>>> round(1.23, 1)
1.2
>>> sin(2)
Traceback ...
>>> from math import sin
>>> sin(90)
0.8939966636005579
>>> from math import *
>>> pi
3.141592653589793
>>> sin(pi/2)
1.0
>>> import math
>>> math.pi
3.141592653589793
```

# Wichtige Tastenkombinationen für IDLE

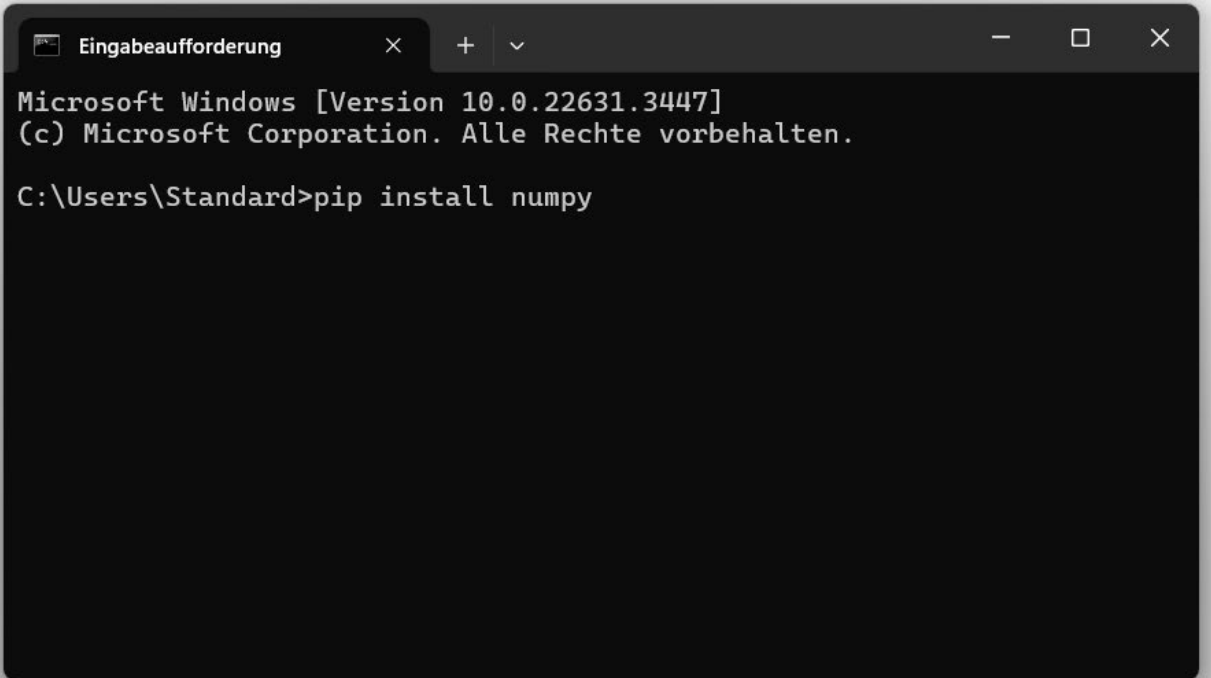
Tastenkombination	Bedeutung
<Alt> p	Vorige Eingabe (previous)
<Alt> n	Nächste Eingabe (next)
<Strg> c	Ausführung der Anweisung abbrechen

# Vorbereitung: Zusatzmodule

Sind alle Module verfügbar?

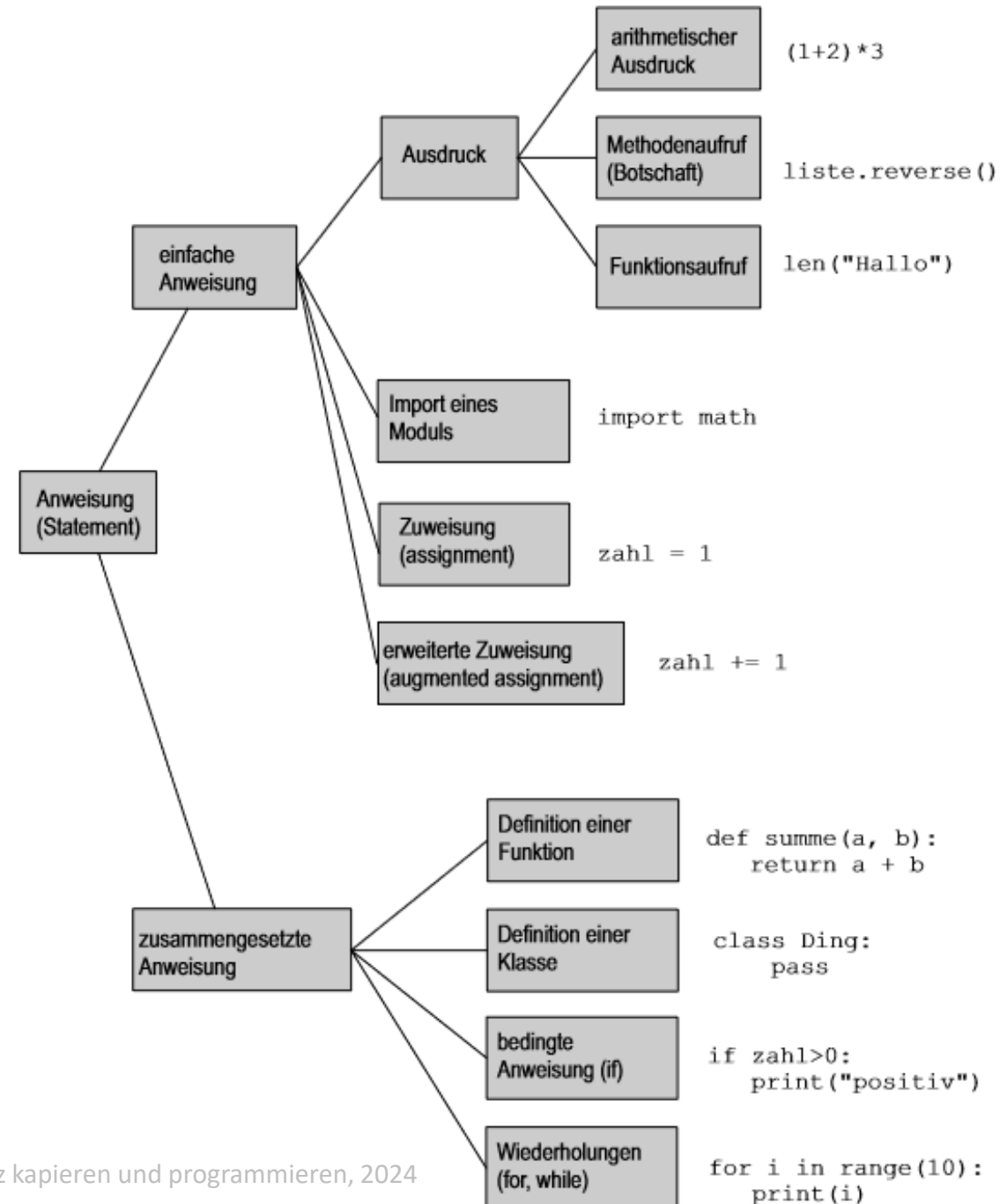
```
>>> import matplotlib  
>>> import numpy  
>>> import cv2
```

```
pip install matplotlib  
pip install numpy  
pip install opencv-python
```



```
Eingabeaufforderung  
Microsoft Windows [Version 10.0.22631.3447]  
(c) Microsoft Corporation. Alle Rechte vorbehalten.  
C:\Users\Standard>pip install numpy
```

# Anweisungen im Überblick





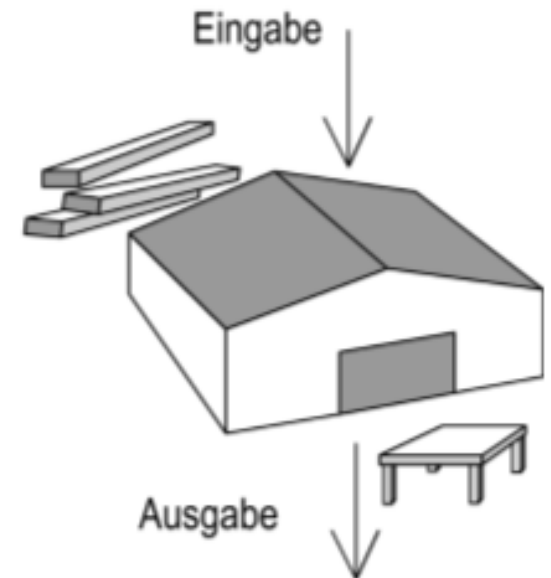
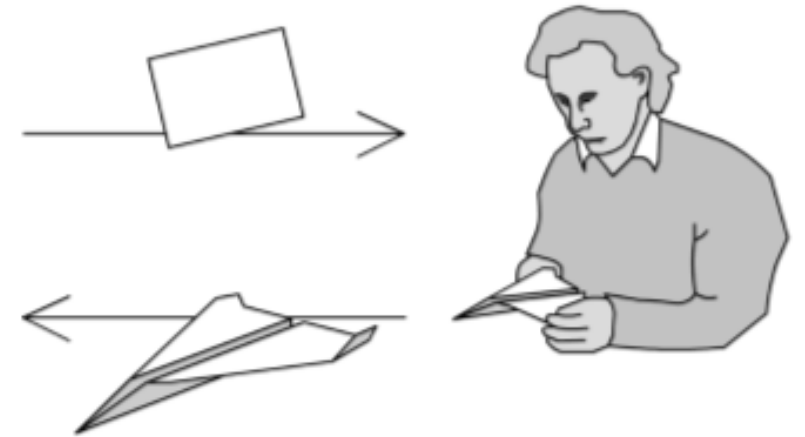
# Funktionsaufruf

Name der  
Funktion

Argument,  
Parameter

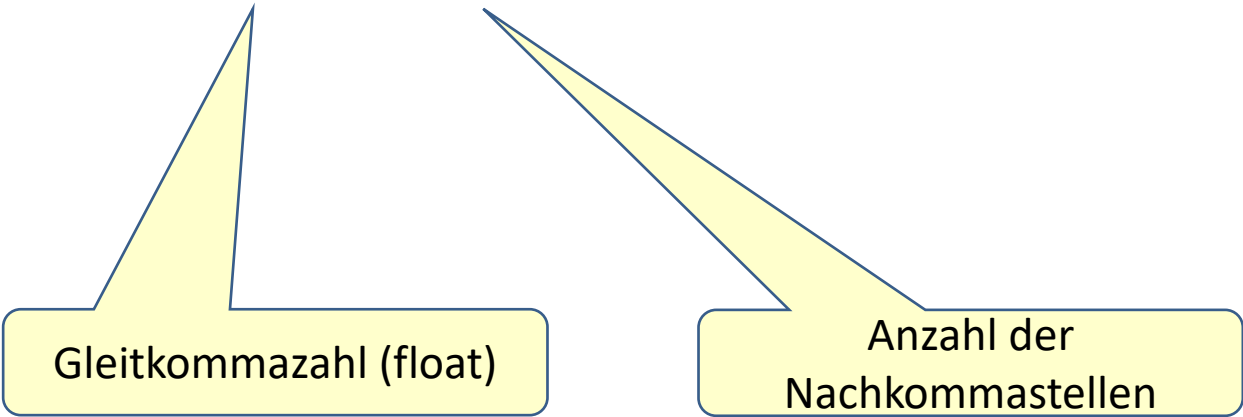
```
>>> len("Python")  
6
```

Zurückgegebener  
Wert



# Funktionsaufruf mit mehreren Argumenten

```
>>> round(1.234, 2)  
1.23
```



Gleitkommazahl (float)

Anzahl der  
Nachkommastellen

# Übung 1.2 Ausdruckanweisungen – die Python-Shell als Taschenrechner

Wie viel ist  
 $(4/7 + 3)**23$ ?

5192365780800.827



Foto cc by 2010 TIFFANY DAWN NICHOLSON in Wikimedia Commons

# Aufgabe 1

## Die Python-Shell als Taschenrechner

Probieren Sie in der Python Shell einige Anweisungen aus.

```
>>> 2 * 3
>>> (2 + 200) * 3
>>> 10/2
>>> 10/3
>>> 10//2
>>> 10//3
>>> 2**3
>>> 4 ** 0.5
>>> 23**45
>>> 10%3
>>> 11%3
>>> 12%3
>>> 2 + -2
>>> 2,3 * 5
>>> 2.3 * 5
>>> 2 +*3
>>> 2 *+ 3
>>> abs(-2)
>>> round(1.23)
>>> round(-1.5)
>>> round(1.234, 2)
>>> len("Python")
>>> len(123)
>>> help(len)
```

### Fragen

1. Welche unterschiedlichen Bedeutungen haben die Operatoren / und //?
2. `len()` ist ein Beispiel für eine Funktion. Was berechnet diese Funktion?
3. Welche Anweisungen liefern ein für Sie überraschendes Ergebnis? Haben Sie eine Erklärung?
4. Erklären Sie den Modulo-Operator % .

# Aufgabe 2

## Ausdruckenweisungen mit Funktionen aus dem Modul math

Mathematische Funktionen und Konstanten des Modul `math` (Auswahl)

Name	Erklärung
<code>cos(x)</code>	Cosinus von x
<code>e</code>	Die Eulersche Zahl $e \approx 2.7182...$
<code>log(x)</code>	Natürlicher Logarithmus von x (Basis e)
<code>log10(x)</code>	Logarithmus von x zur Basis 10
<code>pi</code>	Die Kreiszahl $\pi \approx 3.1415...$
<code>sin(x)</code>	Sinus von x
<code>sqrt(x)</code>	Quadratwurzel von x
<code>tan(x)</code>	Tangens von x

Importieren Sie alle Funktionen und Konstanten aus dem Modul `math`. Dazu geben Sie folgende Anweisung ein:

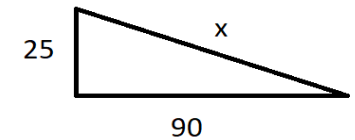
```
>>> from math import *
```

Geben Sie zu den folgenden Formeln Ausdrücke an und werten Sie sie aus:

1. Fläche eines Kreises mit Radius 5:  $A = \pi 5^2$

2. Längste Seite in einem rechtwinkligen Dreieck

$$x = \sqrt{25^2 + 90^2}$$



# Aufgabe 3

## Logische Ausdrücke (boolesche Ausdrücke)

Logische Ausdrücke (boolesche Ausdrücke) liefern bei ihrer Auswertung einen Wahrheitswert: `True` oder `False`.

Probieren Sie die folgenden Anweisungen aus und erklären Sie das Ergebnis:

```
>>> 1 > 2
>>> 1 != 1
>>> 1 == 1
>>> 1 = 1
>>> not False
>>> not True
>>> True or not True
>>> (1 > 2) or (1 < 2)
>>> (1 > 2) and (2 > 1)
>>> 1 < 2 <= 3
```

Operator	Erklärung	Beispiel	Wahrheitswert
<	kleiner	10 < 20	True
		10 < 10	False
<=	kleiner oder gleich	10 <= 20	True
		10 <= 10	True
>	größer	5.0 > 5.0	False
>=	größer oder gleich	"alle" >= "alt"	False
==	gleich	"gleich" == "gleich"	True
!=	ungleich	2 != 3	True
is	identisch	2 is 2	True
		[1] is [1]	False
is not	nicht identisch	1 is not 2	True

# Lösungen zu Übung 1.2

# Aufgabe 1

## Fragen

1. Welche unterschiedlichen Bedeutungen haben die Operatoren / und //?
2. `len()` ist ein Beispiel für eine Funktion. Was berechnet diese Funktion?
3. Welche Anweisungen liefern ein für Sie überraschendes Ergebnis? Haben Sie eine Erklärung?
4. Erklären Sie den Modulo-Operator %.

Exakte Division,  
liefert float-Zahl  
5.0

```
>>> 2 * 3
>>> (2 + 200) * 3
>>> 10/2
>>> 10/3
>>> 10//2
>>> 10//3
>>> 2**3
>>> 4 ** 0.5
>>> 23**45
>>> 10%3
>>> 11%3
>>> 12%3
>>> 2 + -2
>>> 2,3 * 5
>>> 2.3 * 5
>>> 2 +*3
>>> 2 *+ 3
>>> abs(-2)
>>> round(1.23)
>>> round(-1.5)
>>> round(1.234, 2)
>>> len("123")
>>> len(123)
>>> help(len)
```

Ganzzahlige  
Division  
3

Tupel  
(2, 15)

Runden auf 2  
Nachkommastellen  
1.23

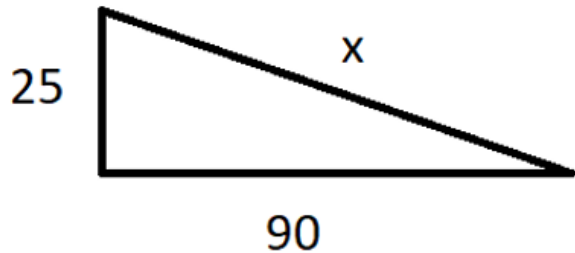
Anzahl der Items in  
einer Kollektion  
3

Zahl (keine  
Kollektion)



# Aufgabe 2

Fläche eines Kreises mit Radius 5:  $A = \pi 5^2$



$$x = \sqrt{25^2 + 90^2}$$

```
>>> from math import *  
>>> pi * 5 ** 2  
78.53981633974483
```

```
>>> from math import *  
>>> sqrt(25**2 + 90**2)  
93.40770846134703
```

# Aufgabe 3

```
>>> not False
```

```
True
```

```
>>> not 123
```

```
False
```

```
>>> (1 > 2) or (1 < 2)
```

```
True
```

```
>>> (1 > 2) and (2 > 1)
```

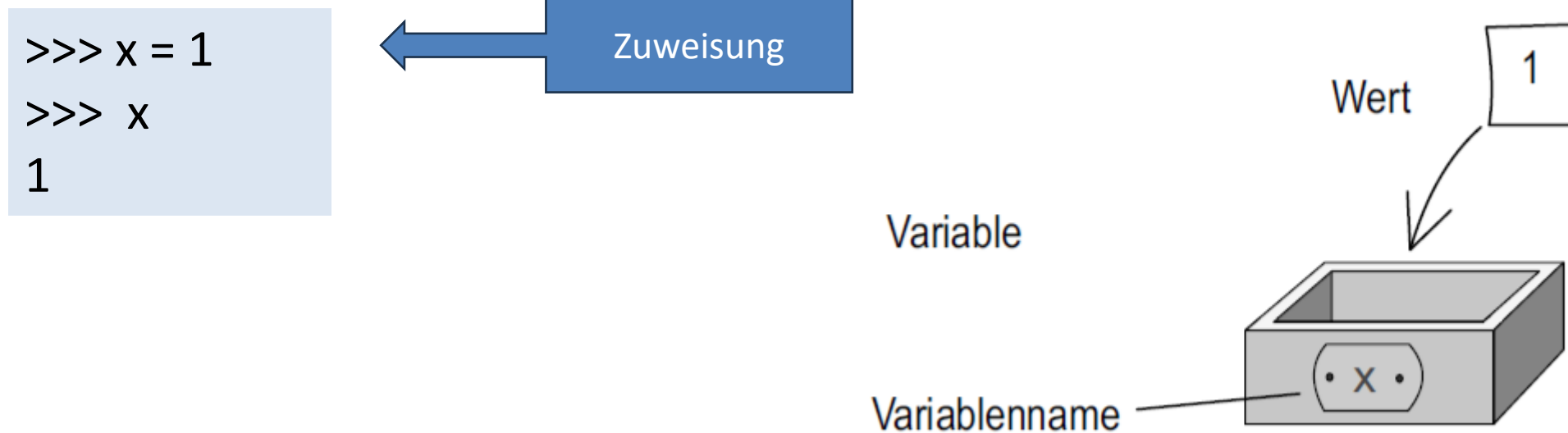
```
False
```

```
>>> 1 < 2 <= 3
```

```
True
```

a	b	not a	a or b	a and b
False	False	True	False	False
False	True	True	True	False
True	False	False	True	False
True	True	False	True	True

# 1.3 Variablen und Datentypen



# Die Wirkung von Zuweisungen - Wertetabelle

Anweisung	x	y
$x = 1$	1	
$x = x + 2$	3	
$y = x$	3	3

# Mehrere Zuweisungen auf einen Schlag

Anweisung	x	y
<code>x = y = 1</code>	1	1
<code>x, y = 20, 30</code>	20	30
<code>x, y = y, x</code>	30	20



# Erweiterte Zuweisungen

Anweisung	x
<code>x = 1</code>	1
<code>x += 1</code>	2
<code>x *= 5</code>	10

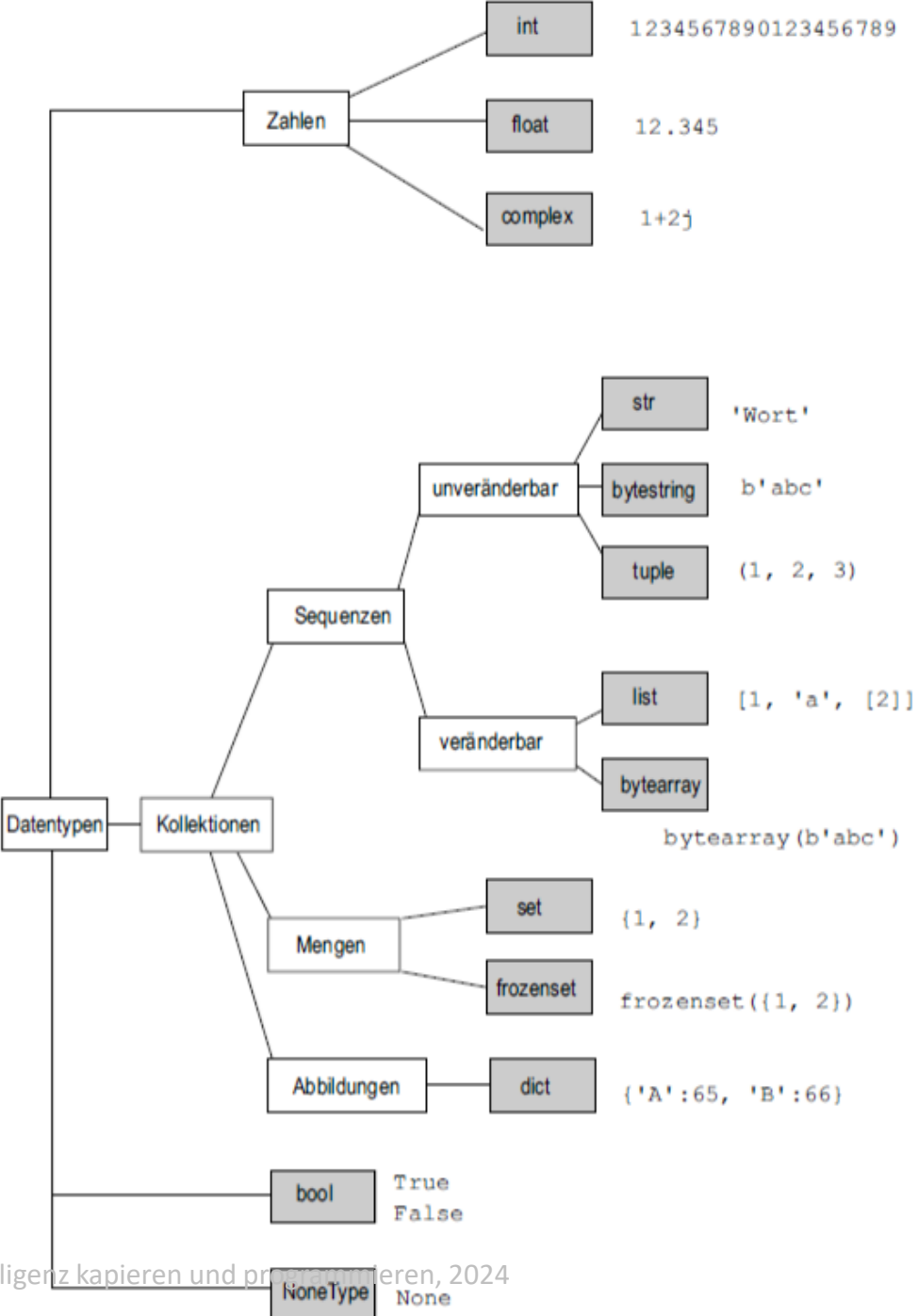
```
x = x + 1
```

```
x = x * 5
```

# Einfache Datentypen

Datentyp	Englische Bezeichnung	Python-Typbezeichnung	Beispiele
Ganze Zahl	integer	int	123, -234
Gleitkommazahl	floating point number	float	1.234 3.4E-12
Komplexe Zahl	complex number	complex	1 + 2j
Zeichenkette	string	str	"Hallo" 'Hallo'
Wahrheitswert	Boolean	bool	True False

# Typhierarchie





# Einem Wert kann man den Typ ansehen

42

1.3 E-2

42.0

"Ente"

True

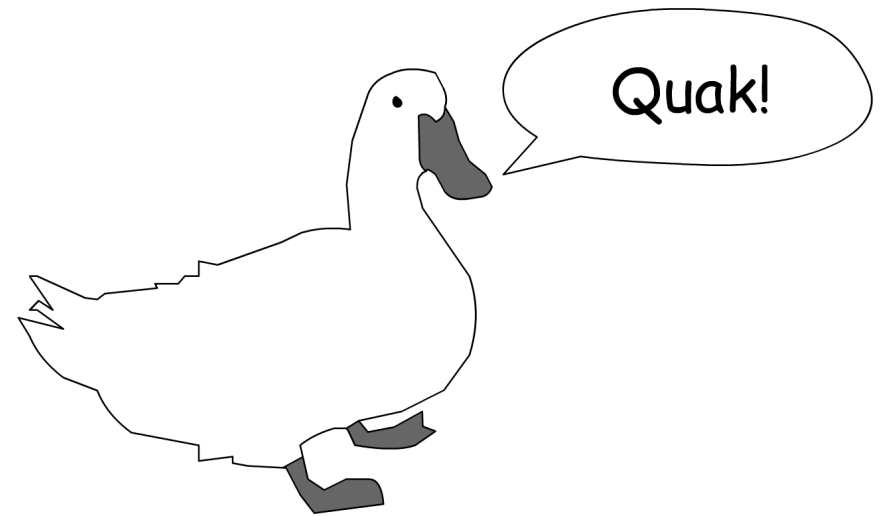
12.3 + 2.1j

## Typ feststellen

```
>>> type(42)  
<class 'int'>
```

»When I see a bird that walks like a duck and swims like a duck and quacks like a duck, I call that bird a duck.«

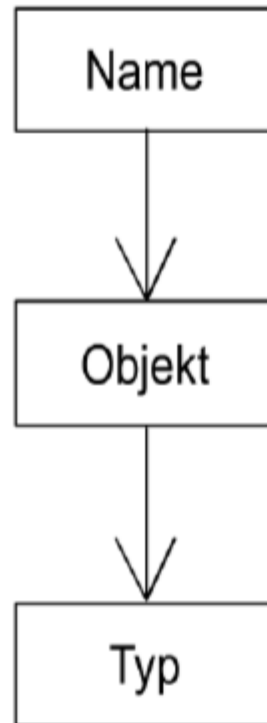
*James Whitcomb Riley (1849–1916)*



# Dynamisches Typisieren

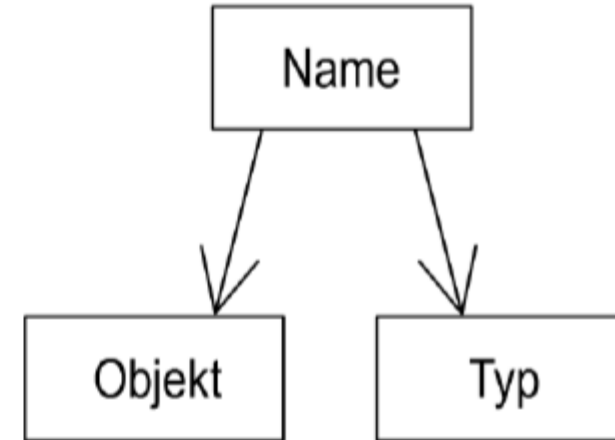
Python

```
a = 1
```



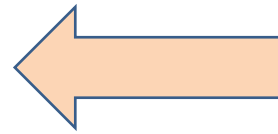
Java

```
int a = 1;
```



# „Typumwandlungen“ (Casting)

```
>>> str(123)
'123'
>>> int('123')
123
>>> float(123)
123.0
>>> complex(123)
(123+0j)
```



Es wird ein neues  
Objekt des Typs  
`int` erzeugt.

# Syntaxregeln für Namen (Identifizier)

Namen bestehen aus Buchstaben, Ziffern und Unterstrichen. Sie müssen mit einem Buchstaben oder Unterstrich beginnen.

Python Schlüsselwörter dürfen nicht verwendet werden.

## Erlaubt oder nicht erlaubt?

höhe  
x1  
volumen  
Volumen  
zahl\_1  
\_zahl  
~~1\_zahl~~  
Ω  
~~import~~

Erlaubt, aber Üblicherweise werden Variablennamen klein geschrieben

Name muss mit Buchstabe oder Unterstrich beginnen

# Schlüsselwörter

Dürfen nicht als Namen  
verwendet werden

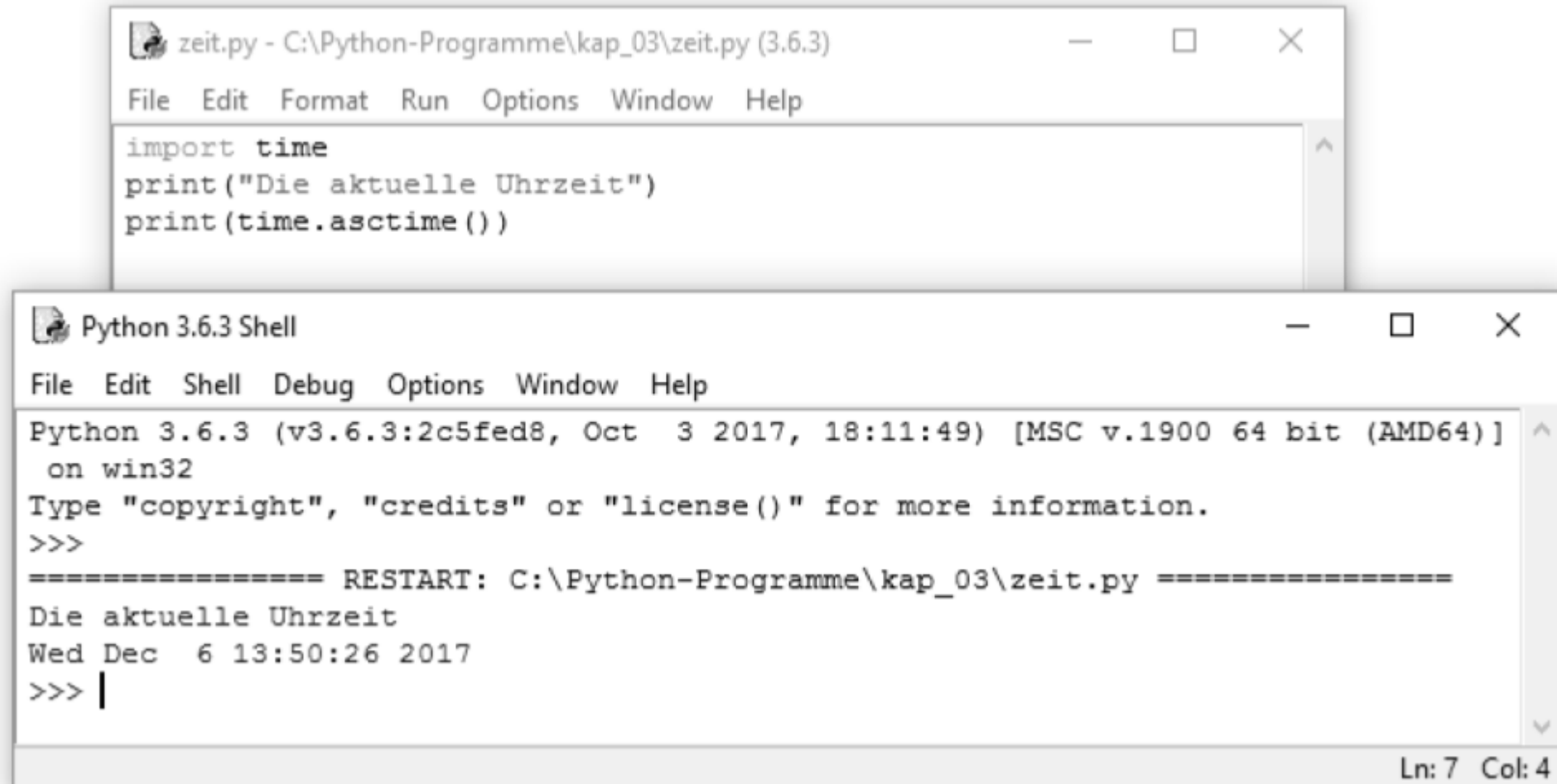
False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	break
except	in	raise		

# 1.4 Interaktive Programme



# Das erste Skript

Live



The image shows two overlapping windows from a Windows environment. The top window is a text editor titled 'zeit.py - C:\Python-Programme\kap\_03\zeit.py (3.6.3)'. It contains a Python script with three lines: `import time`, `print("Die aktuelle Uhrzeit")`, and `print(time.asctime())`. The bottom window is a 'Python 3.6.3 Shell'. It displays the output of running the script. The output includes the Python version and build information, followed by a restart message for the script 'C:\Python-Programme\kap\_03\zeit.py'. The script's output is displayed as 'Die aktuelle Uhrzeit' on one line and 'Wed Dec 6 13:50:26 2017' on the next line. The shell prompt '>>>>' is visible at the end of the output.

```
zeit.py - C:\Python-Programme\kap_03\zeit.py (3.6.3)
File Edit Format Run Options Window Help
import time
print("Die aktuelle Uhrzeit")
print(time.asctime())

Python 3.6.3 Shell
File Edit Shell Debug Options Window Help
Python 3.6.3 (v3.6.3:2c5fed8, Oct 3 2017, 18:11:49) [MSC v.1900 64 bit (AMD64)]
on win32
Type "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: C:\Python-Programme\kap_03\zeit.py =====
Die aktuelle Uhrzeit
Wed Dec 6 13:50:26 2017
>>> |
```

Ln: 7 Col: 4

# Mit dem Editor arbeiten

Vorbereitung: Projektordner erstellen

- Editorfenster öffnen (File | New File)
- Skript erstellen
- Skript im Projektordner speichern
- Skript ausführen (verschiedene Methoden)

Live



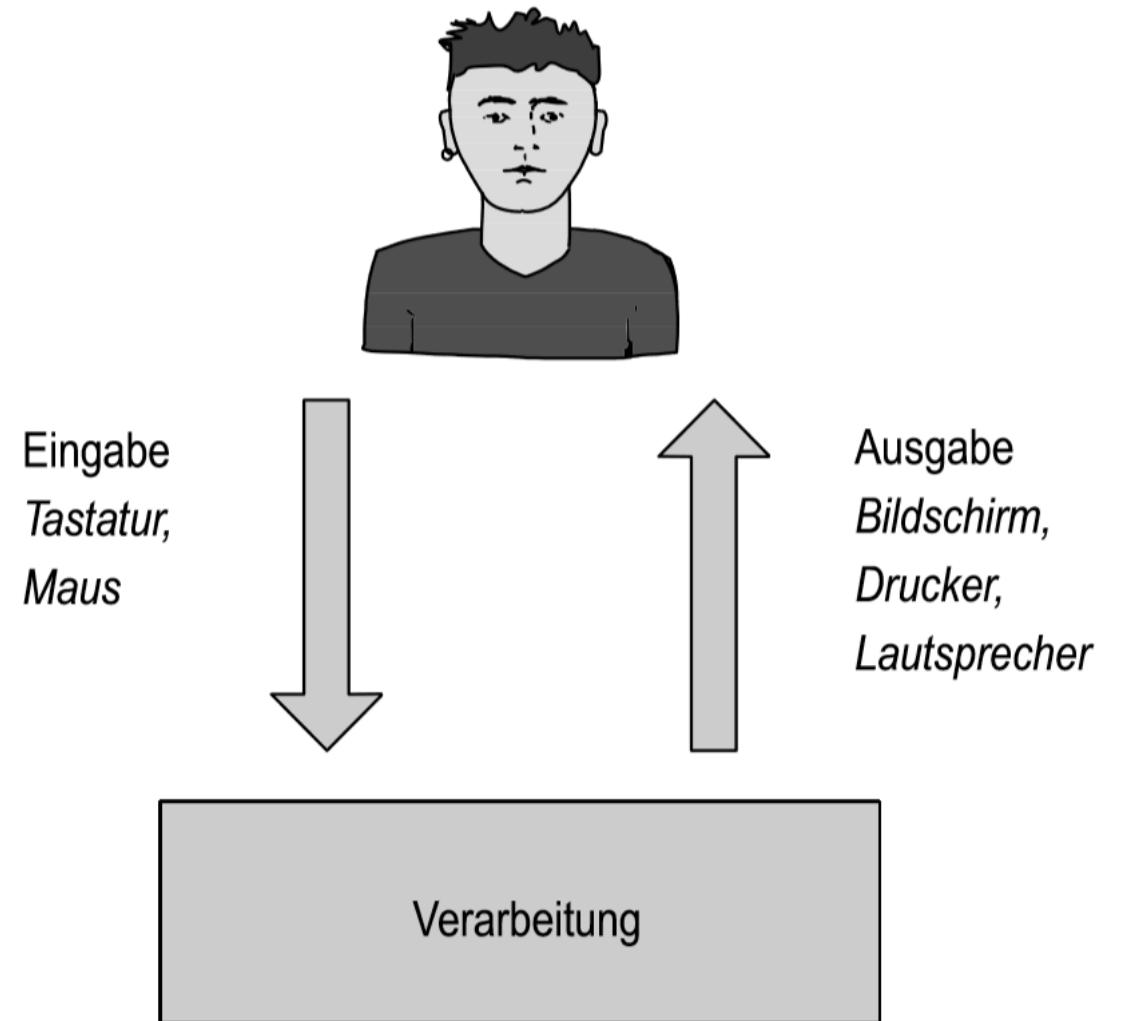
# Kommentare

```
# Ausgabe von Datum und Zeit  
import time  
print("Die aktuelle Uhrzeit")  
print(time.asctime())
```

Wird vom Interpreter  
ignoriert



# Das EVA- Prinzip



# Währungsrechner

```
WECHSELKURS = 0.92
# Eingabe
eingabe = input('Betrag in Dollar: ,')
# Verarbeitung
dollars = float(eingabe)
euros = WECHSELKURS * dollars
# Ausgabe
print('Wert in Euro: ', round(euros, 2))
input()
```

Prompt



Casting



# Kurzanleitung:

## Mit IDLE ein Programm schreiben und ausführen

### Vorbereitung

Erstellen Sie einen Projektordner für Ihre Python-Programme

### Programm im Editorfenster schreiben

- Öffnen Sie Idle (Python 3).
- Wählen Sie in der oberen Menüleiste den Befehl *File/New File*. Es öffnet sich das Editorfenster. Das Fenster hat den Namen *Untitled*.
- Schreiben Sie das Skript.

### Programm speichern

- Klicken Sie auf *File/Speichern unter*. Ihr Programm soll in Ihrem Projektordner gespeichert werden. Wechseln Sie in Ihren Projektordner.
- Geben Sie einen sinnvollen Dateinamen ein (z.B. *hallo.py*)
- Speichern Sie die neueste Version Ihres Programmtextes mit *File/Save* (oder mit der Tastenkombination <Strg>+ <S>)

### Programm ausführen

Klicken Sie auf *Run/Run Module* (oder drücken Sie die Taste <F5>)

# Übung 1.4 Rechenhilfe (5 min)

Wandeln Sie das nebenstehende Starter-Projekt ab und schreiben Sie ein benutzungsfreundliches, interaktives Programm, das das Volumen eines Gegenstandes aus dem Alltag oder den Inhalt eines Gefäßes berechnet und ausgibt (Volumen eines Glases, eines Schuhkartons etc.)

```
WECHSELKURS = 0.92
# Eingabe
eingabe = input('Betrag in Dollar: ,')
# Verarbeitung
dollars = float(eingabe)
euros = WECHSELKURS * dollars
# Ausgabe
print('Wert in Euro: ', round(euros, 2))
input()
```

# 1.5 Chatbots und Turing-Test

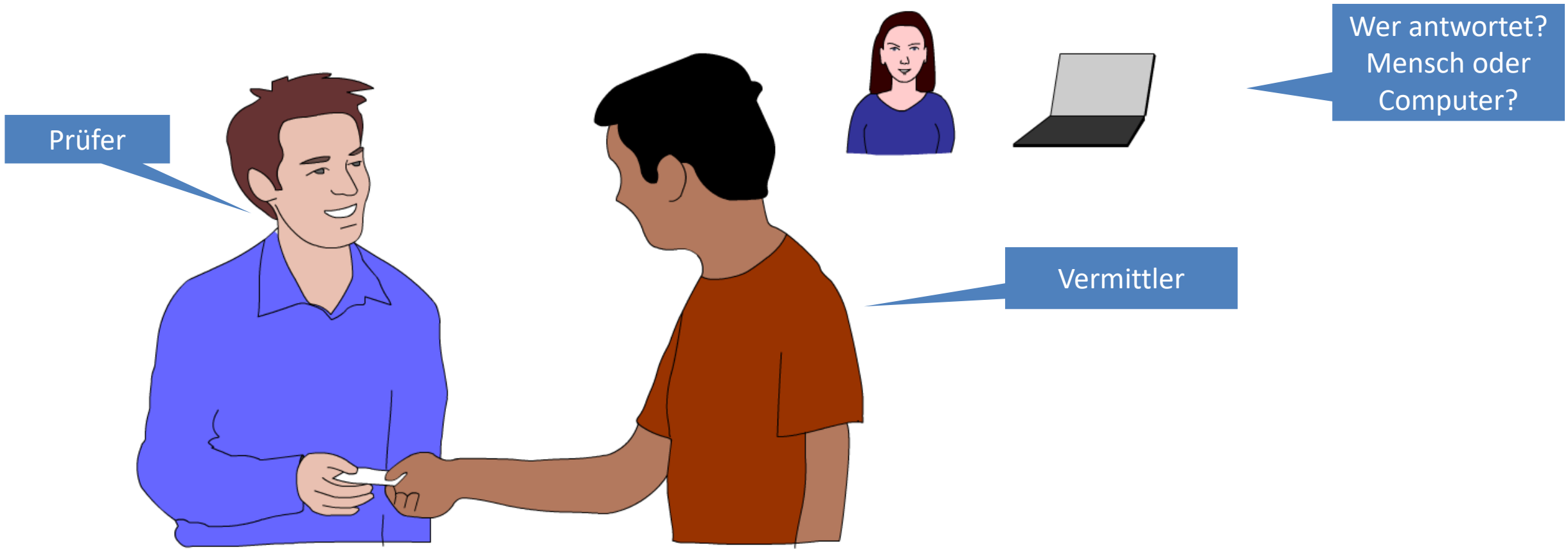
# Können Maschinen denken?

Alan Turing 1950: Can Machines Think?

Grundidee: Eine Maschine ist intelligent, wenn man ihr Verhalten nicht von dem Verhalten eines Menschen unterscheiden kann.



# Turing-Test (ursprünglich „Imitation Game“)





# Eliza

Joseph Weizenbaum 1966

Eliza im Internet:

Medical Artificial Intelligence

<http://www.med-ai.com/models/eliza.html.de>.



Bild: cc-by-sa 2005 Ulrich Hansen in Wikimedia Commons

# Zwischenfrage (1 min)

Wann ist Ihnen in letzter Zeit ein automatisch produzierter Text begegnet?

# Einfache Chatbots



Schön,  
dass Sie  
hier sind ...

Reinigungsroboter im Supermarkt



Bleiben Sie stehen,  
ich möchte Ihnen  
etwas erzählen ...

Sprechende Laternen in Celle

# Projekt: Freundlicher Putzautomat

```
# freundlich.py  
from random import choice
```

```
PHRASEN = ['Schön, dass Sie heute hier sind.',  
           'Ich hoffe, ich störe Sie nicht.',  
           'Ich reinige für Sie den Supermarkt',  
           'Gefällt Ihnen unser Angebot?']
```

```
while True:  
    input()  
    text = choice(PHRASEN)  
    print(text)
```

Liste

Endlosschleife

Zufällige Auswahl



# Erweiterung: Den Computer zum Sprechen bringen


Vorbereitung: espeak installieren:

- Downloadseite besuchen: <https://espeak.sourceforge.net/download.html>
- Installationsdatei (z.B. setup\_espeak-1.48.04.exe) herunterladen und ausführen.
- Pfad der Datei espeak.exe finden

```
import subprocess
from random import choice
PHRASEN = ['Schön, dass Sie heute hier sind.',
           'Ich hoffe, ich störe Sie nicht.',
           'Ich reinige für Sie den Supermarkt',
           'Gefällt Ihnen unser Angebot?']
```

```
BEFEHL = '"C:\Program Files (x86)\eSpeak\command_line\espeak.exe" -vde "{}"'
```

```
while True:
    input()
    text = choice(PHRASEN)
    befehl = BEFEHL.format(text)
    print(befehl)
    subprocess.run(befehl, shell=True)
```



Hier den Pfad auf  
dem eigenen System  
verwenden

# Projekt Mini-Eliza

## Schritt 1: Auf eine Eingabe reagieren

```
# eliza.py
print('Eliza: Hallo, ich bin Eliza. Was hast du auf dem Herzen?')
eingabe = input('Du: ')
if 'hass' in eingabe:
    print('Eliza: Hass kann Wertvolles zerstören.')
else:
    print('Eliza: Kannst du mir das Problem näher erklären?')
```

Bedingung  
(wahr oder falsch)

## Schritt 2: Mehr Intelligenz durch verschachtelte if-else-Anweisungen

```
# eliza
print('Eliza: Hallo, ich bin Eliza. Was hast du auf dem Herzen?')
eingabe = input('Du: ')
if 'hass' in eingabe:
    print('Eliza: Hass kann Wertvolles zerstören.')
else:
    if 'liebe' in eingabe:
        print('Eliza: Liebe ist etwas Wunderbares.')
    else:
        if 'schlaf' in eingabe:
            print('Eliza: Schlaf ist wichtig.')
        else:
            print('Eliza: Kannst du mir das Problem näher erklären?')
```


### Schritt 3: Mit elif die technische Qualität verbessern

```
# eliza
print('Eliza: Hallo, ich bin Eliza. Was hast du auf dem Herzen?')
eingabe = input('Du: ')
if 'hass' in eingabe:
    print('Eliza: Hass kann Wertvolles zerstören.')
elif 'liebe' in eingabe:
    print('Eliza: Liebe ist etwas Wunderbares.')
elif 'schlaf' in eingabe:
    print('Eliza: Schlaf ist wichtig.')
else:
    print('Eliza: Kannst du mir das Problem näher erklären?')
```



## Schritt 4: Von der einfachen Reaktion zum Gespräch

```
# eliza
print('Eliza: Hallo, ich bin Eliza. Was hast du auf dem Herzen?')
eingabe = 'x'
while eingabe != '':
    eingabe = input('Du: ')
    if 'hass' in eingabe:
        print('Eliza: Hass kann Wertvolles zerstören.')
    elif 'liebe' in eingabe:
        print('Eliza: Liebe ist etwas Wunderbares.')
    elif 'schlaf' in eingabe:
        print('Eliza: Schlaf ist wichtig.')
    elif eingabe != '':
        print('Eliza: Kannst du mir das Problem näher erklären?')
print('Es war wunderbar, mit dir zu reden. Bis bald!')
```



Bedingung  
(wahr oder falsch)

# Übung 1.5

Hier sind zwei Projektideen:

## *1. Intelligente E-Mail-Assistentin*

*Die intelligente E-Mail-Assistentin überprüft den Text einer E-Mail und gibt eine Warnung aus, wenn man einen Anhang vergessen hat oder etwas anderes nicht in Ordnung zu sein scheint.*

## *2. Intelligenter Kundenbetreuer*

*Der Kundenbetreuer führt einen Dialog mit einem Kunden, der sich beschweren will. Er reagiert professionell und höflich, ohne wirklich auf die Beschwerden einzugehen. Das Ziel ist, dass der Kunde von sich aus das Gespräch beendet. Er wird abgewiegelt.*

Implementieren Sie eines dieser Projekte oder denken Sie sich ein ganz anderes Thema für einen Chatbot aus.

[https://docs.google.com/document/d/140INslEWA\\_AwUwtpVMCZqtH7\\_eOkU8rmvfrgWqE5M3E/edit?usp=sharing](https://docs.google.com/document/d/140INslEWA_AwUwtpVMCZqtH7_eOkU8rmvfrgWqE5M3E/edit?usp=sharing)

# Rückblick

- Zwei pädagogische Ansätze spielen in diesem Workshop eine besondere Rolle: Konstruktivismus (Papert, Resnick) und fundamentale Ideen der Informatik (Bruner, Schwill)
- Für die Programmierung digitaler Artefakte verwenden wir Python. Vorteile: kurze verständliche Programmtexte, leicht zu lernen und mächtige frei verfügbare Module.
- Im interaktiven Modus kann man einzelne Python-Befehle ausprobieren.
- Ein interaktives Programm übernimmt Daten z.B. über die Tastatur (`input()`), verarbeitet sie und liefert eine Ausgabe (`print()`).
- Nach Alan Turing kann man eine Maschine intelligent nennen, wenn ihr Verhalten nicht vom Verhalten eines Menschen zu unterscheiden ist („Turingtest“).
- Ein Chatbot ist ein Programm, mit dem man ein Gespräch führen kann.