



# python

Erste Schritte

Niels Lohmann  
Stefanie Behrens

Lutz Hellmig  
Karsten Wolf

Universität  
Rostock



Traditio et Innovatio



# python

Erste Schritte

## Organisatorisches

1

Niels Lohmann  
Stefanie Behrens

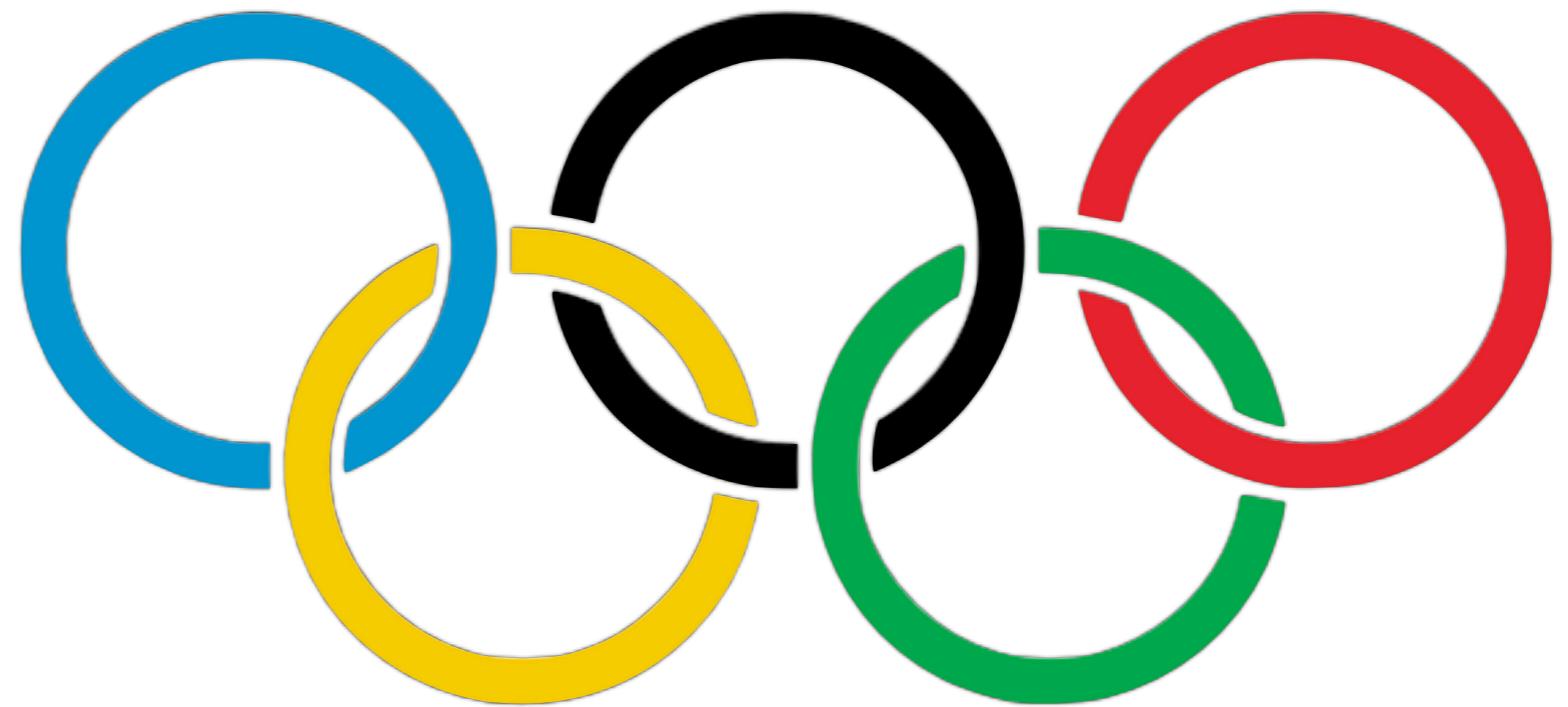
Lutz Hellmig  
Karsten Wolf

Universität  
Rostock



Traditio et Innovatio

# ORGANISATORISCHES



## TAGESABLAUF

Heute: Erste Schritte mit Python

10:00-12:30 **Das kleine Python-ABC**

13:00-15:15 **Erste Programme**

15:30-18:00 **Teilen und Herrschen**

## WER BIN ICH?

Niels Lohmann

Informatiker an der Universität Rostock

Programmiererfahrungen seit der Schulzeit

BASIC, Pascal, Delphi, Java, Prolog, C,  
VHDL, C++, MATLAB, PHP, JavaScript,  
Haskell, Python, BASH



derzeit für >30 Programme mit  
>400.000 Codezeilen zuständig

# ORGANISATORISCHES

## WER SEID IHR?

Habt ihr **Programmiererfahrung**?

Welche **Programmiersprachen** kennt ihr?

Habt ihr schon **eigene Projekte** umgesetzt?

Möchtest Du später im Bereich  
**Informatik** arbeiten/studieren?





# python

Erste Schritte

## Programmiersprachen

2

Niels Lohmann  
Stefanie Behrens

Lutz Hellmig  
Karsten Wolf

Universität  
Rostock



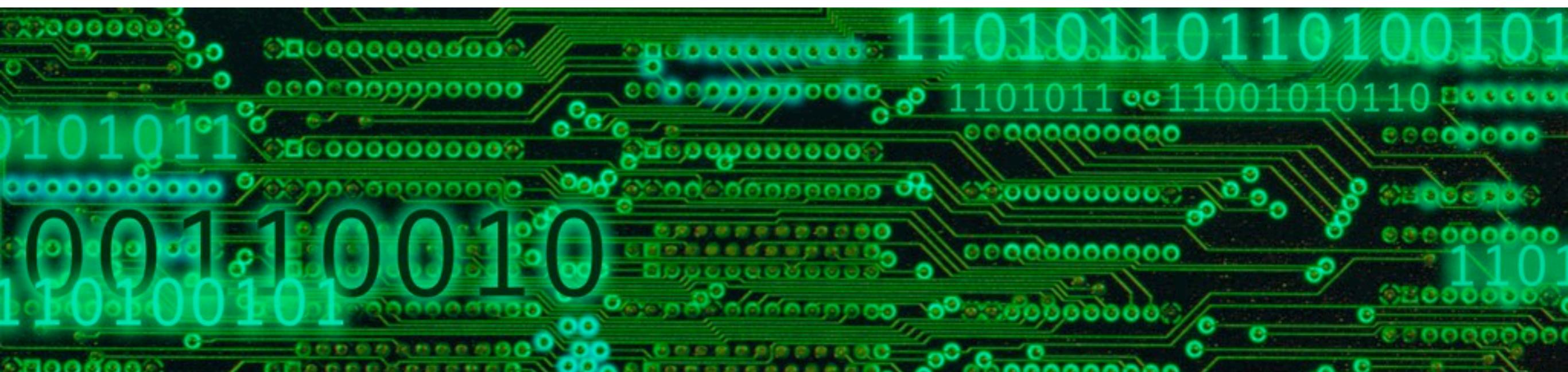
Traditio et Innovatio

# ALGORITHMUS VS. PROGRAMMIEREN

Algorithmus = Lösung eines **Problemes**

Programmieren = ein Problem so **aufbereiten**,  
dass Computer es lösen kann

Programmieren ≠ einfach nur Code aufschreiben!



# ALGORITHMUS VS. PROGRAMMIEREN

8400

Programmiersprache: nur  
Mittel zum Zweck!

**kein Selbstzweck!**

ein guter Informatiker lernt  
schnell neue Sprachen

mein erster Kontakt mit  
**Python**: vor ein paar  
Monaten...



# UNSERE OLYMPISCHE SPRACHE



relativ neue Programmiersprache (1991)

sehr einfach und doch sehr mächtig

optimiert auf Lesbarkeit - sehr gut für den Einstieg

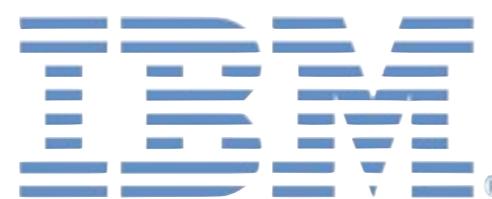
Anbindungen an viele andere Sprachen vorhanden

# UNSERE OLYMPISCHE SPRACHE

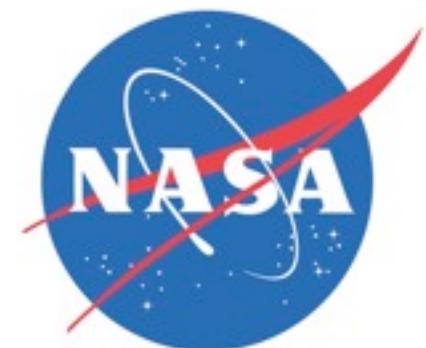


# python

sehr weit verbreitet



INDUSTRIAL  
LIGHT & MAGIC  
A LUCASFILM COMPANY

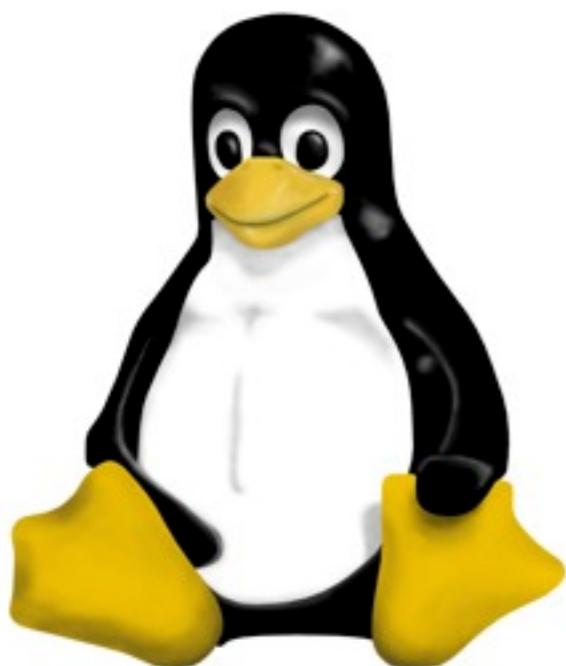


# UNSERE OLYMPISCHE SPRACHE



python

plattformunabhängig



Mac



# UNSERE OLYMPISCHE SPRACHE

**...UND WARUM NICHT**



**Java ?**

gleiche Voraussetzungen für alle

schwieriger zu lernen

mehr "Drumherum" notwendig (Compiler, IDE, ...)

schlechte Erfahrungen im letzten Jahr



# python

Erste Schritte

## Der Python-Interpreter

3

Niels Lohmann  
Stefanie Behrens

Lutz Hellmig  
Karsten Wolf

Universität  
Rostock



Traditio et Innovatio

# INTERPRETER

der Chat  
mit Python



**IDLE**

Python Shell

```
Python 3.1.2 (r312:79360M, Mar 24 2010, 01:33:18)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> |
```

Ln: 4 Col: 4

# DATENTYPEN

**int** ganze Zahlen

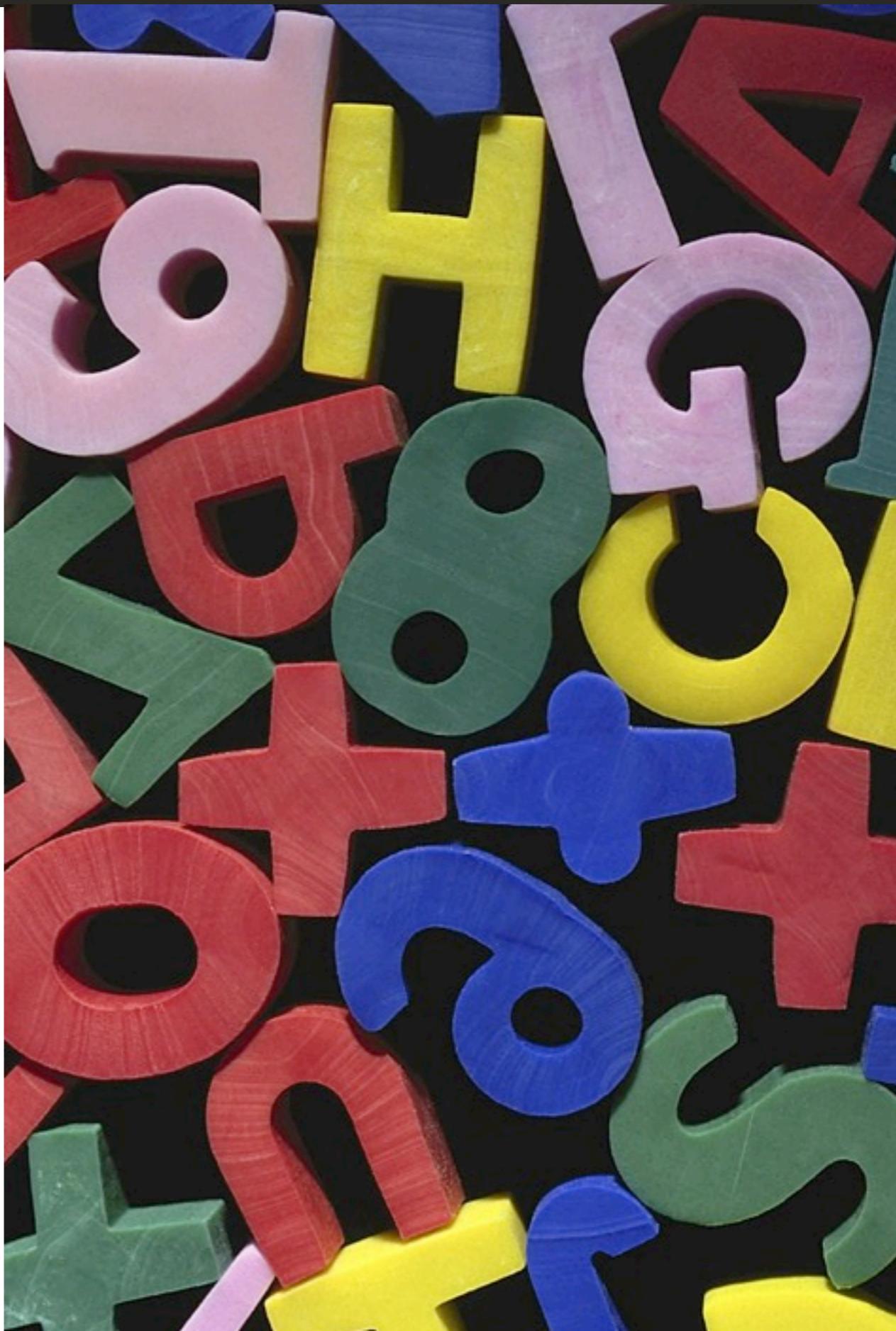
**float** Gleitkommazahlen

**str** Zeichenketten

**tupel** Tupel/Vektoren

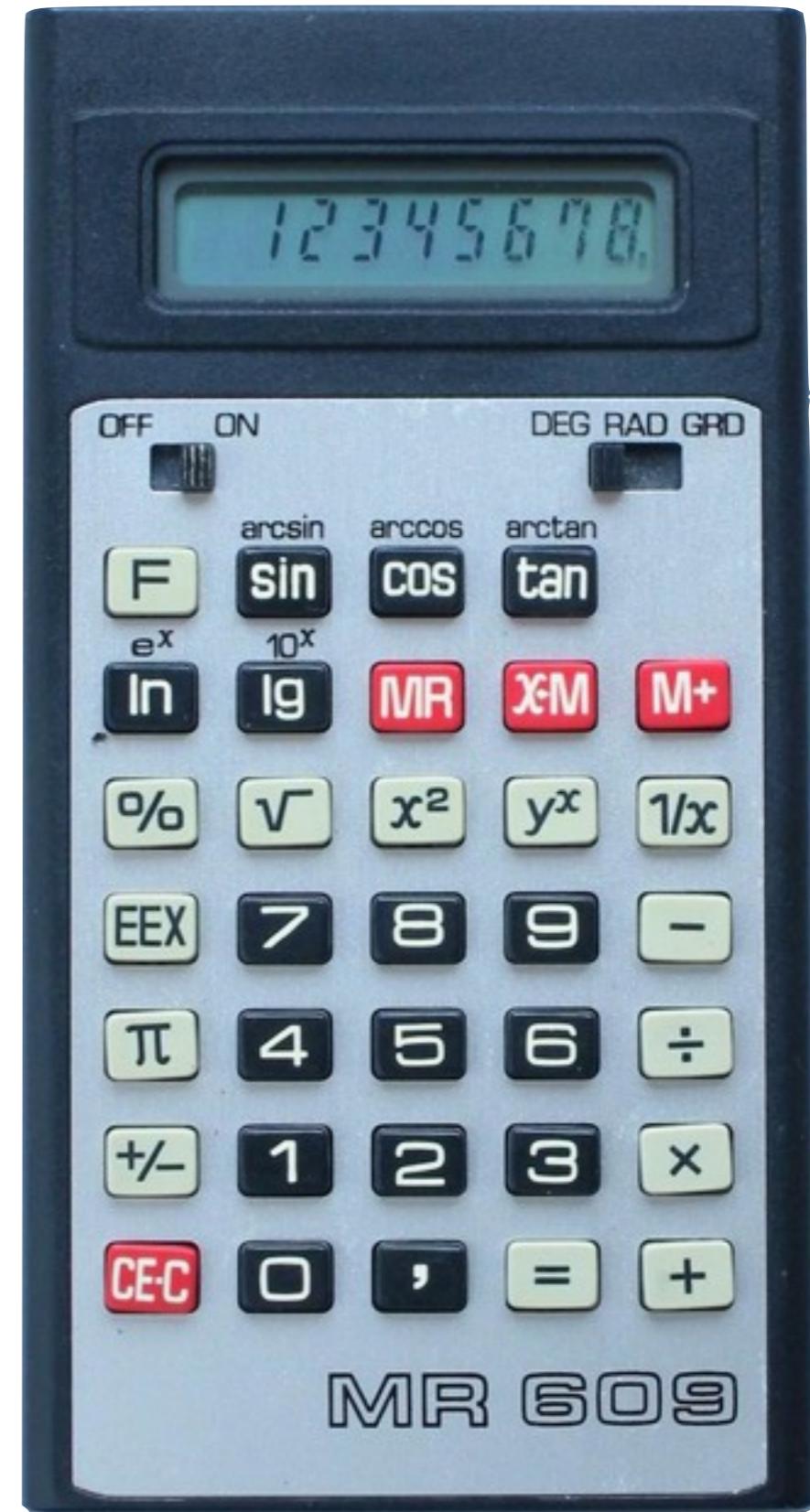
**set** Mengen

**list** Listen

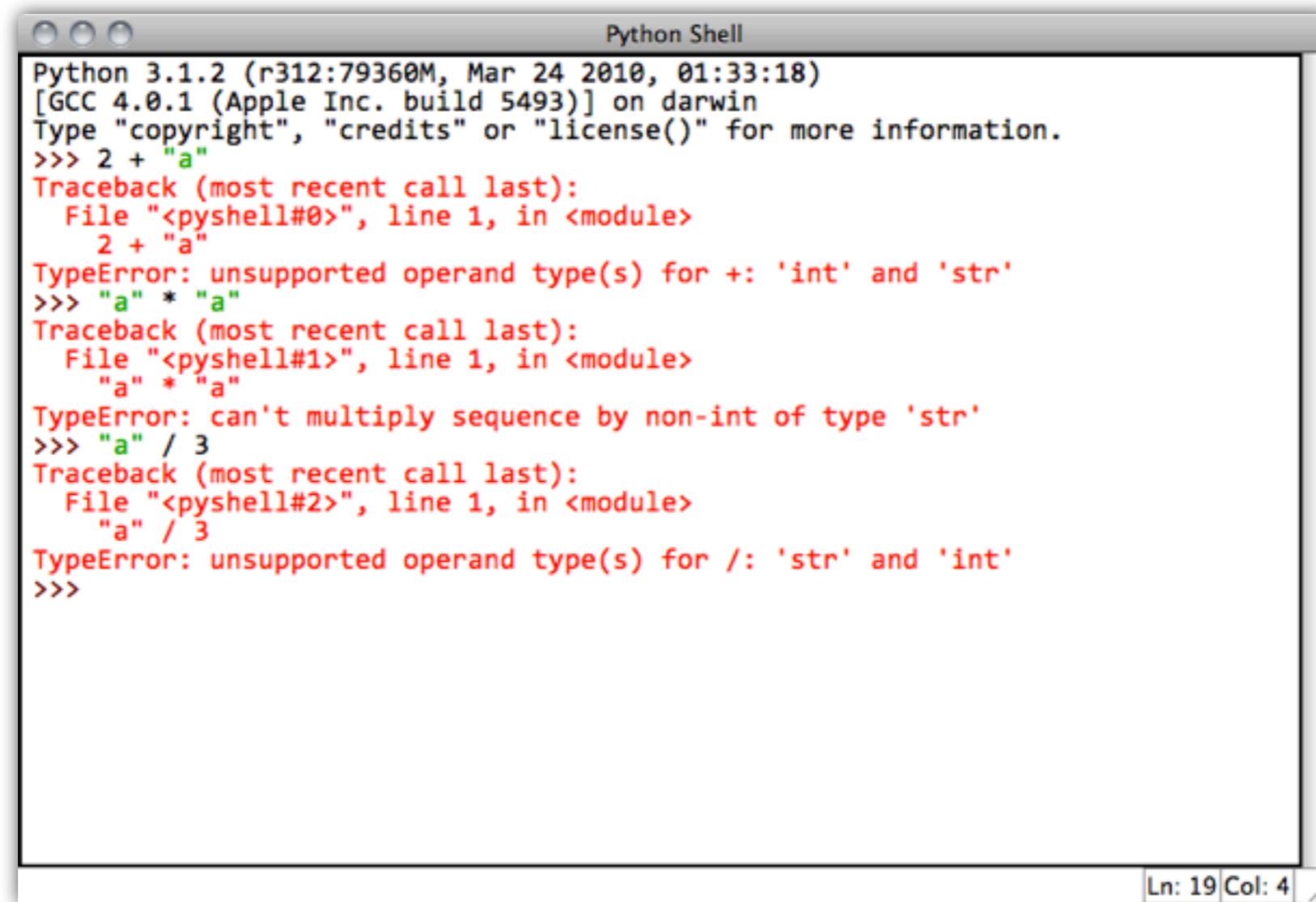


# OPERATIONEN

- + Addition
- Subtraktion
- \*
- / Division
- // ganzzahlige Division
- % Modulo (Division mit Rest)
- \*\* Potenzieren
- ( ) Klammern
  
- +
- \* Verkettung von Strings  
Wiederholung von Strings



# FEHLERMELDUNGEN



The screenshot shows a Python Shell window with the following content:

```
Python Shell
Python 3.1.2 (r312:79360M, Mar 24 2010, 01:33:18)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>> 2 + "a"
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    2 + "a"
TypeError: unsupported operand type(s) for +: 'int' and 'str'
>>> "a" * "a"
Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    "a" * "a"
TypeError: can't multiply sequence by non-int of type 'str'
>>> "a" / 3
Traceback (most recent call last):
  File "<pyshell#2>", line 1, in <module>
    "a" / 3
TypeError: unsupported operand type(s) for /: 'str' and 'int'
>>>
```

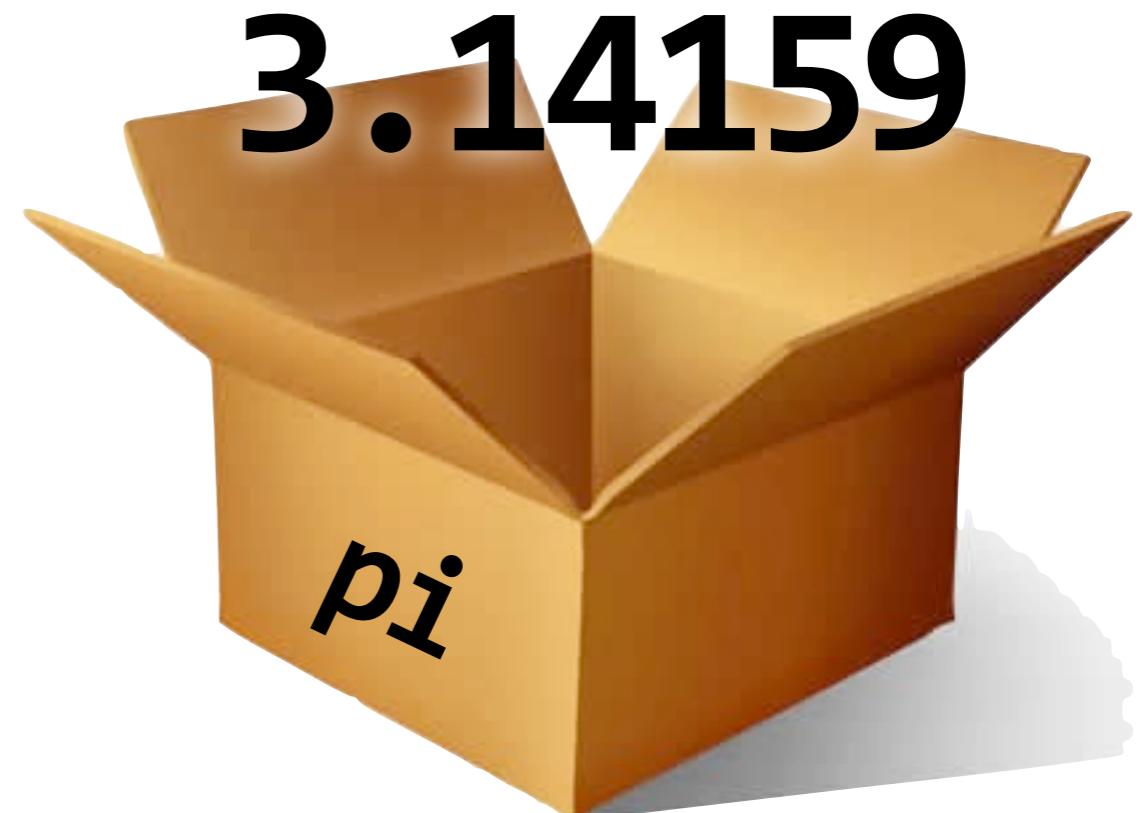
The window has a title bar "Python Shell" and a status bar at the bottom right showing "Ln: 19 Col: 4".

- genau durchlesen!
- Interpreter gibt (oft) gute Hinweise!
- auch **Google** hilft mitunter sehr

# SPEICHERN VON WERTEN

Werte können in Variablen gespeichert werden

Variablen haben einen Namen (= Platzhalter)



Variablennamen: a-Z A-Z \_ 0-9

Achtung: Schlüsselworte; Groß-/Kleinschreibung

# SPEICHERN VON WERTEN

Variable existiert, solange der Interpreter läuft

erzeugen von Variablen durch **Zuweisung**

**Name = "Niels"**

**"Niels"**

Variable steht immer links

Rechte Seite wird vor dem  
Speichern vollständig  
ausgerechnet:

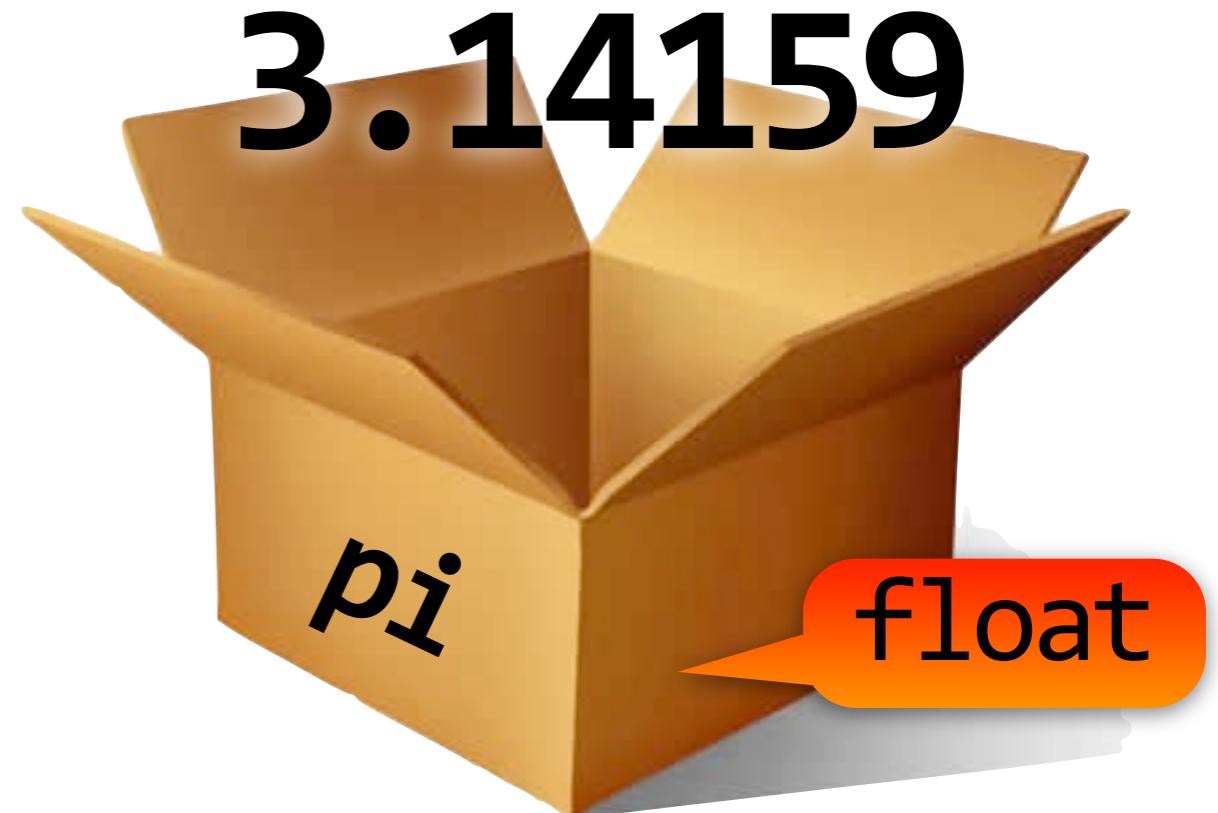
**Preis = (10.40 + 2.99) \* Mehrwertsteuer**



# TYPEN

Variablen haben **Typen**

Typ ergibt sich aus gespeichertem Wert



Typen können mit **type**-Befehl angezeigt werden

# VERGLEICHE

Werte und Variablen können verglichen werden

10.333

var1

==

!=

>

<

>=

<=

12

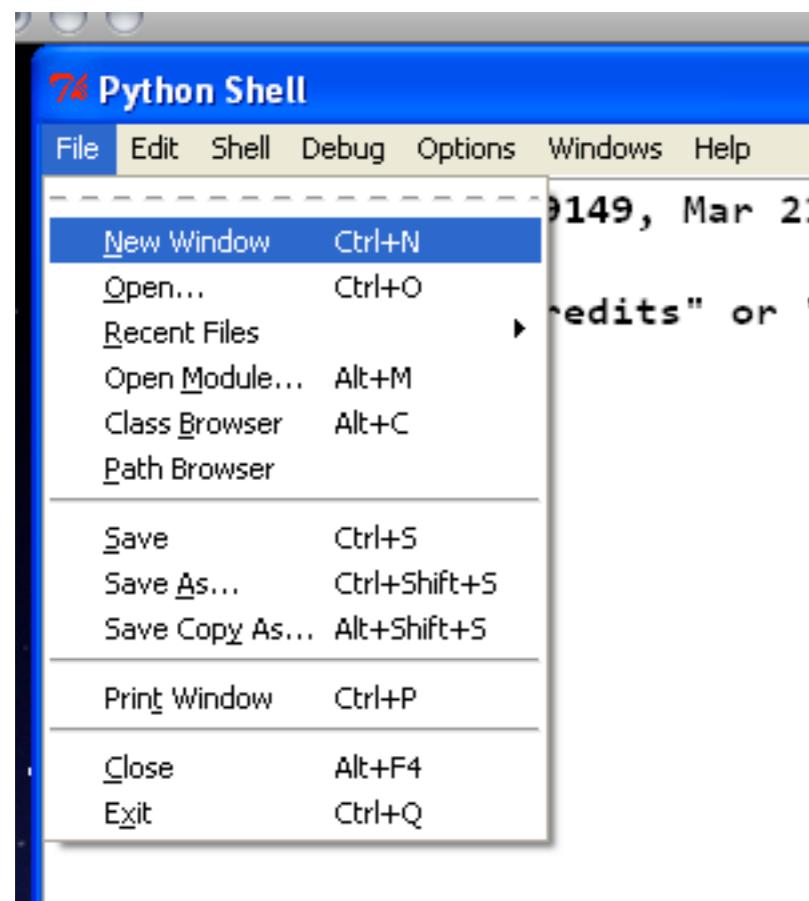
var2

Ergebnis ist wahr (**True**) oder falsch (**False**)

# PROGRAMMDATEIEN

Speichern einer Interpretersitzung

in IDLE über **New Window**



Dateiendung muss **.py** sein!

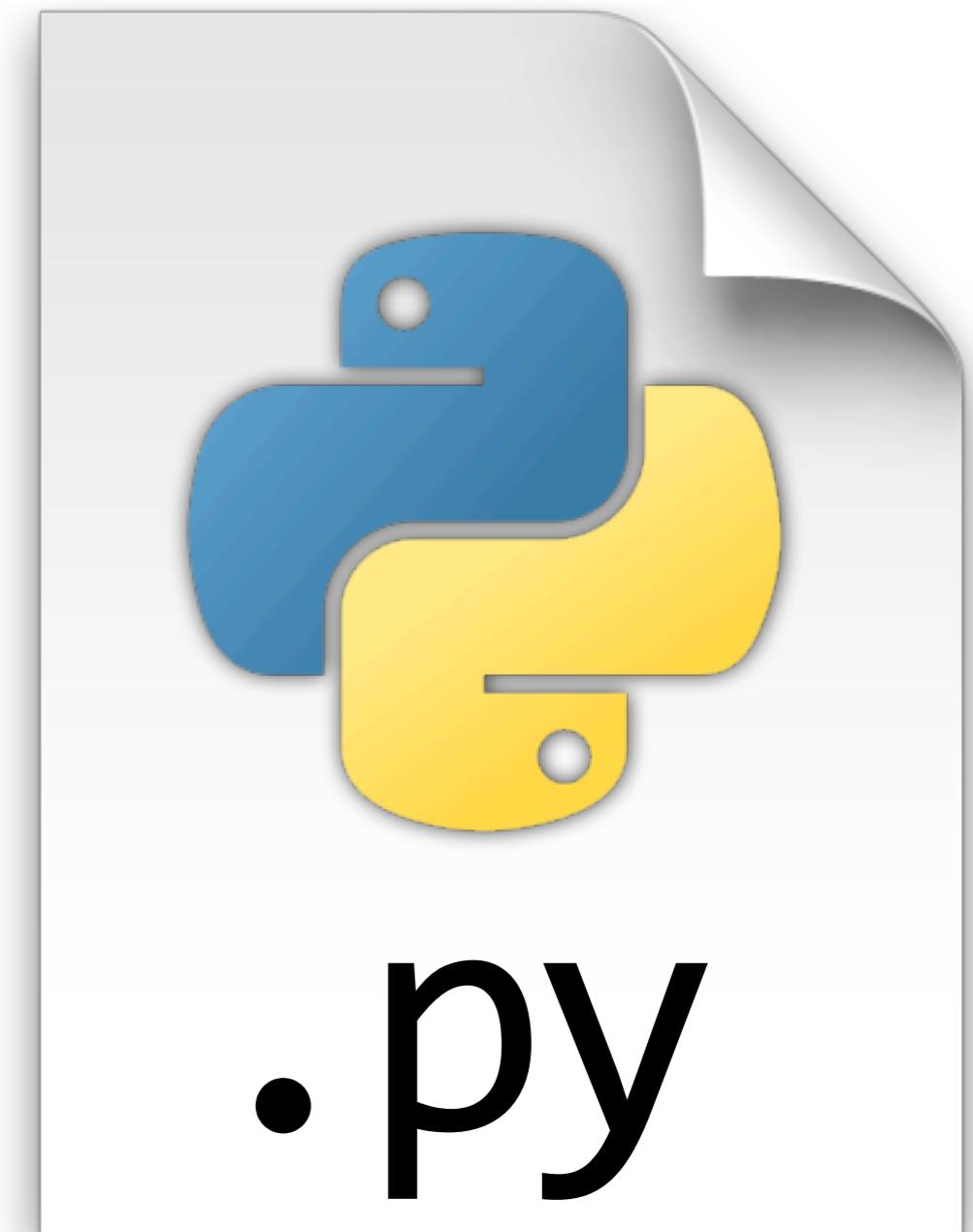
# PROGRAMMDATEIEN

Features im Editor:

- Syntax-Highlighting
- Hilfe bei Klammern
- Code-Vervollständigung
- Automatische Hilfe
- genauere Fehlermeldungen
- ...

Ausführen mit **F5**

Ausgabe von Werten muss  
nun explizit mit **print**-Befehl passieren



# GUTER PROGRAMMIERSTIL

guter Stil hilft beim Wiederverwenden  
und Verstehen von Programmcode

spart Zeit, Geld und Nerven!

## EIN PAAR REGELN

- vernünftige und aussagekräftige Variablennamen
- sinnvolle Kommentare
  - nicht zu viel, nicht zu wenig
  - nicht das Offensichtliche kommentieren
- Code gliedern



# python

Erste Schritte

## Das erste Python-Programm

4

Niels Lohmann  
Stefanie Behrens

Lutz Hellmig  
Karsten Wolf

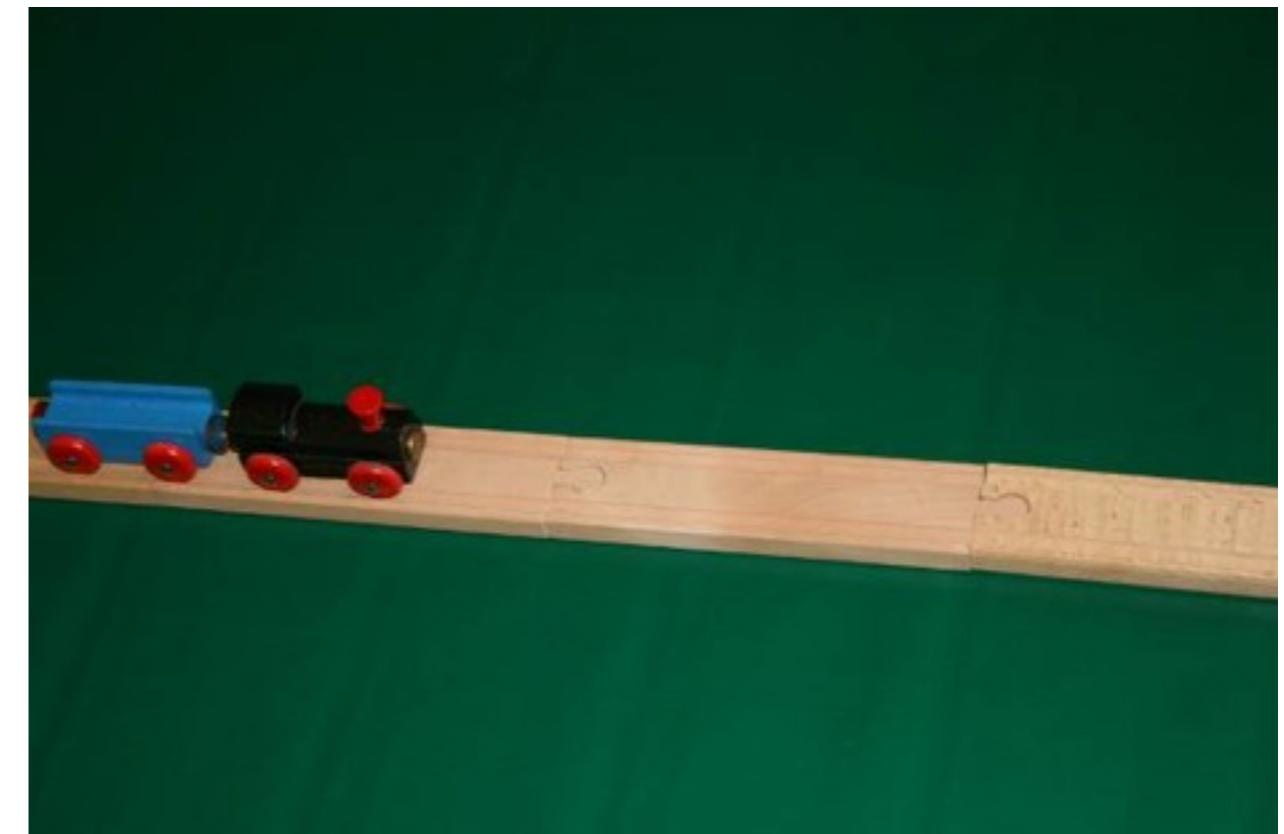
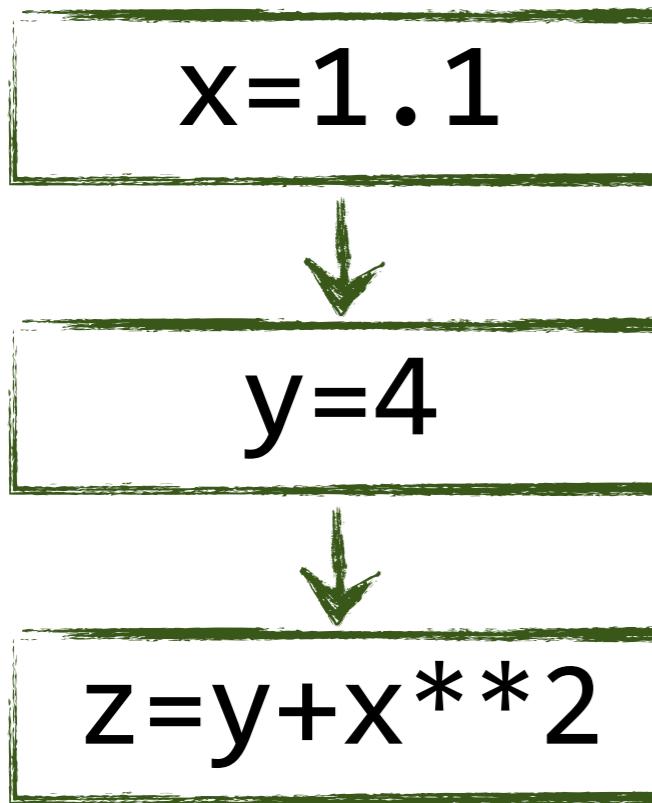
Universität  
Rostock



Traditio et Innovatio

# SEQUENZ

**Sequenz** = bedingungslose Hintereinanderausführung von Code



Mit Sequenzen können beliebige mathematische Formeln berechnet werden:

$$z = y + x^2$$

# 1. PROGRAMM: GRILLENZIRPEN

Eine Grille zirpt ziemlich regelmäßig, und die Frequenz des Zirpens hängt von der Temperatur ab. Die Zahl der Zirplaute in acht Sekunden plus acht ist gleich der Temperatur (in °C). Schreiben ein Programm, dem man die Anzahl des Zirpens in einer Minute eingibt und das die Temperatur berechnet und ausgibt.



# 1. PROGRAMM: GRILLENZIRPEN

Die Zahl der Zirplaute in acht Sekunden plus acht ist gleich der Temperatur in Celsius.

Eingabe

lies Anzahl Zirplaute

Verarbeitung

berechne Temperatur

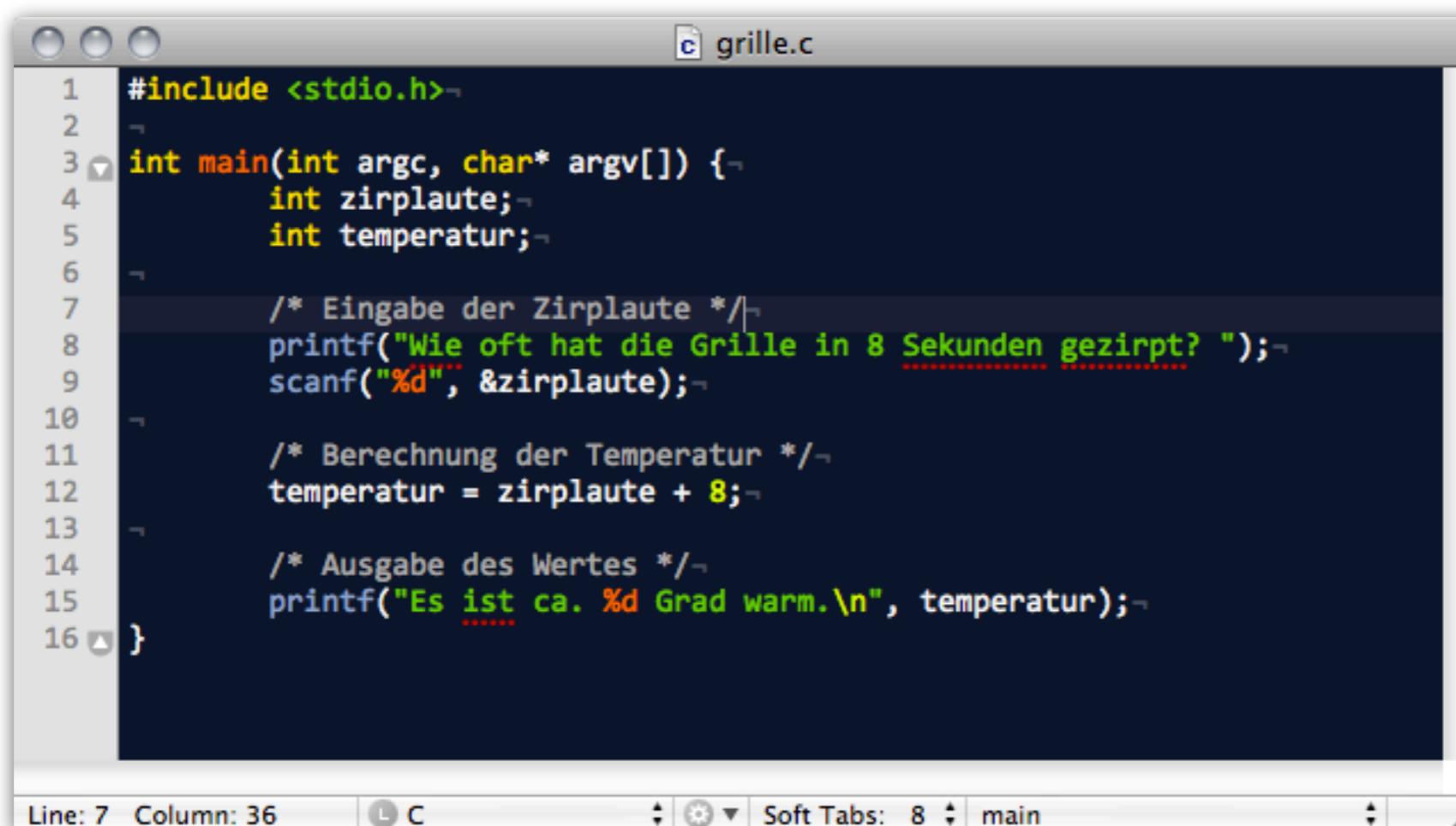
Ausgabe

gib Temperatur aus



# 1. PROGRAMM: GRILLENZIRPEN

Zum Vergleich: dasselbe Programm in C

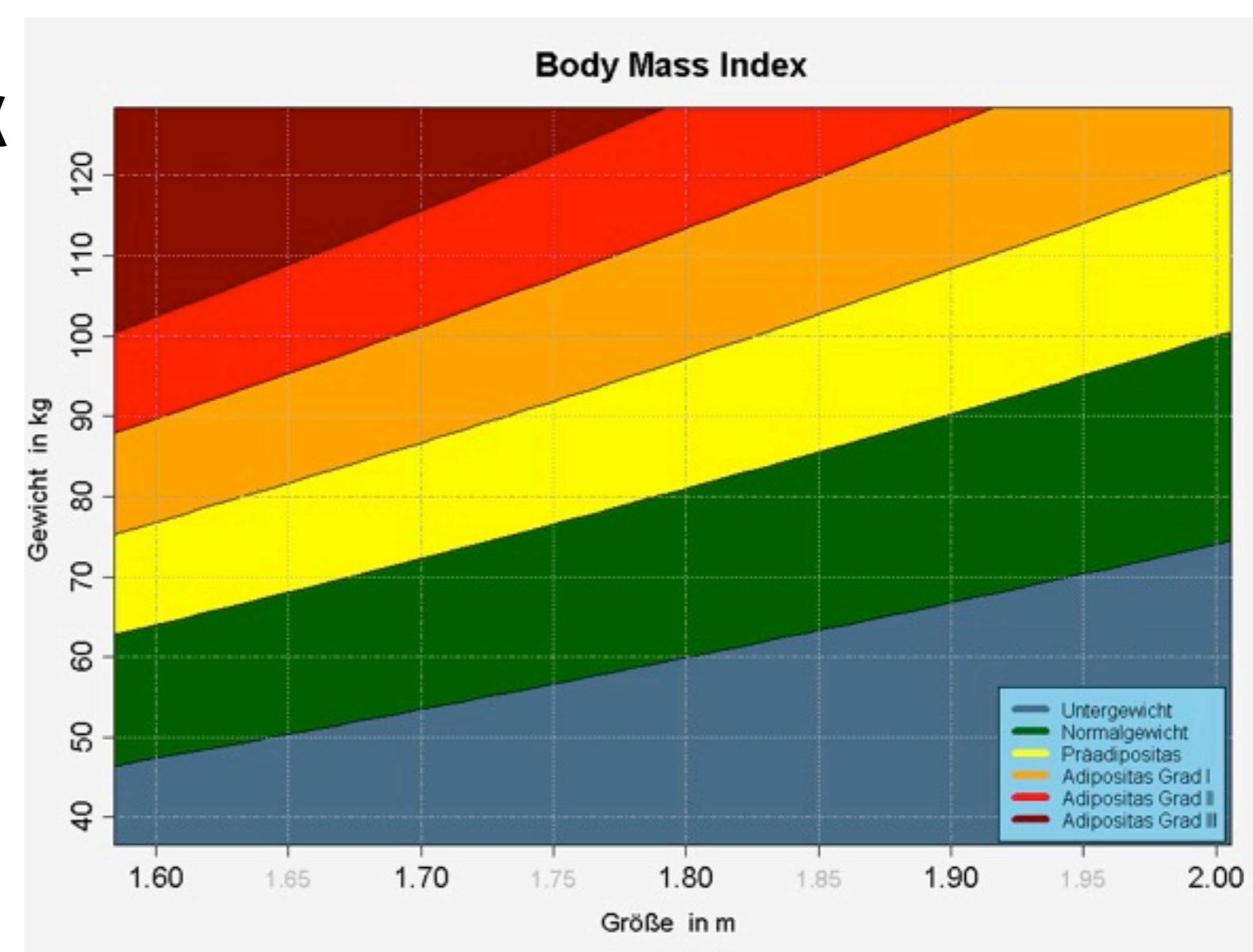


```
grille.c
1 #include <stdio.h>
2
3 int main(int argc, char* argv[]) {
4     int zirplaute;
5     int temperatur;
6
7     /* Eingabe der Zirplaute */
8     printf("Wie oft hat die Grille in 8 Sekunden gezirpt? ");
9     scanf("%d", &zirplaute);
10
11    /* Berechnung der Temperatur */
12    temperatur = zirplaute + 8;
13
14    /* Ausgabe des Wertes */
15    printf("Es ist ca. %d Grad warm.\n", temperatur);
16}
```

Line: 7 Column: 36    C    Soft Tabs: 8    main

## 2. PROGRAMM: BMI

Der **Body-Mass-Index** (BMI) – ist eine Maßzahl für die Bewertung des Gewichts (Körpermasse) eines Menschen. Der BMI wird folgendermaßen berechnet:



$$BMI = \frac{\text{Gewicht in kg}}{(\text{Größe in m})^2}$$

## 2. PROGRAMM: BMI

**E**

lies Größe, Gewicht



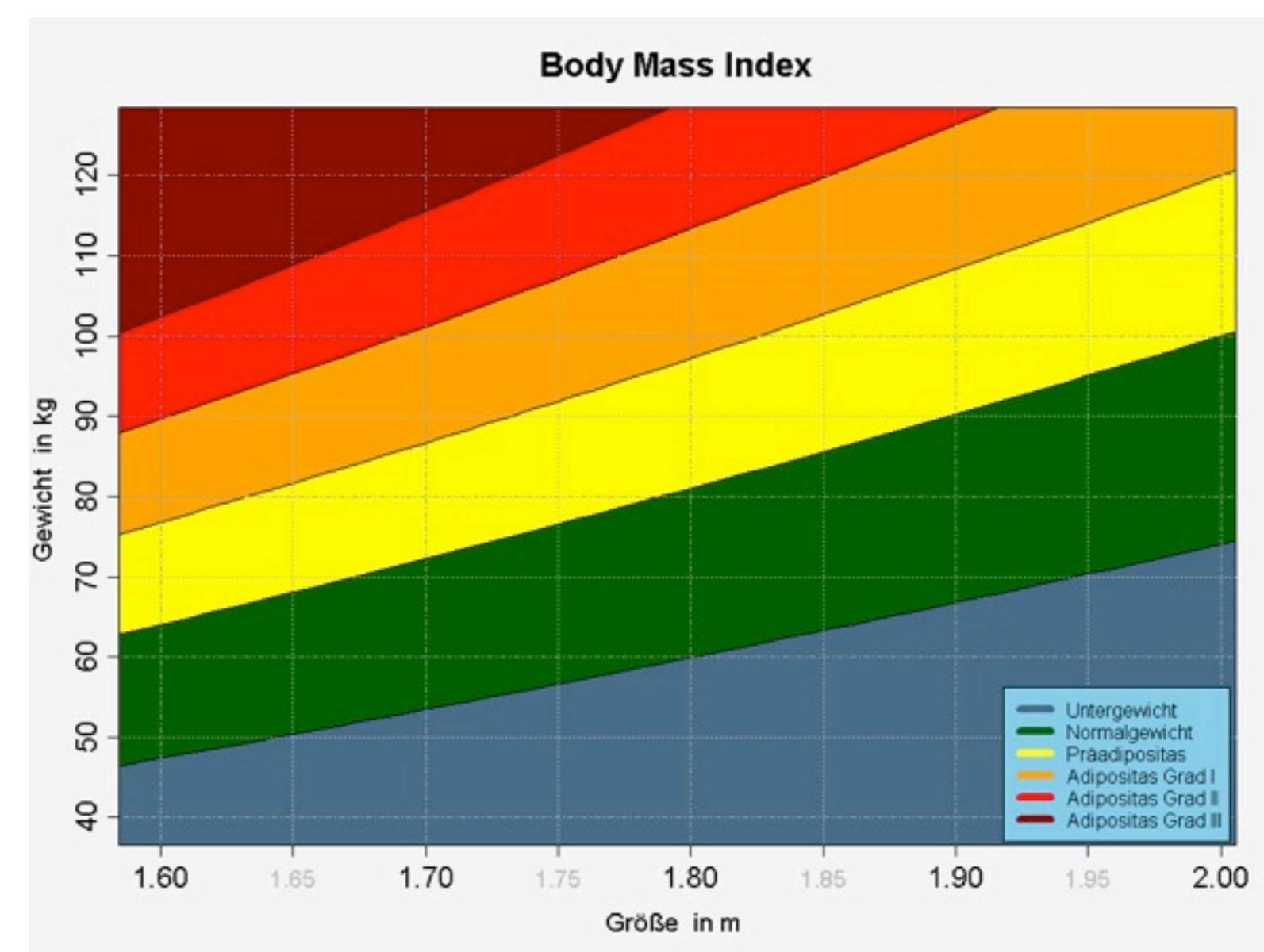
**V**

berechne BMI



**A**

gib BMI aus



$$BMI = \frac{\text{Gewicht in kg}}{(\text{Größe in m})^2}$$

### 3. PROGRAMM: DREIECKSFLÄCHE

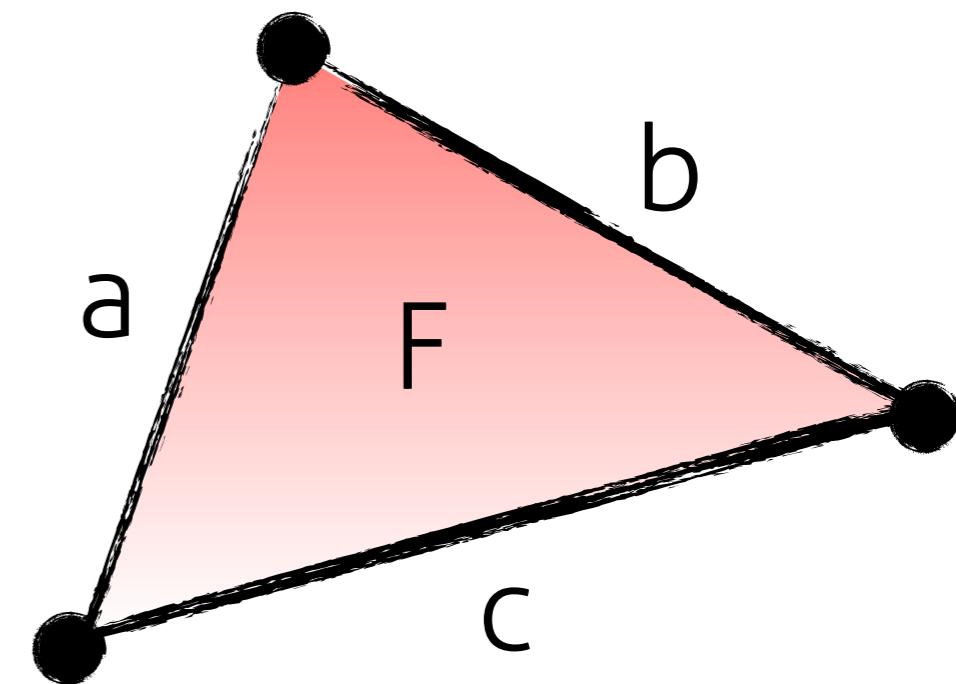
Die **Heronische Formel** gibt den Flächeninhalt  $F$  eines Dreiecks mit den Seitenlängen  $a$ ,  $b$  und  $c$  als

$$F = \sqrt{s \cdot (s - a) \cdot (s - b) \cdot (s - c)}$$

an, wobei  $s = \frac{a + b + c}{2}$

der halbe Umfang des Dreiecks ist.

Entwickle ein Programm, welches mit vom Nutzer eingegebenen Seitenlängen den Flächeninhalt des Dreiecks berechnet.



### 3. PROGRAMM: DREIECKSFLÄCHE

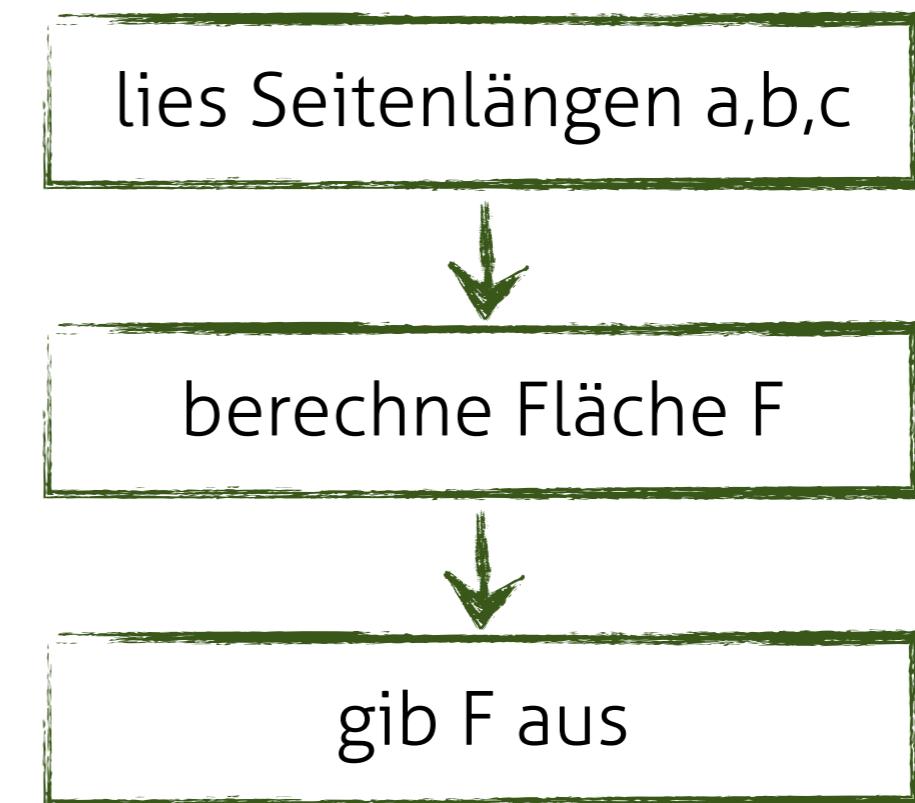
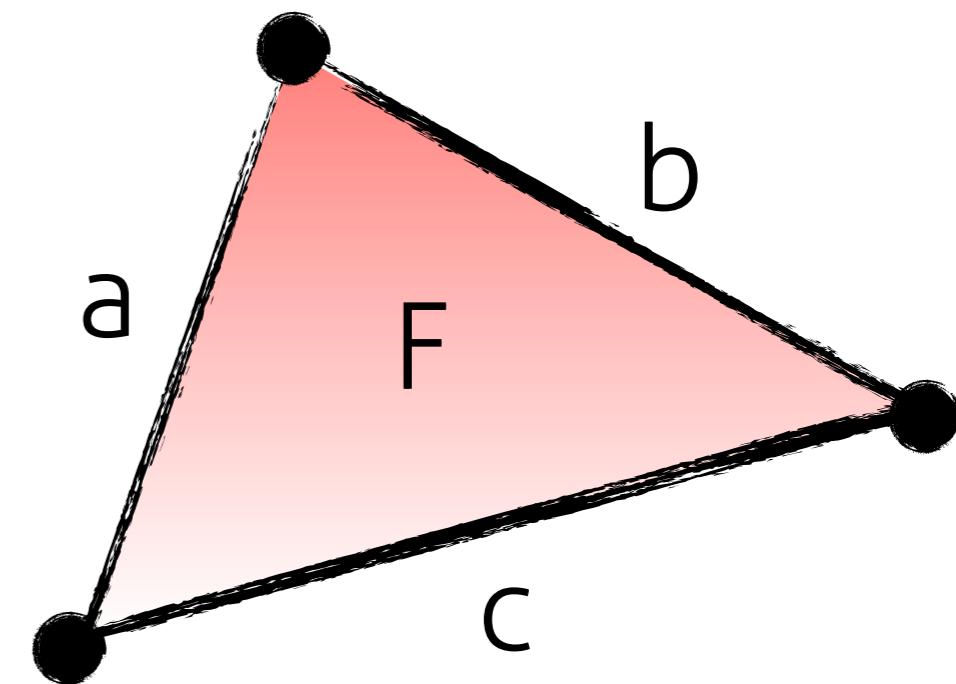
Die **Heronische Formel** gibt den Flächeninhalt  $F$  eines Dreiecks mit den Seitenlängen  $a, b$  und  $c$  als

$$F = \sqrt{s \cdot (s - a) \cdot (s - b) \cdot (s - c)}$$

an, wobei  $s = \frac{a + b + c}{2}$

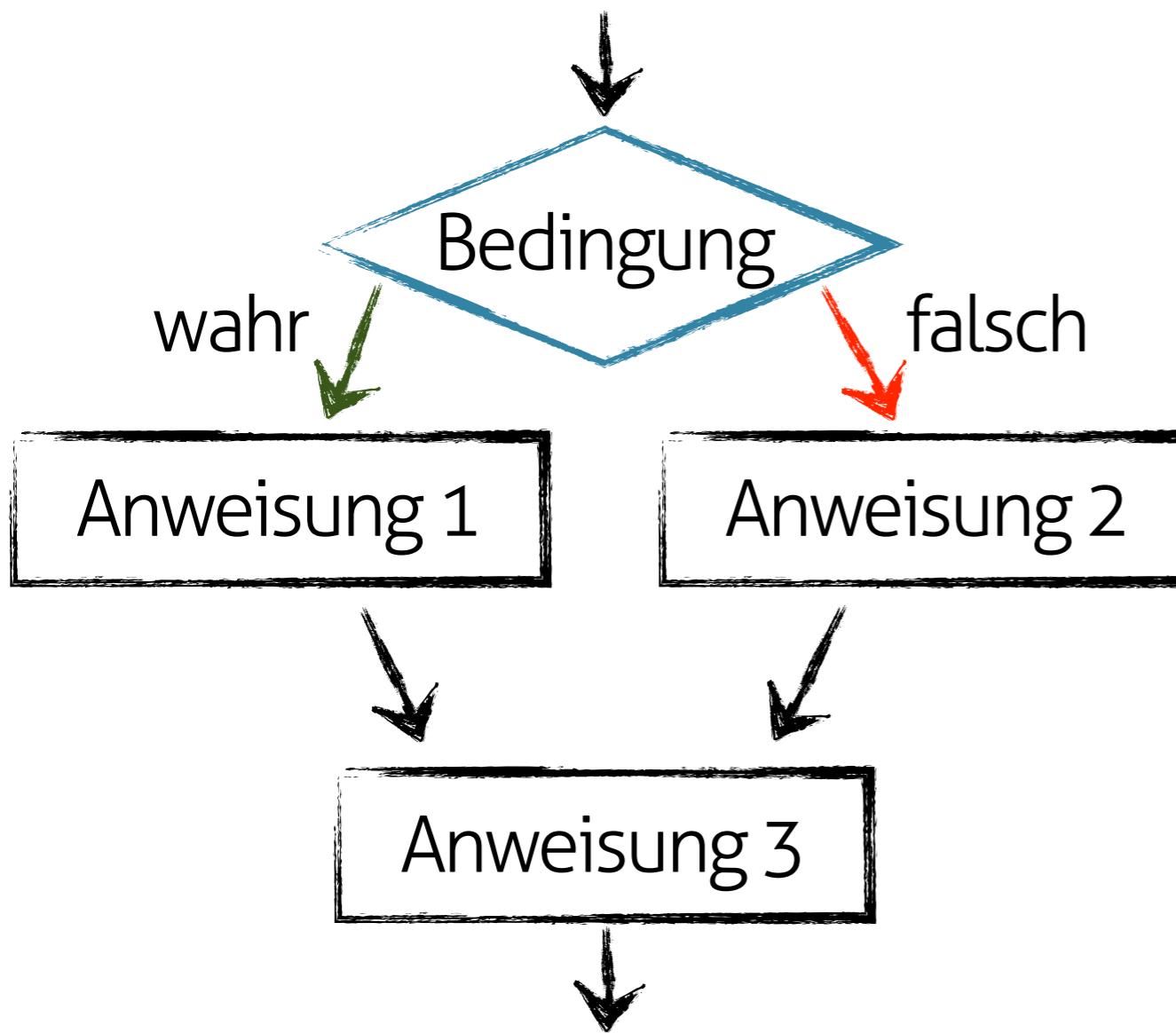
der halbe Umfang des Dreiecks ist.

```
import math  
math.sqrt(81)
```



# VERZWEIGUNG

- Alternative
- Fallunterscheidung
- bedingte Ausführung



erst zur Laufzeit des Programmes wird festgelegt, welche Anweisungen ausgeführt werden

# BEDINGUNGEN

Vergleichsoperatoren: == != > < >= <=

**Achtung:** nicht = statt == verwenden!



Verknüpfungen von Bedingungen:

**and** beide Bedingungen müssen wahr sein

**or** es reicht, wenn eine Bedingung wahr ist

**not** Ergebnis umkehren

# EINRÜCKUNG IN PYTHON

## Python

```
if x > 0:  
    y = 1  
    z = 0  
  
else:  
    y = 0  
    z = 1  
  
x = z
```

## C, C++, Java

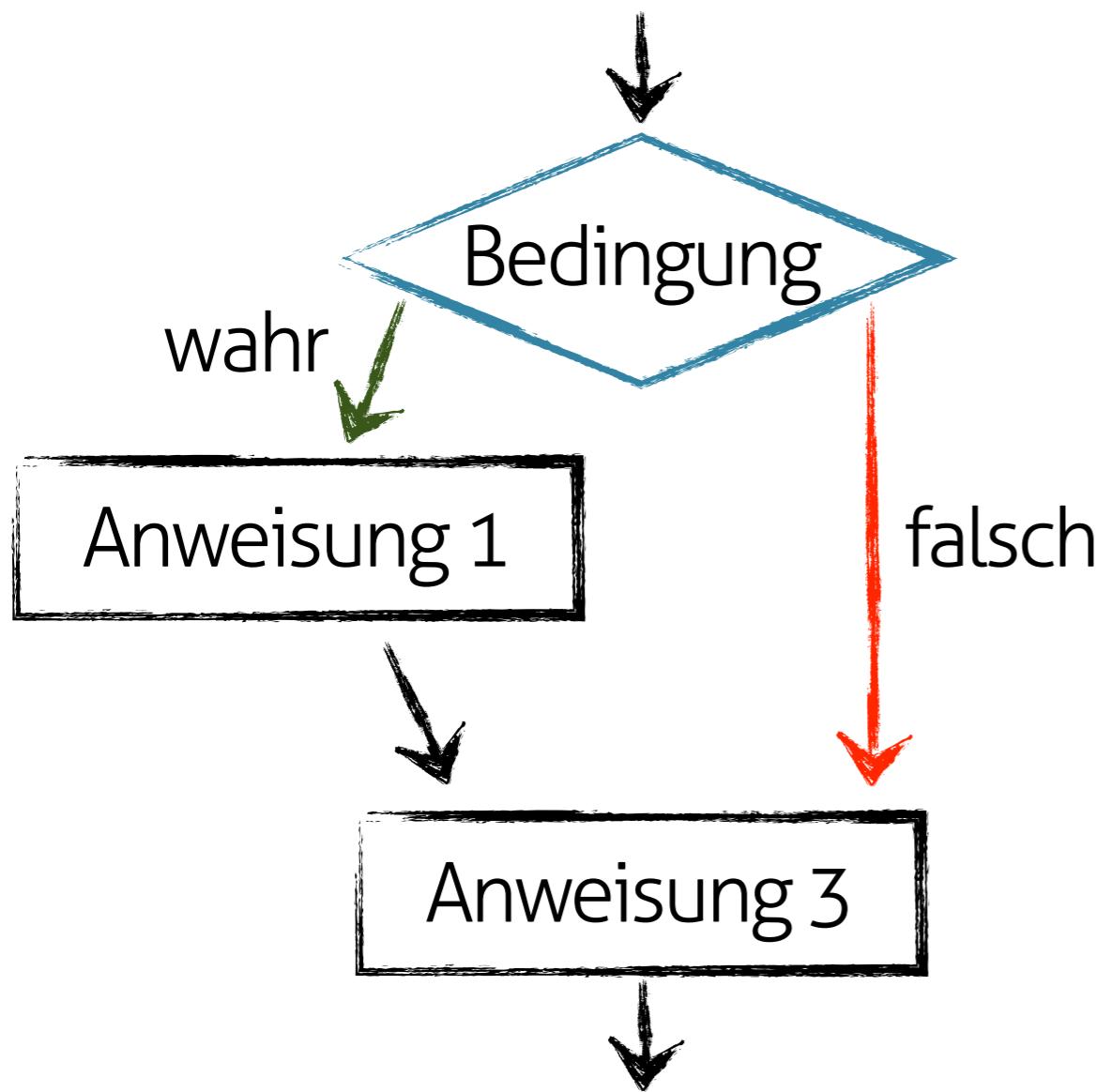
```
if (x > 0) {  
    y = 1; z = 0;  
} else {  
    y = 0;  
    z = 1;  
}  
  
x = z
```

Einrückung essentiell  
keine Klammern / Semikola

Einrückung = Kosmetik  
Blöcke über Klammern

# VERZWEIGUNGEN

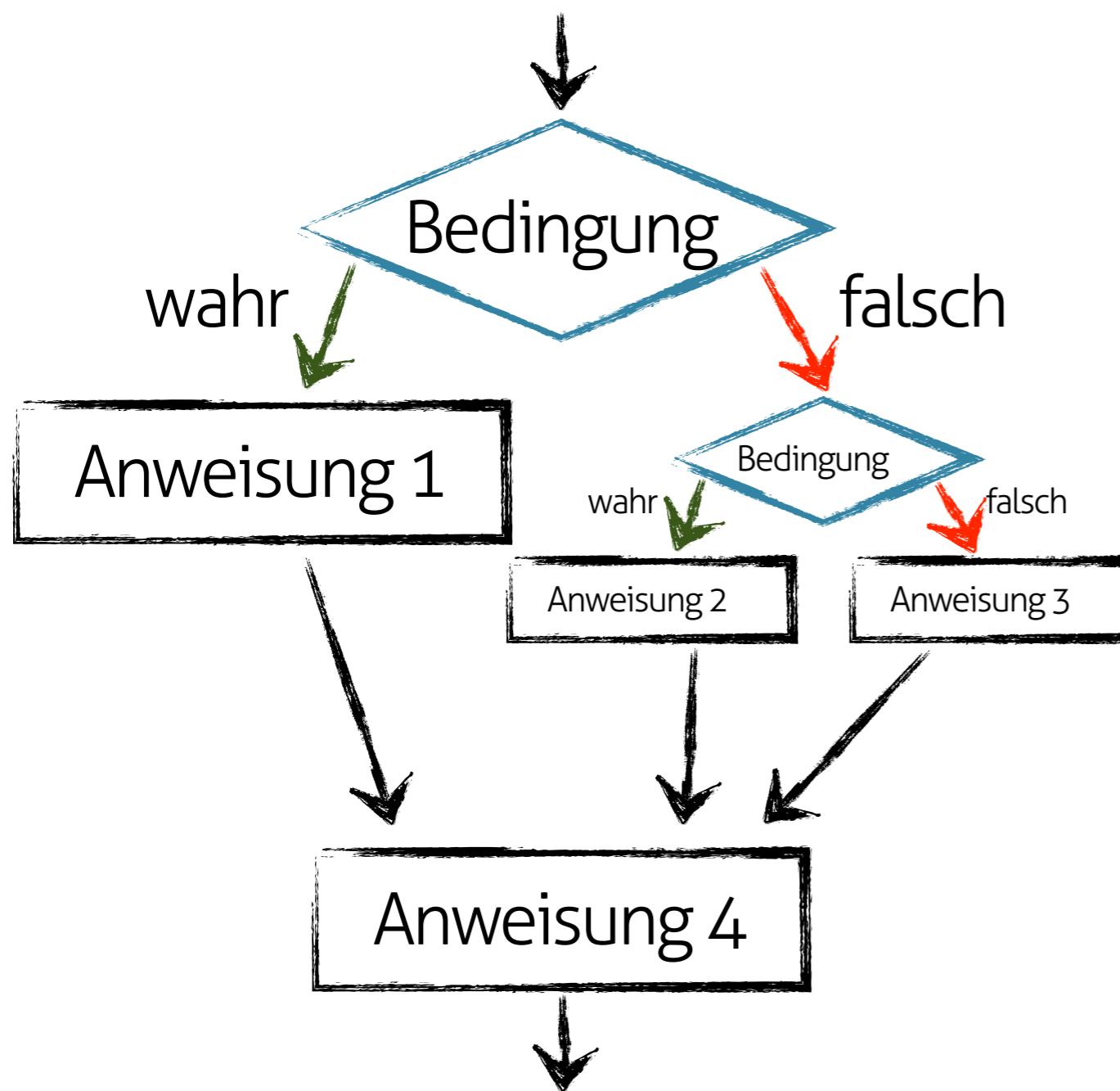
der **else**-Zweig ist optional



falls Bedingung nicht  
erfüllt wird, wird nach  
der Verzweigung  
fortgefahren

# VERZWEIGUNGEN

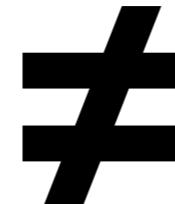
Verzweigungen können geschachtelt werden



eine Verzweigung ist schließlich auch nur eine Anweisung

# SCHACHTELUNG IN PYTHON

```
if x > 0:  
    if y > 0:  
        z = 0  
  
else:  
    y = 0  
    z = 1  
  
x = z
```



```
if x > 0:  
    if y > 0:  
        z = 0  
  
else:  
    y = 0  
    z = 1  
  
x = z
```

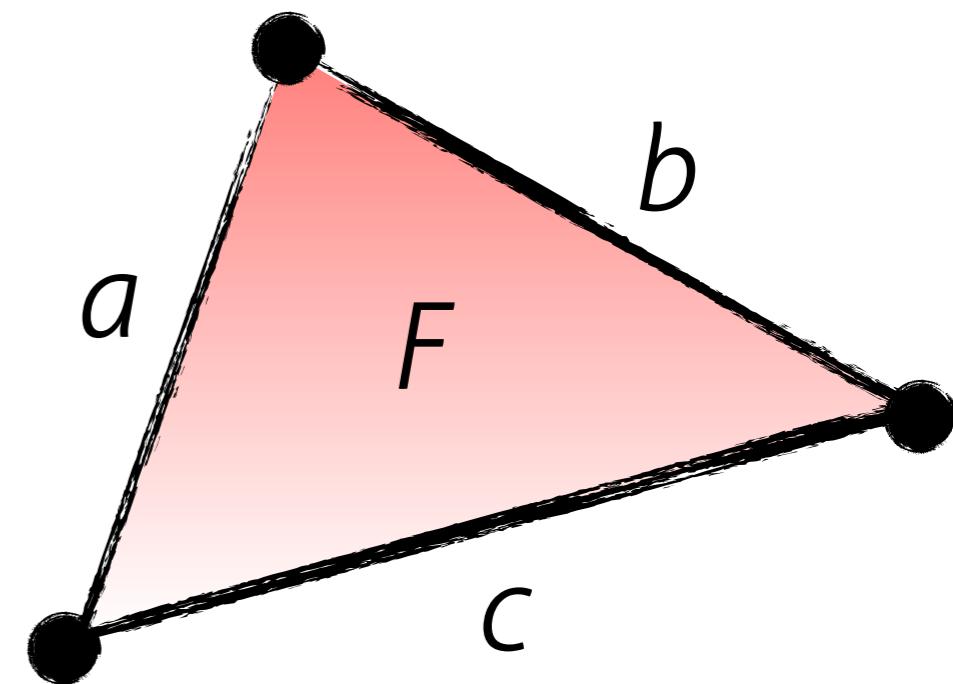
### 3. PROGRAMM: DREIECKSFLÄCHE (FORTSETZUNG)

Die **Heronische Formel** gibt den Flächeninhalt  $F$  eines Dreiecks mit den Seitenlängen  $a, b$  und  $c$  als

$$F = \sqrt{s \cdot (s - a) \cdot (s - b) \cdot (s - c)}$$

wobei  $s = \frac{a + b + c}{2}$

der halbe Umfang des Dreiecks ist.



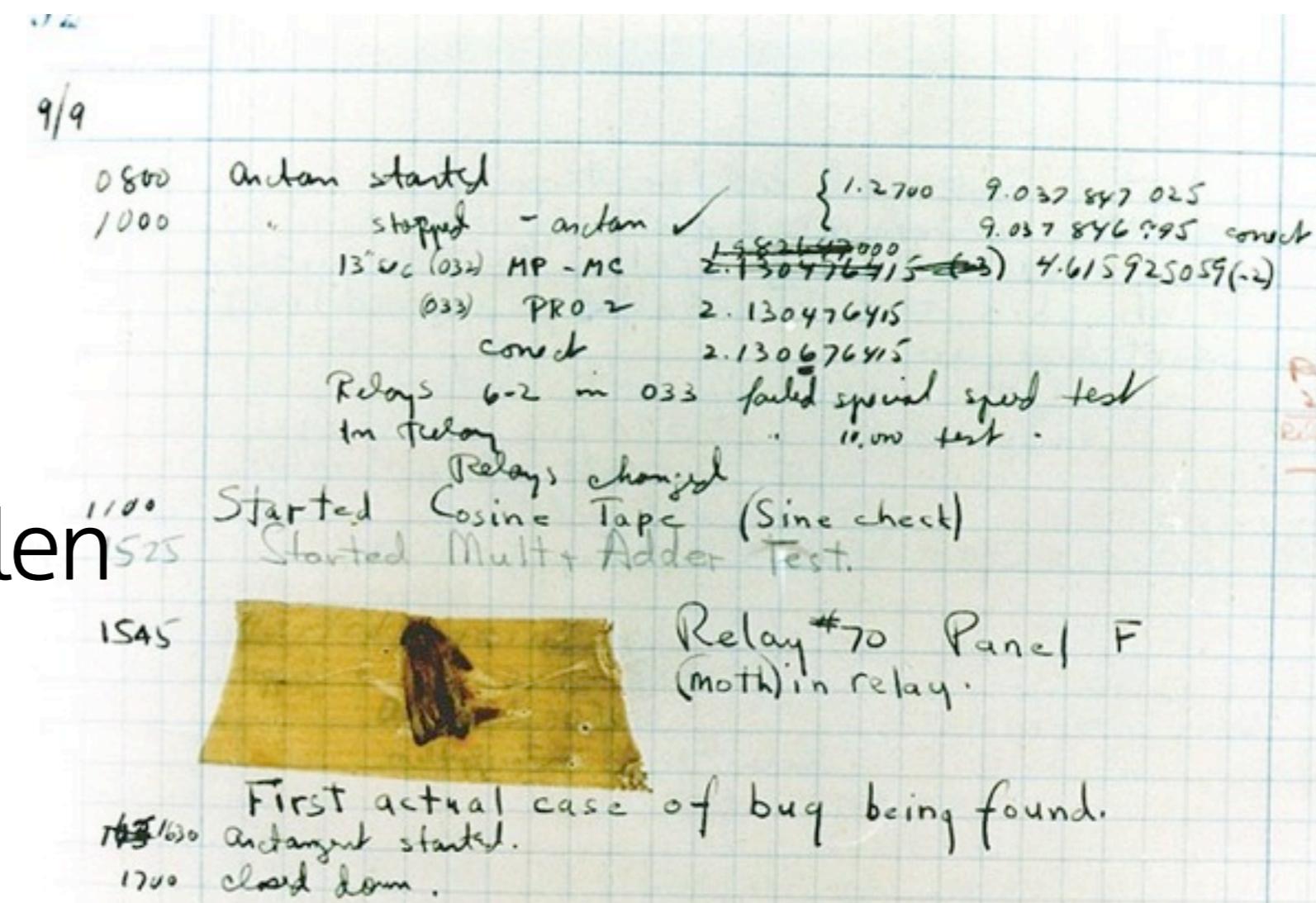
**Ergänze das Programm zur Berechnung der Dreiecksfläche, indem du negative Eingabewerte abfängst.**

# PROBLEME FINDEN

Fehler in Programmen sind oft sehr gut versteckt

Es gibt sehr viele Arten von Fehlern:

- logische Fehler
- falsche Datentypen
- Zeichendreher
- = statt ==
- uninitialized Variablen
- vergessene Fälle
- ...



## 4. PROGRAMM: ZAHLENRATEN

Das Projekt "**Zahlenraten**" soll nach und nach entstehen.

Anfangs soll sich das Programm die ganze Zahl 23 denken.

Der Nutzer gibt seinen Tipp ein und das Programm vergleicht die Eingabe mit der zu erratenden Zahl.

Dem Nutzer wird mitgeteilt, ob er die "gedachte" Zahl erraten hat, oder ob er zuviel oder zuwenig geraten hat.



# ZUFALLSZAHLEN

# Python kann Zufallszahlen generieren:

```
import random  
random.seed()  
roulette = random.randint(0, 36)
```

# Ergebnis:

(ganze) Zufallszahl  
zwischen 0 und 36



# 4. PROGRAMM: ZAHLENRATEN (FORTSETZUNG)

**Die zu erratende Zahl  
soll zufällig generiert werden.**

Der Nutzer gibt seinen Tipp ein und das Programm vergleicht die Eingabe mit der zu erratenden Zahl.

Dem Nutzer wird mitgeteilt, ob er die "gedachte" Zahl erraten hat.





# python

Erste Schritte

## Schleifen und Listen

5

Niels Lohmann  
Stefanie Behrens

Lutz Hellmig  
Karsten Wolf

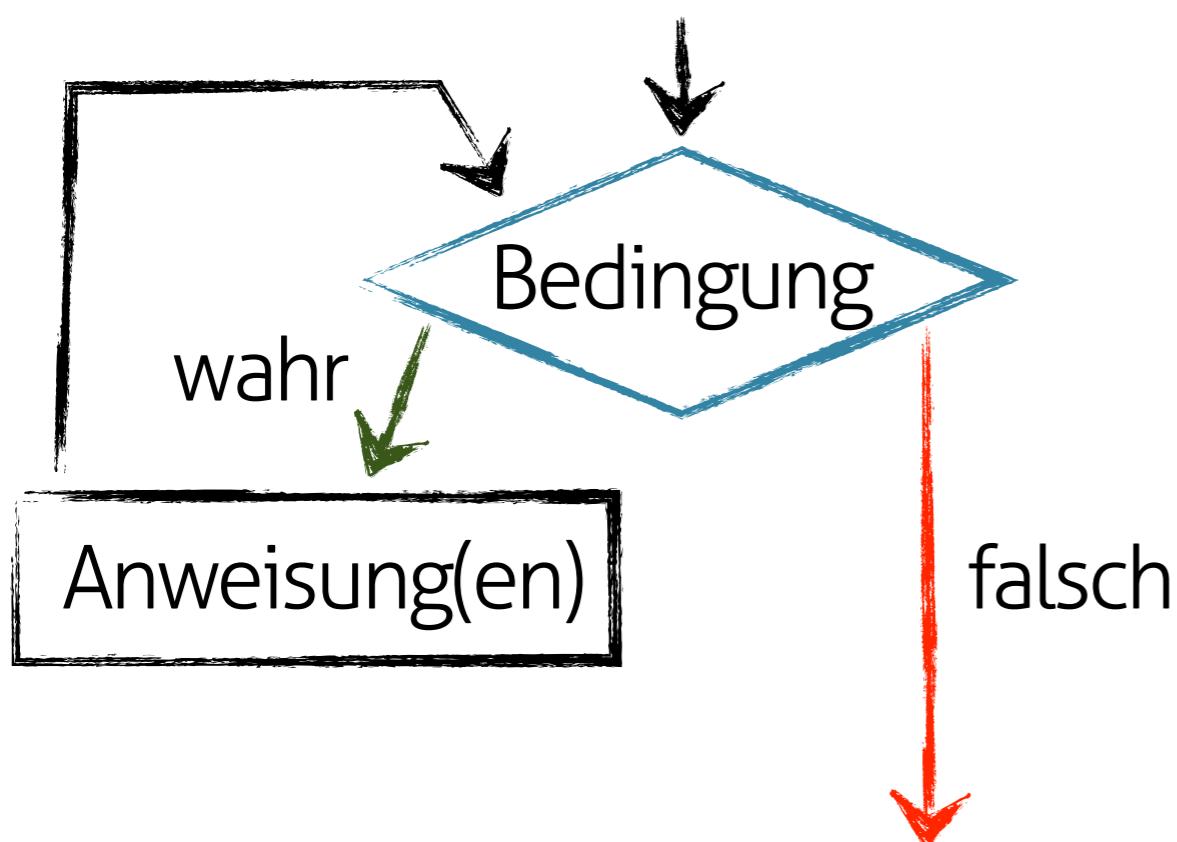
Universität  
Rostock



Traditio et Innovatio

# WHILE-SCHLEIFEN

wiederholte Ausführung  
von Anweisungen



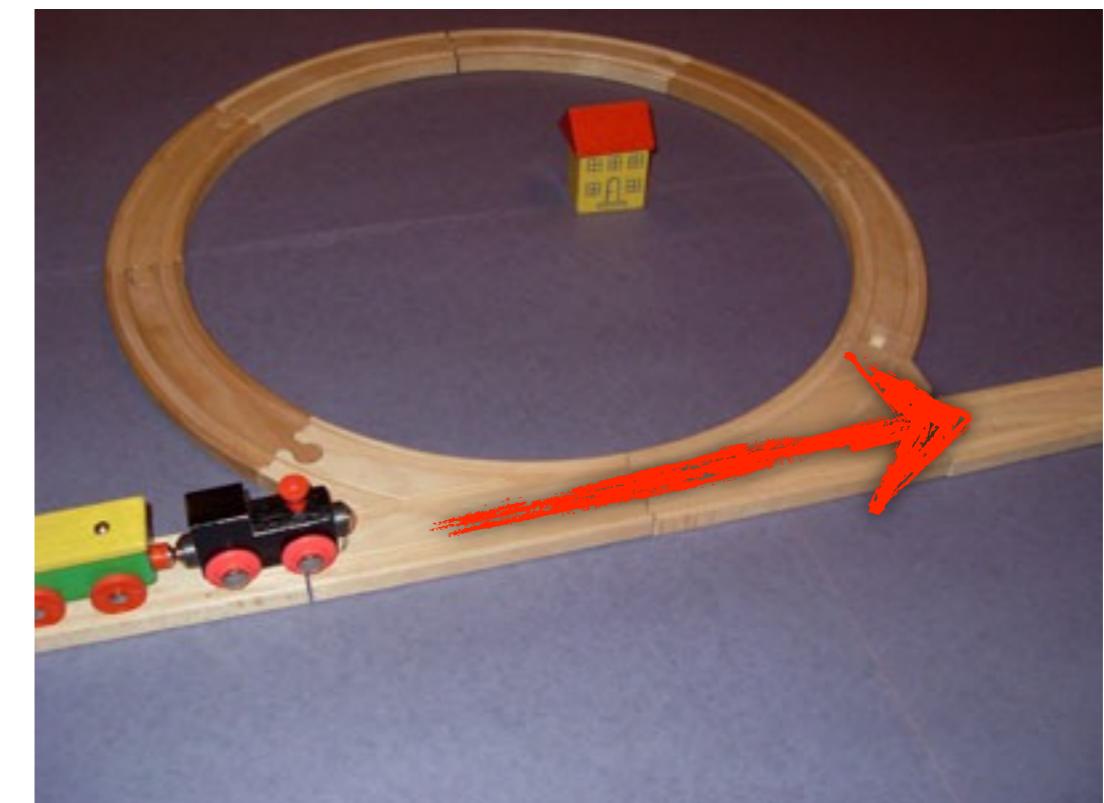
solange die  
Bedingung gilt, wird  
die Anweisung  
ausgeführt

# SCHLEIFEN

auch Schleifen können  
geschachtelt  
werden

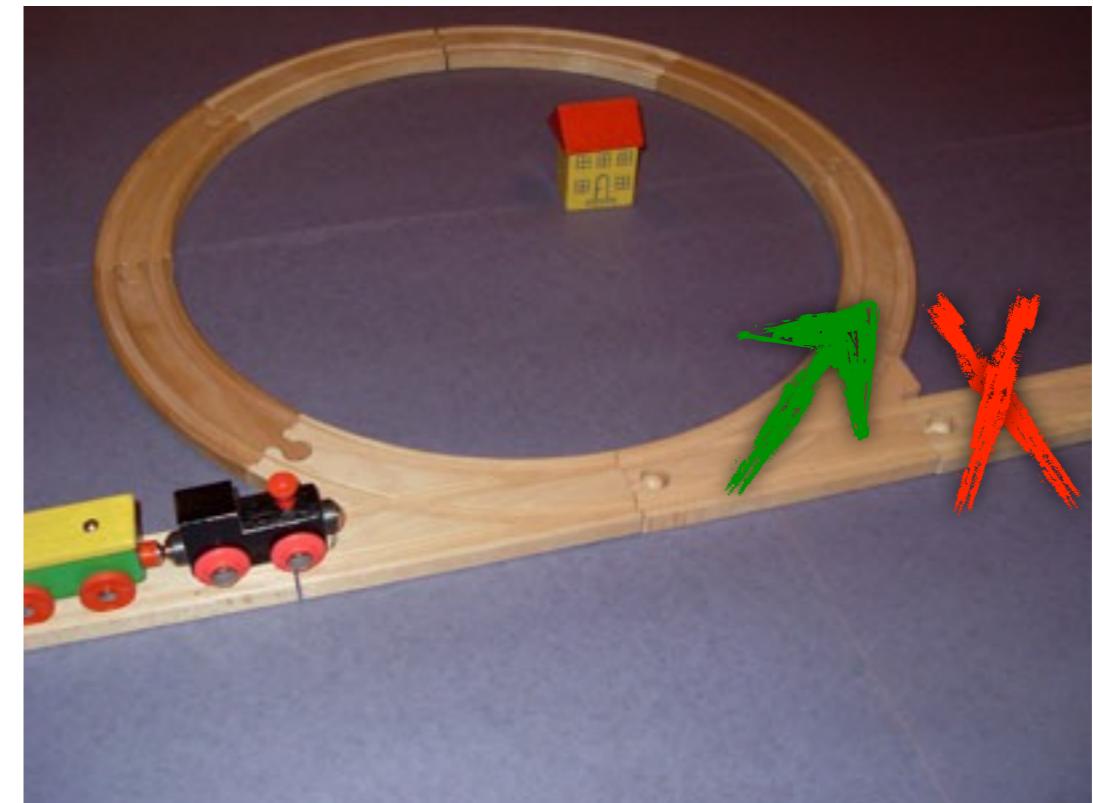


gilt Bedingung zu  
Anfang nicht,  
wird die Schleife  
übersprungen



# ENDLOSSCHLEIFEN

gilt Bedingung immer,  
wird die Schleife nie  
verlassen  
**= Endlosschleife**



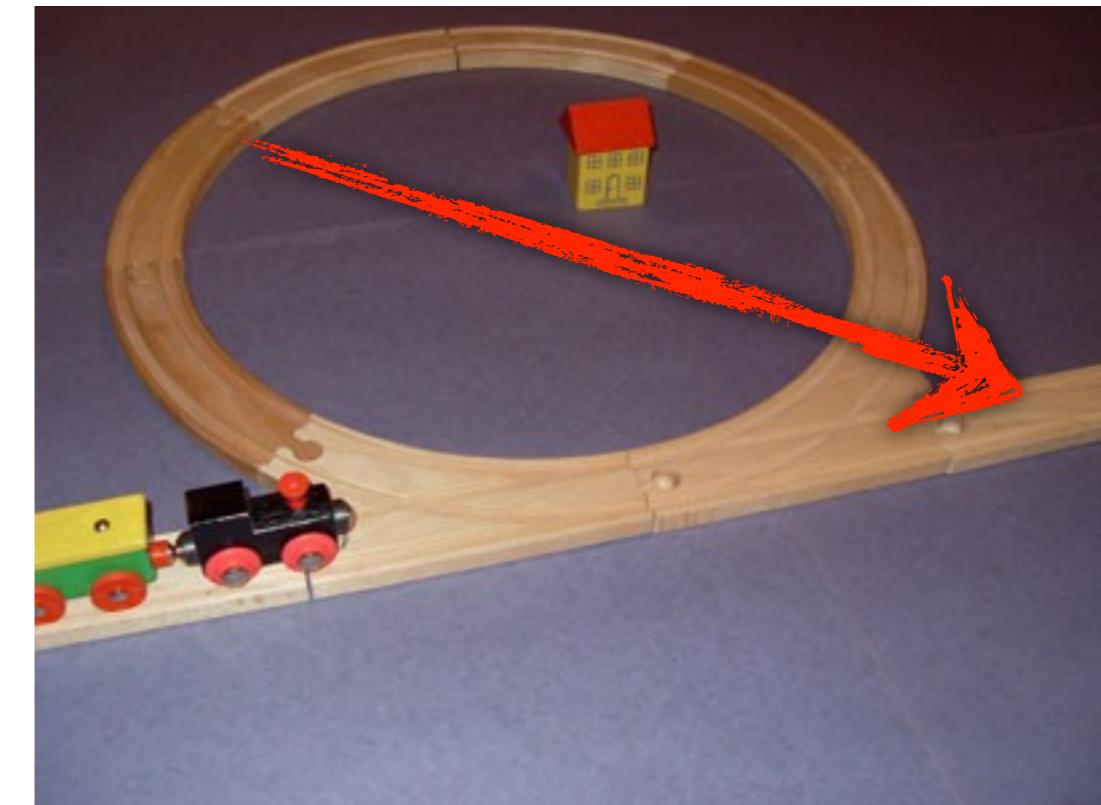
oft Grund für  
Endlosschleifen:  
Variablen in der  
Bedingung werden  
nie geändert



# SPRÜNGE IN SCHLEIFEN

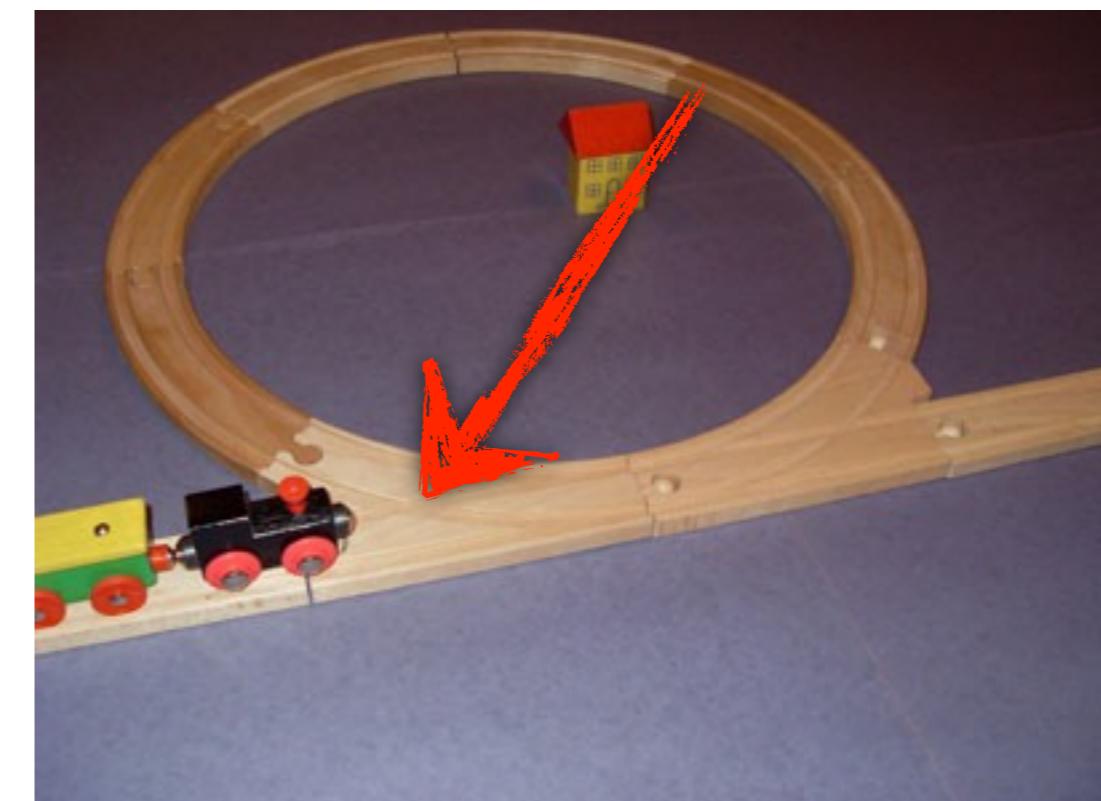
**break**

springt hinter  
die Schleife



**continue**

springt in die  
nächste Runde



# FOR-SCHLEIFEN

Anweisung wird  
mehrfach ausgeführt  
(wie oft, ist schon klar)

jede Runde  
bekommt eine Nummer



Nummer wird in Zählvariable verwaltet

Zählvariable durchläuft ein Intervall von Werten

guter Stil: Zählvariable mit **i**, **j**, **k**, ... benennen

# INTERVALLE

**range(10)** = 0 1 2 3 4 5 6 8 9

**range(3, 7)** = 3 4 5 6

**range(1, 10, 2)** = 1 3 5 7 9

**range(10, 0, -1)** = 10 9 8 7 6 5 4 3 2 1

**range(10, 3, 1)** =

## 4. PROGRAMM: ZAHLENRATEN (FORTSETZUNG)

Die zu erratende Zahl soll zufällig generiert werden.

Der Nutzer gibt seinen Tipp ein und das Programm vergleicht die Eingabe mit der zu erratenden Zahl.



**Der Nutzer darf maximal fünfmal raten.** Dem Nutzer wird mitgeteilt, ob er die "gedachte" Zahl erraten hat.

**Nachdem das Ergebnis der Raterunde mitgeteilt wurde, wird der Nutzer gefragt, ob er erneut spielen will. Bei "Ja" beginnt das Spiel von vorn.**

# LISTEN

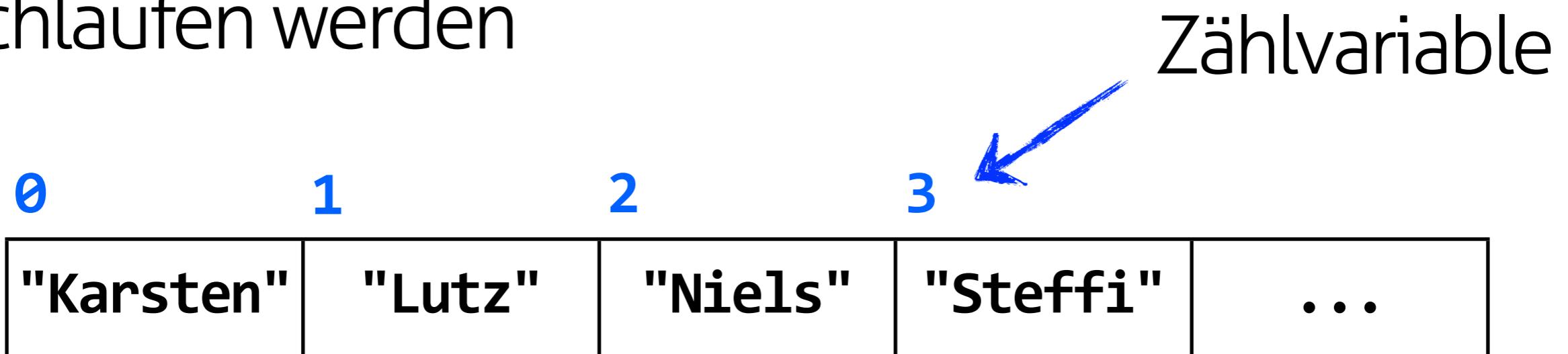
Datentyp, um mehrere Werte zu speichern  
jeder Wert hat einen **Index**

0	1	2	3	
"Karsten"	"Lutz"	"Niels"	"Steffi"	...



# LISTEN

können sehr gut mit **for**-Schleifen  
durchlaufen werden



# FUNKTIONEN AUF LISTEN

**len(Namen)**

Anzahl

**Namen.count("Karsten")**

Anzahl mit Wert

**Namen.reverse()**

Liste umdrehen

**Namen.sort()**

Liste sortieren

**Namen.append("Andreas")**

Wert anfügen

**Namen.index("Andreas")**

1. Index von Wert

**Namen.insert(0, "Susi")**

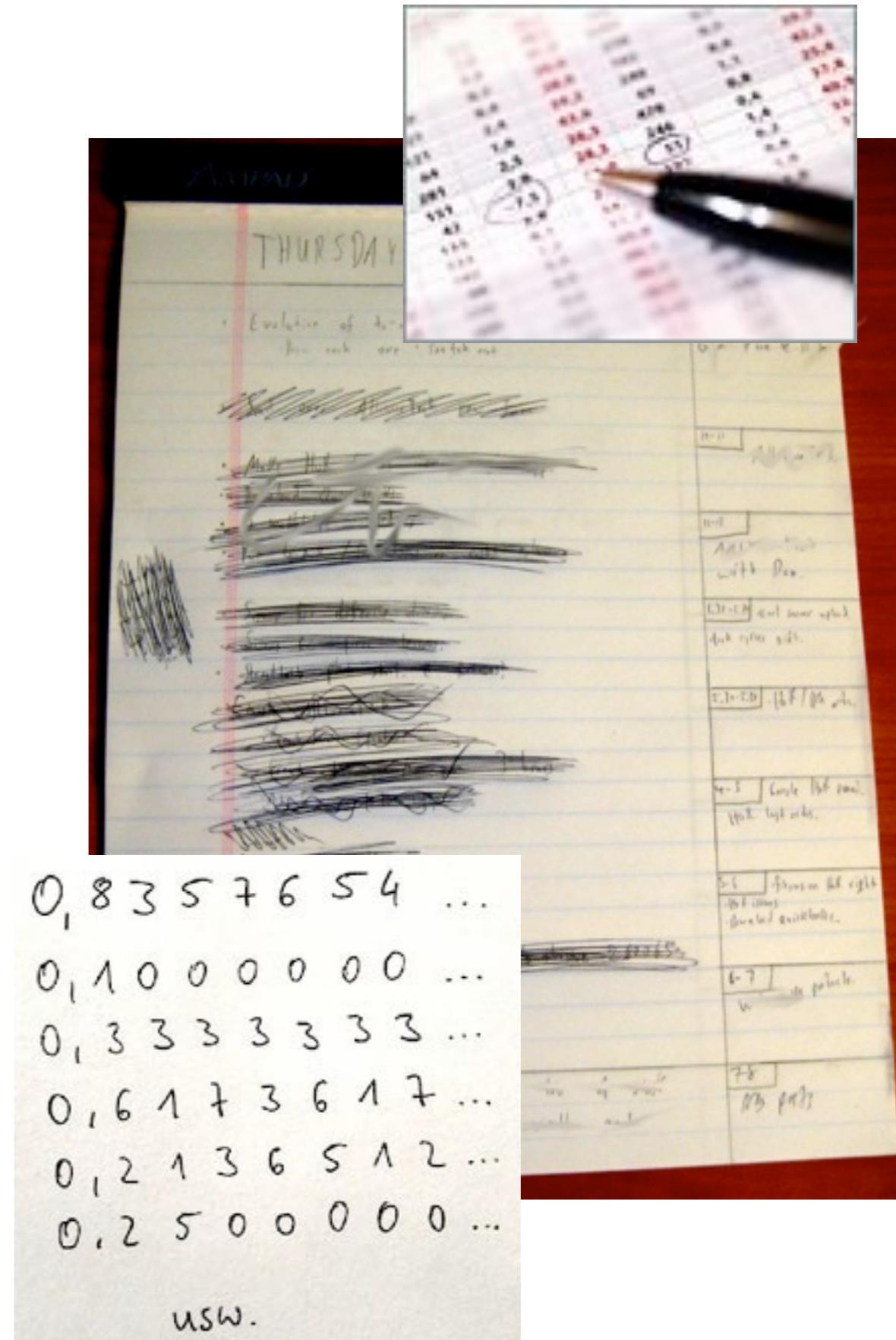
einfügen vor Index

**Namen.remove("Lutz")**

1. Vorkommen löschen

# AUFGABEN MIT LISTEN

- Aufbau einer Liste mit 10 Zufallszahlen zwischen 1 und 50.
- Ausgabe der Summe der ungeraden Zahlen in dieser Liste
- Ausgabe des zweitgrößten Wertes dieser Liste?
- Ausgabe des Durchschnittes der Werte der Liste?





# python

Erste Schritte

## Weitere Programmiertechniken

Niels Lohmann  
Stefanie Behrens

Lutz Hellmig  
Karsten Wolf

Universität  
Rostock



Traditio et Innovatio

# TEILEN UND HERRSCHEN

kleine Probleme sind einfacher als große

**Teilen und Herrschen:**  
große Probleme in kleine aufteilen

beim Programmieren über Kapselung in **Funktionen**



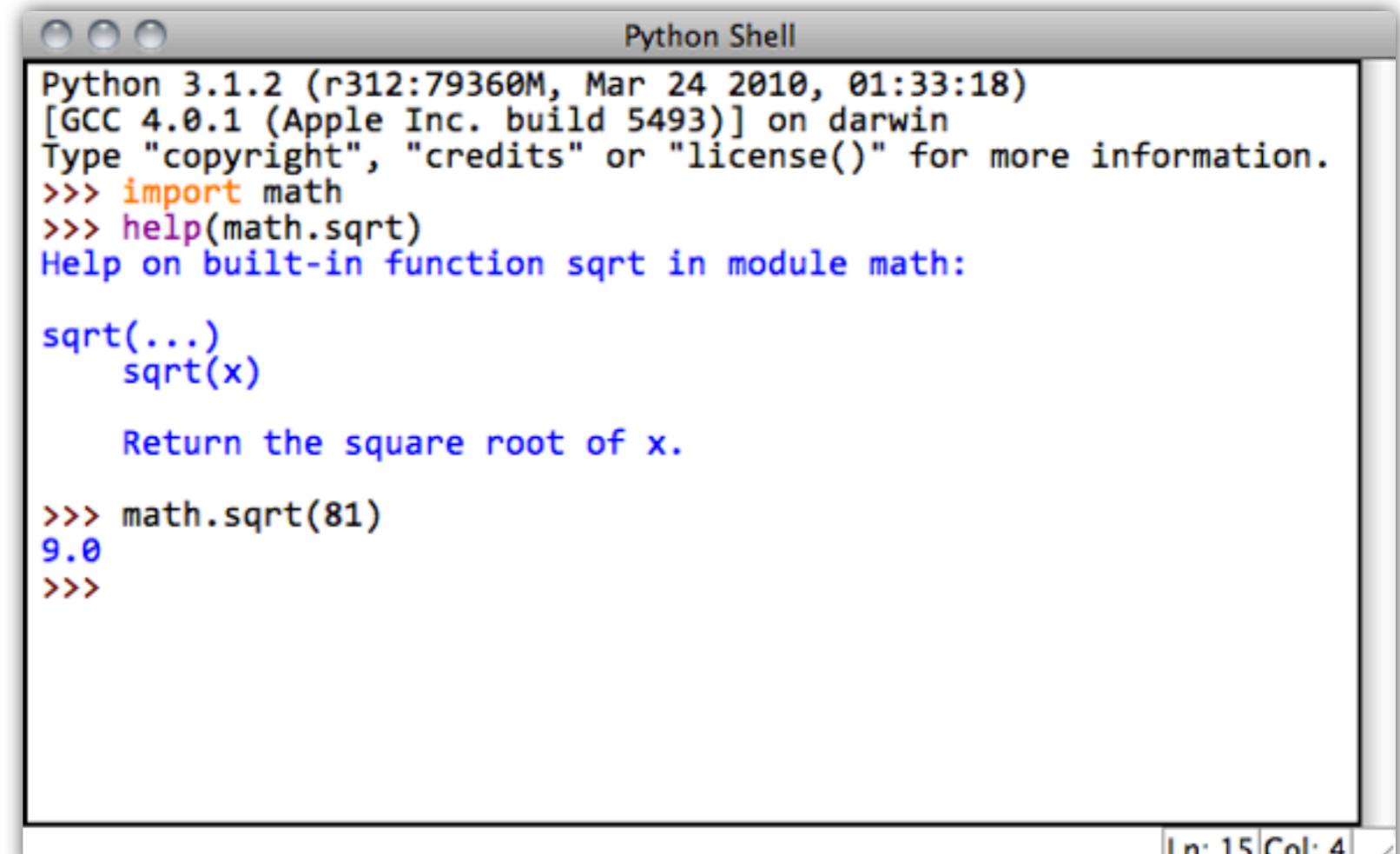
# FUNKTIONEN

schon gesehen: **math.sqrt** Funktion

Hilfe sagt **was**,  
aber nicht **wie**

Nutzung ohne  
weiteres Wissen  
möglich

eigenes Programm  
ist viel übersichtlicher, weil wir  
nicht das Rad neu erfinden müssen



The screenshot shows a Python Shell window with the following content:

```
Python 3.1.2 (r312:79360M, Mar 24 2010, 01:33:18)
[GCC 4.0.1 (Apple Inc. build 5493)] on darwin
Type "copyright", "credits" or "license()" for more information.

>>> import math
>>> help(math.sqrt)
Help on built-in function sqrt in module math:

sqrt(...)
    sqrt(x)

    Return the square root of x.

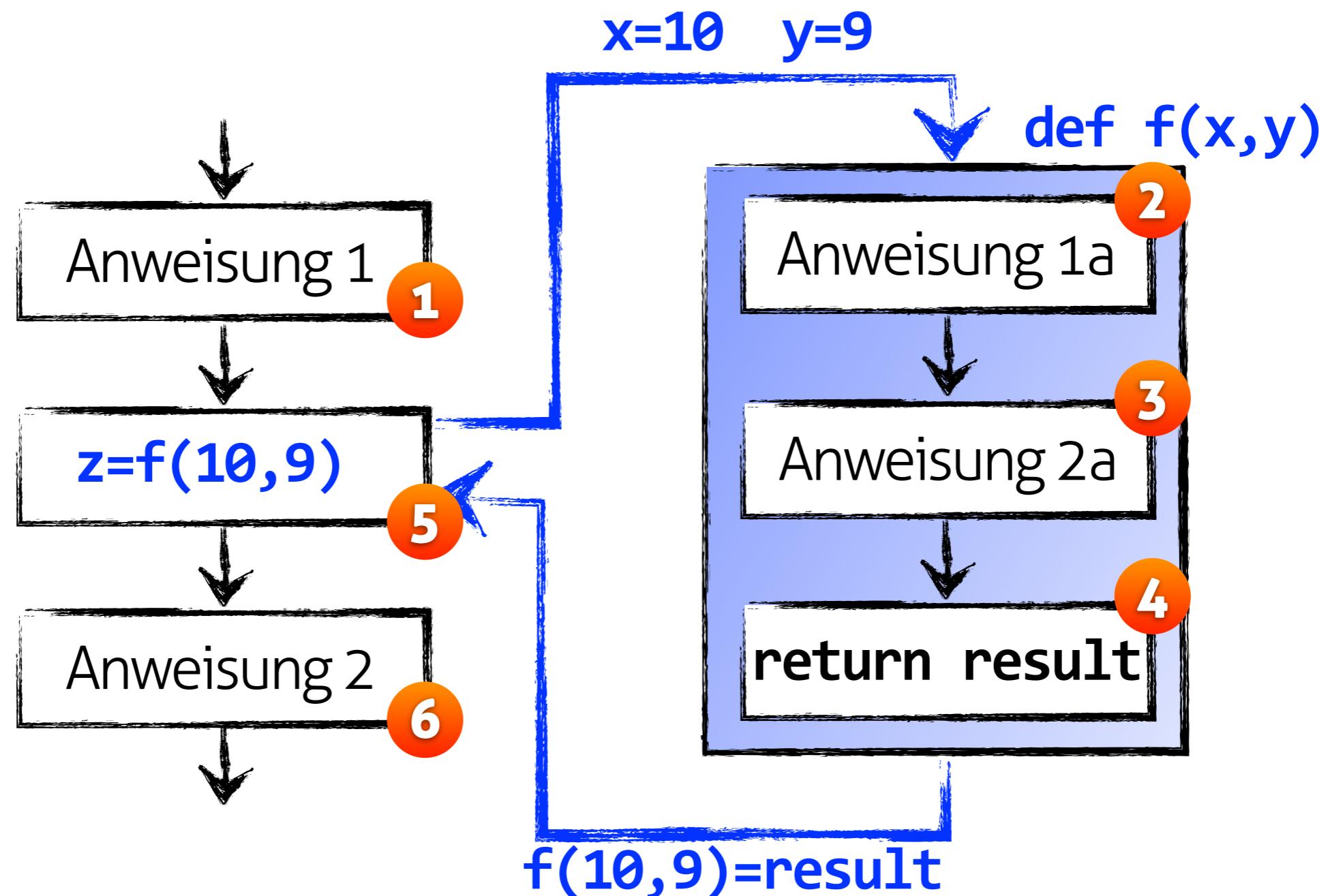
>>> math.sqrt(81)
9.0
>>>
```

The window title is "Python Shell". The status bar at the bottom right shows "Ln: 15 Col: 4".

# FUNKTIONEN MIT STIL

- sinnvolle Namen vergeben  
gut: **calculateListAverage(x)** oder **bmi(w, l)**  
schlecht: **iterate(a,b,c)** oder **do()**
- Funktion mit Kommentar beginnen  
**"""Returns the maximum of three numbers"""**
- möglichst keine unnötzen Ein-/Ausgaben in Funktionen
- Funktionen möglichst nicht schachteln

# FUNKTIONSAUFRUF

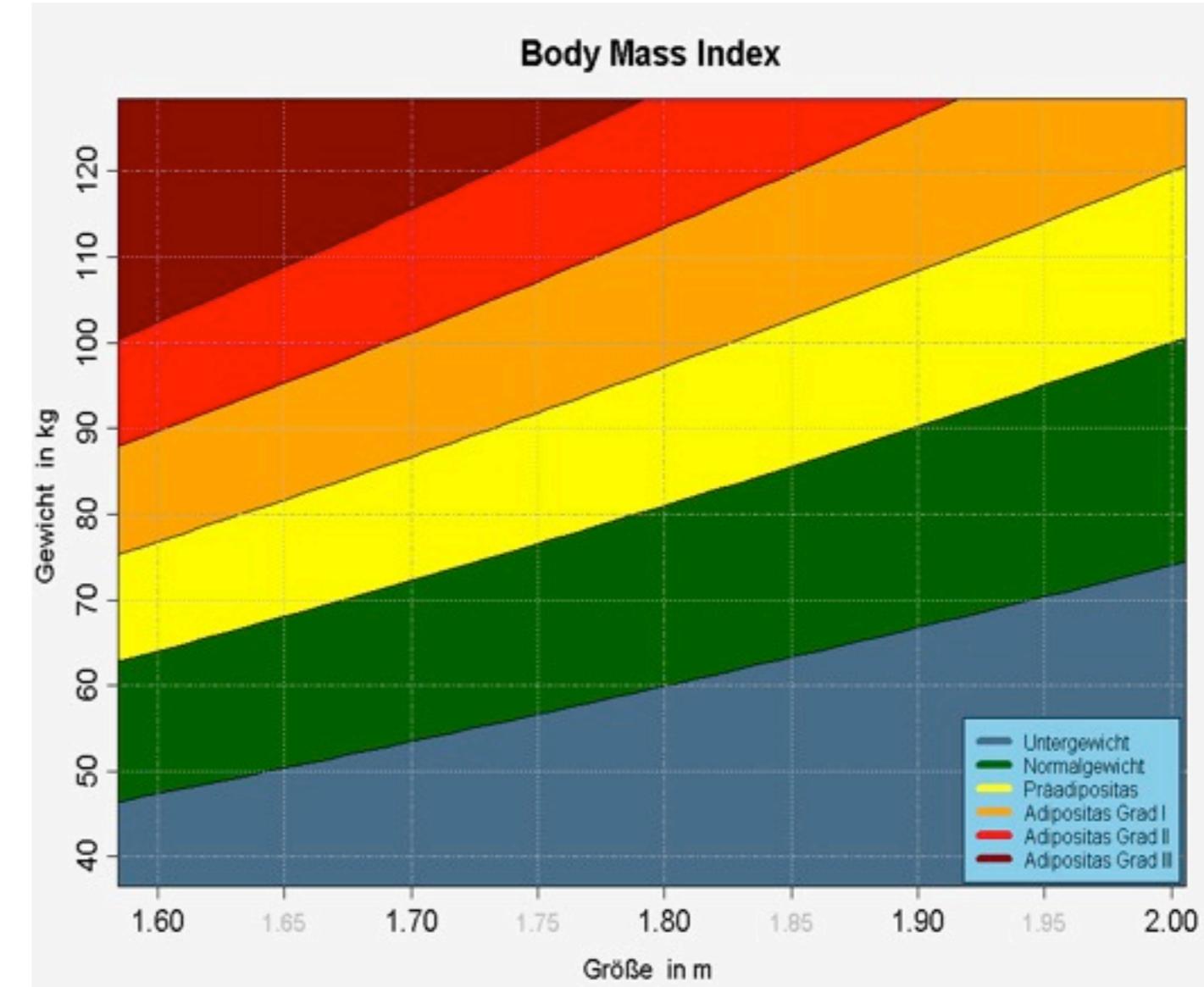


## 2. PROGRAMM: BMI (FORTSETZUNG)

Räume Dein BMI-Programm so auf, dass die Berechnung innerhalb einer Funktion

**bmi(gewicht, groesse)**

passiert.



$$BMI = \frac{\text{Gewicht in kg}}{(\text{Größe in m})^2}$$



# python

Erste Schritte

Niels Lohmann  
Stefanie Behrens

Lutz Hellmig  
Karsten Wolf

Universität  
Rostock



Traditio et Innovatio