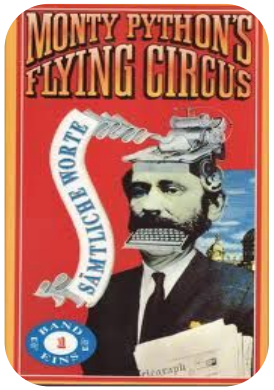
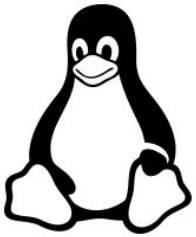
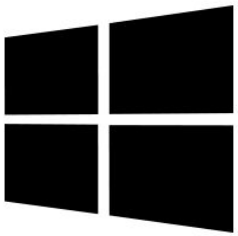
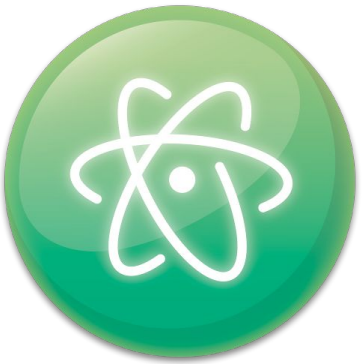
Grundlagen der Programmierung









wichtige Information

Struktur

Dr. Iulian Benta | Tomescu Vlad | Dr. Cătălin Rusu

Workload (in Stunden):

Vorlesung: 2
Seminar/Labor: 2 + 2

MS TEAMS: 7hl9agw (Grundlagen der Programmierung)

Email: rusu@cs.ubbcluj.ro
Fragen und Feedback sind immer erwünscht

Prüfungsform

- **K**lausur (20%)
- **Z**wischenprüfung (20%)
- **L**ab (30%)
- **P**raktische Prüfung (30%)

Minimale Leistungsstandards

K, Z, P, L >= 5 (inkl. aller Labors)

Anwesenheit

- Seminar: **10/14**
- Labor: **12/14**
- Kurs: **...**

- ohne diese Anwesenheit darf man keinen Klausur ablegen
- man darf das Seminar nur mit seiner Gruppe besuchen
- Abwesenheiten muss man immer begründen (vom Arzt)

Ziele

- Wie schreibt man Python-Programme
- Grundlegende Konzepte der objektorientierten Programmierung verstehen und selbstständig nutzen
- Herausforderungen und Lösungsansätze des Softwareentwicklungsprozesses verstehen
- Verstehen der Vorgehens- und Denkweisen von Informatikern
- Erster Einblick in Fähigkeiten und Möglichkeiten der Informatik in IT-Projekten

Kursinhalt

- Hello Python
- Software Entwicklung Intro
- Prozedurale Programmierung
- Modulare Programmierung
- Objektorientierte Programmierung
- Softwarearchitektur
- Vereinheitlichte Modellierungssprache (UML)
- Software Testing
- Rekursive Programmierung
- Komplexitätstheorie
- Suchalgorithmen
- Sortieralgorithmen
- Divide-et-Impera
- Backtracking

Dieses Semester

1. Hello Python

Was ist Informatik?

Informatik

ist die **Wissenschaft**

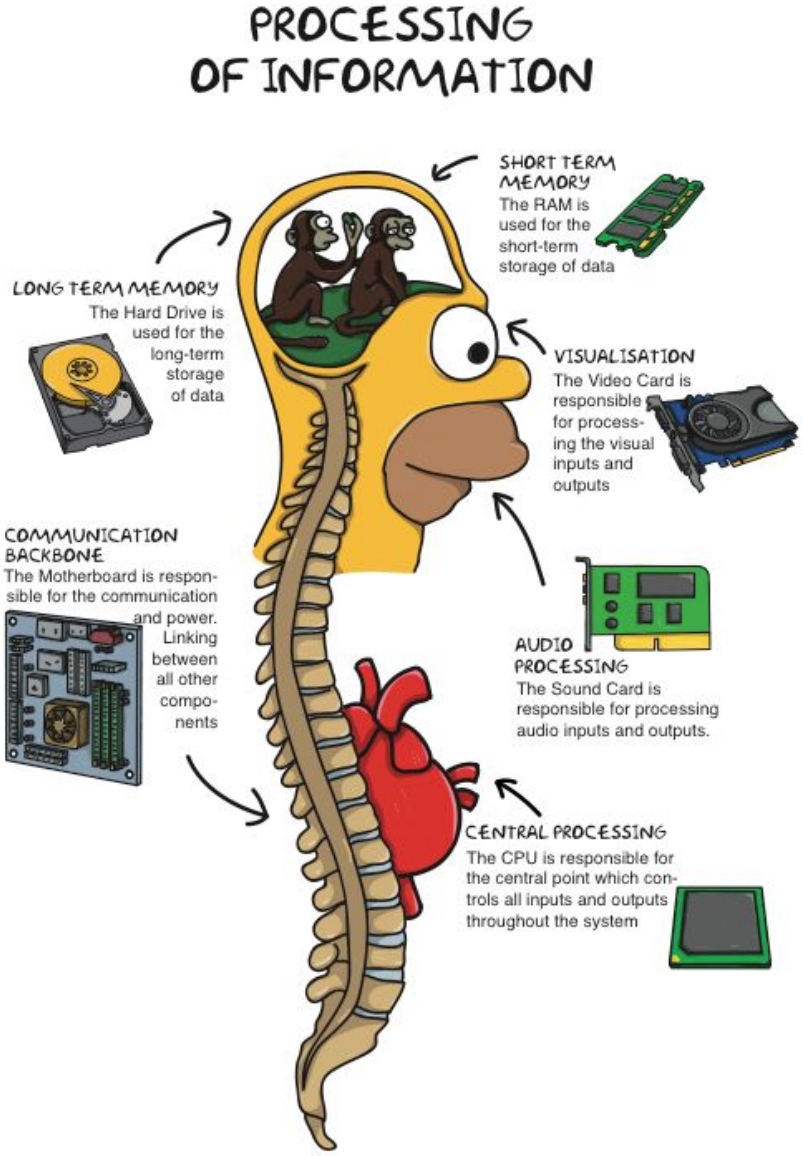
von

Daten

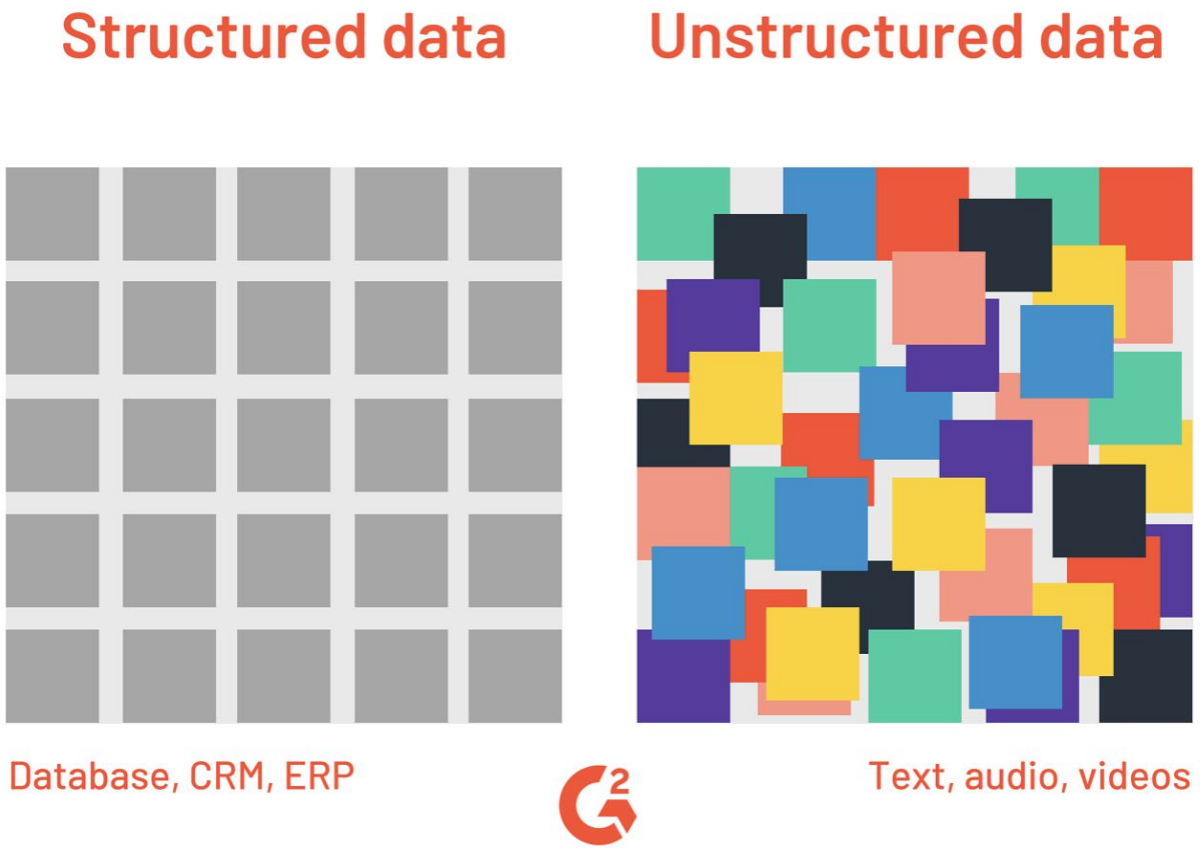
Daten =
Digitale Repräsentation von Information

- Strukturierte
- Semistrukturierte
- Unstrukturierte

- **Datenverarbeitung?**



Daten



Strukturierte Daten sind so organisiert und formatiert, dass sie in relationalen Datenbanken leicht durchsucht werden können

Unstrukturierte Daten haben kein vordefiniertes Format oder keine vordefinierte Organisation

Was ist Informatik?

Informatik

ist die **Wissenschaft**

von

- der systematischen Verarbeitung und Speicherung von Informationen,
- besonders der automatischen Verarbeitung mit Hilfe von Computern

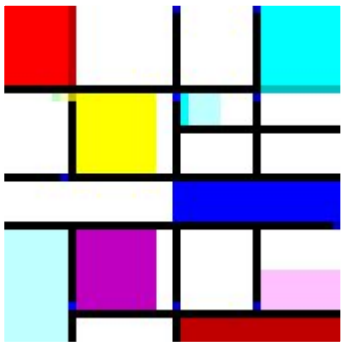

var b = count_
("#limit_val").val()), f
d < f && (f = d,
for (var
for (g = 0;g < c.length;g++) {
b.splice(e, 1);
for (c
b.splice(e, 1);
use_class ? $("#word-list-
'<span class="count-
title="Top keywords"
for (
}
for (
push(d.word),


python

Python?

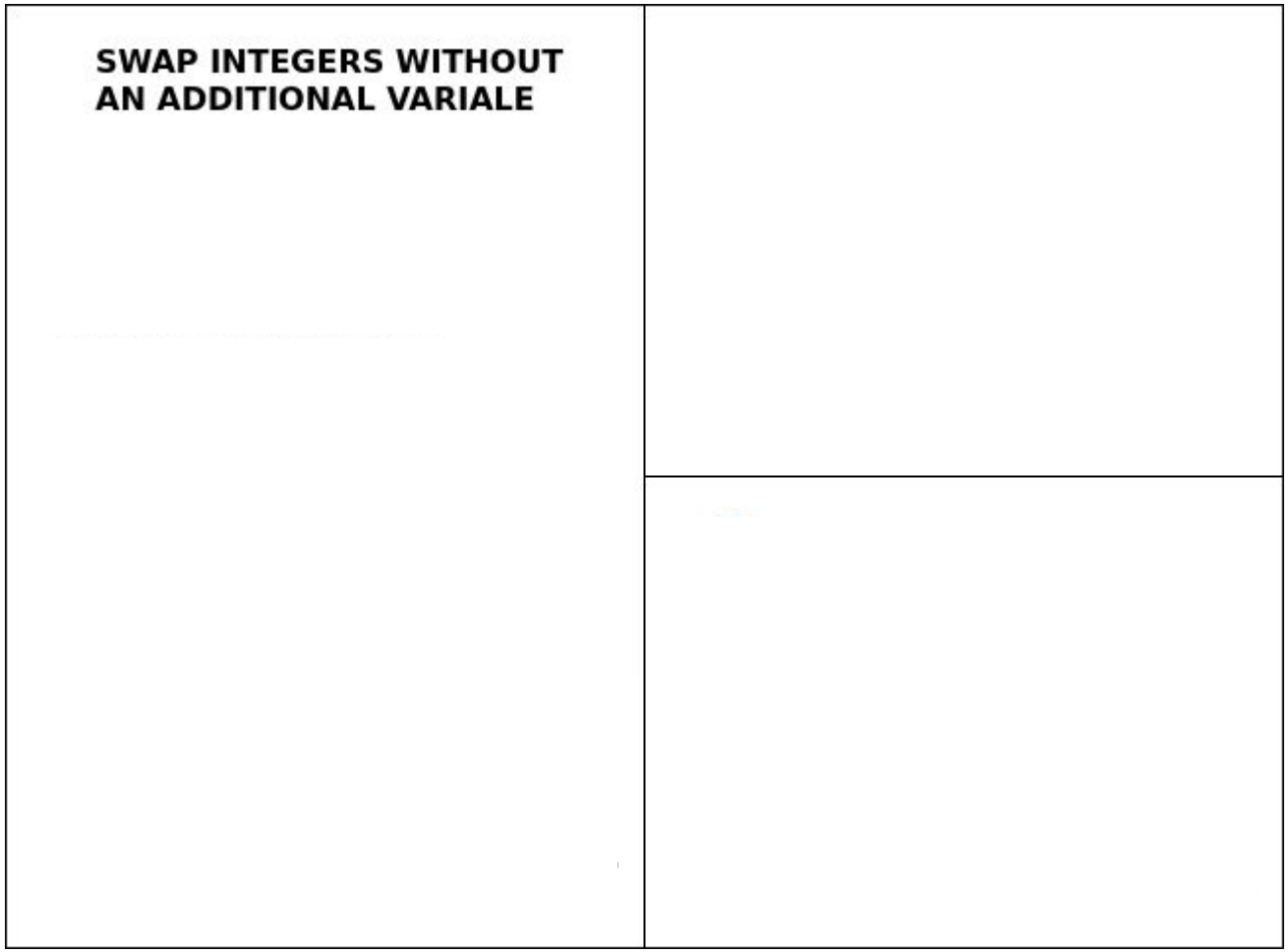
```
🛁 🔡 🍇
  🐖 🐻 → 🔡 🍇
    🍫 firstLetter 🔪 🦮 0 1
    🍫 rest 🔪 🦮 1 🐤 🦮
    🍎 ✨ rest firstLetter "ay" ✨
  🍉
🍉

🏁 🍇
  😀 🐻 "cat"
  😀 🐻 "development"
  😀 🐻 "computer"
🍉
```

```
IT'S SHOWTIME
TALK TO THE HAND "Hello, World!"
YOU HAVE BEEN TERMINATED
```



```
++++++++++[>+++++++>++++++++++>+++<<<-]>++.>+.+++++++
..+++.>++.<<+++++++++++++++.>.+++.------.--------.>+.
```

Python?



Take control of your job search.

Stack Overflow Jobs puts developers first. No recruiter spam or fake job listings.

Browse jobs

Find your next developer.

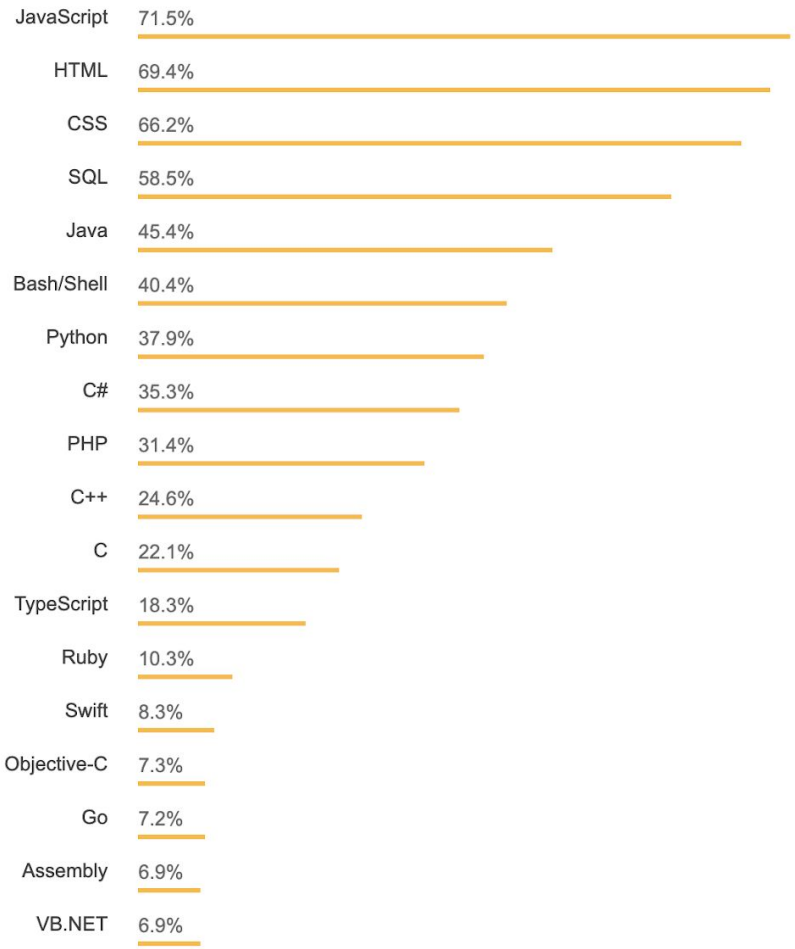
Source, attract and recruit developers on the platform they trust most.

Learn more

Most Popular Technologies

Programming, Scripting, and Markup Languages

All Respondents | Professional Developers



Python

eine Programmiersprache, welche

- Einsteigerfreundlich und leicht zu lernen ist
- Viele Möglichkeiten bietet ohne unübersichtlich zu werden
- Mehr als ein Programmierparadigma unterstützt
- Mit wenigen Keywords auskommt

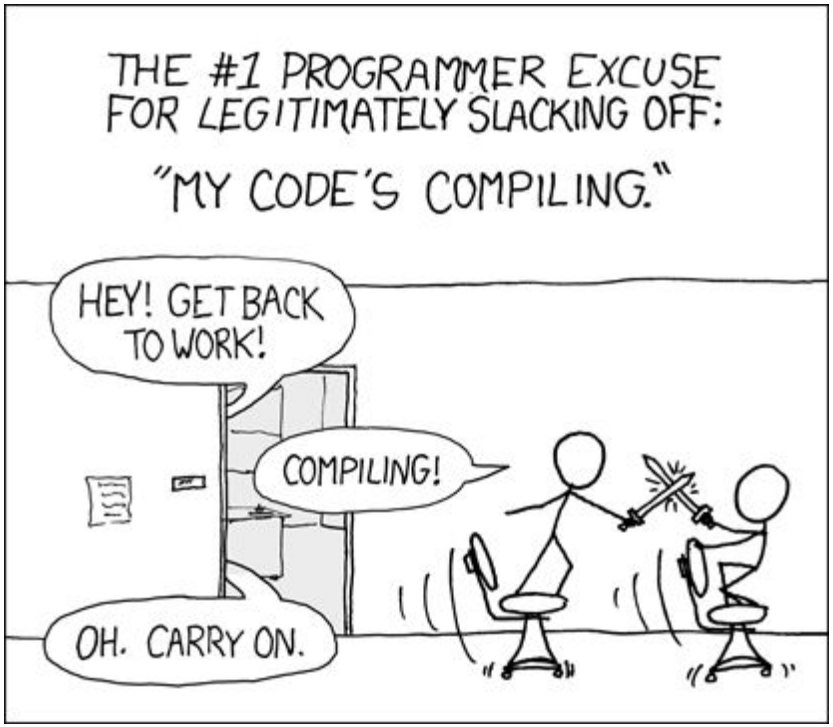
Python ist leicht zu lernen

- ist meist wohl strukturiert
- intuitiv
- gut lesbar

```
127     found_pub=crossref_lookup.lookup(xml)
128     if len (found_pub):
129         most_probable = max(found_pub,key=lambda item:item[2])[1]
130 
131     return most_probable
132 
133 def main ():
134     random.seed(64)
135     prepare_data ()
136 
137     pop = toolbox.population(n=50)
138     CXPB, MUTPB, NGEN = 0.5, 0.2, 20
139 
140     print("Start of evolution")
141 
142     # Evaluate the entire population
143     fitnesses = list(map(toolbox.evaluate, pop))
144     for ind, fit in zip(pop, fitnesses):
145         ind.fitness.values = fit
146 
```

Python ist eine Interpretersprache

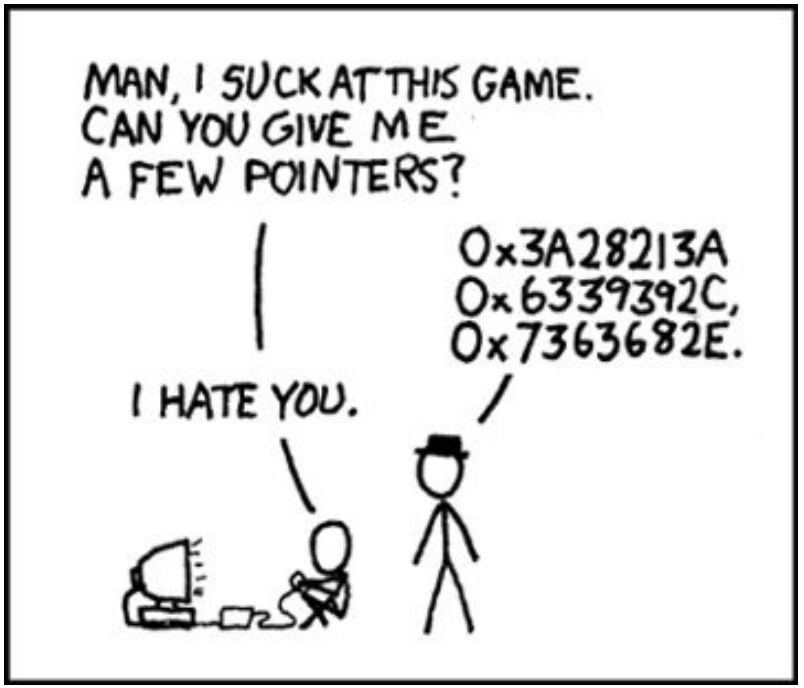
- mit interaktiver Shell
- erzeugt Python-Bytecode
- nutzt Stackbasierte VM
- gut dokumentiert!



http://xkcd.com/303/

Python ist eine moderne Sprache

- Objektorientiert
- Skalierbar
- OS unabhängig
- Reich an Libraries
- Erweiterbar



http://xkcd.com/138/

Zen of Python - Ein Mantra

- ***Beautiful*** is better than ugly
- ***Explicit*** is better than implicit
- ***Simple*** is better than complex
- ***Flat*** is better than nested
- ***Sparse*** is better than dense
- ***Readability*** counts
- ...

Python Grundlagen

- man kann Python-Programme interaktiv eintippen (mit der interaktiven Shell)

```
>>> print ("Hallo Welt!")
Hallo Welt!
>>>
```

```
>>> a = int(input ("a: "))
a: 4
>>> b = int(input ("b: "))
b: 6
>>> c = a + b
>>> print (c)
10
>>>
```

Eigene Syntaxelemente

- **Kommentare:** beginnen mit einem Doppelkreuz-Zeichen **#**
- **Name:** erlaubt sind die Buchstaben A - Z und a - z, die Zahlen 0 - 9, sowie der Unterstrich "_"
- **Literale:** direkte Darstellung der Werte von Basistypen

```
>>> STRING = "# Dies ist kein Kommentar."
```

Grund-Datentypen

- **Integer**
  - `>>> type(1)`
  - `<type 'int'>`
- **(sehr) lange Integer**
  - `>>> type(1L)`
  - `<type 'long'>`
- **Gleitkommazahlen**
  - `>>> type(1.0)`
  - `<type 'float'>`
- **Komplexe Zahlen**
  - `>>> type(1 + 2j)`
  - `<type 'complex'>`

- **Standardoperationen**
  - Addition `+`
  - Subtraction `-`
  - Division `/`
  - Integerdivision `//`
  - Multiplikation `*`
  - Expotentieren `**`
  - Modulo `%`
- **Built-in Funktionen**
  - `round`, `pow`, etc.

Numerische Operationen

Operation	Abkürzung
`x = x + y`	`x += y`
`x = x - y`	`x -= y`
`x = x * y`	`x *= y`
`x = x / y`	`x /= y`
`x = x % y`	`x %= y`
`x = x**y`	`x **= y`
`x = x//y`	`x //= y`

Vergleichsoperation

- `x == y`
- `x != y`
- `x < y`
- `x <= y`
- `x > y`
- `x >= y`

Wahrheitswerte

- bool ist der Typ der Wahrheitswerte `True` und `False`

- Operationen:
  - not a (Negation)
  - a and b (Konjunktion)
  - a or b (Disjunktion)

Ausdrücke und Variablen

Variable: abstrakter Behälter für eine Größe, welche im Verlauf eines Rechenprozesses auftritt

- Name
- Adresse
- in C++: int x;

Python stellt keine Variablen bereit.

Ausdruck: eine Kombination von Operanden (Werten, Variablen) und Operatoren.

```
>>> 2*3-4
2
>>> 2*(3-4)
-2
```

Anweisungen

Programm: eine Abfolge von Anweisungen. Ein Programm ist dabei aus Anweisungsblöcken aufgebaut

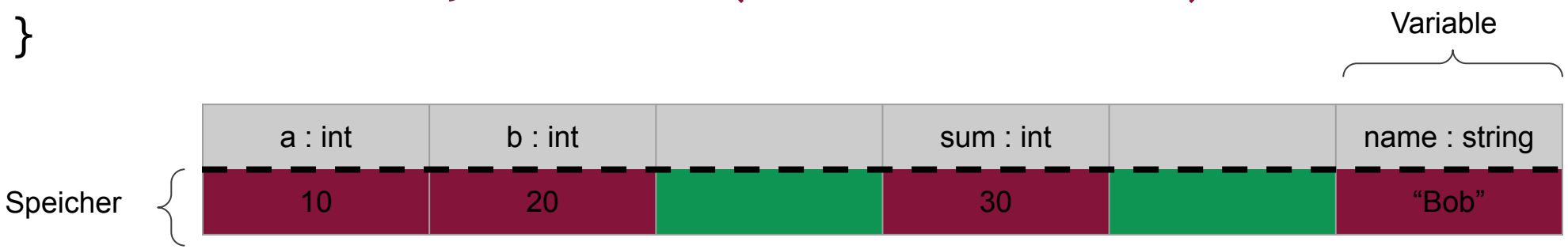
Zuweisung: die Verbindung zwischen einem Namen und dem Wert

```
>>> x = 1
>>>
>>> x = x + 2
>>>
```

- x ist der Name
- 1 ist ein Objekt vom Typ-Int
- x ist mit einem NEUEN Objekt verbunden, dessen Wert x + 2 ist

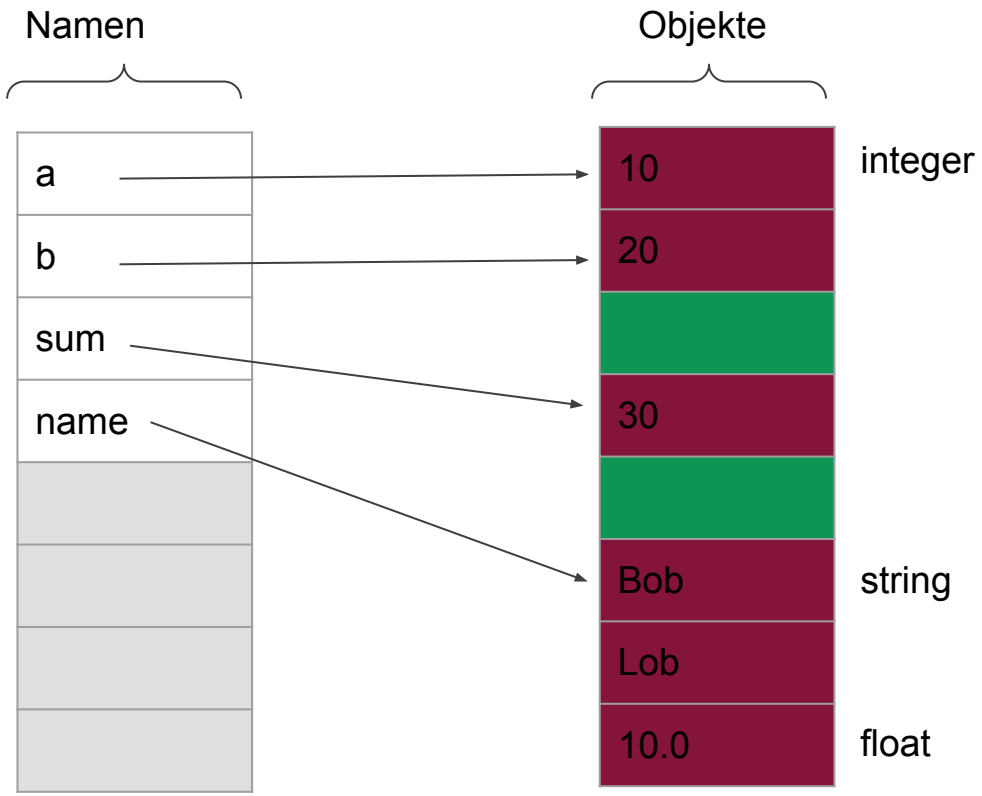
Python und Variablen - C++ Beispiel

```
int main () {
    int a = 10;
    int b = 20;

    int sum = b + a;

    string name = “Bob”;

    std::cout << name;

    a = “Dob” //Fehler (a ist int)
    float a = 10.0; //Fehler (a existiert schon)
}
```



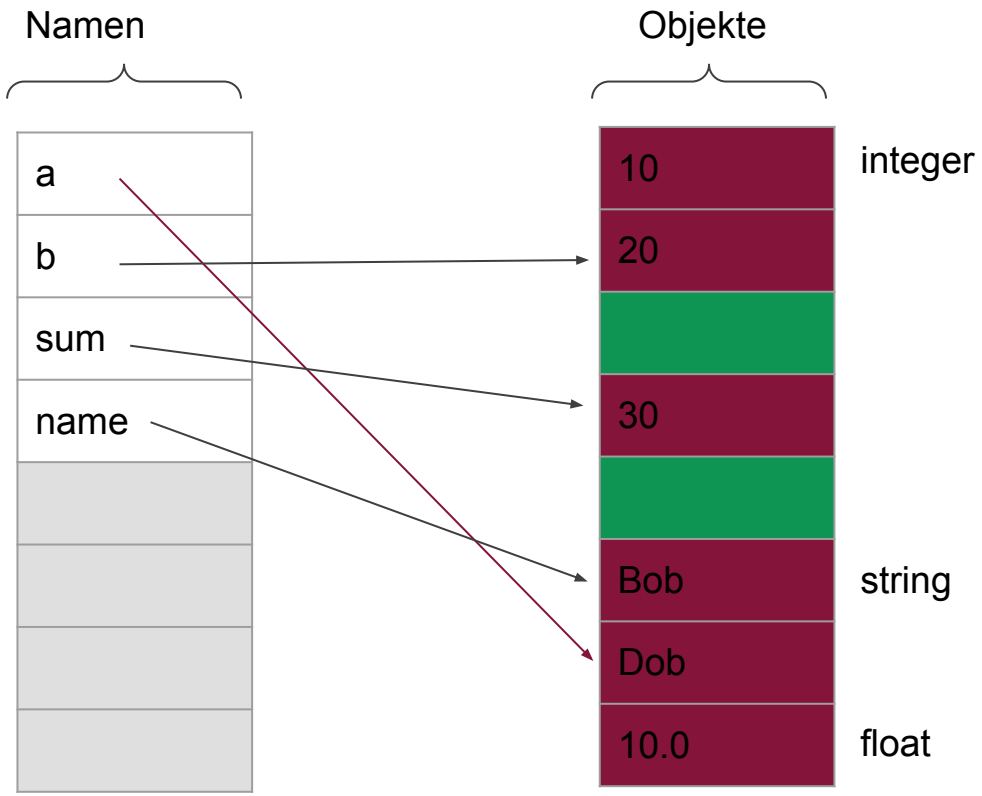
Python und Variablen

```
a = 10
b = 20

sum = b + a

name = “Bob”

print (name)

a = “Dob” #OK
a = 10.0  #OK
```



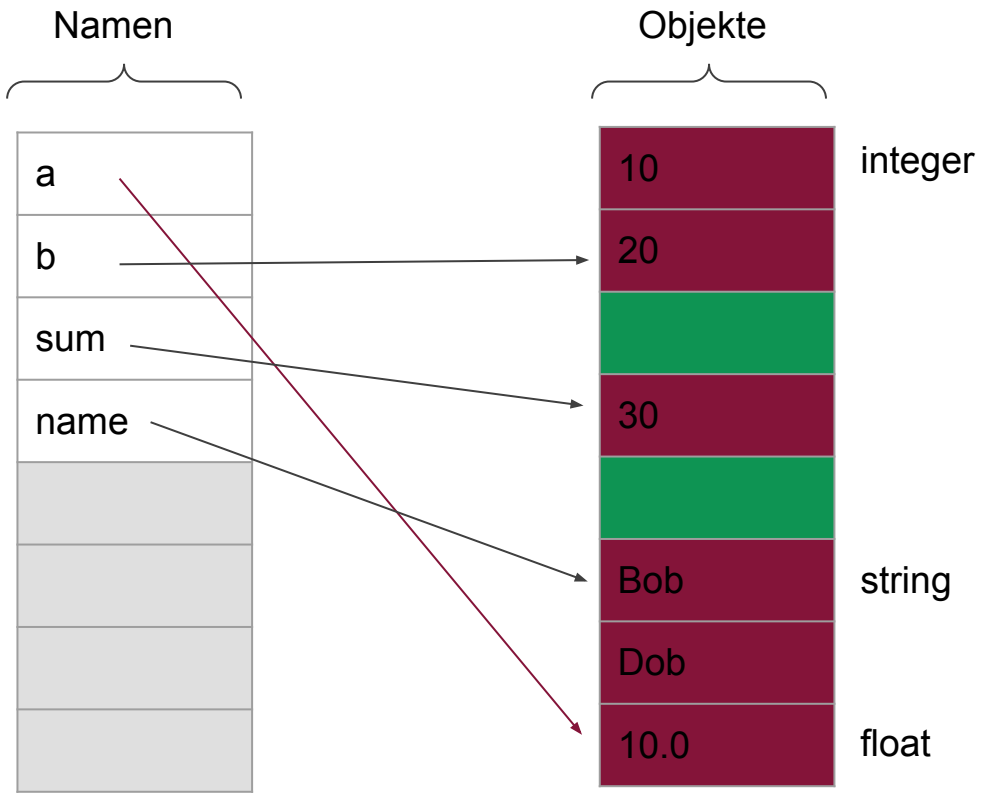
Python und Variablen

```
a = 10
b = 20

sum = b + a

name = “Bob”

print (name)

a = “Dob” #OK
a = 10.0  #OK
```



Python und Variablen

```
a = 10
b = 20

sum = b + a

name = “Bob”

print (name)

a = “Dob” #OK
a = 10.0  #OK
```



If - anweisung

```
if expression1:
   anweisung1
[elif expression2: anweisung2]
   ...
[else: anweisung]
```

- Die Ausdrücke `expression1, expression2, ...` werden in angegebener Reihenfolge ausgewertet
- bis einer zutrifft
  - Dann wird die entsprechende Anweisung ausgeführt.
- Wenn keiner der Ausdrücke zutrifft, wird die else-Anweisung ausgeführt.

Einrückungen und Blöcke

- Leerzeichen sind wichtig:
  - die Anweisungen in einer if-Anweisung müssen eingerückt werden
- d.h. die Anweisungen sind zu einem Block gruppiert
- Anweisungen des gleichen Blocks müssen mit der gleichen Anzahl des gleichen Typs Leerzeichen eingerückt sein
- Leerzeichen:
  - Space, Tabulator

```
a = 10
if a == 10:
    b = 20
    print (a+b)
c = 30

print(a)
print(b) #warum funktioniert?
```

Block

Block

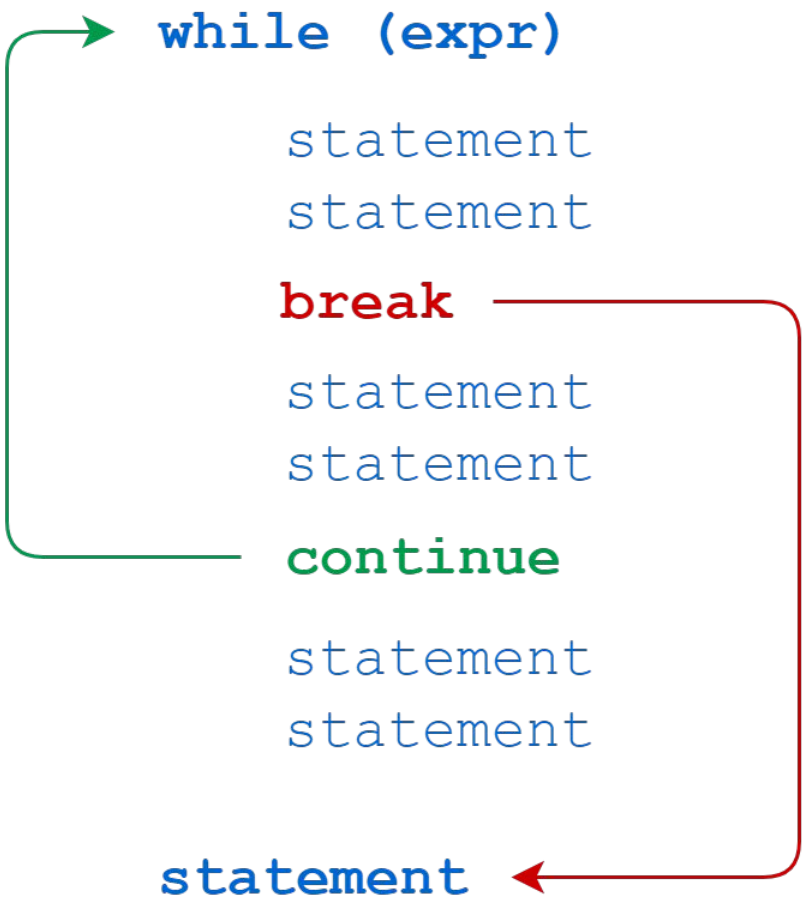
While - Schleife

```
while expression:
    block
```

1. Der Ausdruck `expression` wird ausgewertet
2. Trifft er zu, wird `block` ausgeführt
3. Danach `expression ist wieder ausgewertet (Schritt 1)`

break & continue

- Die break-Anweisung
  - verlässt die aktuelle Schleife
  - `expr` (die Bedingung) wird nicht ausgewertet
- Die continue-Anweisung
  - überspringt den Rest des Blocks
  - wertet `expr` neu aus
  - und setzt ggf. die Schleife fort



Sequenzielle Datentypen

Zur Kategorie der sequenziellen Datentypen gehören

- **str** und **unicode** für die Verarbeitung von Zeichenketten
- **list** und **tuple** für die Speicherung beliebiger Instanzen
  - eine **list** nach ihrer Erzeugung verändert werden kann (mutable)
  - ein **tuple** ist nach der Erzeugung nicht mehr veränderbar (immutable)
- **dict** für eine Zuordnung zwischen Objektpaaren
- **set** für ein ungeordneter Zusammenschluss von Elementen, wobei jedes Element nur einmal vorkommen kann

Strings

- `str1 = "abc"`
- `str2 = 'abc'`
- `str3 = """ abc """`
- `str4 = ("abc" "def")`

Escape-Sequenz

- `\a` erzeugt Signalton
- `\b` Backspace
- `\f` Seitenvorschub
- `\n` Linefeed
- `\r` Carriage Return
- `\t` horizonal Tab
- `\v` vertikal Tab
- `\" \' \\` Escaping `"'\`

```
Catalins-iMac:~ cat$ python3
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 26 2018, 23:26:24)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> name = "bob"
>>> type(name)
<class 'str'>
>>> print(name)
bob
>>> sname = name+"!!"
>>> len(sname)
5
>>> sname
'bob!!'
>>> name[0]
'b'
>>>
```

Formatierung mit Strings

Syntax

```
"...%n...%m..." % (Wert1, Wert2)
```

Beispiele

```
>>> a = 'H'
>>> b = 'ello World'
>>> "%c%s" % (a,b)
'Hello World'
```

Format

`d, i`	*Integer mit Vorzeichen*
`f`	*Float (Dezimaldarstellung)*
`g, G`	*Float (wiss. mit Exponent)*
`u`	*Integer ohne Vorzeichen*
`x`	*Hexzahl ohne Vorzeichen*
`o`	*Oktalzahl ohne Vorzeichen*
`e, E`	*Float (Exponentendarst.)*
`c`	*Zeichen (Länge 1)*
`s, r`	*String*
`%`	*Prozentzeichen*

List

Eine **list** kann Elemente unterschiedlichen Datentyps enthalten

- Syntax `[Wert_1, ..., Wert_n]`
- Eine Liste kann auch nach ihrer Erzeugung verändert werden
- Die Funktion **`len()`** bestimmt die Anzahl der Elemente der Liste
- Listen können auch Listen enthalten, auch sich selbst
- Hinzugefügt werden Werte mit dem **`+-Operator`** und den Funktionen **`append()`** und **`insert()`**
- Zugriff auf Elemente mit **`[]-Operator`**

Tuple

Im Gegensatz zu **Listen** sind **Tuple** immutable

- d.h. jede Änderung erzeugt ein neues Objekt
- Syntax `(Wert_1, ..., Wert_n)`
- Sie sind damit besonders geeignet, um Konstanten zu repräsentieren
- Ein **Tupel** wird in runde Klammern geschrieben (packing)
- **min()** bestimmt das Minimum eines Tupels, **max()** das Maximum
- Nesting
- unpacking

Tuple vs List

```
Catalins-iMac:~ cat$ python3
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 26 2018, 23:26:24)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> l = [1,2,3,4]
>>> l.append(10)
>>> l
[1, 2, 3, 4, 10]
>>> l[2] = 33
>>> l
[1, 2, 33, 4, 10]
>>> v = l + [101]
>>> v
[1, 2, 33, 4, 10, 101]
>>> a = l[2]
>>> a
33
>>> t = (1,2,3)
>>> t.append(10)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
AttributeError: 'tuple' object has no attribute 'append'
>>> t[0] = 101
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>>
```

Dictionary

Mit dict wird eine Zuordnung zwischen Objektpaaren hergestellt

- Syntax `{ Key_1: Value1, Key_2: Value2, ... }`
- müssen die Keys nicht ganze Zahlen (aber Liste?)
- Dictionaries sind iterierbare Objekte
- Die Länge eines Dictionaries `d` kann über `len(d)` abgefragt werden
- Mit del `d[k]` wird das Element mit Schlüssel `k` gelöscht
- mit `k in d` kann geprüft werden, ob sich der Schlüssel `k` in `d` befindet

Dictionary

```
Catalins-iMac:~ cat$ python3
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 26 2018, 23:26:24)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> d = {}
>>> d['a'] = 2
>>> d
{'a': 2}
>>> d['a']
2
>>> d.keys()
dict_keys(['a'])
>>> d.values()
dict_values([2])
>>> d['b'] = [1,2,3]
>>> d
{'a': 2, 'b': [1, 2, 3]}
>>> d['b'][1]
2
>>>
```

Set

Eine Menge ist ein ungeordneter Zusammenschluss von Elementen, wobei jedes Element nur einmal vorkommen kann

- Syntax `{Wert_1, ..., Wert_n}`
- gibt es für mutable Mengen den Typ **set**
- für immutable Mengen den Typ frozenset
- `len(m)` liefert die Anzahl der Elemente in m
- `x in m` ist True, wenn x in m enthalten ist
- `m<=t` ist True, wenn m eine Teilmenge von t ist

Set

```
Catalins-iMac:~ cat$ python3
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 26 2018, 23:26:24)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> s = {1,2,3}
>>> s[0]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'set' object does not support indexing
>>> l = [1,2,3,4,3
... ]
>>> l
[1, 2, 3, 4, 3]
>>> ns = set(l)
>>> ns
{1, 2, 3, 4}
>>> 2 in ns
True
>>> 5 in ns
False
>>>
```