

Python Lernreise

Tutorial #1

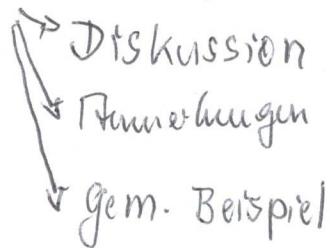
Kontakt@uc-it.de

"Kurs"

9⁰⁰ - 12⁰⁰ | 13⁰⁰ - 17⁰⁰

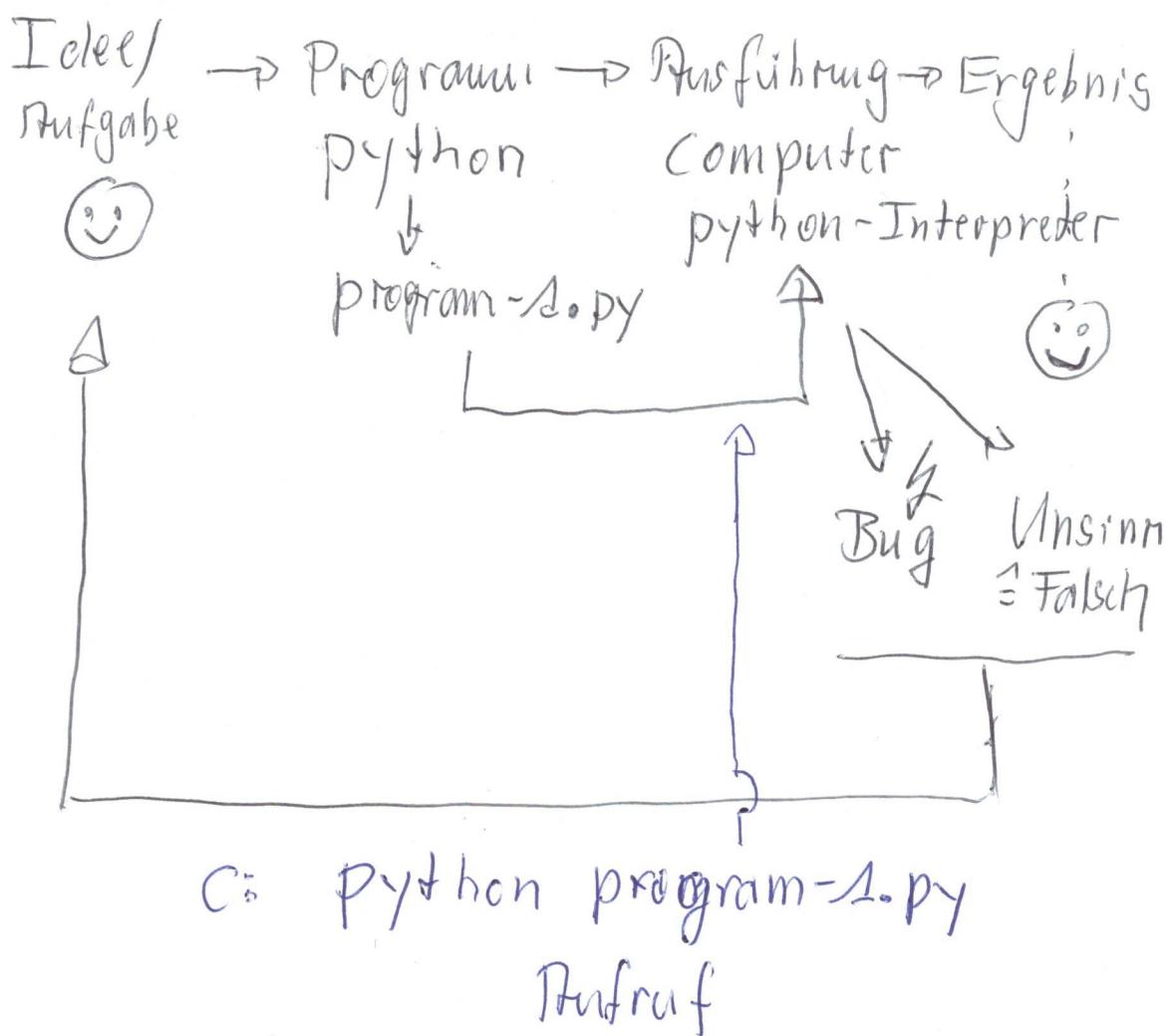
50 min - Einheit

Einheit → Themen

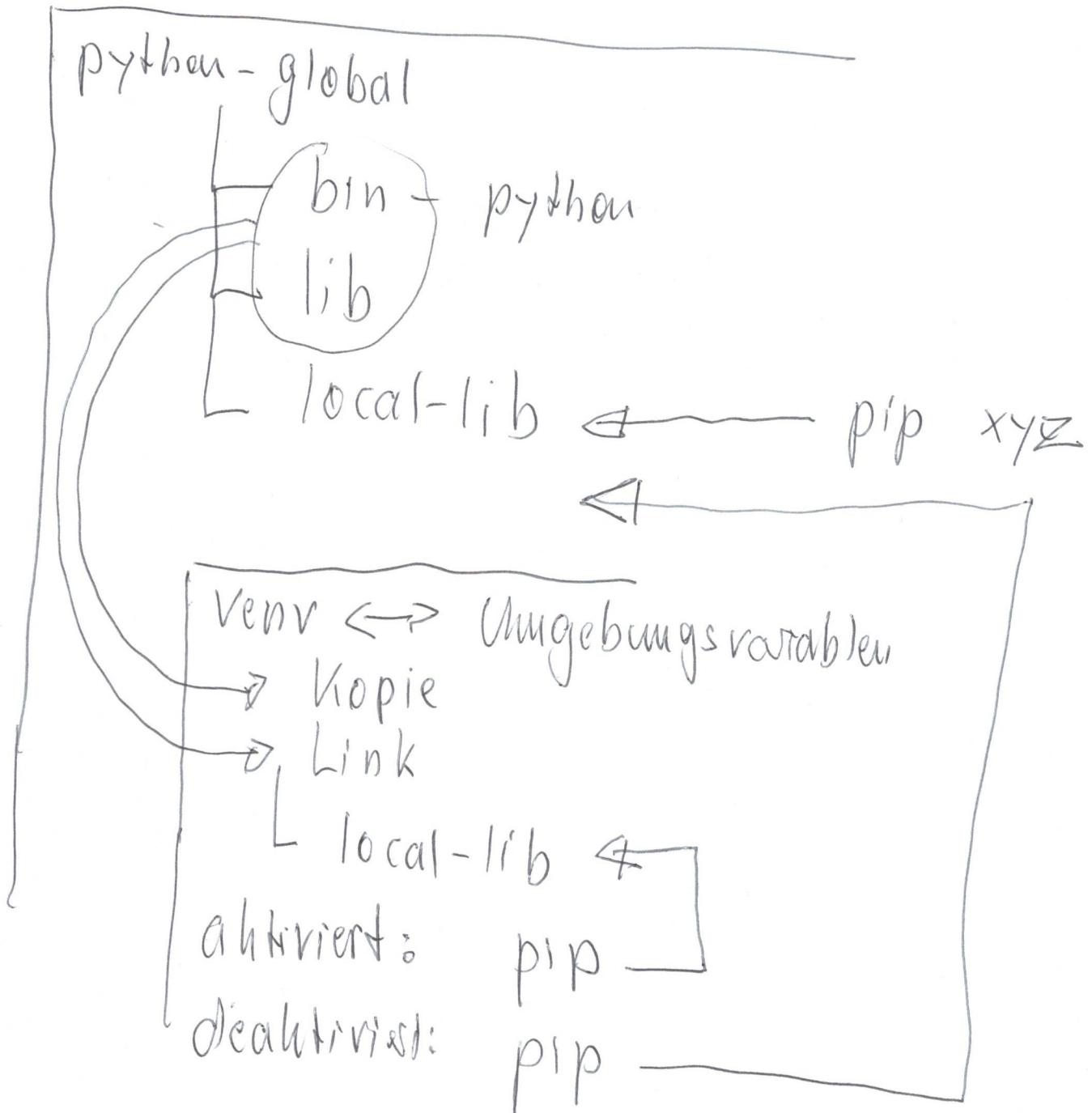


Python

- Programmiersprache



Venv



Datentypen

name → Objekt vom Typ x
↓
zergt

Namensvorlage
Konvention

Pep 8

x: String
Int
Float
Bool
Liste
Hash
Set

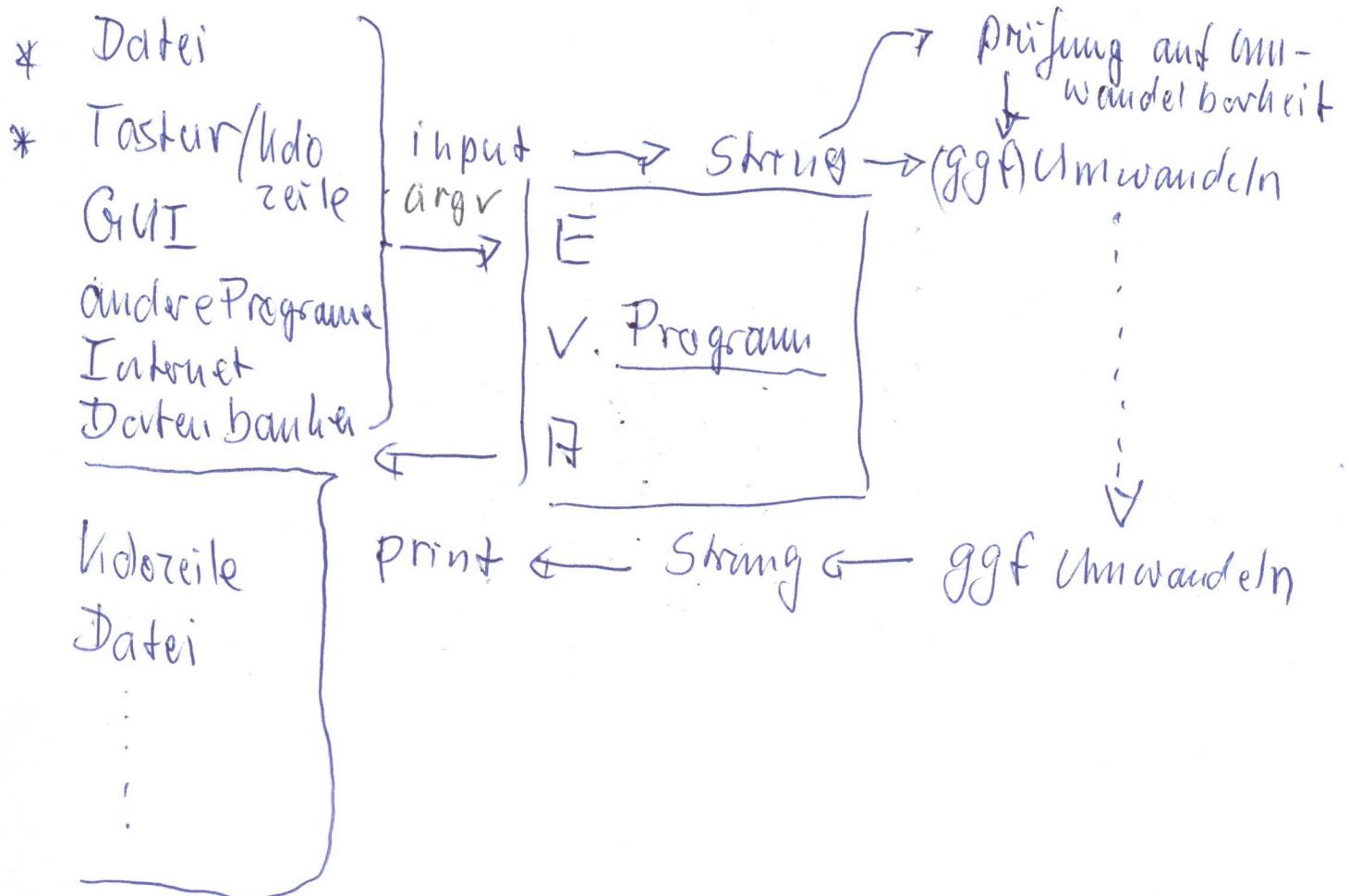
Klassenbasiert
Funktion

Veränderbar: Int, Float, Bool, Liste,
Hash, Set

Unveränderbar: String

Typstrenge: Gemischte Ausdrücke nicht erlaubt
jeder Typ/Objekt hat eigene
Methoden

Input



Fehlerbehandlung

„Shit happens“... aber nicht immer

- a) Teste auf alles was schiefgehen könnte
 - Aufwändig!!!
 - langsam
 - meist unnötig
 - es wird immer was übersehen!
- b) Schau' mir mal
 - Versuch (+try)
 - wenn klappt ✓
 - Wenn es failt → aufgemessenes
↓
Exceptions Stromkreis
Errors (except)

Kommandozeile

=

prog-1.py

`#!/usr/bin/env Python3`

`import sys`

`print(sys.argv)`

`[22, 23]`

`python prog-1.py 22 23`

`prog-1.py 22 23`

4

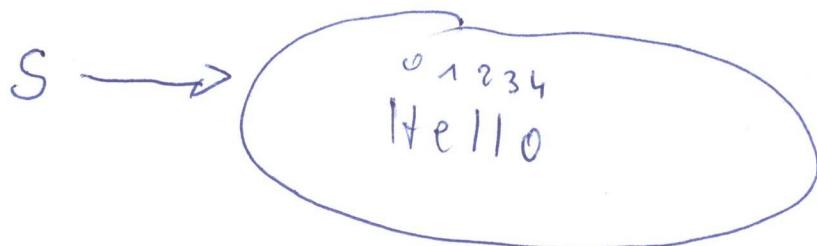
[

22, 23]

]

Strings

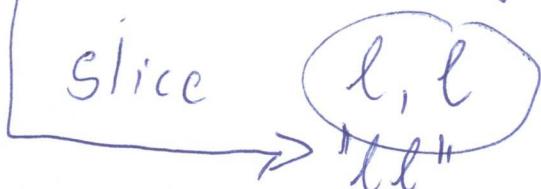
$S = "Hello"$



$S[0] \Rightarrow "H"$

$S[4] \Rightarrow S[\text{len}(S)-1] \Rightarrow S[-1] \Rightarrow "o"$

$S[2:4] \equiv [2, 3]_4$



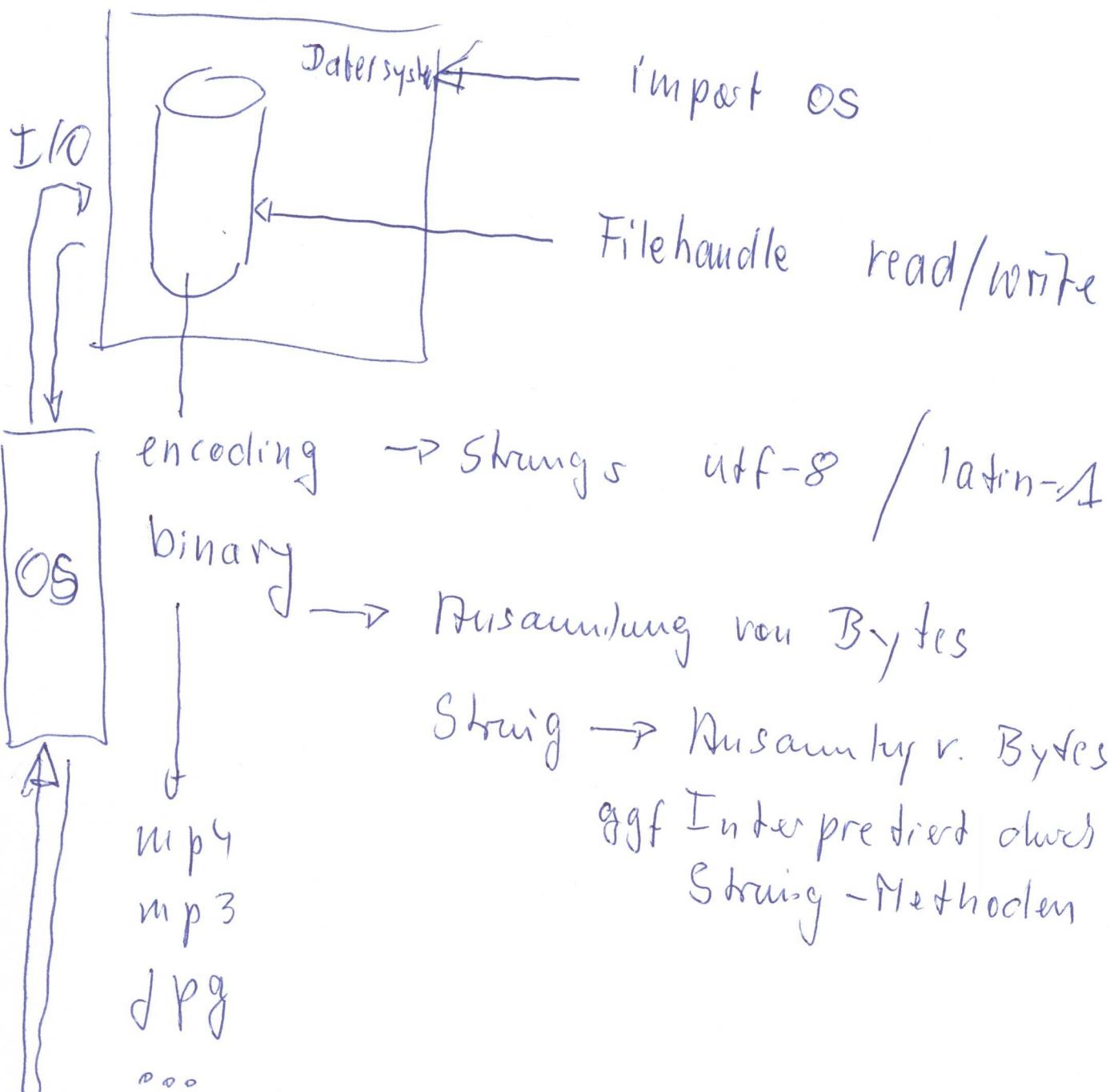
$S[0:] \rightarrow \text{hello}$

$S[1:] \rightarrow \text{ello}$

$S[:4] \rightarrow \text{hell}$

Bsp Start = 1 2 3
 0 1 2 3 4 5 [6:]

Datei



`fh = open → FileObject / FileHandle`



Datei lesen / Schreiben

fd = open(...)

inhalt = fd.read()

fd.close()



am Besten mit try ... finally

abgekürzt:

with open(...) as fcl:

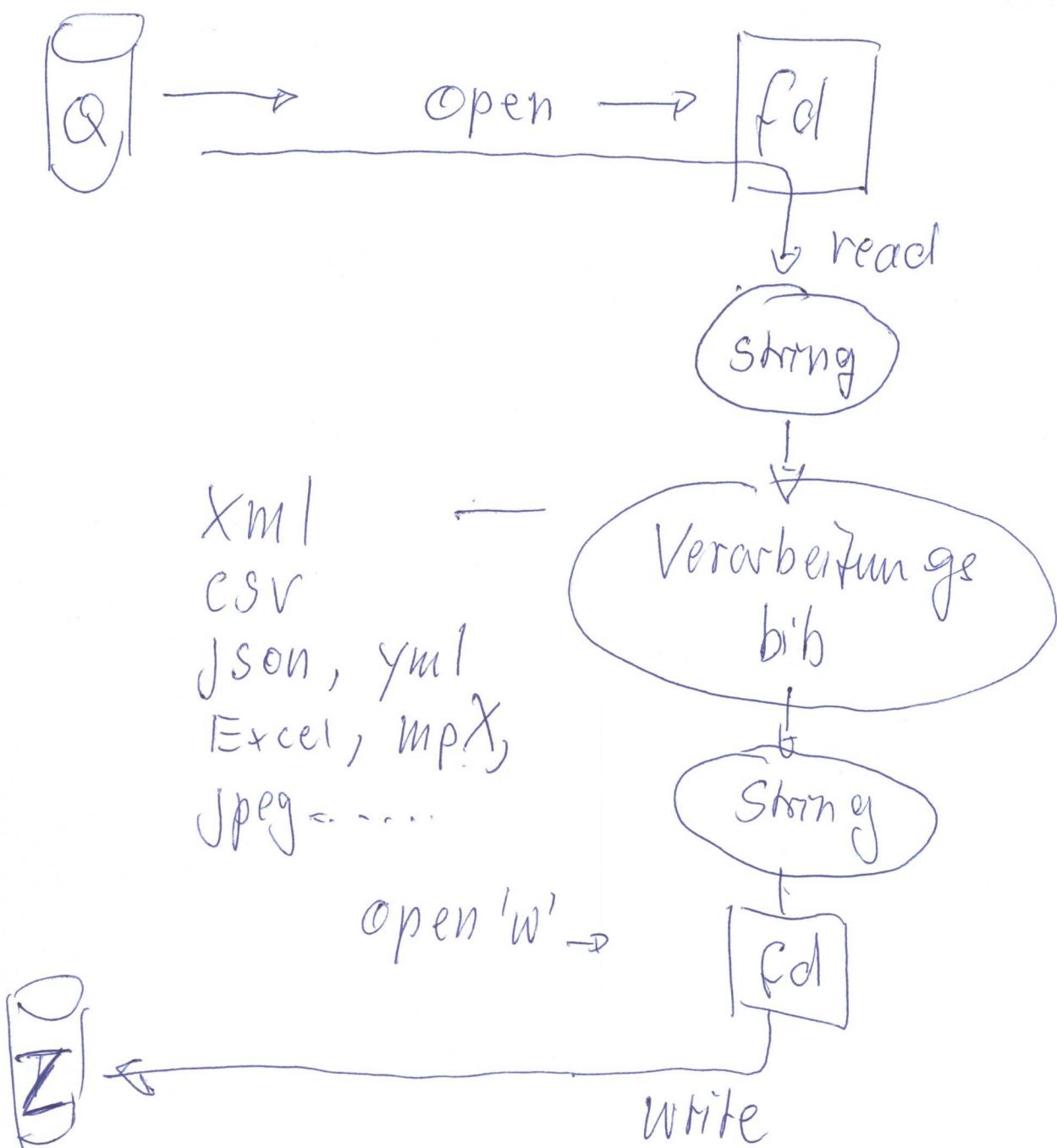
inhalt = fd.read()

umplatzit

finally

fd.close()

Dateiinhalt



String

$s \rightarrow \left\{ \begin{array}{c} "Hello, World" \\ | 01234567891011 \end{array} \right\}$ immutable

$\text{len}(s) \rightarrow 11$

Index $\rightarrow 0 - (\text{len}(s)-1)$

Slices: Teilstück des Strings
ist selbst ein String

Bsp $s[0] \rightarrow "H"$ kein Slice

$s[0:1] \rightarrow "H"$ Slice

$s[von:bis:schritt] \rightarrow "...."$

$s[-1] \rightarrow$ der Letzte $\rightarrow "d"$ kein Slice

$s[11:] \rightarrow "d"$ Slice

$s[:]$ full copy

$s[:3] \rightarrow "Hel"$

$s[9:] \rightarrow "rel"$

Liste

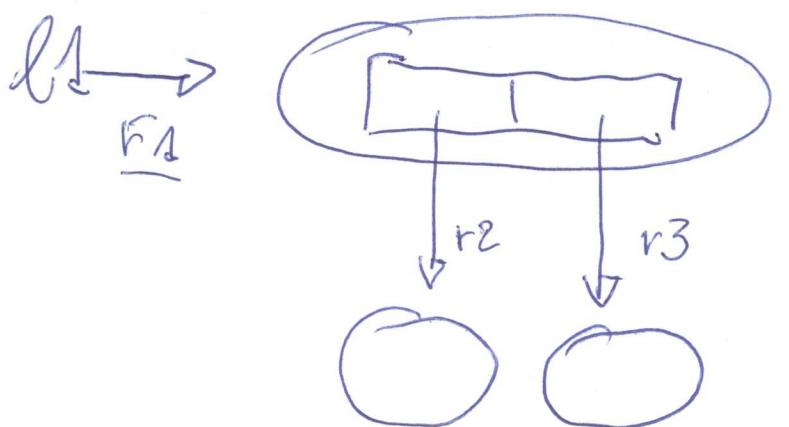
$l = []$ leere Liste

$l = \text{list}(\text{Daten})$ Daten werden „verlistet“

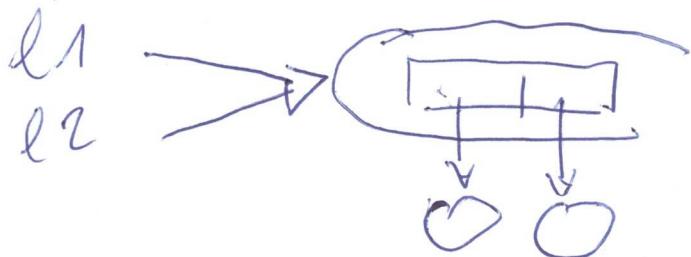
Index zugriff

Mutable

Slicing



$l_2 = l_1 \rightarrow r_1$ wird kopiert

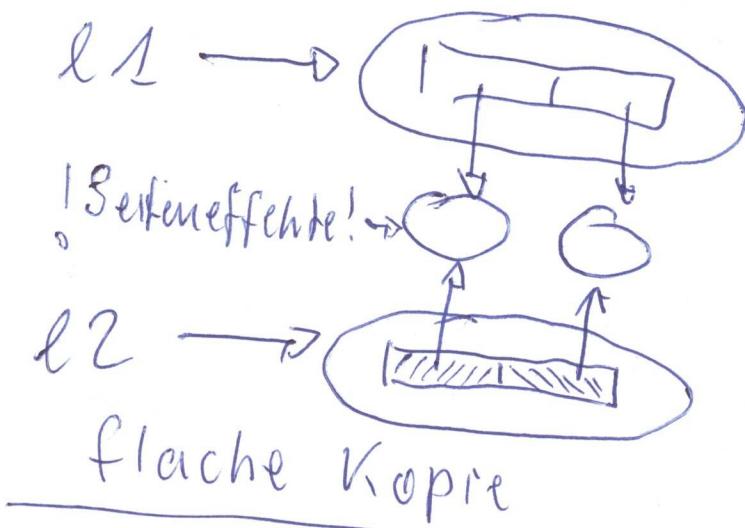


Liste

Kopie

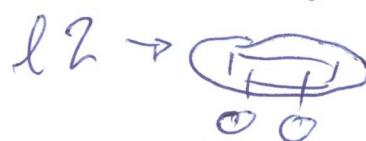
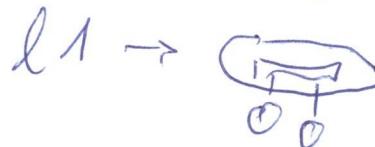
$$l2 = l1[:]$$

$$l2 = l1.\text{copy}()$$

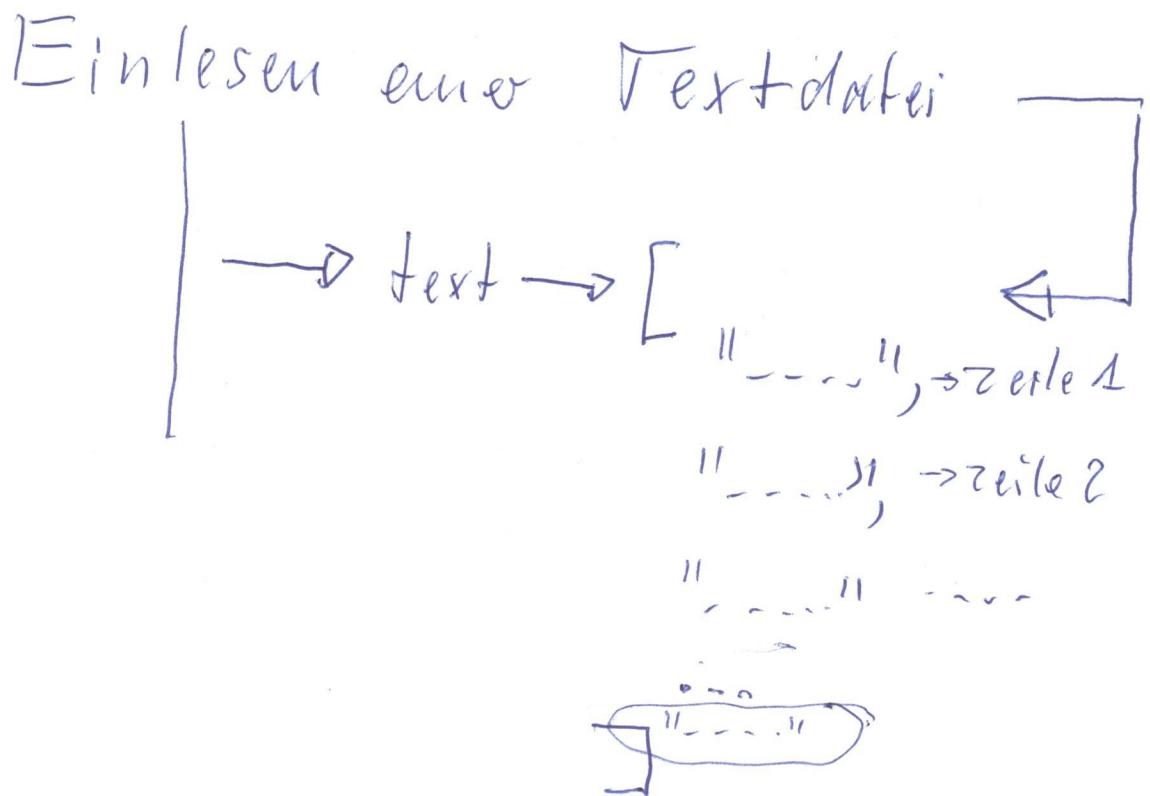


Impart copy

$$l2 = \text{copy.deepcopy}(l1)$$



Aufgabe



Ausgabe der letzten Zeile

Set / Tuple

Tuple: $\hat{=}$ immutable List

$$\begin{array}{l} l = [1, 2, 3] \\ l[0] = 5 \end{array} \quad \boxed{\checkmark}$$

$$\begin{array}{l} t = (1, 2, 3) \\ t[0] = 5 \end{array} \quad \boxed{\times}$$

$l.append(99) \rightarrow [5, 2, 3, 99]$ \checkmark

$t.append(99)$ $\boxed{\times}$ append gibt es nicht

Umwandlungen erlaubt

$$l = list(t)$$

$$t = tuple(l)$$

Set

"Menge": Elémente, non ordered,
unique, mutable

$m = \text{set}(\text{liste})$

$m = \{1, 2, 3\}$ ↗ unordn oder
dict auch $\{3\}$ nutzt

Mengenoperatoren als Symbol
bzw Methode

z.B. $s1.\text{union}(s2)$

$s1 | s2$

Dictionary

hash / assoz. array /

⇒ key-value (fun fact: viele NoSQL DB sind das auch)

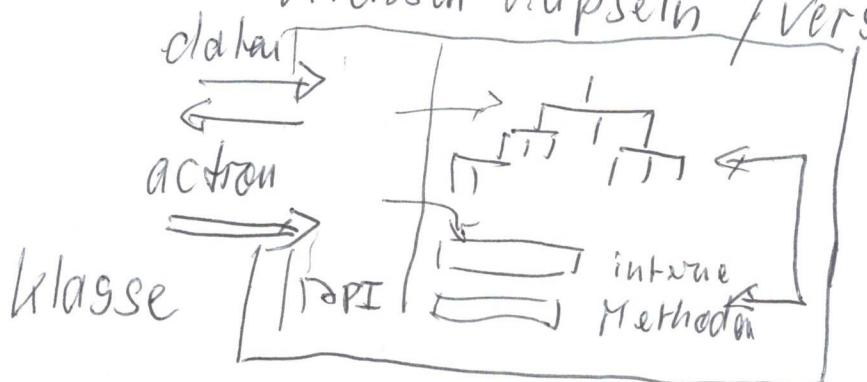
String list, dict, set, tuple, None,
Basis Typen, bel. andere Objekt

z.B. → CSV-Verwaltung

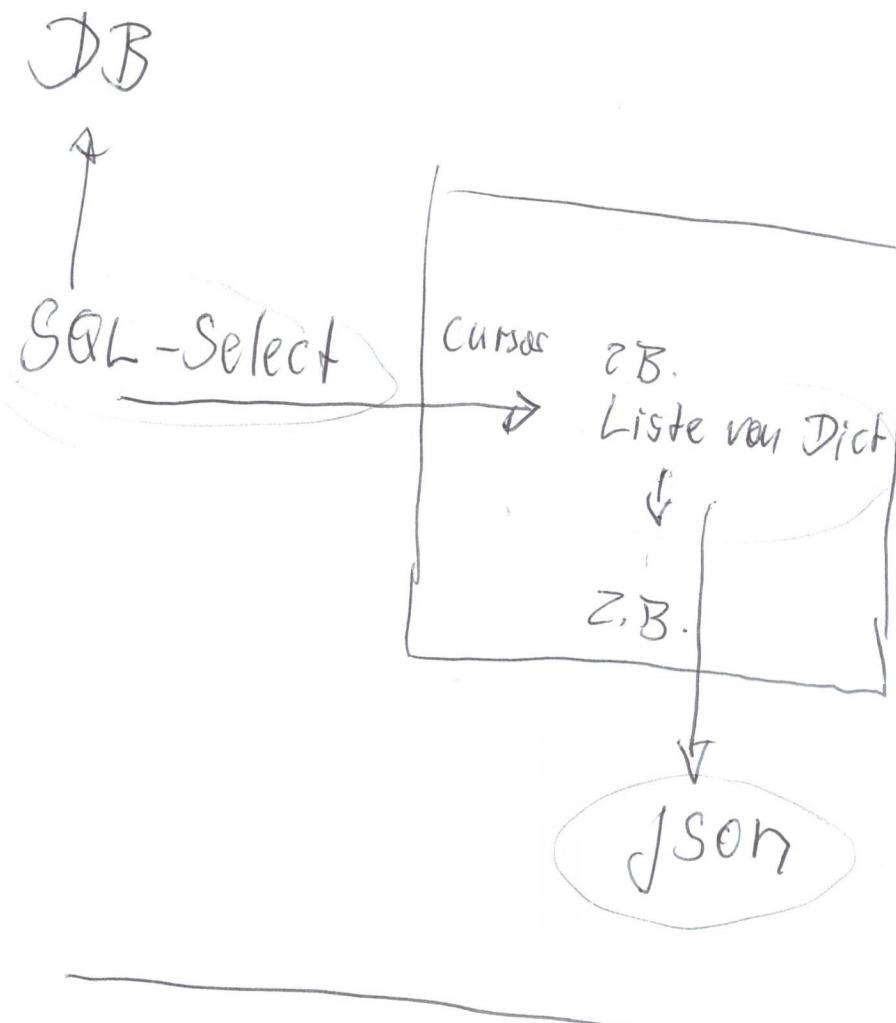
Bäume / Trees / Tabellen

Komplexe Strukturen

→ Achtung: Kopieren kann zur Komplexe Datenstrukturen ~~Qual~~ werden mit Klassen kapseln / verstecken



Dictionary



Dictionary

$d = \{ \text{key1: value}, \text{key2: value}, \dots \}$

$d = \text{dict}([(k_1, v), (k_2, v), (k_3, v)])$

$d[k]$ Zugriff,

Wenn k existiert ✓

Wenn nicht ↗ Exception

$d.get(k, default)$

k existent ✓

Wenn nicht → default

$d[k] = \text{value}$

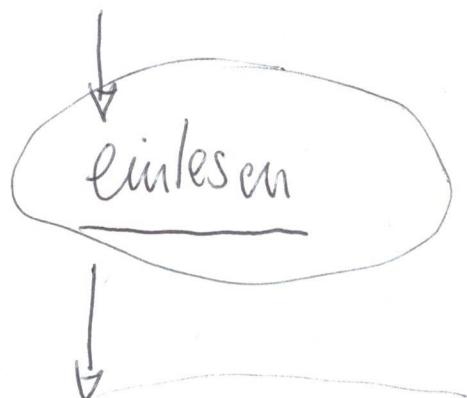
Wenn k existiert - überschreiben

Wenn nicht - Anlage

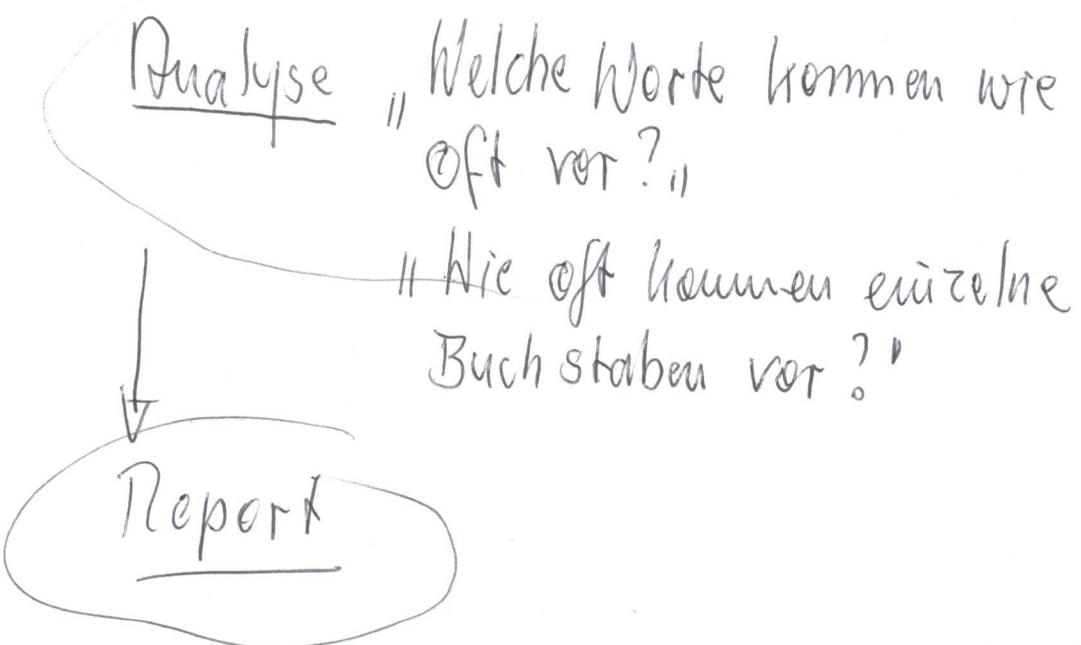
Aufgabe

Datei mit Text

E



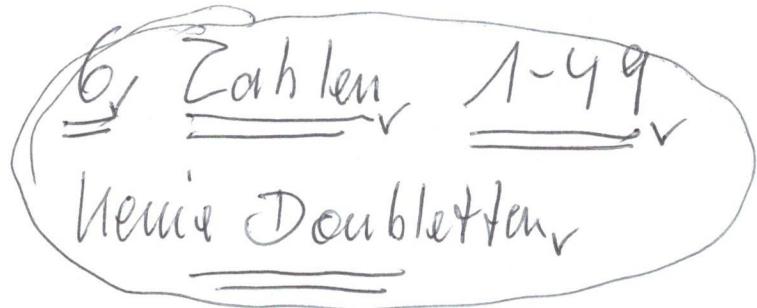
V



A

Aufgabe

Eingabe



wenn Zahlen ok

6 Zahlen zufällig erzeugen

Vergleichen

Bereiten