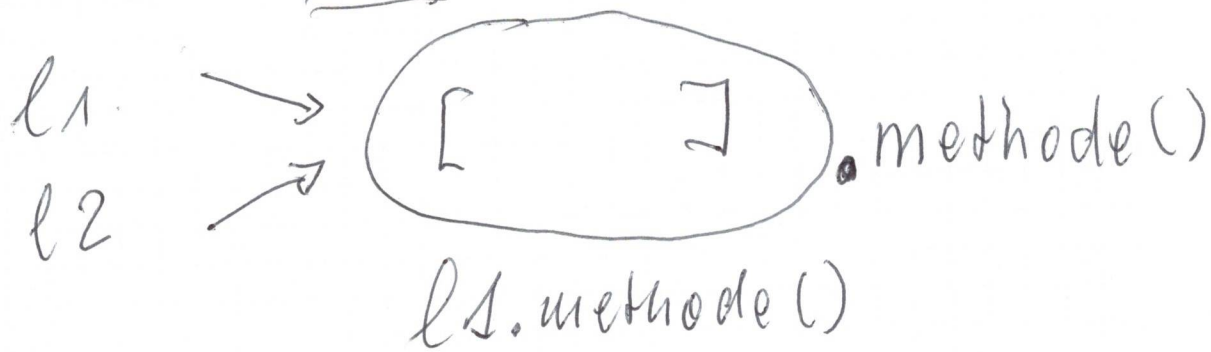
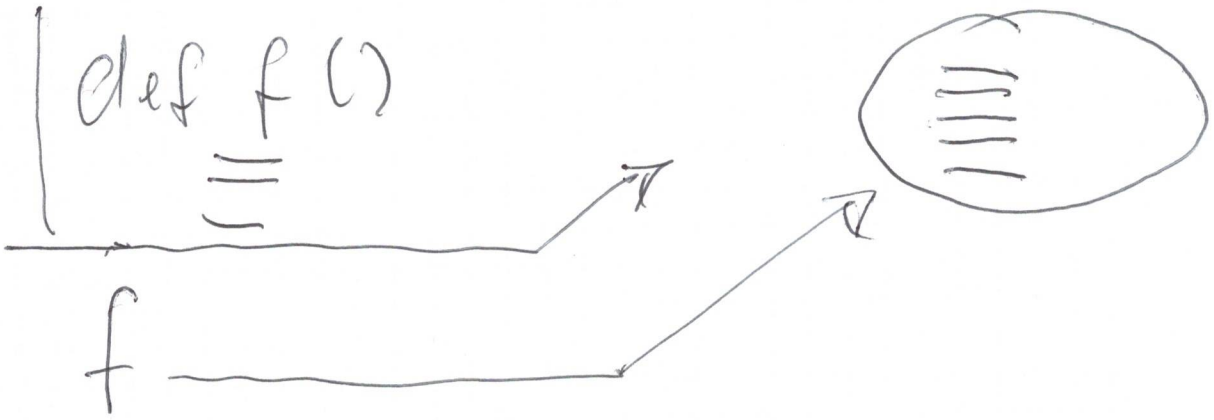
OOP - python



l1.methode()

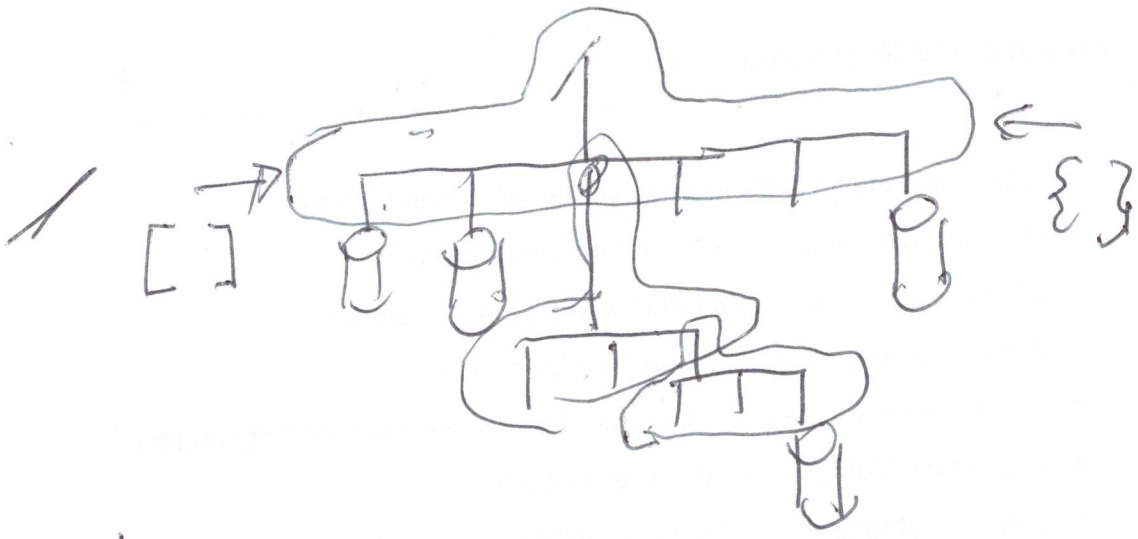


f

f()

l = f

l()



List

Liste ⟷ Tupel

`l = []`

`t = ()`

`l = list()`

`l = produziere_liste()`

veränderbar

nicht veränderbar

i = 0 1 2

`l = [1, 2, 3]`

`l[0] = 4711` ✓

`l[3] = ...` ↯

`l.append(...)`

`t = (1, 2, 3)`

`t[0] = 4711` ↯

`t[3] = ...` ↯

0 1 2 0~(n-1)

~~`l.append(None).append(None).append(...)`~~

3 4 5

l = [ ] ← append( )

l = list( )

---

l = [...] * n

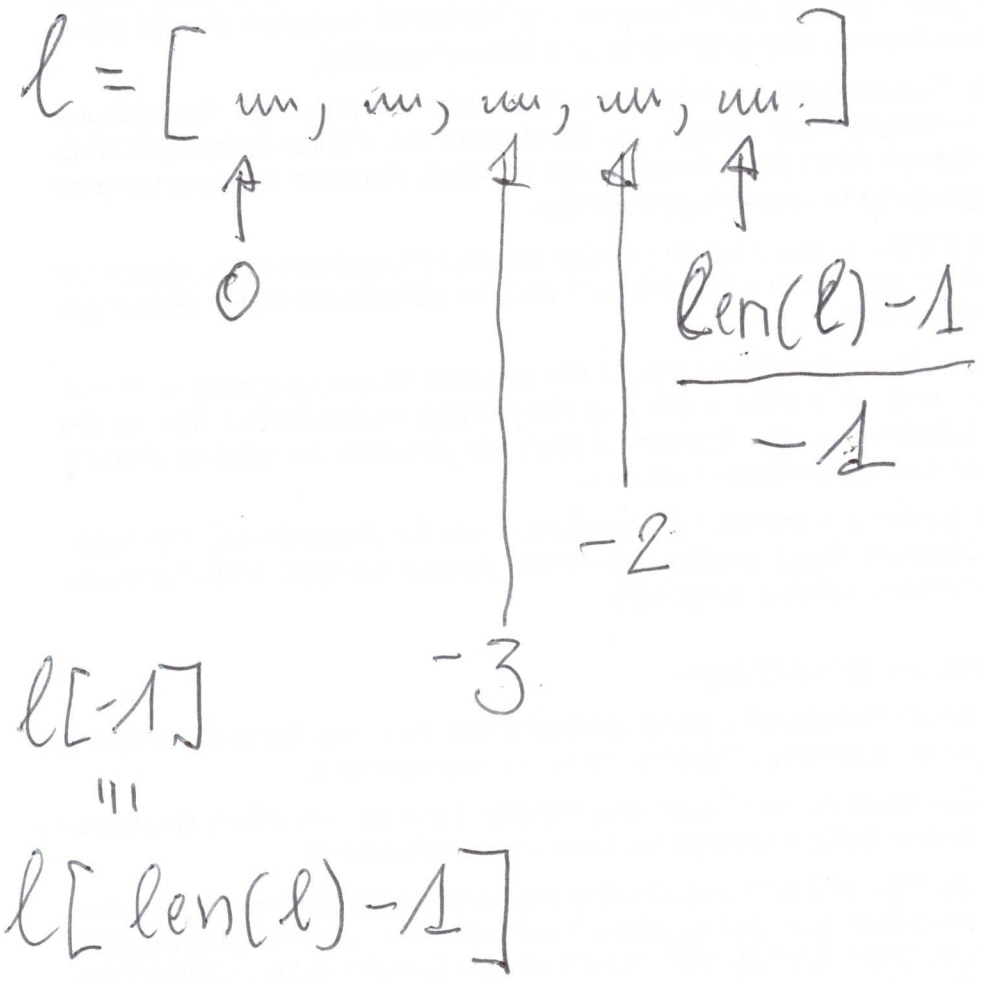
l = [None] * n

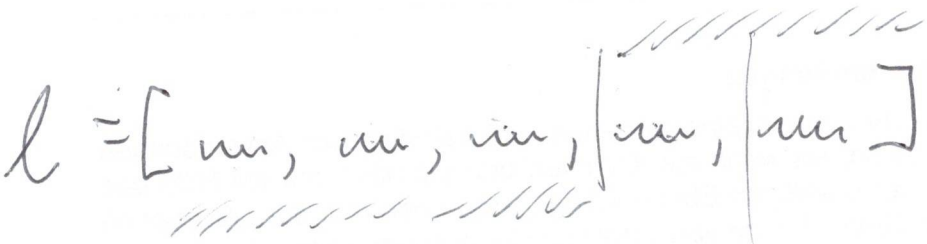
l = [ø] * n

l = list(range(1, 100))

---

list comprehension


l = [...]
0
len(l) - 1
-1
-2
-3
l[-1]
=
l[len(l) - 1]



l[von : bis : schritt] slicing

l[0 : 2] = l[0], l[1]

l[3 :]

l[: 3]

l[:] als Kopie

l1 = l → l1 → [...]
           l ↗

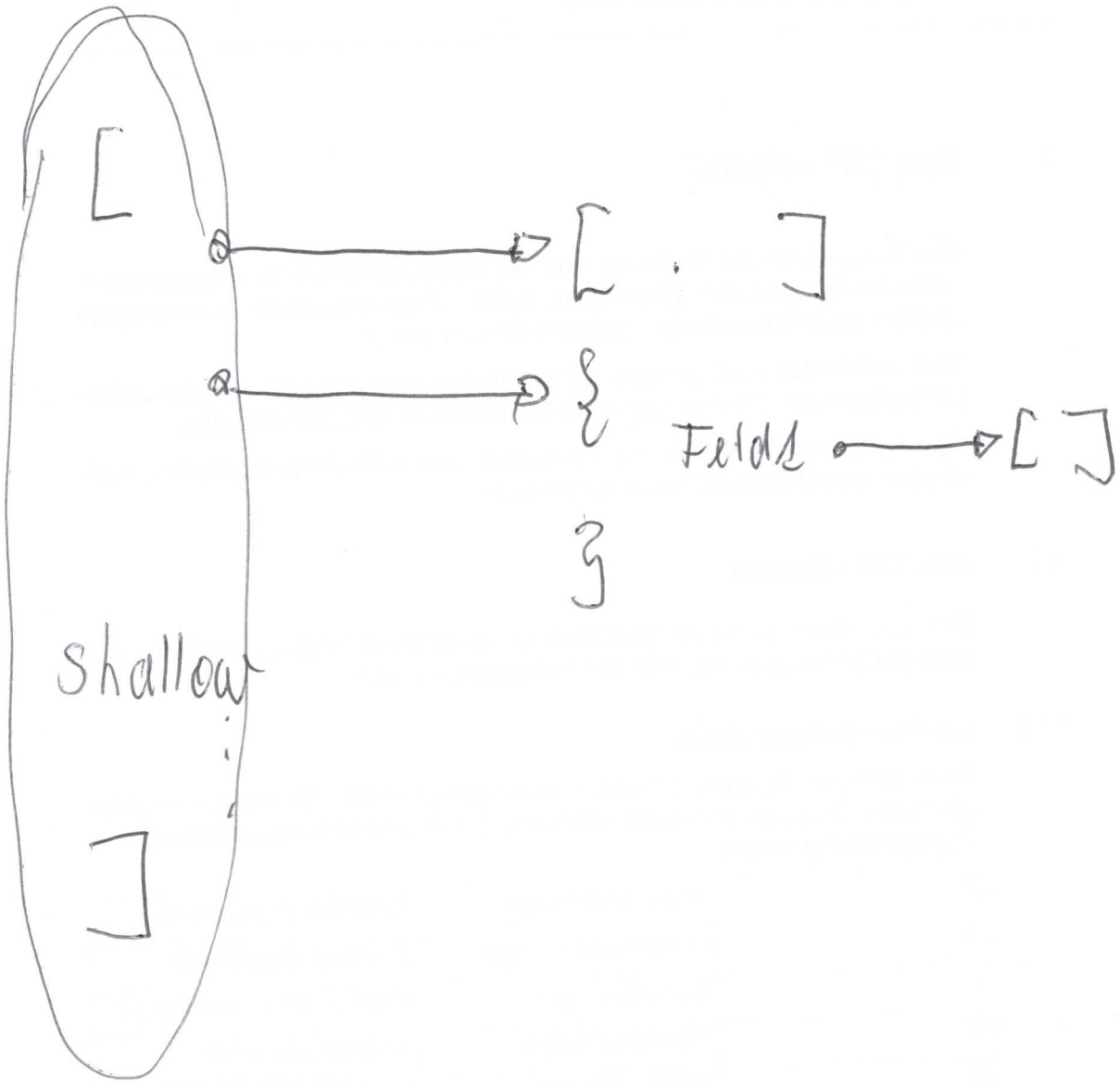
---

l1 = l[:] l1 → [ ]
           l → [ ]

$V_1 = [\ ]$

$V_2 = V_1$

$V_3 = V_1$ / $V_3 = V_2$


[
[.]
{
Feld1
[ ]
}
shallow
]

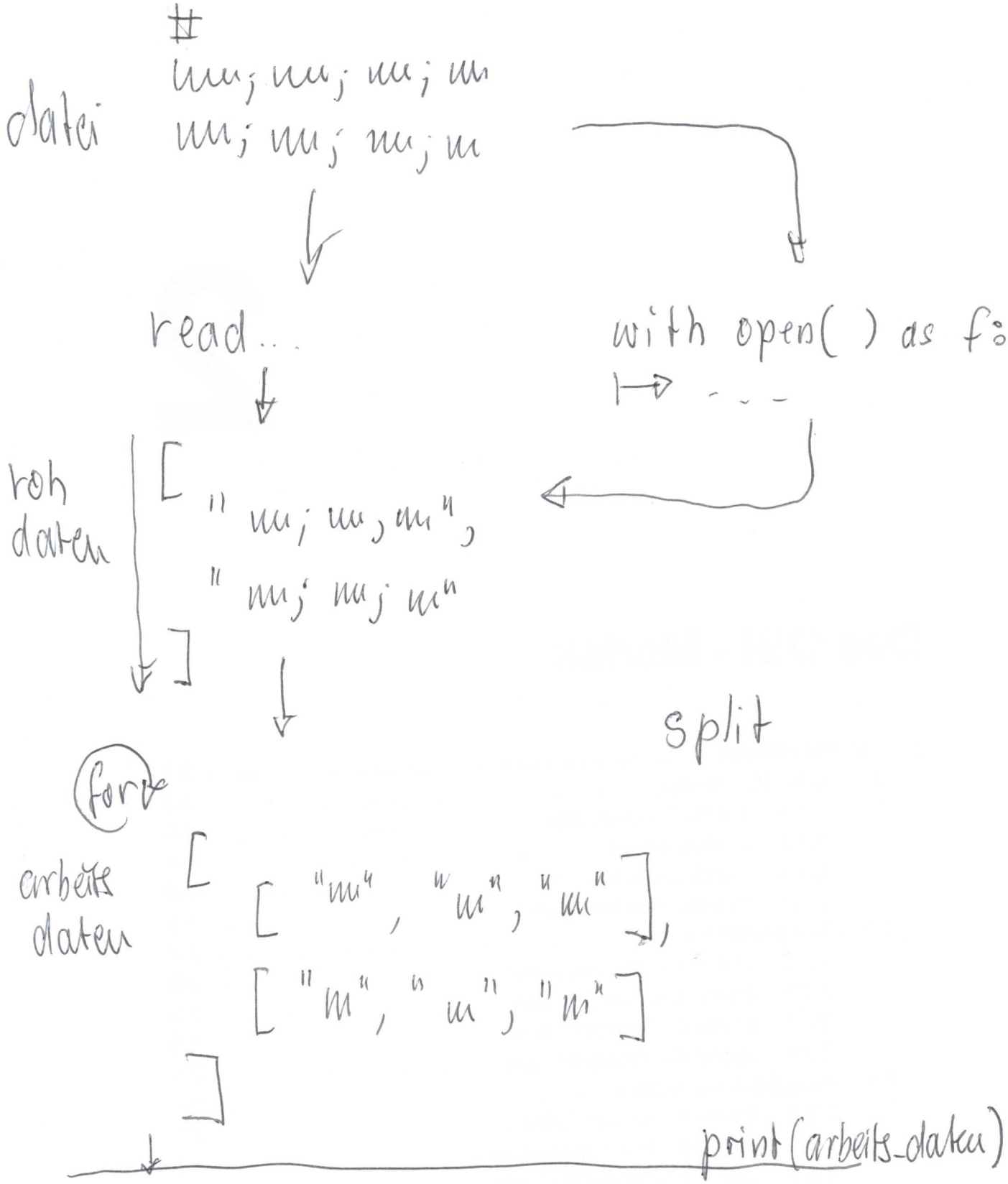
```
d = {}
d = dict()
```

```
d = {"key": value,
     "key": value}
```

```
d = dict(key = value,
         key = value)
```

```
= dict([("key", value),
        ("key", value)
])
```

```
d["key"] = value
print(d.get("key", alternativwert))

print(k["daddel"])
```


#
m; m; m; m
m; m; m; m
datei
read...
with open( ) as f:
...
roh daten
[
" m; m, m",
" m; m; m"
]
split
for
arbeits daten
[
["m", "m", "m"],
["m", "m", "m"]
]
print(arbeits_daten)

```
def x(w):
    pass

def y(w):
    pass

if __name__ == "__main__":
    for i
    x(w)
    y(w)
```

v3.py

```
import v1
import v2
import v3
```

v3.py

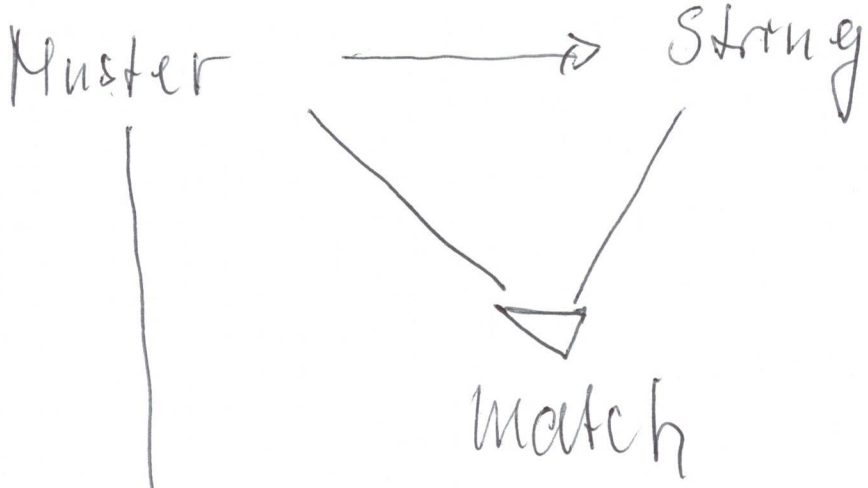
```
var = 4711

def x():
    pass

def y():
    pass

def main():
    x()
    y()
    var2 = "xxx"

main()
```

v1.py

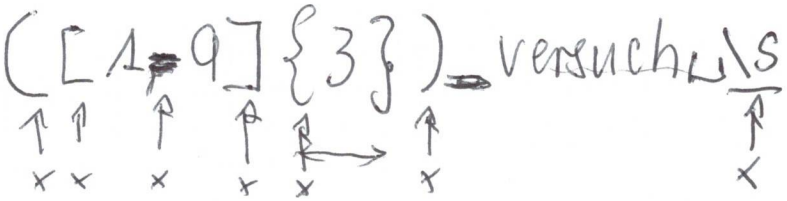
```
def x(m):
    pass

def y(m):
    pass

m = "..."
x(m)
y(m)
```

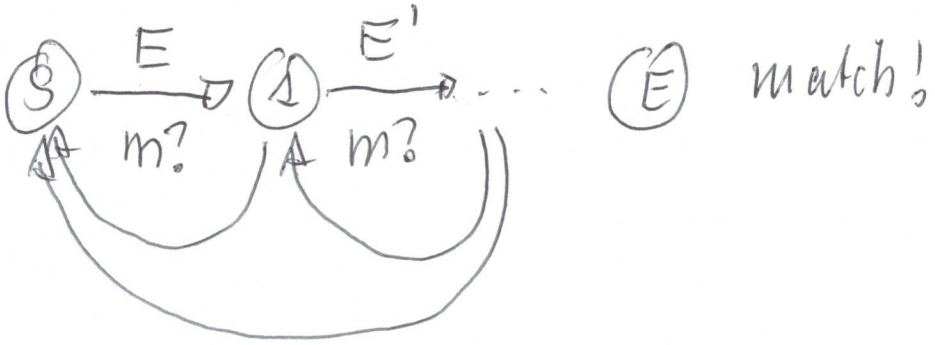
v2.py

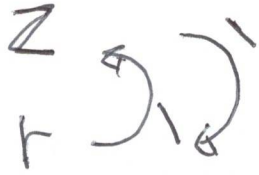
Reguläre Ausdrücke



einfache Zeichen
Sonderzeichen







. bel. Zeichen
\. Punkt

\S Zeichen
\S nicht, Spaces

Quantifier

← * 0-∞

1 *

: *

← + 1-∞

← ? 0/1

'Willi'

← {von, bis}

← {exakt}

← {mindestens,}

^ Anfang
$ Ende

^$

^␣*$

(r) Gruppen

[ ] Zeichenklasse [a-z]

[0-9]

\d

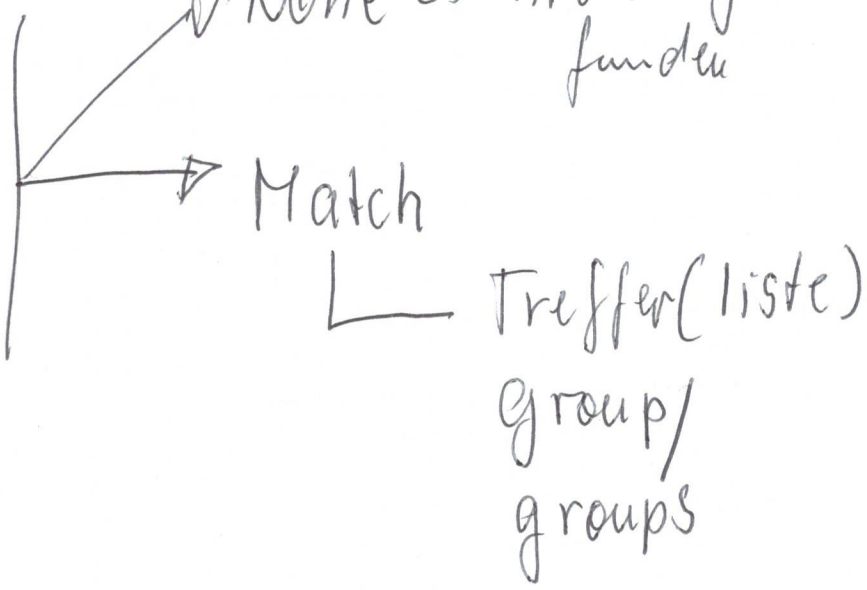
[a-zA-Z]

[a-zA-Z0-9]

r1|r2 oder

import re

re.find
- findall
- search
- sl



C++ / Java public / private

python ∅

Konvention!

class X

| name
| def aktion } „public"

_name
def_aktion } „private"

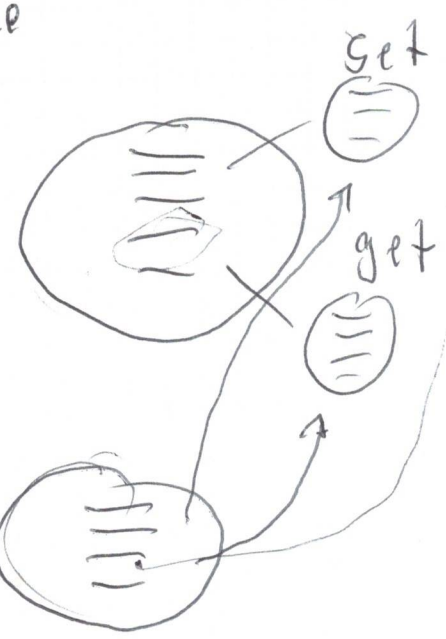
__name von aussen unsichtbar

__name__ „magic"
„dunders"

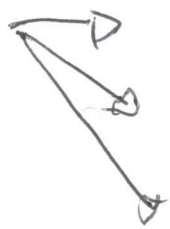
„Aussenwelt"

```
class X:
    name = None
    def set(self, wert):
        self.name = wert

    def get(self):
        return self.name
```

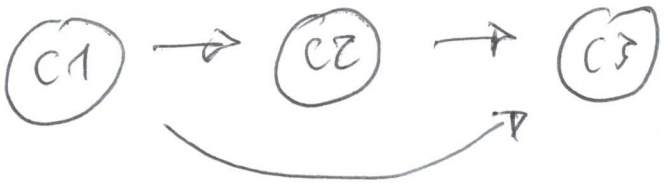
```
p = X()
p2 = X()
p2.set(p.get())
```



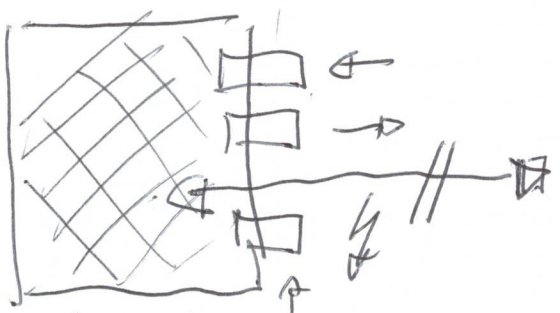
class



01
02
03

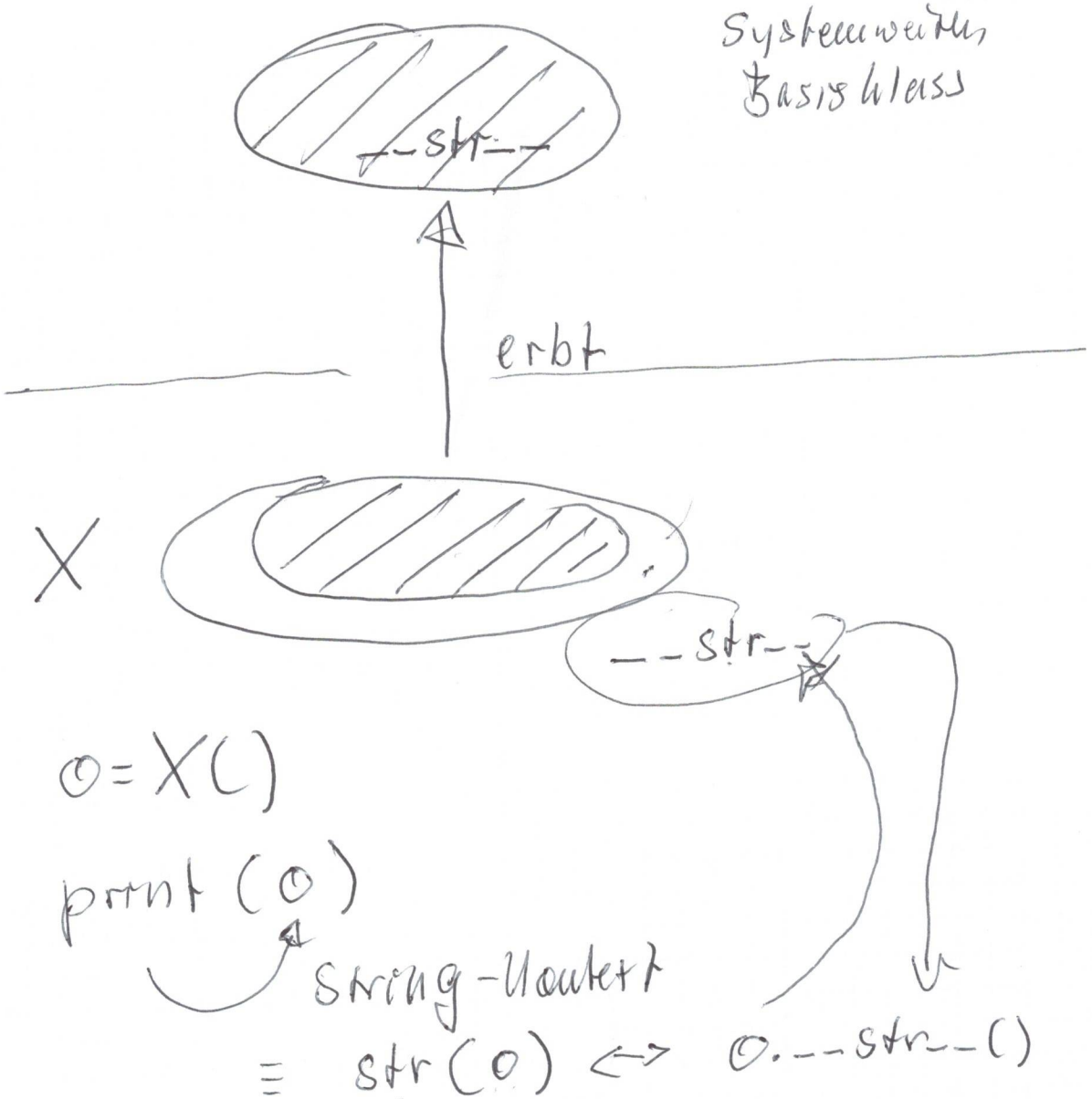


class
init
attribute <-> Daten
methoden <-> Aktionen



info hiding
daten + aktion

API


Systemweite Basisklasse
__str__
erbt
X
__str__
o = X()
print(o)
String-Umwandlung
≡ str(o) ⟷ o.__str__()

Class StockValueReader: (Stock_value_reader)

→ fname (Annahme: csv-Werte)

-- init --→ Datei geladen
intern gespeichert
zur Nutzung vorbereitet

dict{ Datum: TT.MM.JAHR, → [Werte]

Kurse ohne $ Zeichen
als Float

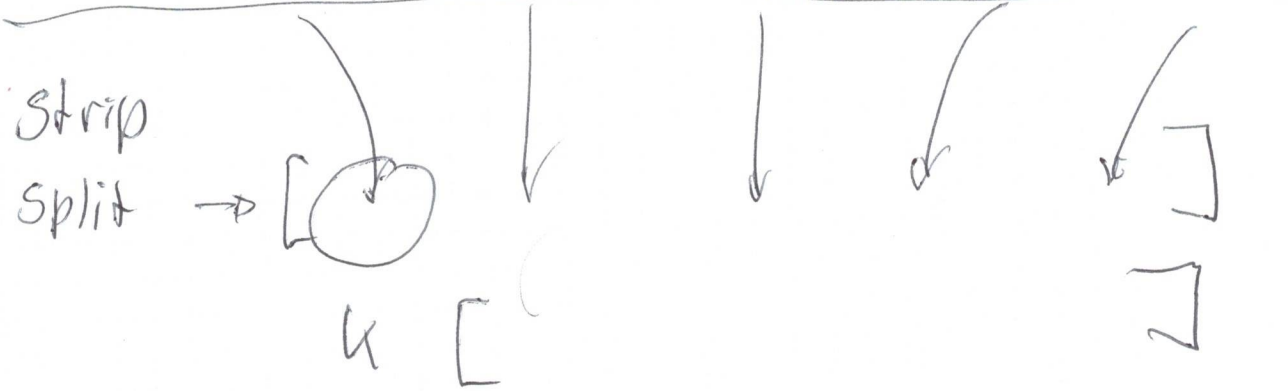
Stückzahlen als Integer

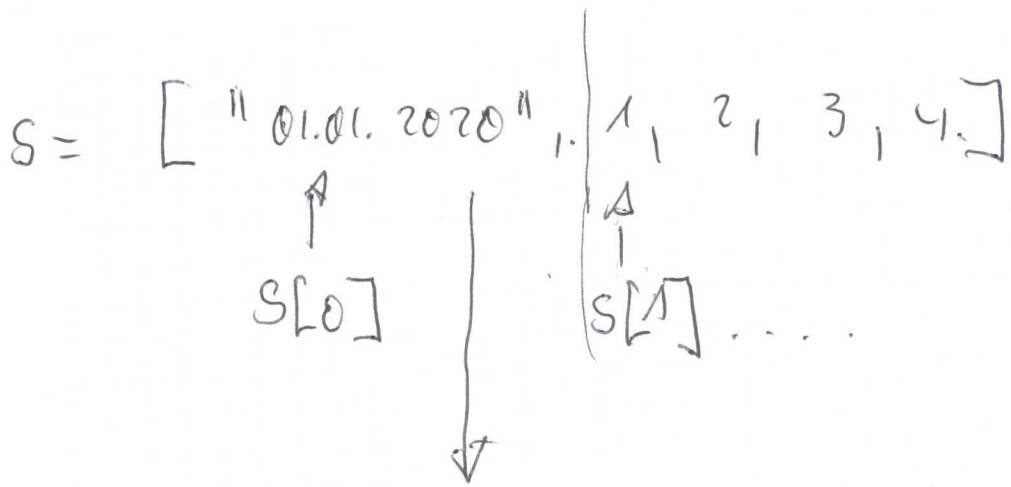
__ str __

[(m), m, m]

key → [value]

2020\01\01, ($5000), ($6000), 1000000, $700





date = S[0]
values = S[1:]

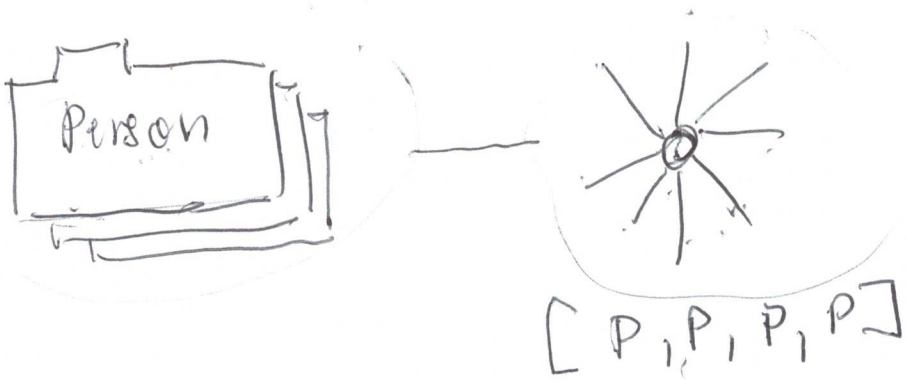
date, *values = S

---

date = ["Y", "m", "d"]

Y = d[0]
m = d[1]
d = d[2]

Y, m, d = date



class

self.var

def methode(self):

self.var

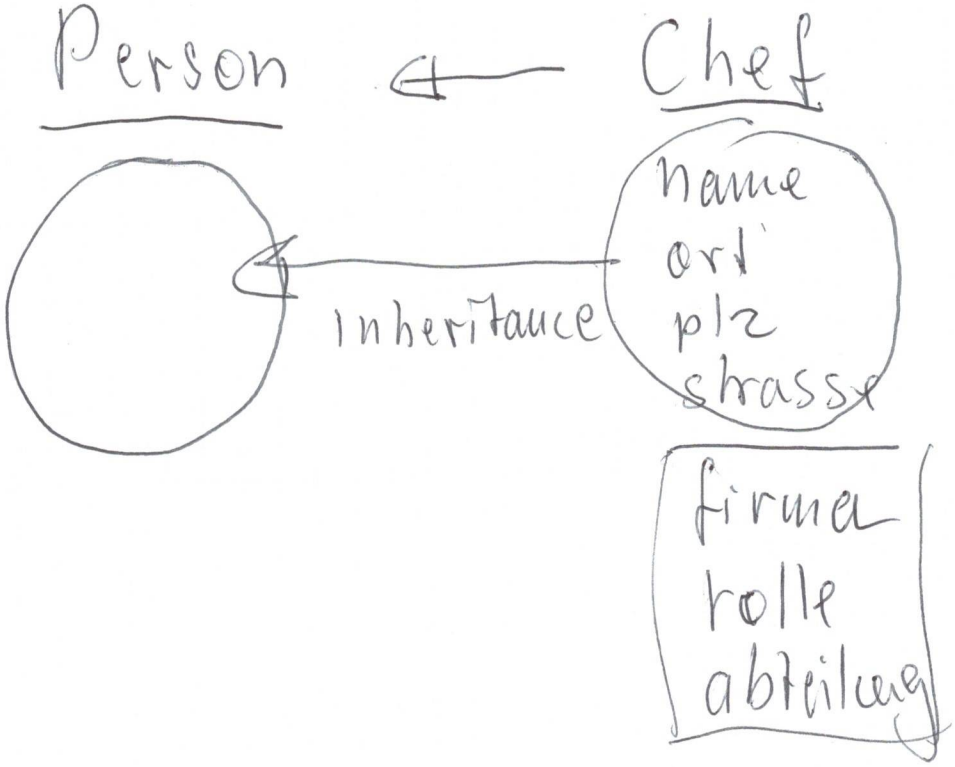
var

def m2(self):

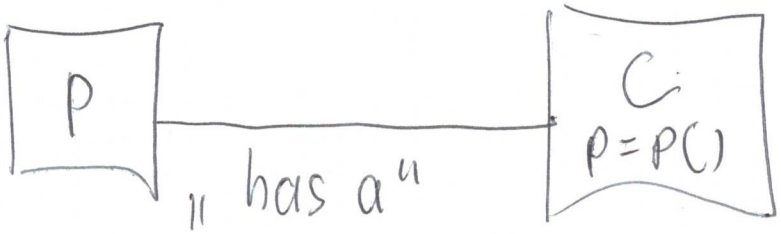
self.var

X

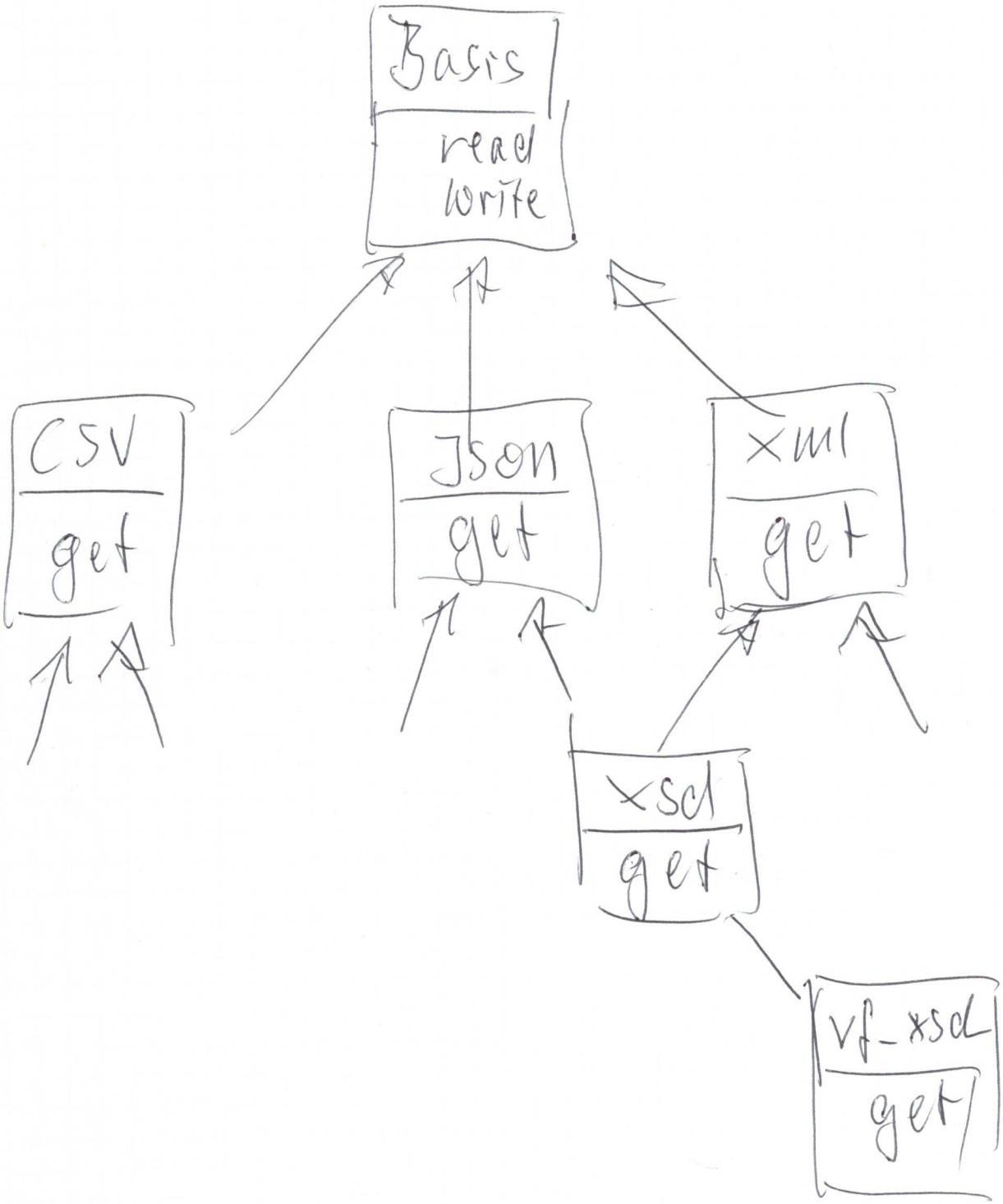
Vererbung


Person
Chef
inheritance
name
ort
plz
strasse
firma
rolle
abteilung


P
C
P
"is a"
Vererbung!


P
"has a"
C
P=P()

Klassenhierarchie



name, vorname, ort, plz, strasse

personen daten
csv

class Person

class RoloDex

load

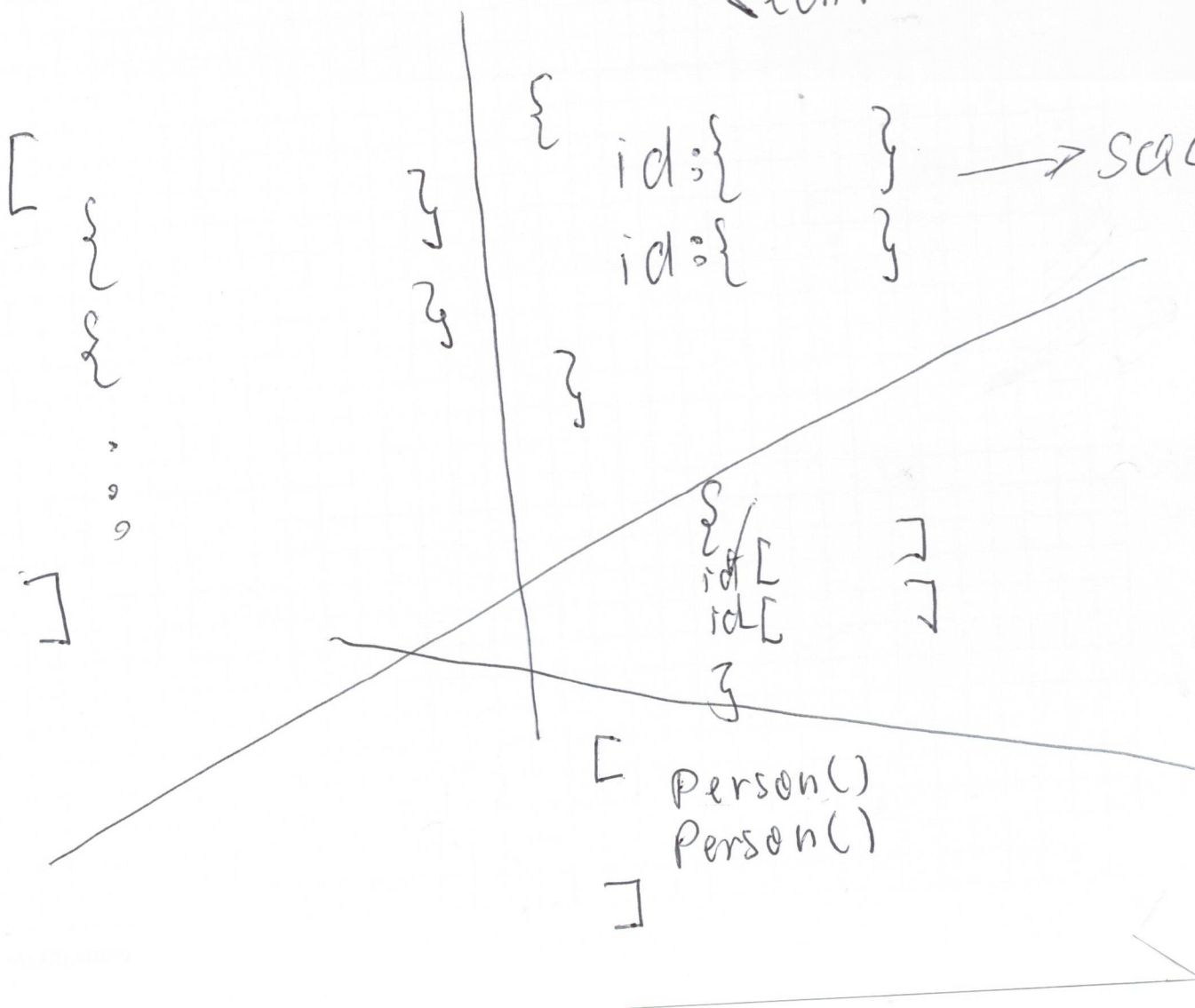
Save

search

add

del

edit



FileTransformer

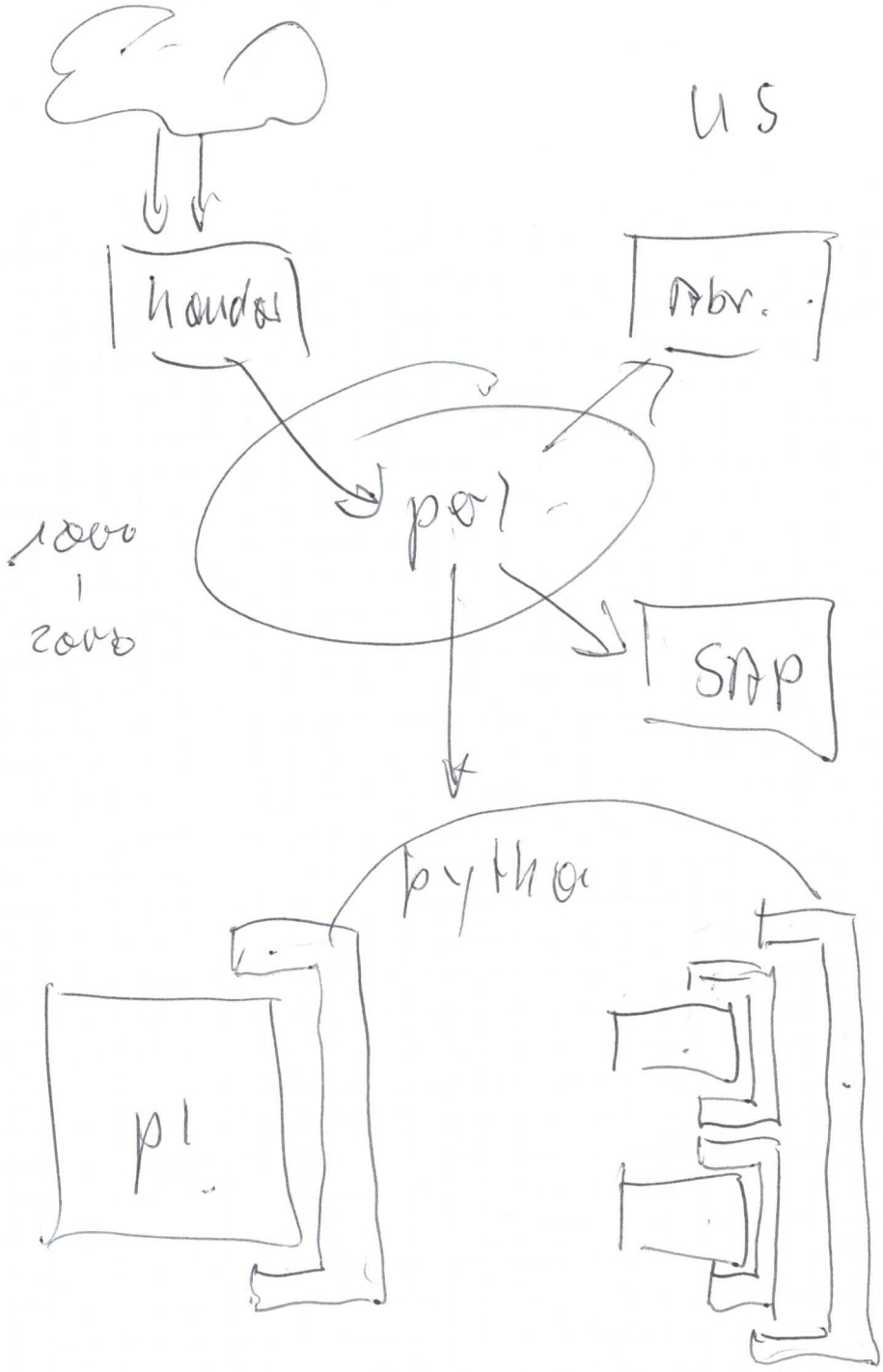
dict

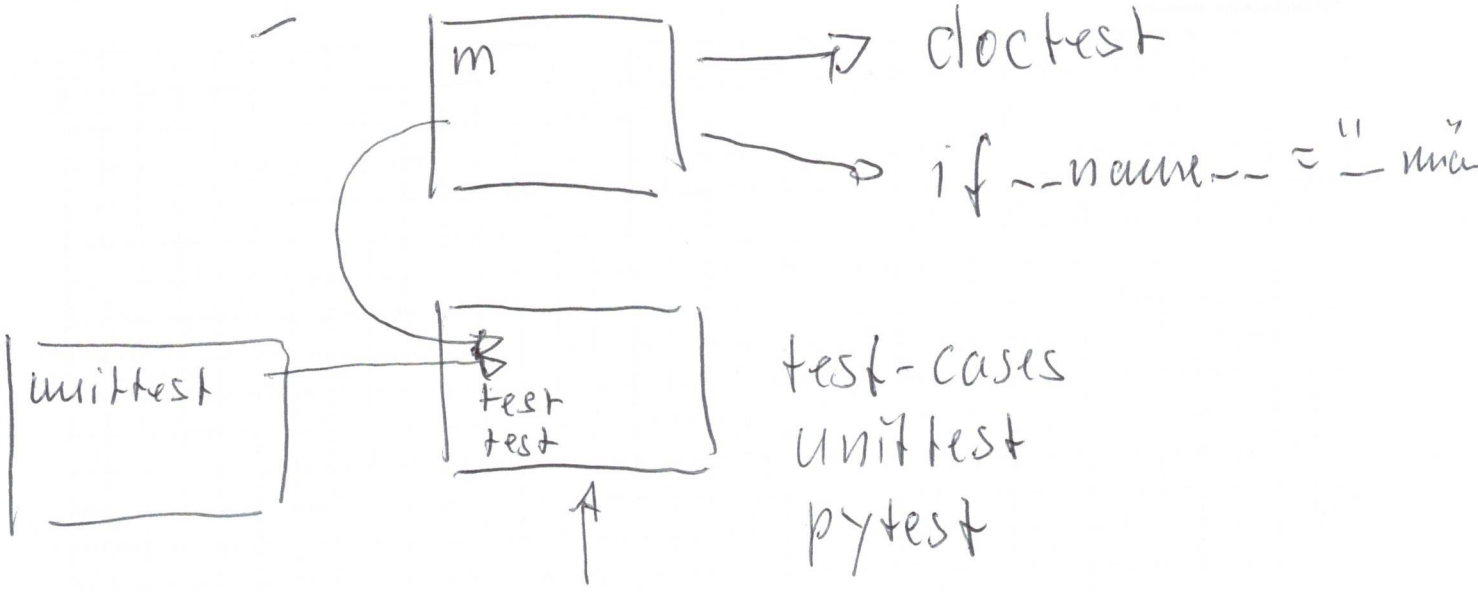
X

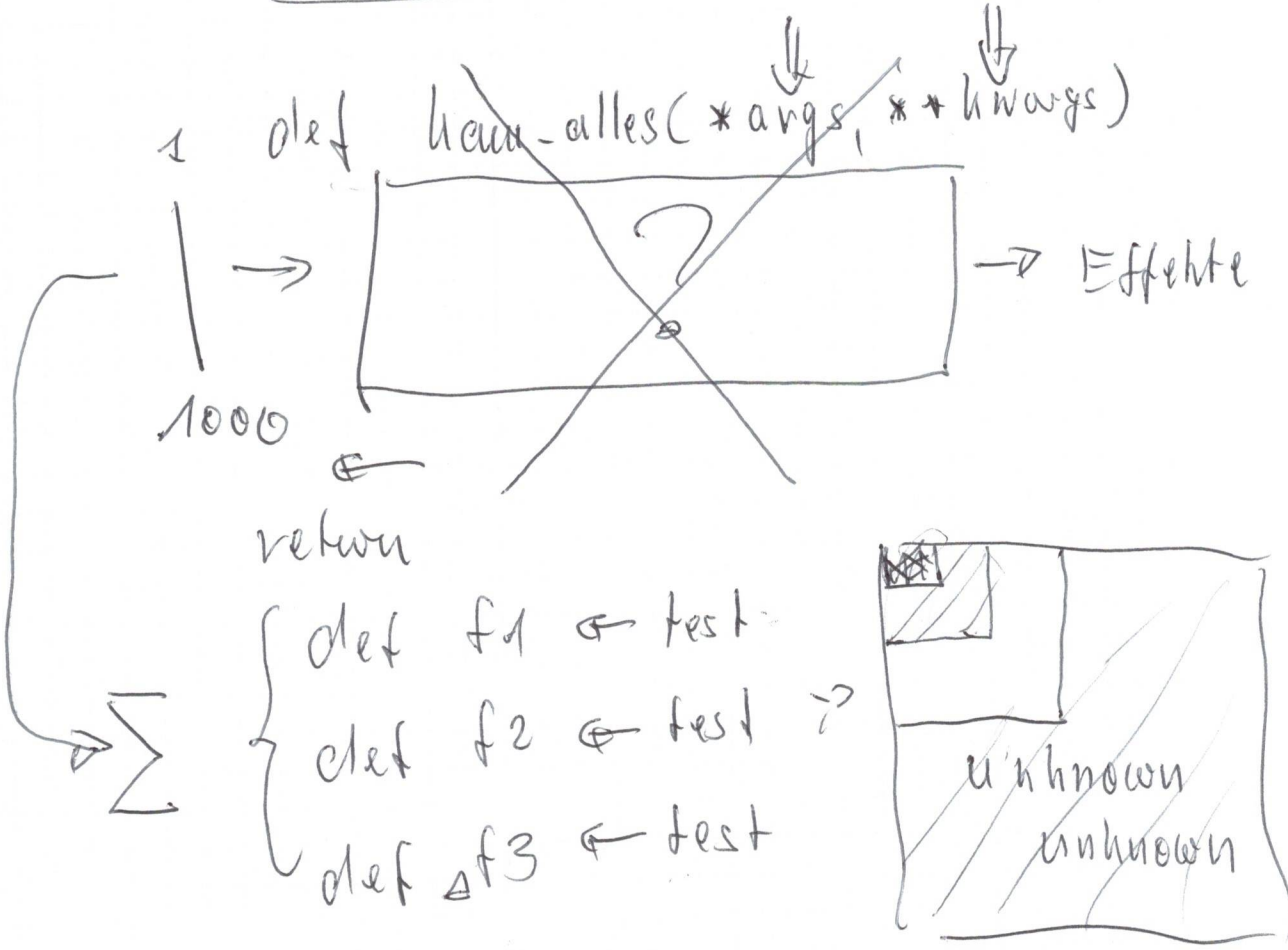
Y

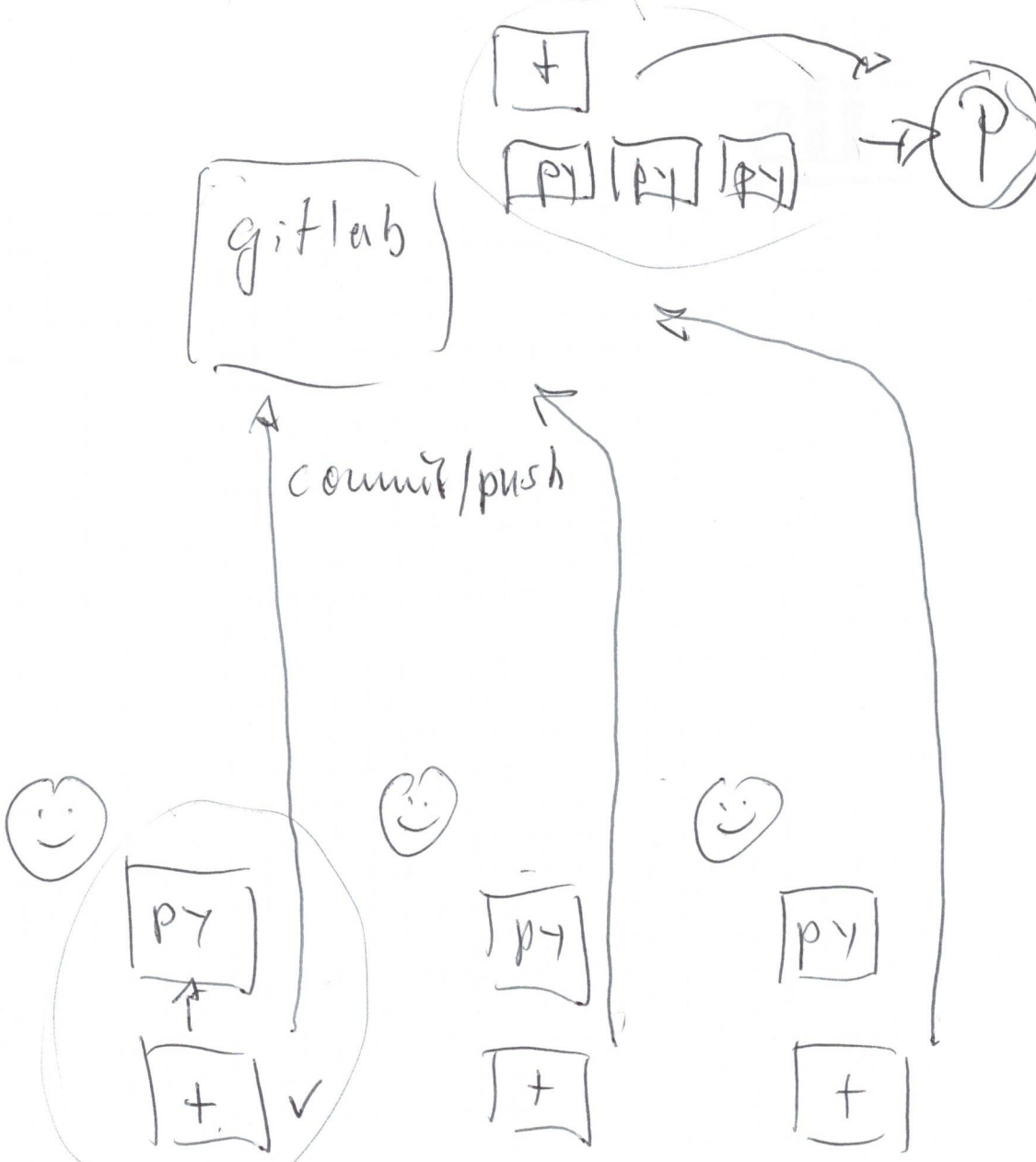
load(fname)

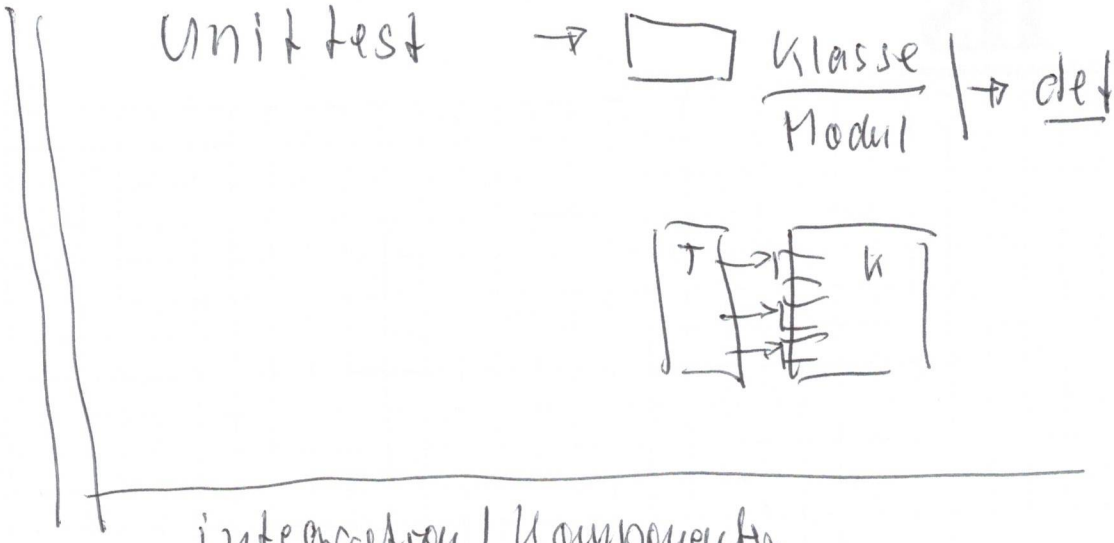
save(fname, ...)


U S
handel
perl
SAP
python
pl

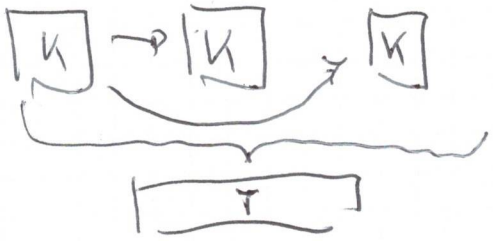

m
doctest
if __name__ == "__ma
unittest
test
test
test-cases
unittest
pytest
python3 test.py


1
def tun_alles(*args, **kwargs)
Effekte
1000
return
def f1 ← test
def f2 ← test
def f3 ← test
unknown
unknown


gitlab
commit/push

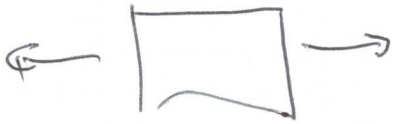


Integration / Komponenten



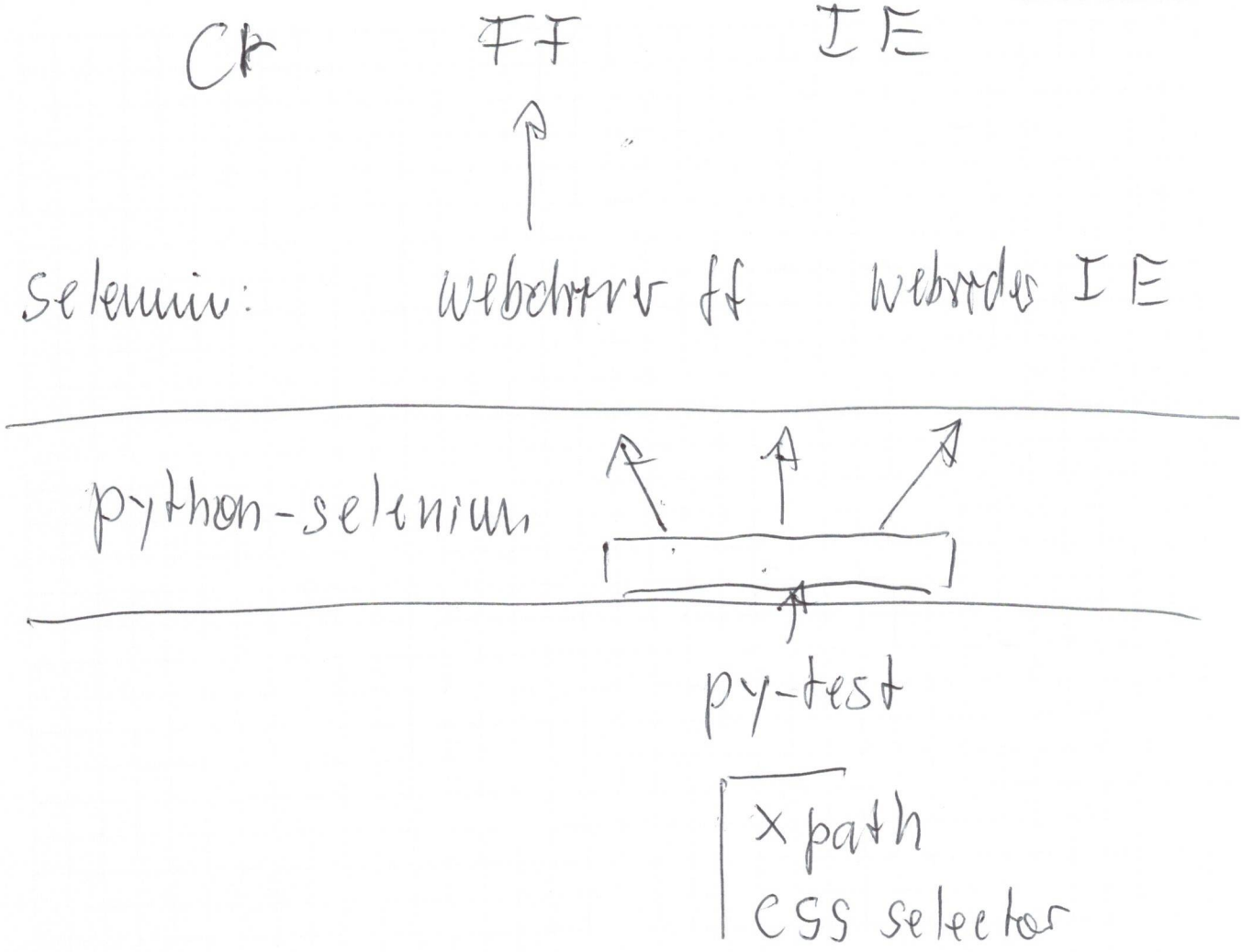
Integration

System → API

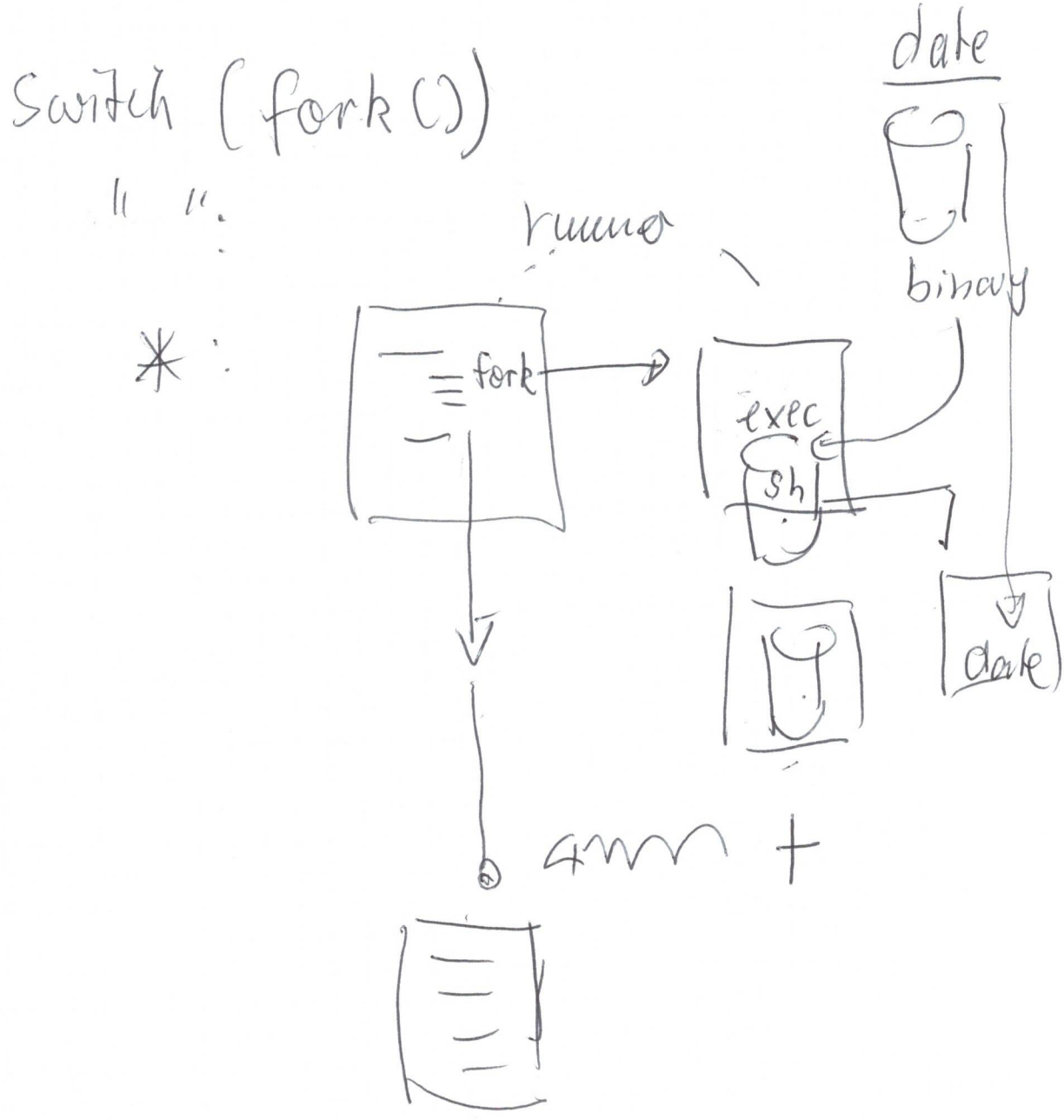


GUI ↔ Web

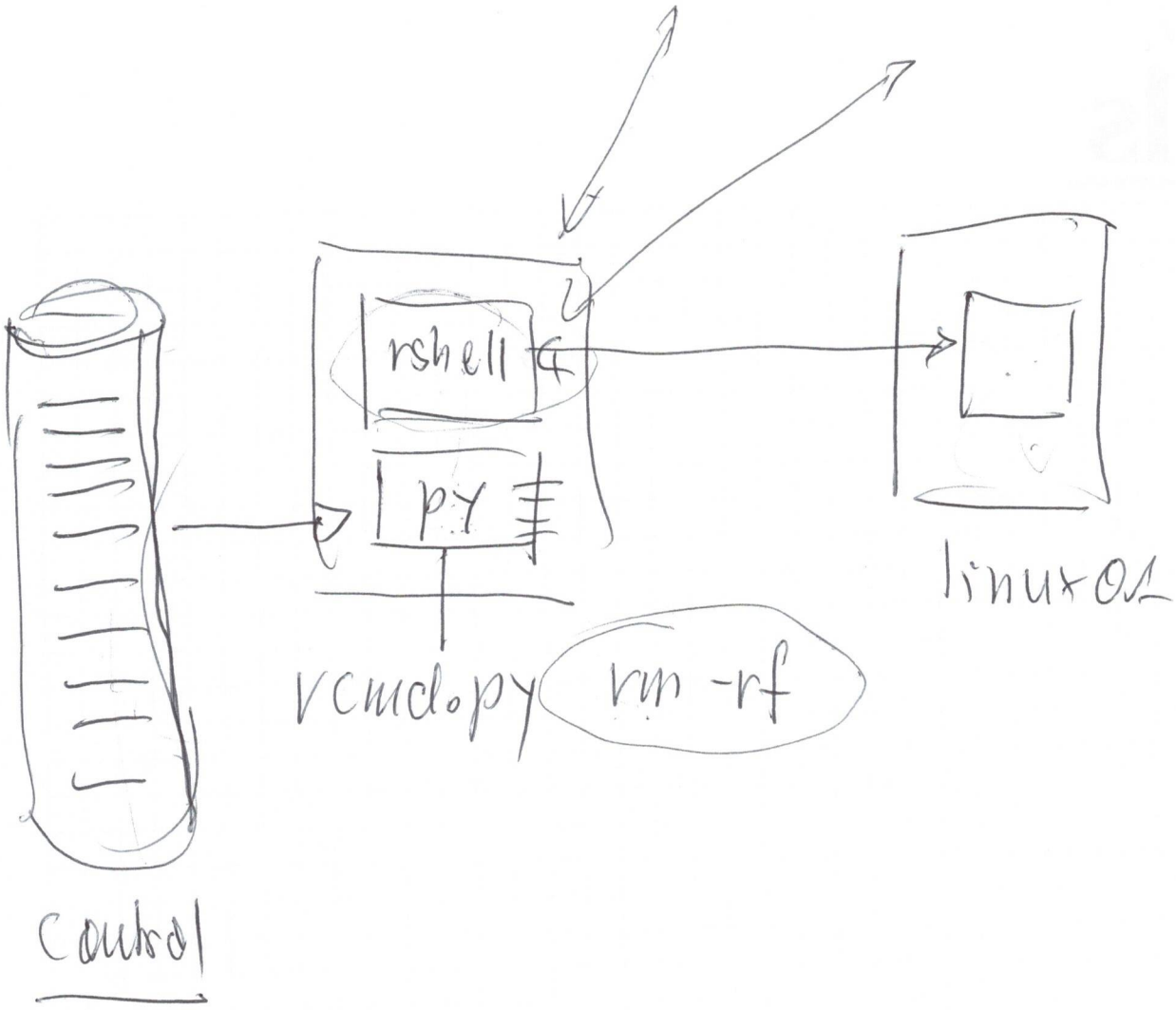
„py-binding" selenium


Ch
FF
IE
selenium:
webdriver ff
webdrvr IE
python-selenium
py-test
xpath
CSS selector

fork
exec* } System calls



sh "a | b | c > datei")


rshell
py
linux01
rcmd.py
rm -rf
Control

z. B

Aufruf: filetree --startfolder /home/code
--datei_name xx.py

Startet bei startfolder
geht durch den Baum durch
wenn datei_name gefunden -> Ausgabe

netstat nutzen
alle Streams / connect ~~PID~~ finden
in Liste speichern
Liste ausgeben

---

kontakt@uc-it.de