Quick-Start Informatik 2011

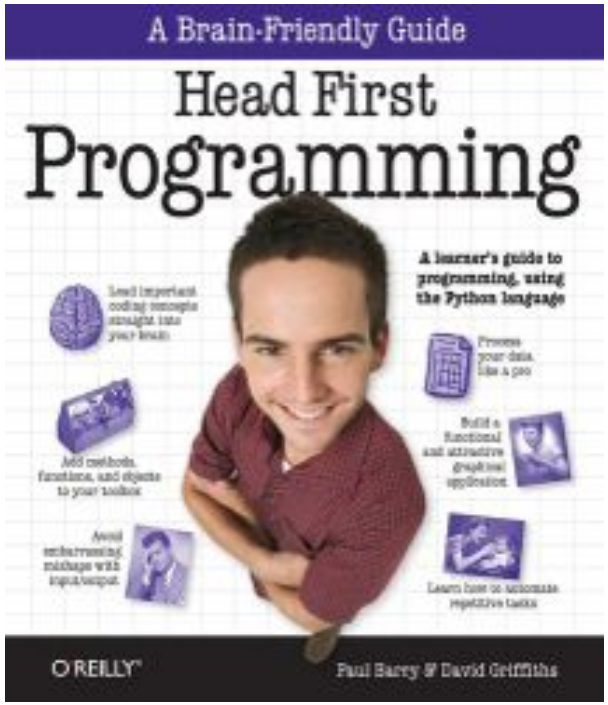
Programmieren in Python
Teil 1

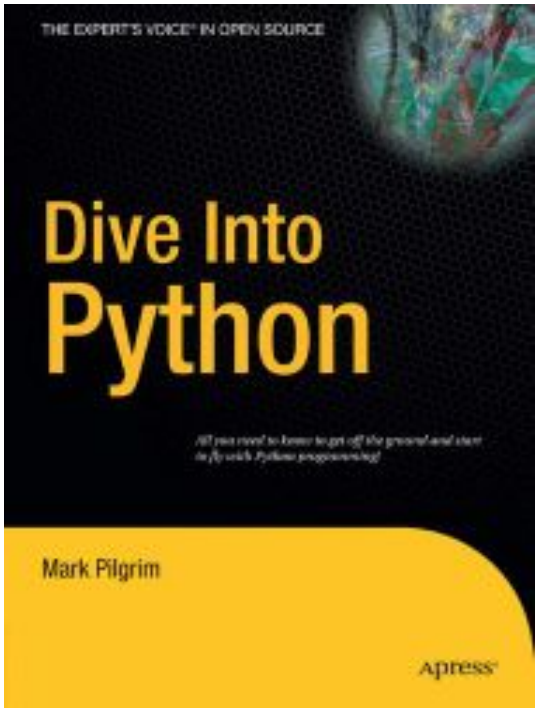
Alles zum Praxisteil

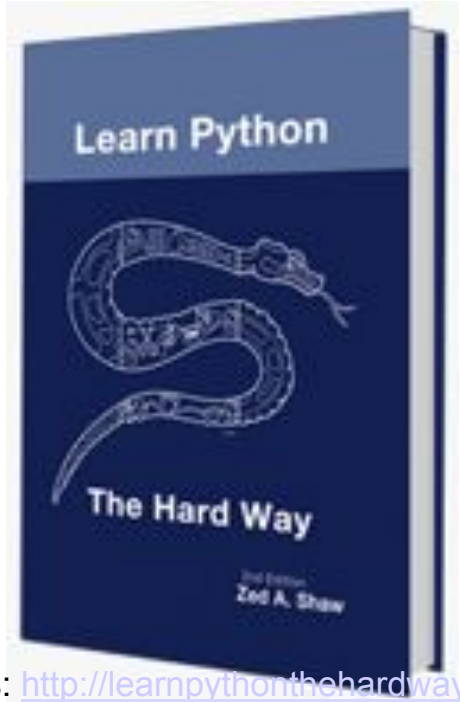
Skript,
Übungen,
Folien...

auf http://vorkurs.cs.uni-frankfurt.de

Bücher:







Gratis: http://learnpythonthehardway.org/

Fragen, fragen, fragen!

Erste Vorlesung

- Programmieren, wie geht das?
- Daten und Datentypen
  - Zahlen
  - Vergleiche und Wahrheitswerte
  - Text
- Variablen
- Bedingungen stellen
- Daten anfordern und Ausgeben

Programmieren

Bild: http://www.eatliver.com/i.php?n=3012

Python

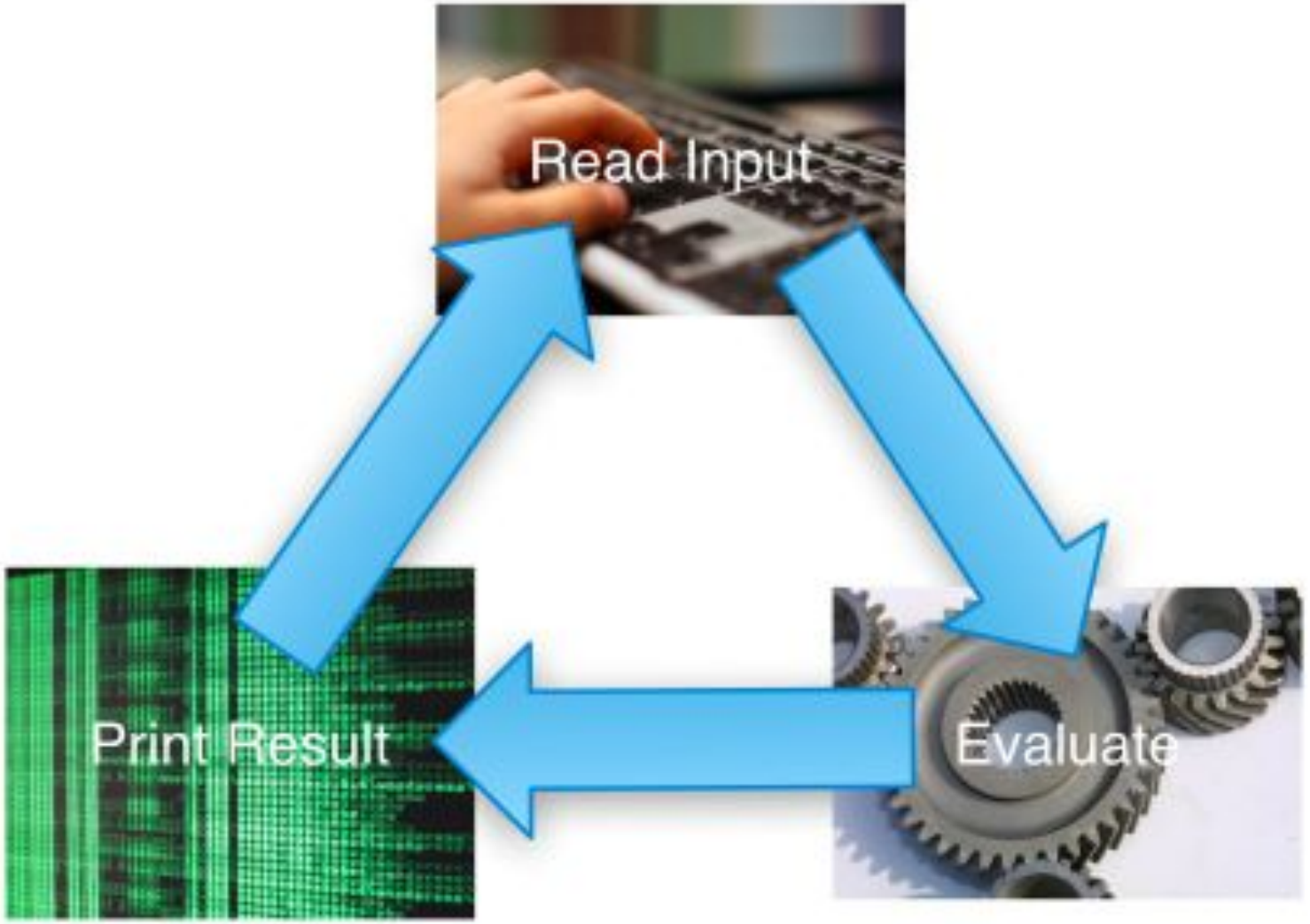
- Höhere Programmiersprache
  - Sie ist für Menschen einfacher zu verstehen, als die "Muttersprache" des Computers
- Aber: der Computer versteht sie nicht (direkt):
  - Wir brauchen einen "Übersetzer"
- http://www.python.org
  - Kostenlos für Win/Mac/Linux

IDLE

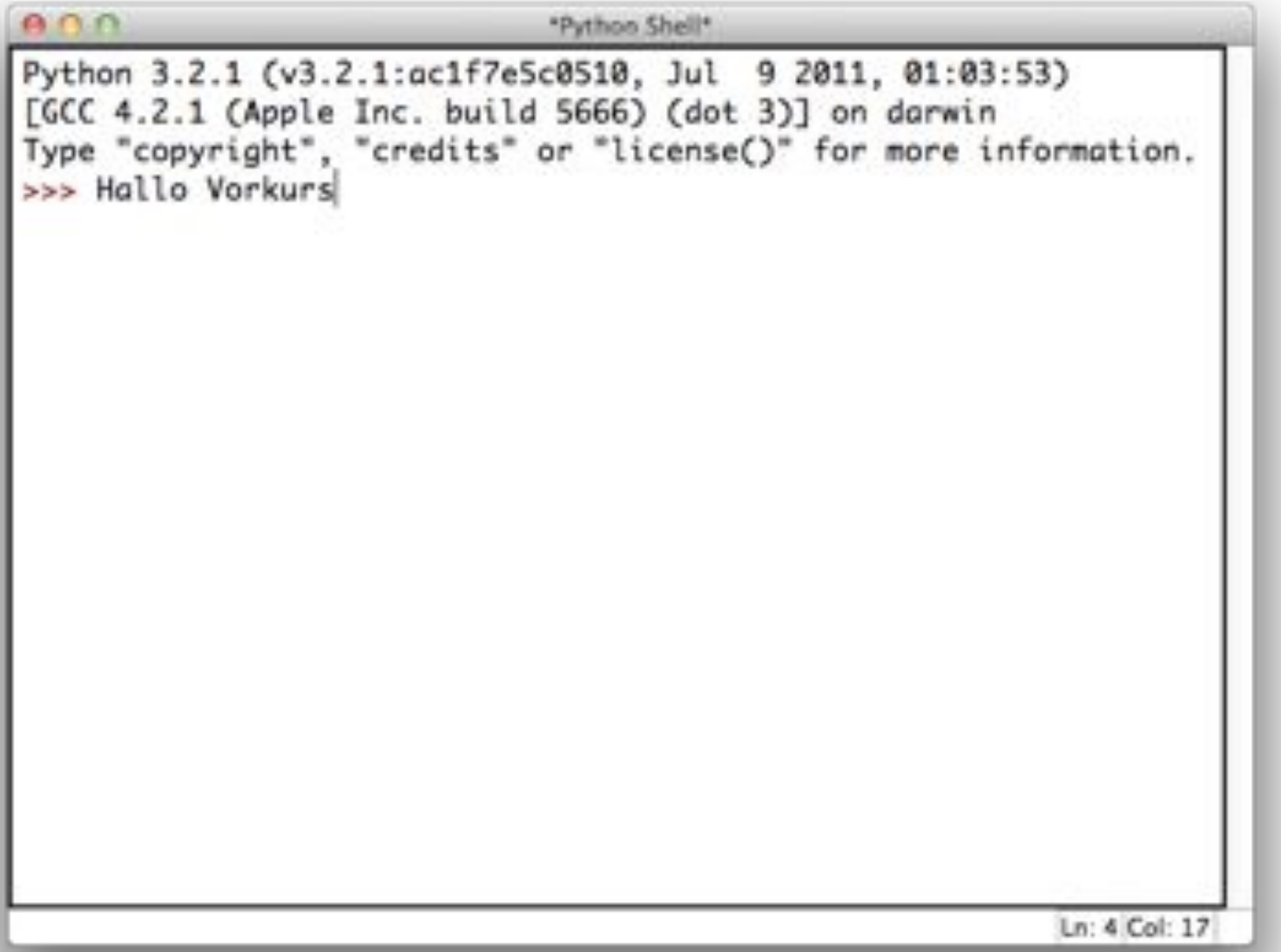

Python Shell
Python 3.2.1 (v3.2.1:ac1f7e5c0510, Jul 9 2011, 01:03:53)
[GCC 4.2.1 (Apple Inc. build 5666) (dot 3)] on darwin
Type "copyright", "credits" or "license()" for more information.
>>>
Python 3.2
IDLE.app
HTML
Python D...tion.html
Python L...cher.app
COMMAND
Update S...ommand
Open in Finder

Diesen Modus nennt man "REPL"

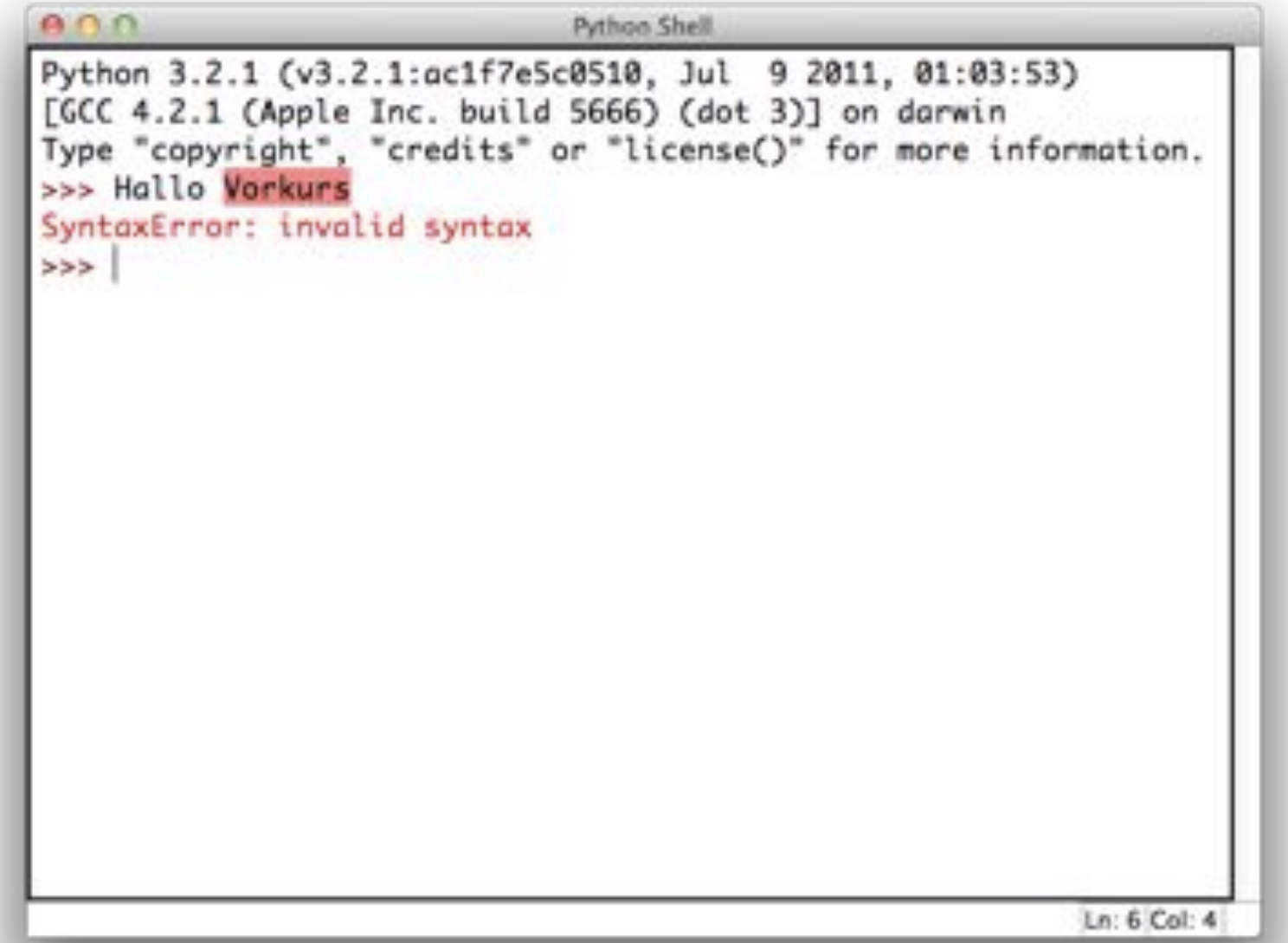
(kurz für **R**ead **E**valuate **P**rint **L**oop)



Mit dem REPL arbeiten



Mit dem REPL arbeiten



Was bedeutet "Invalid Syntax"?

Sprachen haben Syntax und Semantik

Syntax: Wie Sätze gebaut sein müssen, damit sie (in dieser Sprache) richtig sind
Semantik: Bestimmt was der Satz bedeutet.

'Hallo Vorkurs' ist also wohl falsch gebaut

Was bedeutet "Invalid Syntax"?

Fehler können vor allem am Anfang frustrieren.
Nicht aufgeben - das geht jedem so!

Hier findet ihr Hilfe:

Google bing stackoverflow ..

stackoverflow.com

Vorkurs Forum: http://fsinf-forum.de/viewforum.php?f=45

Python Programme

Willst du ein Programm weitergeben/mehrmals verwenden, speichere den Python Code in einer Textdatei mit Endung *.py*

SayHello.py

IDLE Demo

Zusammenfassung

- Programmieren = Sprache lernen die in "Computersprache" übersetzt werden kann
- Python übersetzt uns IDLE
  - Im REPL
  - Aus einer Textdatei

Datentypen

Datentypen

Ein Computer funktioniert (stark vereinfacht) so:

Was kann der Computer alles mit einem Datum machen?

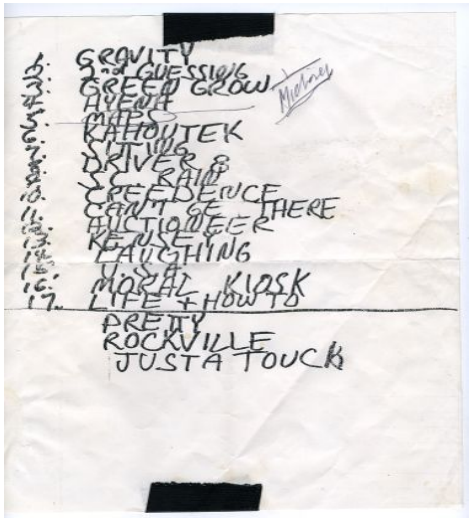
Das hängt davon ab, von welchem Datentyp es ist...

Datentypen

Python unterstützt viele Datentypen:

Zahlen

Text



Listen von Dingen

und weitere ...

Zahlen

Zahlen (Integer und Float)

Zahlen ohne Komma
Sog. `int` (Ganzzahl)

```
>>> 4
```

Zahlen mit 'Komma'
Sog. `float` (Fließkommazahl)

```
>>> 4.0
```

Richtig echter Python Code!!

Warum Punkt und kein Komma??
Der englische Sprachraum (und deshalb viele Programmiersprachen) verwendet den **Punkt** als Dezimaltrenner, **kein Komma**!

Rechnen mit Zahlen

Addition	`>>> 67 + 2` `69`
Subtraktion	`>>> 5 - 2` `3`
Multiplikation	`>>> 4 * 6` `24`
Potenz	`>>> 2 ** 5` `32`
Division	`>>> 4 / 6` `0.6666666666666`
Division (mit Rest)	`>>> 4 // 6` `0`
Modulo (Rest d. Division)	`>>> 4 % 6` `4`

http://www.flickr.com/photos/e2/2333259787

Vorrang

Bei mehr als zwei Rechnungen haben gewisse Operatoren (+, *, -, /, ...) Vorrang.

Wie in Mathe gilt Punkt vor Strich:

```
>>> 3 + 4 * 5
23
```

... und Klammern haben Vorrang

```
>>> (3 + 4) * 5
35
```

http://www.flickr.com/photos/henteaser/274494784/

Alle Vorrangregeln
http://docs.python.org/reference/expressions.html#summary
Tipp: Nicht auswendig lernen. Klammern setzen.

Von Kommazahl zu Ganzzahl

Manchmal braucht man eine bestimmten Typ von Zahl:

Kommazahl zu Ganzzahl

```
>>> int(5 / 2)
2
```

Parameter

Ganzzahl zu Kommazahl

```
>>> float(5 - 2)
3.0
```

`int()` und `float()` sind sog. Funktionen.
Sie transformieren den Parameter in einen anderen Wert!

Vergleiche

Interessanter wird es wenn wir Zahlen miteinander vergleichen...

Vergleiche

Interessanter wird es wenn wir Zahlen miteinander vergleichen...

```
>>> 3 > 4      Ist drei grösser als vier?
False
>>> 3 < 4      Ist drei kleiner als vier?
True
>>> 3 <= 4     Ist drei grösser oder gleich vier?
True
>>> 3 >= 4     Ist drei kleiner oder gleich vier?
False
```

Gleichheit

```
>>> 3 == 4
```
Ist drei gleich vier?

```
False
>>> 3 != 4
```
Ist drei ungleich vier?

```
True
```

Prüfen auf Gleichheit

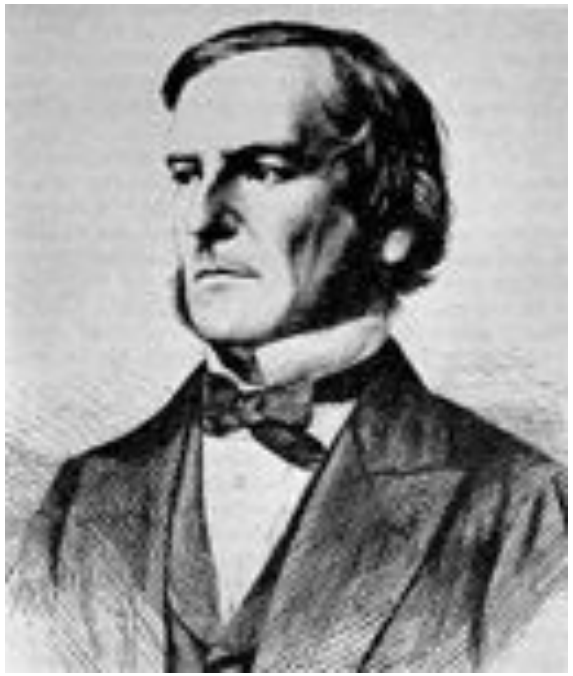
Gleichheit wird in vielen Programmiersprachen mit == geprüft.
Das einzelne = ist hier für eine andere Operation reserviert.

Wahrheitswerte

Wahrheitswerte (bool)

Das Ergebnis eines Vergleichs ist ein Wahrheitswert.

Es gibt nur zwei Wahrheitswerte:
`True` (wahr) und `False` (falsch).



http://en.wikipedia.org/wiki/George_Boole

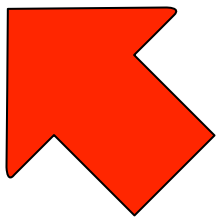
In Python heißt dieser Datentyp bool nach George Boole, der sich viel mit Wahrheitswerten beschäftigte.

Kombinierte Vergleiche

Ausdrücke Kombinieren

```
>>> (4 > 3) and (3 < 5)
True

>>> (4 > 3) and True
True
```



Man kann Wahrheitswerte auch direkt in Aussagen verwenden!

Wahrheitswerte kombinieren

"Und" Kombiniert (Konjunktion)	"Oder" Kombiniert (Disjunktion)
`>>> True and True` `True`	`>>> True or True` `True`
`>>> True and False` `False`	`>>> True or False` `True`
`>>> False and True` `False`	`>>> False or True` `True`
`>>> False and False` `False`	`>>> False or False` `False`

Logisch!

Diese sog. Junktoren (= Verknüpfer) kommen aus der Logik, die sich mit der spannenden Frage nach der Wahrheit auseinandersetzt! Mehr dazu am Tag 4!

Wahrheitswerte negieren

Manchmal möchte man wissen ob etwas nicht zutrifft:

```
>>> not (3 == 4)
True
>>> not (3 < 4)
False
```

Zu diesem "Umdrehen" sagt man auch Negation

Text

Text (Strings)

Wir sagen dem Interpreter, dass etwas Text (und kein Code ist) indem wir es in Hochkomma setzen:

```
>>> 'Hallo Vorkurs!'
'Hallo Vorkurs!'
>>> "Ich bin normaler Text"
'Ich bin normaler Text'
```

Warum String?
Dieser Datentyp wird als String bezeichnet, weil die einzelnen Buchstaben wie an einem Faden in einer Reihenfolge "aufgehängt" sind (vgl. Bild)

Text (Strings)

Wenn ein String über mehr als eine Zeile geht, müssen wir ihn in dreifache Hochkomma setzen!

```
>>> '''Hallo Vorkurs,
ich bin ein langer Text.
Ich gehe über drei Zeilen!'''
'Hallo Vorkurs, \nich bin ein langer Text.
\nIch gehe über drei Zeilen!'
```

Funktioniert genauso mit Strings in "

Text (Strings)

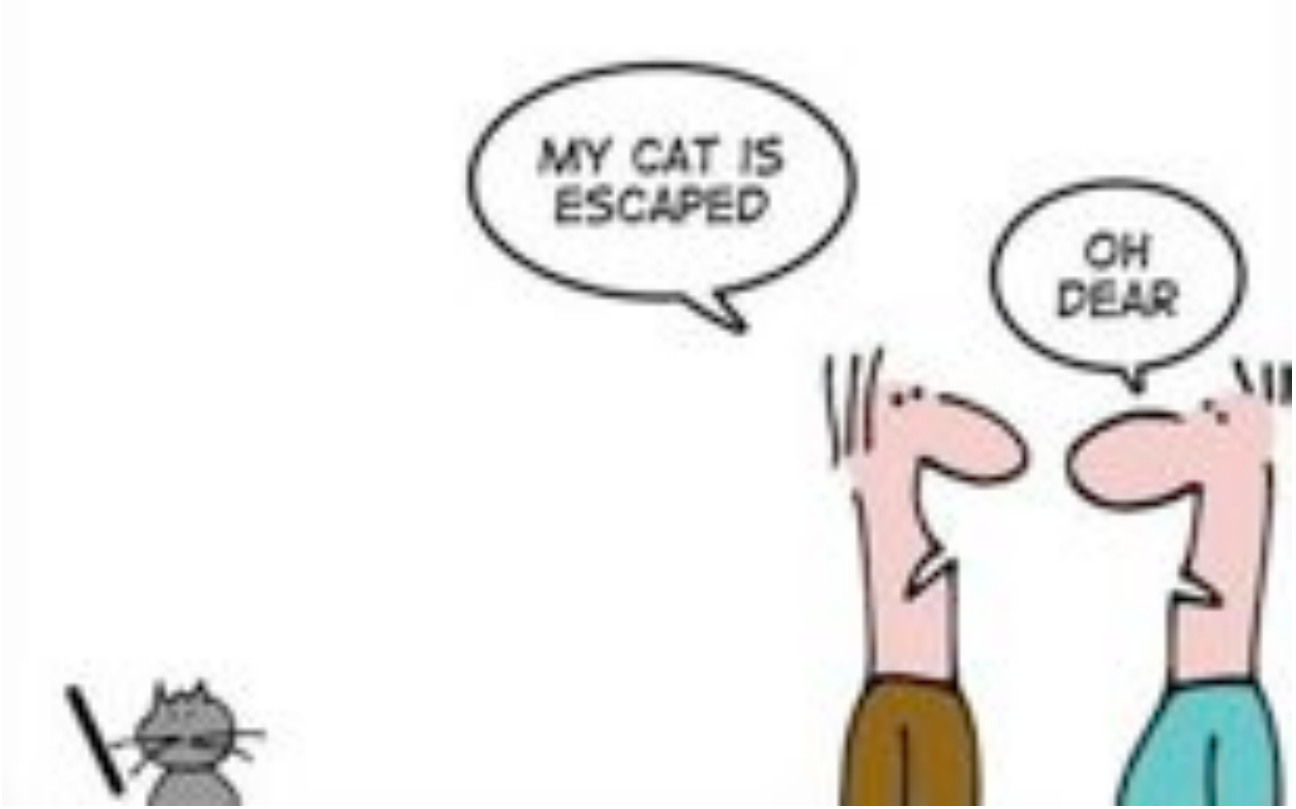
Wenn ein String über mehr als eine Zeile geht, müssen wir ihn in dreifache Hochkomma setzen!

```
>>> '''Hallo Vorkurs,
ich bin ein langer Text.
Ich gehe über drei Zeilen!'''
'Hallo Vorkurs,\nich bin ein langer
Text.\nIch gehe über drei Zeilen!'
```

Was ist das denn???

Escape-Sequenzen

Escape-Sequenzen (\ + Buchstabe) ermöglichen die Eingabe von nicht-druckbaren (Tab, Leerzeichen) oder nicht erlaubten (', ") Zeichen in Strings!



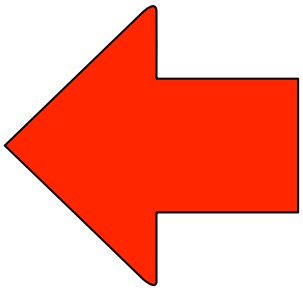
comic by: geekandpoke.typepad.com/

Alle möglichen Escape-Sequenzen findest du hier:
http://docs.python.org/py3k/reference/lexical_analysis.html#string-and-bytes-literals

Zwischenfrage

Wir haben jetzt bereites einige Daten eingegeben und gesehen was wir damit machen können.

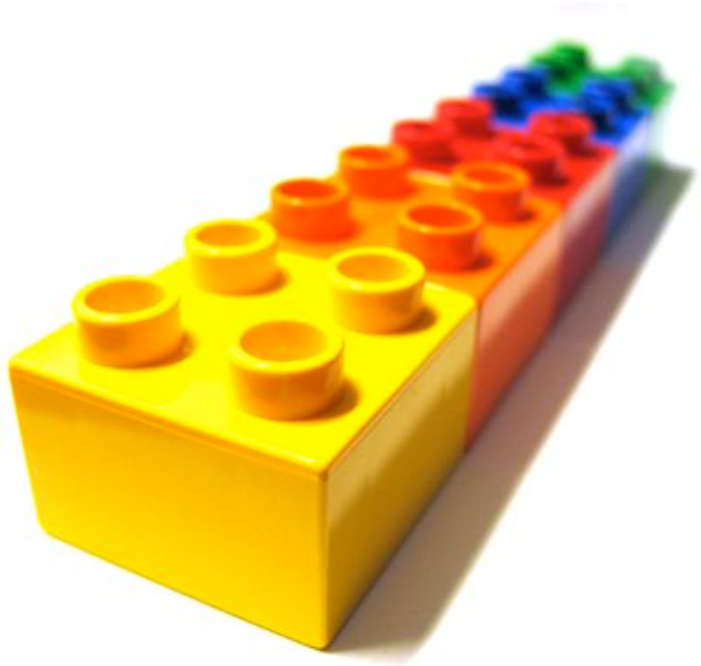
```
>>> 4
>>> 'Hallo!'
>>> True
```



Was ist das hier eigentlich?
Daten? Datentypen?

Das sind Objekte!

Objekte sind die Bausteine in einer Objekt-Orientierten Programmiersprache wie Python.



Was bist du denn für ein Typ?

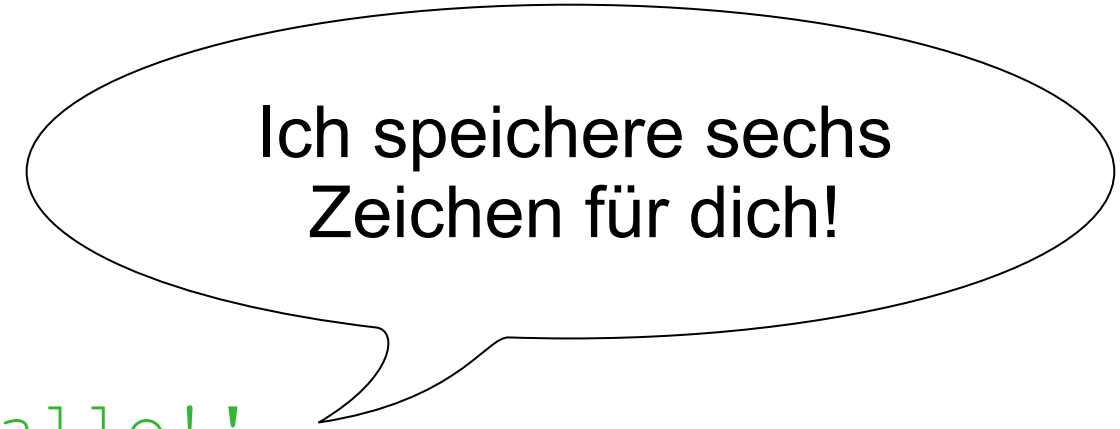
Zurück zum Anfang: Wir wissen Objekte haben einen Typ der bestimmt, was man damit machen kann.

```
>>> type(3)
<class 'int'>
>>> type(3 / 2)
<class 'float'>
>>> type(4 == 3)
<class 'boolean'>
```

Benutze die Funktion `type()` um den Typ eines Ausdrucks oder einer Variable herauszufinden.

Objekte

... wir haben auch gesehen, dass Objekte Daten speichern.




```
>>> 'Hallo!'
```

Objekte

Jetzt werden wir sehen, wie wir:

- Infos über die Daten in einem Objekt bekommen.
- Daten von Objekten verändern.

Objekte haben Methoden, um etwas mit den Daten zu tun.

(String) Methoden

In einem String etwas ersetzen:

```
>>> "I can haz Cheeseburger?".replace(
    "Cheeseburger","cheezburger")
'I can haz cheezburger?'
```

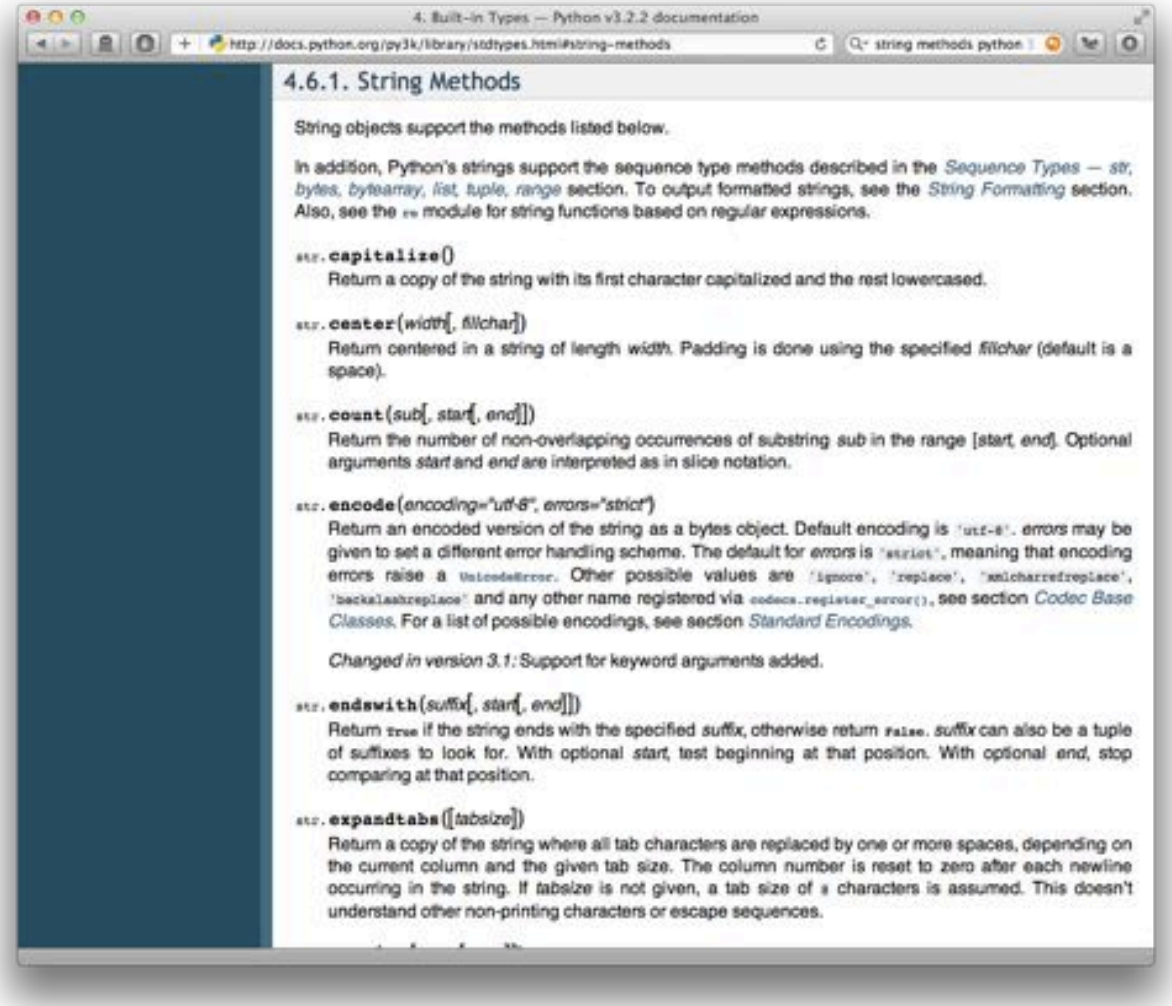
Parameter

Herausfinden, ob in dem String eine Zahl steht:

```
>>> "333".isnumeric()
True
```

Rückgabewert (Return Value)
der Methode

(String) Methoden



Dokumentation

Alle Methoden von Strings findest du in der Dokumentation unter:
http://docs.python.org/py3k/library/stdtypes.html#string-methods

Let's talk about Style

Don't repeat yourself

Es kommt oft vor, dass du bestimmte Werte immer wieder brauchst.
(Stell dir ein Programm vor, das 100 Zahlen auf Teilbarkeit durch 2 testet.)

Wenn sich ein solcher Wert (2) ändert, müssen alle Stellen an der er Vorkommt angepasst werden...
(In unserem Beispiel wohl an 100 Stellen. Yikes!)


Good Code
is DRY
Don't Repeat Yourself

Variablen zur Rettung!

Variablen

Variablen geben einem Wert einen Namen.

http://www.flickr.com/photos/tbuser/2679445409

```
>>> meaning_of_life = 40 + 2
```

Statt den Wert zu wiederholen, kann man dann den Namen an seiner Stelle verwenden:

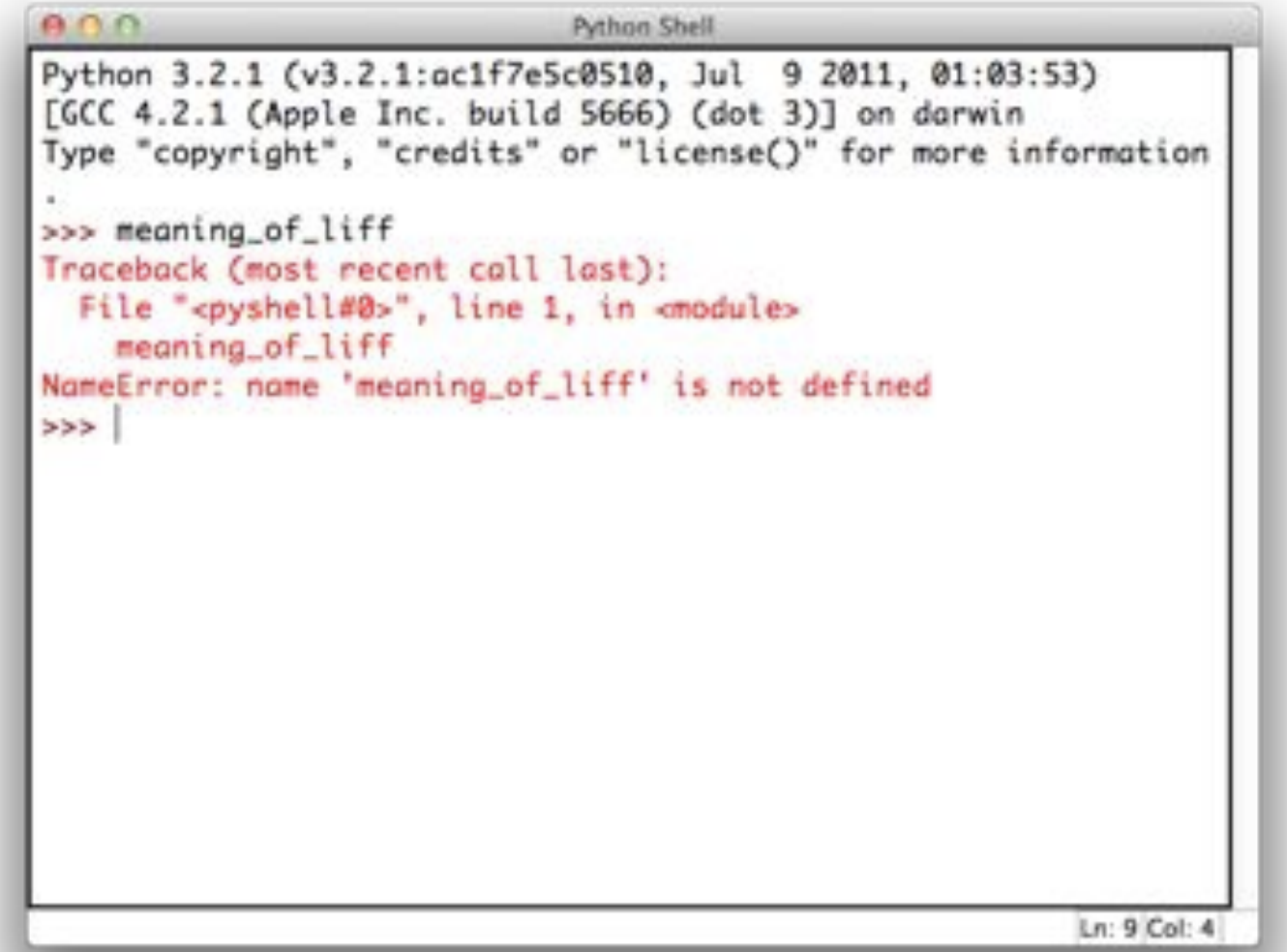
```
>>> 1 + meaning_of_life
43
```

Vorrang

Die Zuweisung **=** hat den niedrigsten Vorrang aller Operatoren. Die "rechte Seite" wird also zuerst ausgewertet und dann das Ergebnis gespeichert!

Fehlermeldungen...

Fragen wir nach einem Namen, den es noch nicht gibt ...



Regeln für Variablennamen

Wählst du Namen sorgfältig, kannst du damit Programme lesbarer und deine Absichten deutlich machen!

```
>>> meaning_of_life = 42   # Guter Name!
>>> var001 = 42   # Mieser Name!
```

Python erlaubt jede Kombination aus Buchstaben, Nummern und Unterstrichen (_), die nicht mit einer Zahl beginnt.

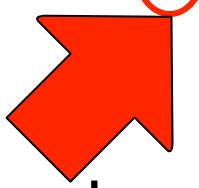
```
>>> 001var = 42   # Fehlermeldung!
```

Reservierte Namen

Einige Namen sind reserviert und können nicht für Variablen verwendet werden: http://docs.python.org/reference/lexical_analysis.html#keywords

Moment mal...

```
>>> meaning_of_life = 42  #Guter Name!
```



Was ist das denn??

Moment mal...

```
>>> meaning_of_life = 42  #Guter Name!
```

Was ist das denn??

Die Raute leitet einen Kommentar ein.
Alles ab der Raute bis zum Zeilenende wird vom Interpreter ignoriert.

Kommentare sind dafür da, Code zu erklären und ihn für Menschen verständlicher zu machen.

Bedingungen stellen

Bedingtes Ausführen

Bisher hatten alle unsere Beispiele etwas gemeinsam.

Der Computer hat die Anweisungen in der Reihenfolge abgearbeitet, in der wir sie eingegeben haben.

```
"Erste Zeile"
"Zweite Zeile"
"Dritte Zeile"
"Vierte Zeile"
```

Bedingtes Ausführen

Gewisse Dinge passieren aber nur unter Bedingungen.
z.B. Wenn du zur dunklen Seite der Macht überläufst, bekommst du Kekse.

In Python:

```
if you.come_to_the_dark_side():
␣␣␣␣you.get_cookies()
```

Wenn `you.come_to_the_dark_side()` nicht True ist, wird `you.get_cookies()` nicht ausgeführt.

http://www.thinkgeek.com/tshirts-apparel/miscellaneous/d0b2/

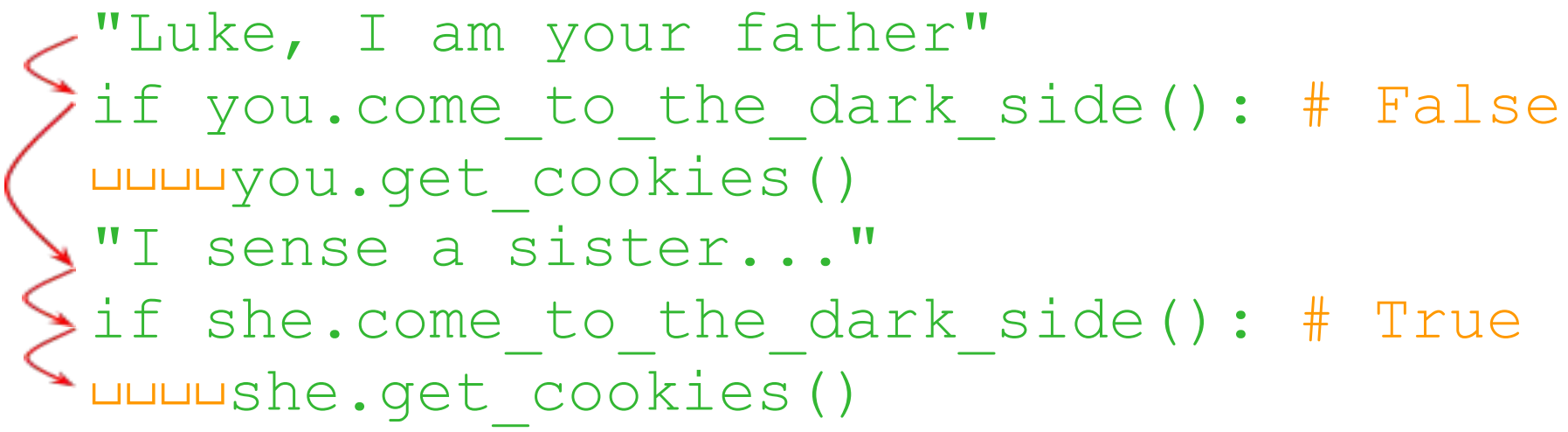
Blöcke

Blöcke markieren den Bereich der bedingt ausgeführt wird:

```
"Vor dem Block"
if True:
␣␣␣␣"Wir sind im Block"
"Nicht mehr im Block"
```

Eingeleitet wird ein Block von einem :
Der Block selbst ist mit vier Leerzeichen (␣␣␣␣) eingerückt.
Er endet, wenn diese Einrückung endet.

Bedingtes Ausführen



Man nennt diese roten Pfeile den Kontrollfluss.
Anweisungen die den Kontrollfluss ändern, heissen Kontrollstrukturen.

Bedingtes Ausführen

Trifft eine Bedingung nicht zu, soll evtl. eine alternative Anweisung ausgeführt werden. Dafür gibt es das Schlüsselwort `else`:

```
if you.come_to_the_dark_side(): # False
␣␣␣␣you.get_cookies()
else:
␣␣␣␣you.do_not_get_cookies()
```

Bedingtes Ausführen

Gibt es mehrere Bedinungen, können wir elif verwenden:

```
x = 3
if x < 0:
    "Negative Zahl"
elif x > 0:
    "Positive Zahl"
else:
    "Die Zahl ist Null"
  "Block ist zuende"
```

http://www.flickr.com/photos/sminor/2843977224/

`elif` ist eine Kurzform für "else if"

(Image origin unknown)

Let's Code!

Live coding

Live coding learnings

Was wir beim Live coding gelernt haben:

Ausgabe mit `print()`

Eingabe mit `input()`
Optional mit Argument: "Prompt", gibt String zurück

Konvertieren von String in Zahl:

```
>>> int("235")
235
>>> float("1.5")
1.5
```

Vielen Dank für eure Aufmerksamkeit

Übungen um 14:00 Uhr in den Fischer-Räumen