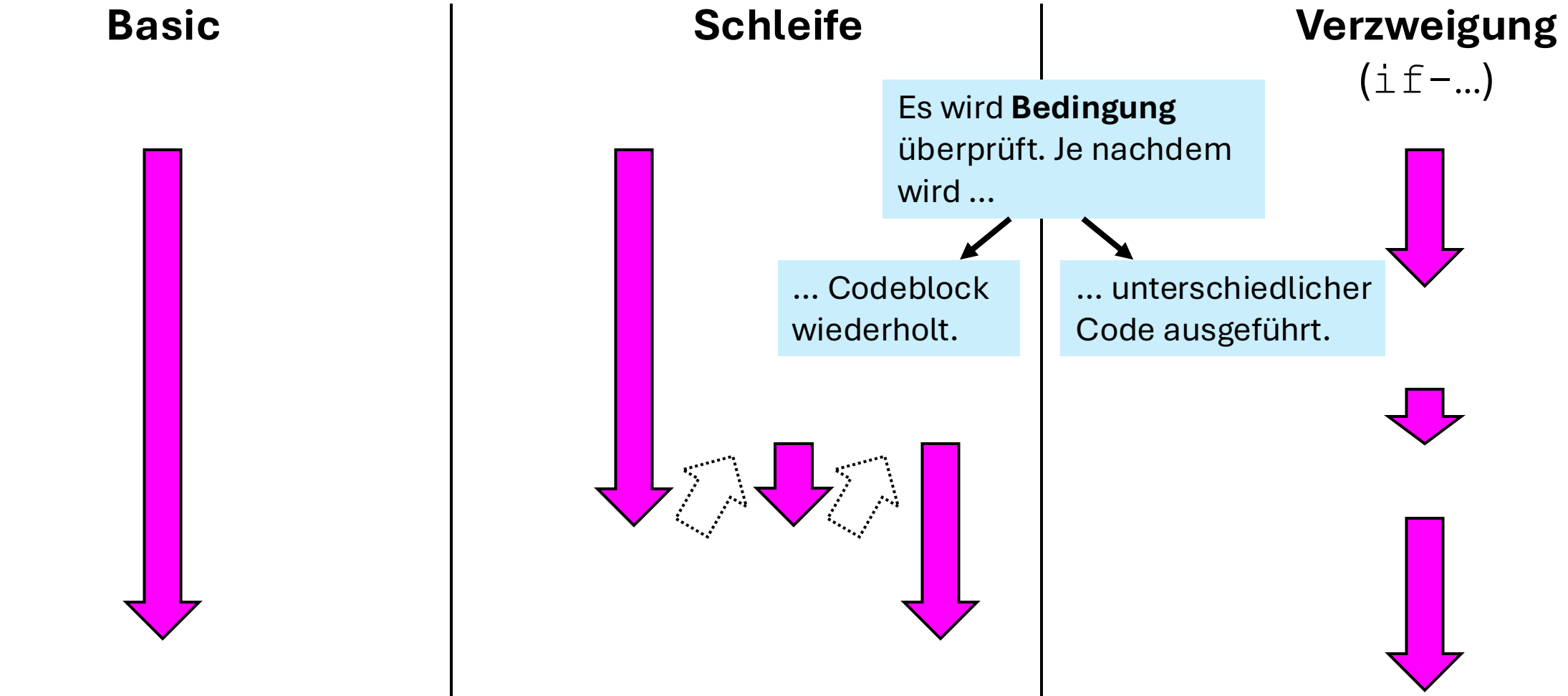
Programmieren I

Bedingungen & Schleifen

Andreas Schärer, KSR

Erinnerung: Code Flow



Bedingungen

Bedingungen

- Vergleichsoperatoren

Vergleichsoperator	Überprüft ob ...
`x == y`	`x` gleich `y` ist.
`x != y`	`x` *un*gleich `y` ist.
`x > y`	`x` grösser als `y` ist.
`x < y`	`x` kleiner als `y` ist.
`x >= y`	`x` grösser oder gleich `y` ist.
`x <= y`	`x` kleiner oder gleich `y` ist.

- Verwenden diese, um **Bedingungen zu überprüfen**
- Beispiel:

```
zahl = 7
print(zahl > 5)
```

Ausgabe: `True`

Bedingung, hat deshalb Wert `True` oder `False`.

- Bedingung kann *entweder* **erfüllt** (`True`) oder **nicht erfüllt** (`False`) sein.
- `True` & `False` sind eigener *Datentyp*: **Boolean** (kurz: **Bool**)

Verzweigungen

- Beispiel (sinnloser Code)
- Was ist Output?
- Sicher?
- Ausgabe: `"Pasta"`

Variable `cond` (condition) hat also Wert False.

Bedingung. Nimmt Wert `False` an, da 2 <= 0 nicht stimmt.

```
x = 7
y = 5

cond = x - y <= 0

if cond:
    print("Pizza")
else:
    print("Pasta")
```

Einfachgleich = vs. Doppelgleich ==

Einfachgleich =

- ist **Zuweisung**
- weist Variable Wert zu
- `alter = 16`

Doppelgleich ==

- ist **Vergleichsoperator**
- wird in *Bedingungen* verwendet
- `if alter == 16:`
  `…`

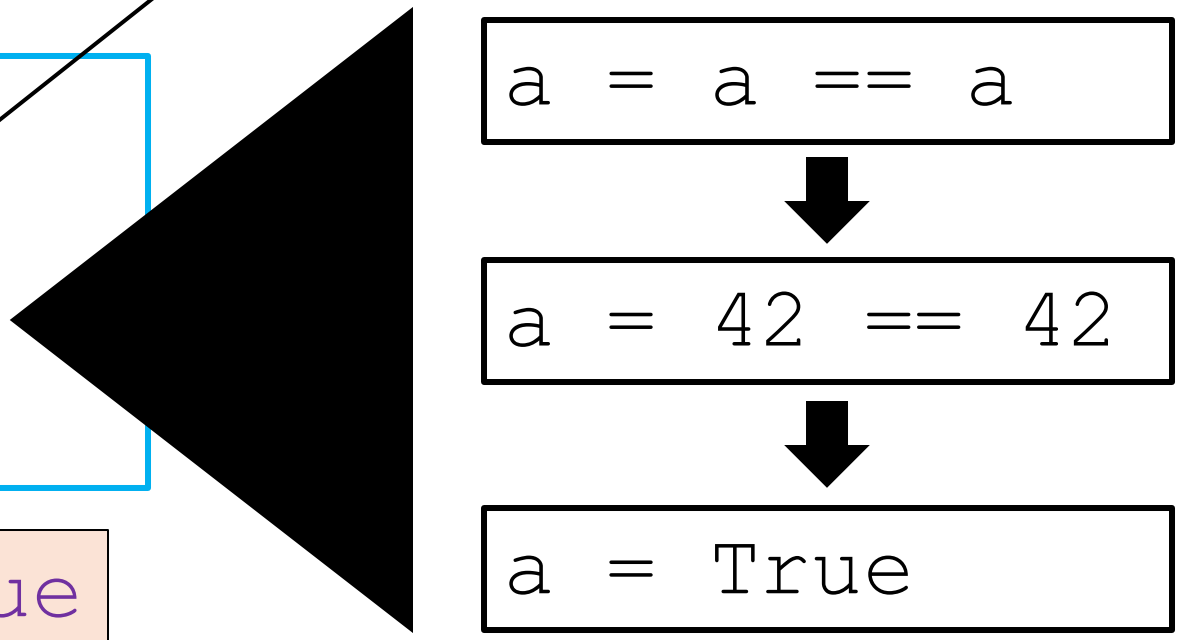
Variable `alter` wird hier *nicht verändert*. Wird nur überprüft, ob sie gerade den Wert 16 hat.

Boolean Quiz

«Vergleichsoperator (==) vor Zuweisung (=)!»

```
# 1)
a = 42
a = a == a
print(a)
```

Ausgabe: `True`

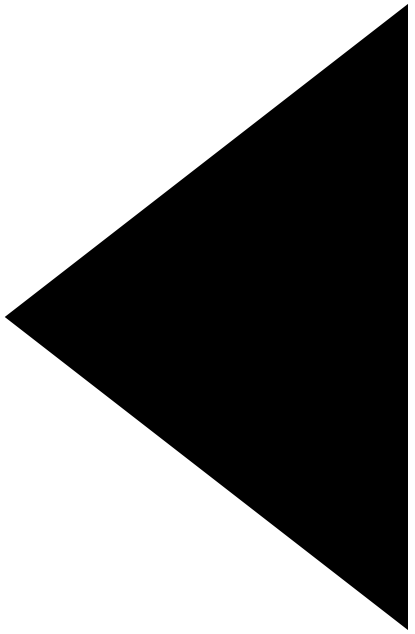


Boolean Quiz

«Vergleichsoperator (==) vor Zuweisung (=)!»

```
# 2)
a = 42
a = a != a - 1
print(a)
```

Ausgabe: `True`



```
a = a != a - 1
```

↓

```
a = 42 != 41
```

↓

```
a = True
```

Schleifen

Repeat-Schleife

```
from gturtle import *

wanda = Turtle()

repeat 4:
    wanda.forward(100)
    wanda.left(90)
```

- Bisher Schleifen mit `repeat` programmiert.
- Problem mit `repeat`:
  - Spezielfeature von WebTigerPython / TigerJython
  - Ist also *kein richtiges* Python
  - Warum gibt denn repeat? Weil sehr einfach, gut für Anfänger
- Wollen deshalb eine ‘richtige’ Python-Schleife kennen lernen
- **While-Schleife**!

While-Schleife

- while bedeutet «während»
- Syntax:

```
while <Bedingung>:
    # Eingerückter Codeblock,
    # der wiederholt werden soll.

# Code NACH while-Schleife
```

- Idee:
  - *Während* die *Bedingung erfüllt* ist, wird der zugehörige Codeblock *wiederholt*.
  - Nach jedem Durchgang wird Bedingung wieder überprüft.
  - Sobald Bedingung nicht mehr erfüllt, springt zu Code nach der Schleife.

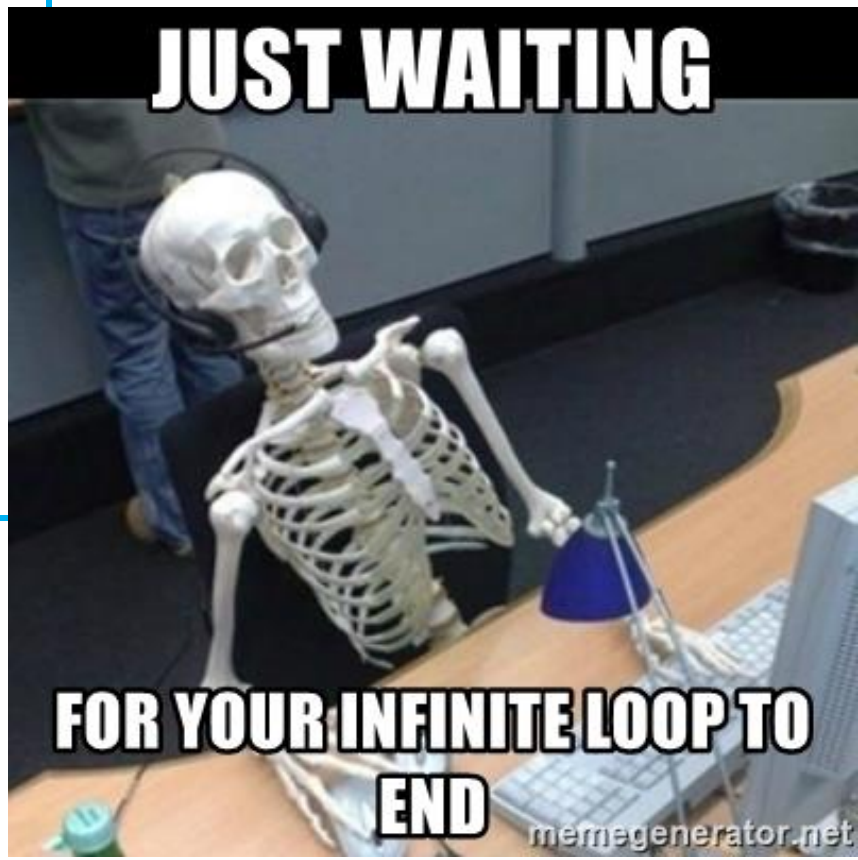
Quadrat mit while

- Turtle-Quadrat mit repeat
- Und jetzt mit while!
- Wollen Codeblock 4x wiederholen.
- Idee:
  - Definiere **Variable** `count = 0` die *mitzählt*, wie Codeblock bereits ausgeführt wurde. Solche Variable nennt auch **Counter**.
  - Nach jedem Durchlauf des Codeblocks wird **Variable um 1 erhöht**, also `count = count + 1`
  - Wollen Codeblock wiederholen, solange `count` noch nicht Wert 4 erreicht hat. **Bedingung** ist als `count < 4`.

```
from gturtle import *

wanda = Turtle()

count = 0
while count < 4:
    wanda.forward(100)
    wanda.left(90)
    count = count + 1
```

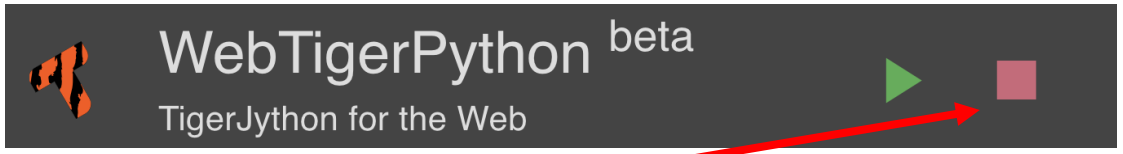
Endlosschleifen

```
from gturtle import *

wanda = Turtle()

count = 0
while count < 4:
    wanda.forward(100)
    wanda.left(90)
```



- Problem hier?

- Zeile `count = count + 1` vergessen.
- Counter `count` bleibt also immer auf Wert 0.
- Somit Bedingung `count < 4` IMMER erfüllt.
- -> **Endlosschleife**!
- Muss Programm manuell *abbrechen*.



No more repeat!

No more repeat!

No more repeat!

Ab sofort:
Striktes repeat-Verbot!