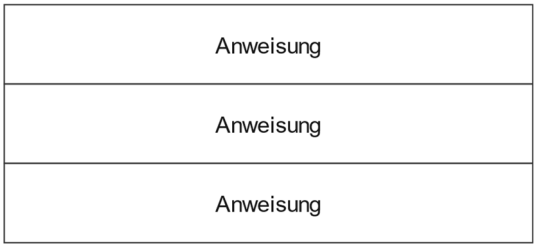
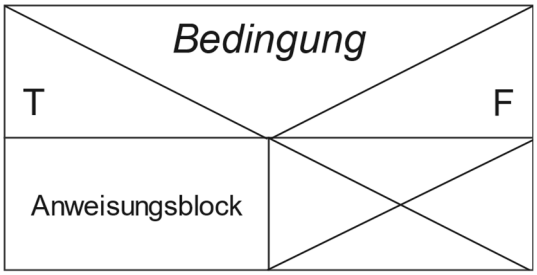
Cheat-Sheet



Anweisungsblock

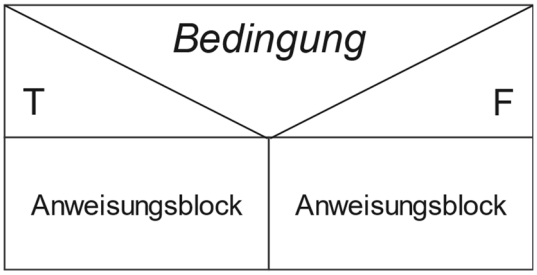
- einzelne Anweisungen werden nacheinander ausgeführt



Bedingte Anweisung

- Ein Anweisungsblock wird nur ausgeführt, wenn eine Bedingung den Wahrheitswert *wahr (true)* zurückgibt.

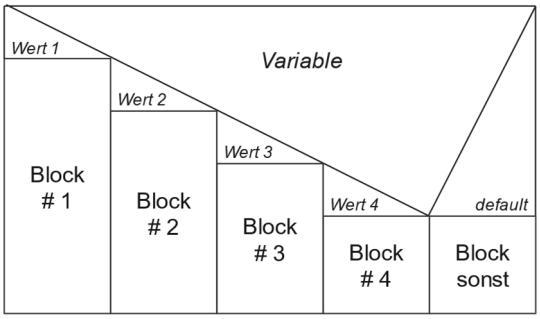
```
if a == 1:
  doSomething(1)
```



Verzweigung

- Trifft eine Bedingung zu, wird der *true*-Zweig ausgeführt, sonst der *false*-Zweig.

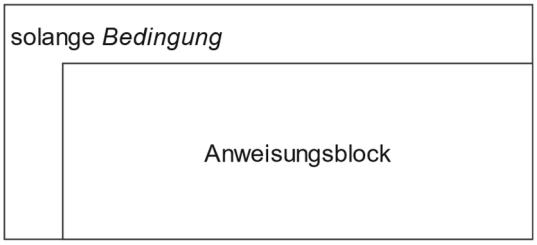
```
if a == 1:
  doSomething(1)#true-Anweisung
else:
  doSomething(2)#false-Anweisung
```



Mehrfachauswahl

- Eine Variable wird auf bestimmte Werte geprüft. Je nach Wert wird ein anderer Anweisungsblock ausgeführt. Für alle nicht explizit ausgeführten Werte wird der *default*-Block

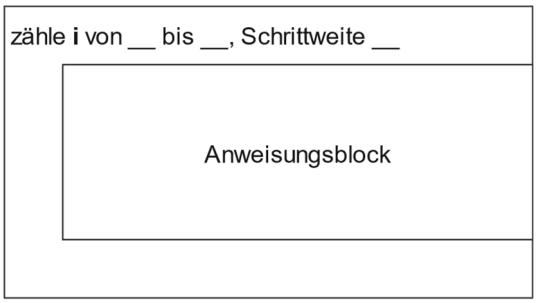
```
if a == 1:
  doSomething(1) #Block #1
elif a== 2:
  doSomething(2) #Block #2
elif a== 3:
  doSomething(3) #Block #3
else:
  doSomething(1000) #Block sonst
```



while-Schleife

- Eine Schleife ausgeführt, solange eine Bedingung gilt

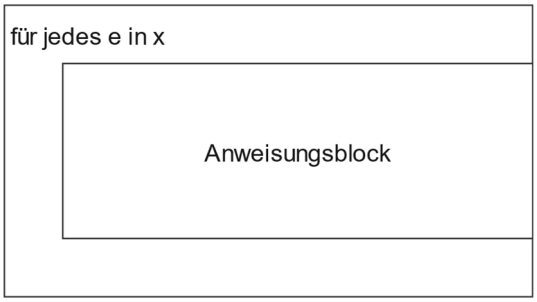
```
while a < 10:
  print(a)
  a = a + 1
```



for-Schleife / Zählschleife

- Eine Variable wird bei jeder Schleifeniteration von einem Startwert bis zu einem Endwert hochgezählt. Es kann optional eine Schrittweite angegeben werden. Ist keine angegeben, ist die Schrittweite 1.

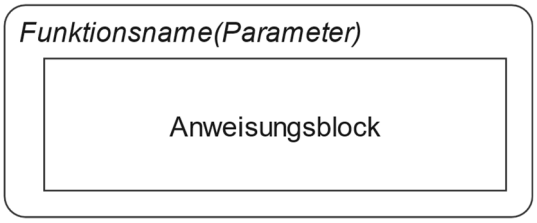
```
for i in range (0,10):
  print(i)
```



foreach-Schleife

- Für jedes Element einer Liste, eines Arrays oder eines Strings wird ein Anweisungsblock ausgeführt

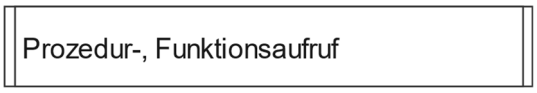
```
word = "Python"
for letter in word:
  print(letter)
```



Definition einer eigenen Funktion (mit Parametern)

- Zur Lösung eines wiederkehrenden Problems kann eine eigene Funktion bzw. Prozedur definiert werden

```
def foobar(para1):
  para1 = para1+1
  return para1
```



Aufruf einer eigenen Prozedur/Funktion in einem Anweisungsblock

Boolsche Grundoperatoren

not

A	not A
0	1
1	0

and

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

or

A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

1 … wahr / ja / true / Python: True
0 … falsch / nein / false / Python: False

Operatorenhierarchie: NAO (not-and-or, d.h., zuerst wird negiert, dann wird and ausgeführt, zum Schluss or). Diese Hierarchie kann durch Klammern gebrochen werden.

Vergleiche

==	ist gleich	a < b	a ist kleiner als b
!=	ungleich	a >= b	a ist größer oder gleich b
a > b	a ist größer als b	a <= b	a ist kleiner oder gleich b