

Einführung in die Programmierung

Ronja Düffel
WS2019/20

24. September 2019

RBI-Account

- Account für das Rechnernetz der Informatik
- RBI-Account \neq HRZ-Account
- zum Arbeiten an und auf den Rechnern des Instituts

Programmieren (vereinfacht)

- 1 Problem beschreiben und analysieren
- 2 Entwicklung und Beschreibung einer Lösung
- 3 Übertragung/Umsetzung in eine Programmiersprache
- 4 Test des Programms

Programmiersprachen

Maschinenprogramme

- können direkt vom Computer verstanden und ausgeführt werden.
- bestehen aus Bit-Folgen (0-en und 1-en),
- für Menschen nahezu unverständlich

Höhere Programmiersprachen

- für Menschen besser zu lesen und zu verstehen
- *Quelltext* = Programm in höherer Programmiersprachen
- für Computer unverständlich
⇒ Quelltext muss in Maschinenprogramm übersetzt werden!



- Höhere Programmiersprache
- Als Lehrsprache entwickelt
 - Übersichtlichkeit
 - leichte Erlernbarkeit
- benutzt zur Übersetzung einen Interpreter

IDLE

- Entwicklungsumgebung für Python
- unterstützt bei der Programmierung
- Interaktive Python-Shell mit
 - Syntax-Highlighting
 - Autovervollständigung
 - Anzeige der Befehlssyntax
- Editor zum Programme schreiben

The screenshot shows a terminal window titled '*Python 3.5.2 Shell*'. The menu bar includes 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area contains the following text:

```
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linux
Type "copyright", "credits" or "license()" for more information.
>>> print("Willkommen zum Vorkurs!")
Willkommen zum Vorkurs!
>>> 
```

A dropdown menu is open, showing a list of commands: 'pow', 'print', 'property', 'quit', 'range', and 'repr'. The status bar at the bottom right indicates 'Ln: 6 Col: 5'.

The screenshot shows the same terminal window. The text area now includes a second line of code:

```
>>> print("Willkommen zum Vorkurs!")
Willkommen zum Vorkurs!
>>> print(
print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
```

A tooltip box highlights the function signature for the `print` function: `print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)`. The status bar at the bottom right indicates 'Ln: 6 Col: 10'.

Datentypen: Wahrheitswerte und Zahlen

Wahrheitswerte Bool

- True False
- Operatoren: and, or und not

Zahlen

- Ganzzahlen int (fehlender Dezimalpunkt)
- Gleitkommazahlen float (Dezimalpunkt)
- Operatoren: +, -, *, /, //, %, **

```
>>> 2.0*3
6.0
>>> 2*3
6
>>> 3.0+2
5.0
>>> 6/3
2.0
>>> 5//2
2
>>> 5%1
0
>>> 5<2
False
>>> 2.0==2
True
>>> |
```

Datentypen: Zeichenketten

Zeichenketten string

- in Anführungszeichen

```
'Und nun zu etwas ganz anderem...'
```

```
''Setz dich, nimm dir'n Keks...''
```

```
'''Er sagte: ''Geht's jetzt los?'' '''
```

```
>>> 'Jeder nur 1 Kreuz'
'Jeder nur 1 Kreuz'
>>> 'Setz dich, nimm dir'n Kecks, mach's dir schön bequem'
SyntaxError: invalid syntax
>>>
>>> "Setz dich, nimm dir'n Kecks, mach's dir schön bequem"
"Setz dich, nimm dir'n Kecks, mach's dir schön bequem"
>>>
>>> '''"Da ist das Untier!"
"Wo, hinter dem Karnickel?"
"Es IST das Karnickel!"""
'"Da ist das Untier!"\n"Wo, hinter dem Karnickel?"\n"Es IST das Karnickel!'"
>>>
```

Built-in-Funktionen (int, float)

- `abs(x)`: gibt den Betrag $|x|$ der Zahl x zurück

```
>>> abs(-4.73)
4.73
```

- `max(a,b)`: gibt das Maximum von a und b zurück

```
>>> max(3.2, 4)
4
```

- `min(a,b)`: gibt das Minimum von a und b zurück

```
>>> min(3.2, 4)
3.2
```

- `round(x,n)`: gibt den gerundeten Wert von x zurück
(n -Nachkommastellen)

```
>>> round(3.84629,2)
3.85
```

Built-in-Funktionen (string)

- `<string>.isalpha()`: prüft, ob String nur aus Buchstaben besteht

```
>>> "hallo".isalpha()
True
>>> "hallo vorkurs".isalpha()
False
```

- `<string>.isdigit()`: prüft, ob String nur aus Ziffern besteht

```
>>> "284".isdigit()
True
>>> "2.84".isdigit()
False
```

- `ord('c')`: gibt die Nummer des Zeichens `c` zurück
- `chr(<int>)`: gibt das Zeichen zur Ganzzahl `i` zurück

```
>>> ord('W')
87
>>> chr(37)
'%'
```

- `+`: fügt zwei Strings zusammen (Operator)

nützliche Funktionen

- `int(x)`: wandelt `x` in einen ganzzahligen Wert um

```
>>> int(2.9)
2
```

- `float(x)`: wandelt `x` in einen Gleitkommawert um

```
>>> float(5)
5.0
```

- `str(x)`: wandelt `x` in einen String um

```
>>> str(3.57)
'3.57'
```

- `input()`: Benutzereingabe lesen

- `print()`: Bildschirmausgabe

Variablen

- Möglichkeit Informationen zu speichern
- bezeichnet einen bestimmten Speicherbereich
- hat immer:
 - einen Namen
 - einen Typ
 - einen Wert
- in Python durch Zuweisung `<Name> = <Ausdruck>` angelegt

! Vergibt man denselben Namen zweimal, so wird der Wert der Variablen überschrieben !

Variablen, Beispiel

```
>>> a = 17
>>> b = 2.98
>>> a + b
19.98
>>>
>>> a = "hallo"
>>> a + b
Traceback (most recent call last):
  File "<pyshell#110>", line 1, in <module>
    a + b
TypeError: Can't convert 'float' object to str implicitly
>>>
>>> b = str(b)
>>> a + b
'hallo2.98'
>>>
>>> c = 17+3
>>> d = c / 5
>>> d
4.0
>>> c
20
>>>
```

Variablennamen

- Variablennamen müssen mit Buchstaben (a,...,z,A,..., Z) oder Unterstrich (_) beginnen.

```
>>> 2fast = 300
SyntaxError: invalid syntax
---
```

- können beliebig lang sein und ab 2.Zeichen Ziffern(0,..., 9) enthalten.
- Groß- und Kleinschreibung ist immer relevant!
- Umlaute (ä,ö,ü,...) sollten nicht verwendet werden
- Sonderzeichen(!,&, ...) sind nicht erlaubt

```
>>> Ernie&Bert = "Friends"
SyntaxError: can't assign to operator
>>>
```

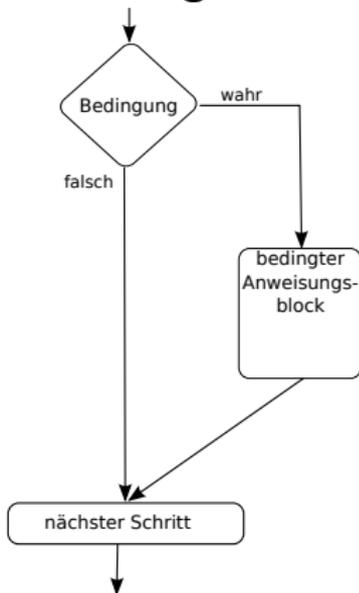
- Schlüsselwörter sind als Variablennamen verboten (z.B. and, or, if, in ...)

Kontrollstrukturen

- steuern den Programmablauf

Verzweigung

if-Anweisung



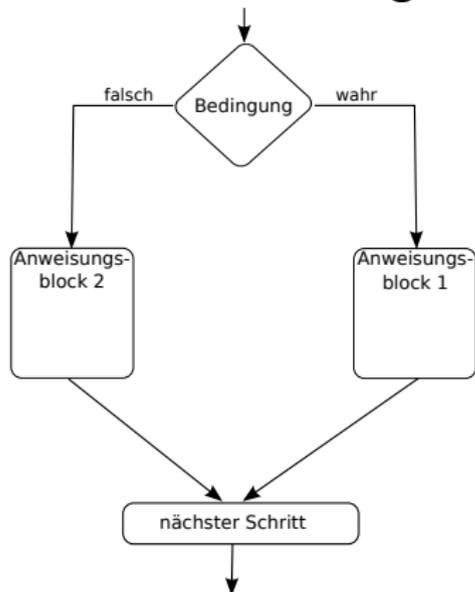
```
>>> # Beispiel für eine if-Anweisung
>>> if 4 > 7:
    print("Hoppla!")
```

```
>>> if not(4 > 7):
    print("Alles ok!")
```

```
Alles ok!
>>>
```

Verzweigung

if...else-Anweisung



```

1 # Beispiel für if...else
2
3 if 4 > 7:
4     print("Hoppla!")
5 else:
6     print("Alles ok!")
7 print("And now for something
   completely different")
  
```

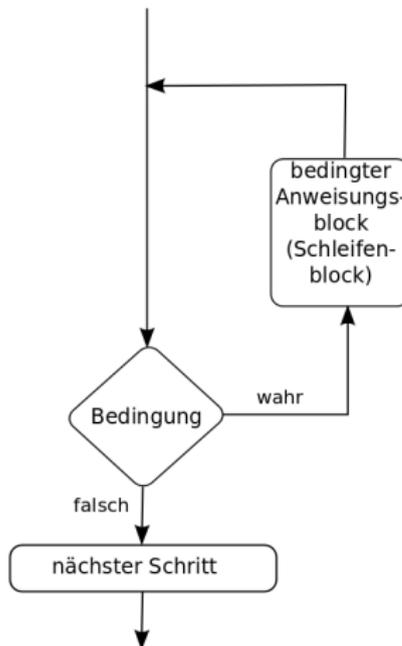
Alles ok!

And now for something completely different

>>>

Schleifen: while-Schleife

while-Schleife



while-Schleifen

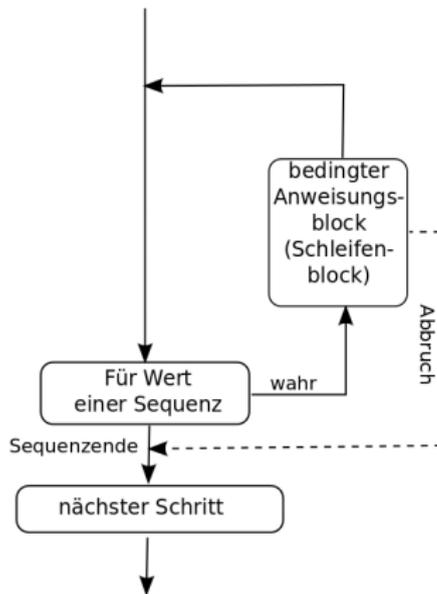
```
1
2 count = 0
3
4 # vorprüfende while-Schleife
5 while count < 9:
6     print(count)
7     count = count + 1
8 print('Ende der vorprüfenden while-Schleife')
9
10 # nachprüfende while-Schleife
11 while True:
12     print(count)
13     count = count + 1
14     if count >= 9:
15         break
16 print('Ende der nachprüfenden while-Schleife')
```

Ausgabe while-Schleifen

```
0  
1  
2  
3  
4  
5  
6  
7  
8  
Ende der vorprüfenden while-Schleife  
9  
Ende der nachprüfenden while-Schleife  
>>>
```

Schleifen: for-Schleife

for-Schleife



for-Schleife

```
1 for i in 'Python':  
2     print(i)
```

```
P  
y  
t  
h  
o  
n  
>>>
```

range()-Funktion

```
range(start, stop[, step])
```

- erzeugt eine Sequenz von Zahlen von start bis stop-1 (!!!)
- wird start nicht angegeben, beginnt die Sequenz bei 0
- wird keine Schrittweite (step) angegeben, wird immer um 1 erhöht

Beispiel

```
1 for i in range(5):  
2     print(i)
```

```
0  
1  
2  
3  
4  
>>> |
```

Beispiel

```
1 for i in range(2,8,2):  
2     print(i)
```

```
2  
4  
6  
>>>
```

Schleifen-Kontrollanweisungen

- `break`: beendet Schleife sofort; Programmfluss springt zur ersten Anweisung nach dem Schleifenrumpf
- `continue`: bricht aktuellen Schleifendurchlauf ab; Programmfluss springt zum Schleifenkopf
- `pass`: wenn Python eine Anweisung verlangt, aber nichts gemacht werden soll

Beispiel continue

```
1 for i in range(5):  
2     if i == 3:  
3         continue  
4     print(i)
```

```
0  
1  
2  
4  
>>>
```

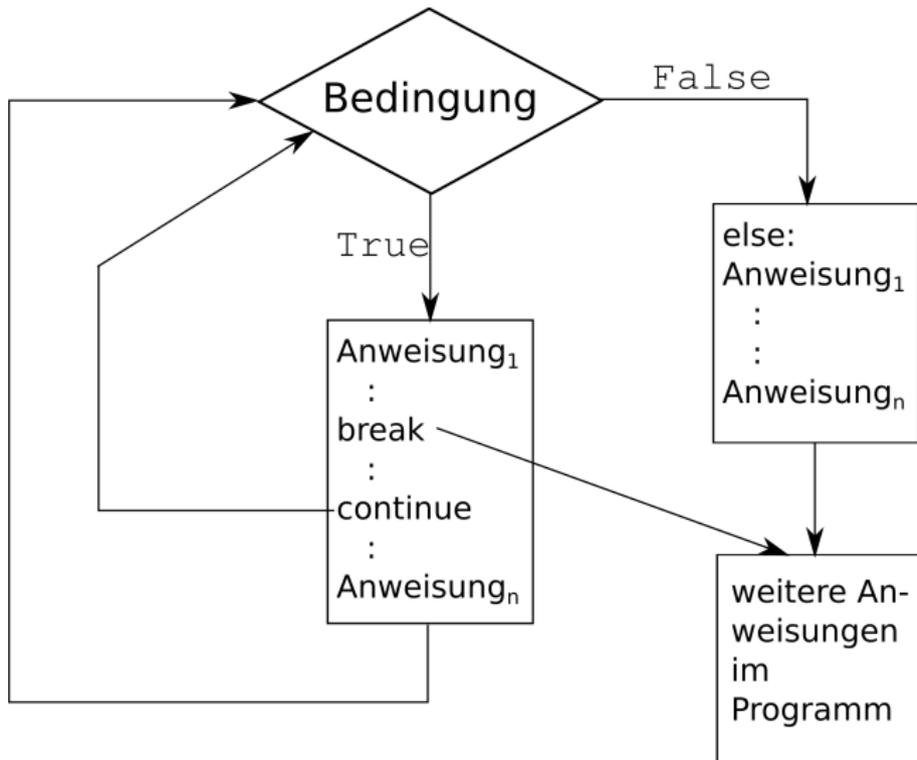
Beispiel break

```
1 #Beispiel: Verwendung von break-Anweisung
2 # Nutzereingabe lesen, bis Ende gewünscht
3
4 while True:
5     eingabe = input("Geben Sie etwas ein: ")
6     if eingabe == 'ende':
7         break
8     print("Die Länge des Strings ist:", len(eingabe))
9 print("Fertig")
```

Ausgabe:

```
Geben Sie etwas ein: Hallo Vorkurs
Die Länge des Strings ist: 13
Geben Sie etwas ein: Keine Ahnung
Die Länge des Strings ist: 12
Geben Sie etwas ein:
Die Länge des Strings ist: 4
Geben Sie etwas ein:
Die Länge des Strings ist: 1
Geben Sie etwas ein: ende
Fertig
```

Kontrollfluss else-Zweig in while-Schleife



Beispiel else-Zweig in while-Schleife

```
1 #Beispiel: else-Zweig in while-Schleife
2 # Ratespiel
3
4 geheimeZahl = 57
5 tipp = 1
6 while tipp != geheimeZahl:
7     tipp = int(input('Rate: '))
8     if tipp > 0:
9         if tipp > geheimeZahl:
10            print('zu gross')
11            elif tipp < geheimeZahl:
12                print('zu klein')
13        else:
14            print('Schade, dass du aufgibst :(')
15            break
16 else:
17     print ('Glückwunsch, du hast die Zahl erraten!')
```

Fragen?

?