

Kontrollstrukturen

Claas de Boer, Tilman Hinnerichs

12.11.2020

Python-Grundlagen

1. Wie man ein Python Programm schreibt
2. Was sind Kontrollstrukturen?
3. Verzweigungen ("If-Statements")
4. While-Schleife
5. Intermezzo: Listen
6. For-Schleife

Wie man ein Python Programm schreibt

Die Python shell

- erlaubt es schnell Code zu testen
- hilft schnell Dinge auszurechnen
- ist kein Python Programm

Erste Schritte

```
1      >>> print("Hello World!")
2      Hello World!
3      >>> 3+3
4      6
5      >>> print(3+3)
6      6
7      >>> 3 * 3
8      9
9      >>> 3 / 3
10     1.0
11     >>> 3 ** 3
12     27
13     >>> 4 % 3
14     1
15
```

Variablenzuweisungen

```
1      >>> a = 5
2      >>> a
3      5
4      >>> b = a+2
5      >>> b
6      7
7      >>> a*b
8      35
```

DIY: Python Programm schreiben

1. Öffnet den Editor eurer Wahl
2. Schreibt euren Programmcode
3. Speichert die Datei als *.py ab
4. Führt die Datei mit dem Interpreter eurer Wahl aus

Oder

1. Macht dies alles zusammen in der Entwicklungsumgebung eurer Wahl, z.B.: <https://repl.it/languages/Python3>

DIY: Python Programm schreiben

1. Öffnet den Editor eurer Wahl
2. Schreibt euren Programmcode
3. Speichert die Datei als *.py ab
4. Führt die Datei mit dem Interpreter eurer Wahl aus

Oder

1. Macht dies alles zusammen in der Entwicklungsumgebung eurer Wahl, z.B.: <https://repl.it/languages/Python3>

Demonstriere Shell vs. Script

Fragen?

Schreibe ein Python Programm, welches

1. die Fläche eines Kreises mit $r = 4cm$ und $\pi = 3.14$ berechnet.
2. die Zahlen von 1 bis 20 addiert.

Was sind Kontrollstrukturen?

Kontrollstrukturen: Wozu?

- Strukturieren Programmverhalten
- für alle imperativen Sprachen ähnlich
- z.B.: Verzweigung, Zählschleife

Python kennt:

- Ein- / Zweiseitige Verzweigung: `if .. : / if .. : .. else: ..`
- Zählschleife: `for .. in .. :`
- Kopfgesteuerte Schleife: `while .. :`

Python kennt:

- Ein- / Zweiseitige Verzweigung: `if .. : / if .. : .. else: ..`
- Zählschleife: `for .. in .. :`
- Kopfgesteuerte Schleife: `while .. :`

Darüber hinaus heute:

- Kommentare
- Komparatoren (`<`, `>`, `==`, `<=`, `>=`)

Verzweigungen (" If-Statements")

Einseitige Verzweigungen

```
1      >>> 5==5
2      True
3      >>> 5>6 and 5==5
4      False
```


Einseitige Verzweigungen

```
1      >>> 5==5
2      True
3      >>> 5>6 and 5==5
4      False
```

```
1      if a == b:
2          print ("a is equal to b!")
3      print ("this is printed every time. bye.")
```

Einseitige Verzweigungen

```
1      >>> 5==5
2      True
3      >>> 5>6 and 5==5
4      False
5
```

```
1      if a == b:
2          print ("a is equal to b!")
3      print ("this is printed every time. bye.")
4
```

Beachte Indentation/Einrückung

```
1      if a == b:
2          print ("a is equal to b!")
3          print ("this is not printed every time. bye.")
4
```

Zweiseitige Verzweigungen

```
1  if a == b:
2      print ("a is equal to b!")
3  else:
4      print ("a is not equal to b!")
5      # this is a comment
6      # be careful: 1 == 1, but 1 != "1" !
7
8  print ("this is printed every time. bye.")
9
```

Verschachtelte Verzweigungen

```
1  if a < b:  
2      print ("a is less than b!")  
3  else:  
4      if a == b:  
5          print ("a == b!")  
6      print ("a >= b!")  
7      # is also executed when a == b!  
8
```

Kurzform elif Verzweigungen

Oder mit elif:

```
1  if a < b:
2      print ("a is less than b!")
3  elif a == b:
4      print ("a == b!")
5  else:
6      print ("a > b!")
7      # is not if when a == b!
8
```

While-Schleife

While-Schleife

→ Wiederholt, solange Bedingung erfüllt ist:

```
1   a = 10
2   while a >= 0:
3       a -= 1
4
```

Vorzeitiges Abbrechen

```
1  a = 10
2  while a > -1:
3      a -= 1
4      if a > 3:
5          continue # jump to next interation
6      print ("countdown:", a) # only printed for 3,2,1
7      if a == 1:
8          break # break out of loop immediately
9
```


- Indentation/Einrückung
- Kommentare
- Vergleichen von Werten
- Boolesche Werte (True, False, and, or)
- Kontrollstrukturen
 - if, else, elif
 - while und continue

Einige Aufgaben 2

Schreibe ein Python Programm, welches

1. für eine Variable a testet, ob diese zwischen 100 und 2000 liegt.
2. für eine Variable a testet, ob diese gerade ist.
3. alle gerade Zahlen zwischen 1 und 10 ausgibt.
4. zwei Variablen addiert. Wenn die Summe zwischen 15 und 20 liegt, gebe 20 zurück.

und printe eine passende Nachricht an den User.

Intermezzo: Listen

Was sind Listen?

Wie alltägliche Listen. In Python:

- `[]`, bzw. `[1,2,3]`
- Methoden: `insert()`, `append()`, `pop()`, `remove()`, `index()`
- siehe auch `help([])`

Wie verwende ich Listen

```
1  a = [1,5,3,2] # create new list
2  a.sort() # a is now sorted
3  print (a[3]) # 4th element of sorted list -> 5
4  highest = a.pop() # removes last element (5) and returns
   it
5  highest += 1
6  a.insert(0, highest) # add 6 at the start
7  print (a) # -> [6,1,2,3]
```

For-Schleife

Klassische Zählschleife gibt es nicht in Python

↪ for durchläuft Listen

```
1   a = [1,2,3]
2   for elem in a:
3       print (elem)
4
```

Wie zählen?

```
1   for index in range(10):  
2       print (index)  
3   # prints numbers 0 to 9  
4
```

```
1   a = [1,2,3]  
2   for index, elem in enumerate(a):  
3       print (index, elem)  
4
```


Einige Aufgaben 3

Schreibe ein Python Programm, welches

1. alle Zahlen von 1 bis 20 addiert
2. alle geraden Zahlen von 1 bis 20 ausgibt
3. eine gegebene Liste in ihrer Reihenfolge invertiert