

Skriptsprachen: Python

Kontrollstrukturen



Jan Krüger, Alexander Sczyrba

Technische Fakultät
Universität Bielefeld

8. Februar 2019

Python

—

Kontrollstrukturen

Kontrollstrukturen

- Kommandozeilenparameter

Kontrollstrukturen

- Kommandozeilenparameter
- Fallunterscheidungen
 - if, elif, else
 - Conditional Expressions

Kontrollstrukturen

- Kommandozeilenparameter
- Fallunterscheidungen
 - if, elif, else
 - Conditional Expressions
- Schleifen
 - while
 - for

Kommandozeilenparameter

- Modul sys importieren → Zugriff auf Systemkomponenten
`import sys`
- `sys.argv` enthaelt Kommandozeilen Parameter
- `sys.argv[0]` enthaelt den Skriptaufruf
- `sys.argv[1..]` enthaelt alle Parameter

Aufgabe

Schreibe ein Skript 'echo.py' das sich analog zum Shell Befehl 'echo' verhaelt.

if, else

```
if <Bedingung>:  
    Anweisung  
    ...  
    Anweisung  
else:  
    Anweisung  
    ...  
    Anweisung
```

elif

- Häufiges Konstrukt:

```
if <Bedingung1>:  
    Anweisung1  
if <Bedingung2>:  
    Anweisung2
```

elif

- Häufiges Konstrukt:

```
if <Bedingung1>:  
    Anweisung1  
if <Bedingung2>:  
    Anweisung2
```

- → **teuer**

elif

- Häufiges Konstrukt:

```
if <Bedingung1>:  
    Anweisung1  
if <Bedingung2>:  
    Anweisung2
```

- → **teuer**

- besser : elif

```
if <Bedingung1>:  
    Anweisung1  
elif <Bedingung2>:  
    Anweisung
```

Conditional Expressions

- Alternative zu if, else Anweisung
- Beispiel:

```
if a == 1:  
    b = 10  
else:  
    b = 1000
```

wird zu

```
b = (10 if a == 1 else 1000)
```

- eleganter (aber unlesbarer) Code
- lazy evaluation

Aufgabe

Implementiere ein Skript das von der Kommandozeile drei Argumente erwartet. Das erste Argument ist eine der folgenden Rechenoperationen `[+,-,*,/]` das auf die beiden folgenden Argumente angewand werden soll. Das Ergebnis soll nach `STDOUT` geschrieben werden.

Beispiele:

```
> rechne.py - 1 2  
> -1  
> rechne.py / 20 4  
> 5
```

while

- `while <Bedingung>:`
 Anweisung
 ...
 Anweisung
- fuehrt die Anweisungen **solange** die Bedingung erfuellt ist

break

- Abbruch der Schleife
- ```
while <Bedingung>:
 ...
 if (Bedingung2):
 break
```

## continue

---

- Abbruch **eines** Schleifendurchlaufs
- ```
while <Bedingung>:  
    ...  
    if <Bedingung2>:  
        continue  
    ...
```

else

- wird **einmal** ausgeführt
- ```
while <Bedingung>:
 ...
else:
 Anweisung
```
- sinnvoll in Verbindung mit break

# for

---

- `for <var> in <Objekt>:`  
    Anweisung  
    ...  
    Anweisung
- Objekt muss *Iterable* sein  
    → *Module/Objekte*
- `break`, `continue` oder `else` verwendbar

## for - Beispiele

---

- ```
for i in range(5):  
    print i
```

range ist eine build-in Funktion

- ```
for c in 'Hello World':
 print c
```

ein String Objekt ist vom Typ *Iterable*

# Lesen von STDIN

---

- einfache Benutzereingabe via STDIN
- `input([prompt]) -> value`
- typische Anwendung:  
`year = input('Geburtsjahr ? ')`

# Aufgabe

---

Ändere die vorherige Aufgabe so, das bei einem Aufruf ohne Argumente, das Skript die Rechenoperation und die beiden Argumente von STDIN abfragt.