

Programmieren in Python - Teil 0

Christian Dietrich

October 15, 2008

Einfuehrung

Was ist Python?

Datentypen und Variabelen

Zahlen

Zeichenketten

Listen

Woerterbuecher und Tupel

Das Hello World

Bedingungen



Figure: Schema einer interpretierten Sprache



Figure: Schema einer interpretierten Sprache

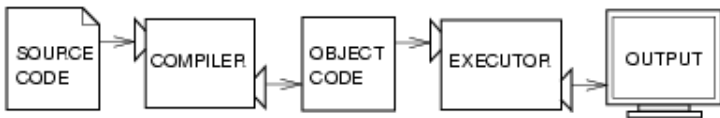


Figure: Schema einer maschinennahen Sprache



Figure: Schema einer interpretierten Sprache

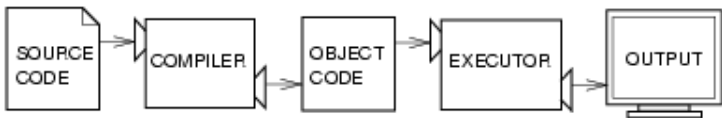


Figure: Schema einer maschinennahen Sprache

► Skriptsprache

Was ist Python

- ▶ Monty Python

Was ist Python

- ▶ Monty Python
- ▶ Klare Syntax, einfach zu verstehen (Hoffentlich)

Was ist Python

- ▶ Monty Python
- ▶ Klare Syntax, einfach zu verstehen (Hoffentlich)
- ▶ Grosze Bibliothek von Funktionen (Graphische Oberflaeche, . . .)

Zahlen

- ▶ Ganze Zahlen

a = 1

b = 5

Zahlen

- ▶ Ganze Zahlen

a = 1

b = 5

- ▶ Flieszkommazahlen

c = 0.4

d = 3.1415

Zahlen

- ▶ Ganze Zahlen

$a = 1$

$b = 5$

- ▶ Flieszkommazahlen

$c = 0.4$

$d = 3.1415$

- ▶ Komplexe Zahlen

$e = 3 + 5j$

$f = 1j$

Strings

- ▶ Eingeschlossen von " oder '
spam = "egg"
parrot = 'Hallo Welt'

Strings

- ▶ Eingeschlossen von " oder '
spam = "egg"
parrot = 'Hallo Welt'
- ▶ Gerne auch Mehrzeillig
egg = """Spam
and
egg and spam and egg"""

Strings

- ▶ Eingeschlossen von " oder '
spam = "egg"
parrot = 'Hallo Welt'
- ▶ Gerne auch Mehrzeilig
egg = """Spam
and
egg and spam and egg"""
- ▶ "/" in Strings
falsch = "'flsch' ist falsch"
richtig = "Dies ist ein \" Zeichen"

Listen 1

- ▶ Zusammengesetzter Datentyp
tin = ["egg", "spam", 23, 42]
bier = ["Hopfen", "Malz", "Wasser"]

Listen 1

- ▶ Zusammengesetzter Datentyp
tin = ["egg", "spam", 23, 42]
bier = ["Hopfen", "Malz", "Wasser"]
- ▶ Zugriff ueber Indices
tin[0]
bier[-1]
tin[1:3]

Listen 1

- ▶ Zusammengesetzter Datentyp
tin = ["egg", "spam", 23, 42]
bier = ["Hopfen", "Malz", "Wasser"]
- ▶ Zugriff ueber Indices
tin[0]
bier[-1]
tin[1:3]
- ▶ Ein String als Liste
a = "Club Mate"
a[1]
a[0:4]

Listen 2

- ▶ Laenge einer Liste
`len(tin)`

Listen 2

- ▶ Laenge einer Liste
`len(tin)`
- ▶ Listen sind veraenderbar
`bier[2] = 'Quellwasser'`
Warnung: Strings koennen nicht uf diese Weise veraendert werden.

Hashes

- ▶ Zusammengesetzter Datentyp, wie eine Liste, jedoch wird nicht nur ueber Zahlen indiziert
`tin = {"parrot": "dead", 23: 42, 2.5: 6.25}`

Hashes

- ▶ Zusammengesetzter Datentyp, wie eine Liste, jedoch wird nicht nur ueber Zahlen indiziert

```
tin = {"parrot": "dead", 23: 42, 2.5: 6.25}
```

- ▶ Als Zugriffsindex kann jeder Zahlentyp dienen

```
tin[23]
```

```
tin["parrot"]
```

```
tin[2.5]
```

Hashes

- ▶ Zusammengesetzter Datentyp, wie eine Liste, jedoch wird nicht nur ueber Zahlen indiziert
`tin = {"parrot": "dead", 23: 42, 2.5: 6.25}`
- ▶ Als Zugriffsindex kann jeder Zahlentyp dienen
`tin[23]`
`tin["parrot"]`
`tin[2.5]`
- ▶ Keys and Values
`tin.keys()`
`tin.values()`

Tupel

- ▶ Tupel sind unveränderbare Listen
tekla = 23, 6, 42, 'gutausschend'
eris = ('mavis', 'stella')

Tupel

- ▶ Tupel sind unveraenderbare Listen
tekla = 23, 6, 42, 'gutausschend'
eris = ('mavis', 'stella')
- ▶ Zugriffsindex wie bei Listen
tekla[0]
eris[1]

Tupel

- ▶ Tupel sind unveränderbare Listen
tekla = 23, 6, 42, 'gutausschend'
eris = ('mavis', 'stella')
- ▶ Zugriffsindex wie bei Listen
tekla[0]
eris[1]
- ▶ Unveränderbar
tekla[0] = 'pouty'

Das Hello World Beispiel

```
>>> print "Hello World"  
Hello World
```

Das Hello World Beispiel

```
>>> print "Hello World"  
Hello World
```

- ▶ print gibt etwas aus

Das Hello World Beispiel

```
>>> print "Hello World"  
Hello World
```

- ▶ print gibt etwas aus
- ▶ Der String "Hello World" wird an print uebergeben

Das Hello World Beispiel

```
>>> print "Hello World"  
Hello World
```

- ▶ print gibt etwas aus
- ▶ Der String "Hello World" wird an print uebergeben

```
>>> print 2+3, tekla[0], "YAPH"
```

Das Hello World Beispiel

```
>>> print "Hello World"  
Hello World
```

- ▶ print gibt etwas aus
- ▶ Der String "Hello World" wird an print uebergeben

```
>>> print 2+3, tekla[0], "YAPH"
```

- ▶ 2+3, tekla[0], "YAPH" sind mehrere Argumente von print

Das Hello World Beispiel

```
>>> print "Hello World"  
Hello World
```

- ▶ print gibt etwas aus
- ▶ Der String "Hello World" wird an print uebergeben

```
>>> print 2+3, tekla[0], "YAPH"
```

- ▶ 2+3, tekla[0], "YAPH" sind mehrere Argumente von print
- ▶ print gibt jedes Element nacheinander aus

Das if-Statement

```
a = 23
if a == 23:
    print "a ist", a
    print "Auch dies wird ausgefuehrt"
```

Das if-Statement

```
a = 23
if a == 23:
    print "a ist", a
    print "Auch dies wird ausgefuehrt"
```

- ▶ Die Einrueckungen sind wichtig bei python

Das if-Statement

```
a = 23  
if a == 23:  
    print "a ist", a  
    print "Auch dies wird ausgefuehrt"
```

- ▶ Die Einrueckungen sind wichtig bei python
- ▶ Jede Einrueckung hat 4 Leerzeichen oder einen Tab

Das if-Statement

```
a = 23
if a == 23:
    print "a ist", a
    print "Auch dies wird ausgefuehrt"
```

- ▶ Die Einrueckungen sind wichtig bei python
- ▶ Jede Einrueckung hat 4 Leerzeichen oder einen Tab
- ▶ == ist ein Operator der auf Gleichheit prueft
Probiere >>> 2+3 == 5
Der Ausdruck 2+3 == 5 gibt True zurueck

Das if-Statement

```
a = 23
if a == 23:
    print "a ist", a
    print "Auch dies wird ausgefuehrt"
```

- ▶ Die Einrueckungen sind wichtig bei python
- ▶ Jede Einrueckung hat 4 Leerzeichen oder einen Tab
- ▶ == ist ein Operator der auf Gleichheit prueft
Probiere >>> 2+3 == 5
Der Ausdruck 2+3 == 5 gibt True zurueck
- ▶ if prueft ob der Ausdruck danach (bis zum :) True ist

Das if-Statement

```
a = 23
if a == 23:
    print "a ist", a
    print "Auch dies wird ausgefuehrt"
```

- ▶ Die Einrueckungen sind wichtig bei python
- ▶ Jede Einrueckung hat 4 Leerzeichen oder einen Tab
- ▶ == ist ein Operator der auf Gleichheit prueft
Probiere >>> 2+3 == 5
Der Ausdruck 2+3 == 5 gibt True zurueck
- ▶ if prueft ob der Ausdruck danach (bis zum :) True ist
- ▶ ist er wahr, so wird der Naechste Block (Die Einrueckung) ausgefuehrt

Das else-Statement

```
a = 23
if a != 23:
    print "a ist", a
else:
    print "Pech gehabt"
```

Das else-Statement

```
a = 23
```

```
if a != 23:
```

```
    print "a ist", a
```

```
else:
```

```
    print "Pech gehabt"
```

- ▶ != prueft auf Ungleichheit

Probiere `>>> 2+3 != 23`

Der Ausdruck `2+3 == 23` gibt True zurueck

Das else-Statement

```
a = 23
```

```
if a != 23:
```

```
    print "a ist", a
```

```
else:
```

```
    print "Pech gehabt"
```

- ▶ != prueft auf Ungleichheit

Probiere `>>> 2+3 != 23`

Der Ausdruck `2+3 == 23` gibt True zurueck

- ▶ if sagt: Stimmt nicht, wird nicht ausgefuehrt

Das else-Statement

```
a = 23
```

```
if a != 23:
```

```
    print "a ist", a
```

```
else:
```

```
    print "Pech gehabt"
```

- ▶ != prueft auf Ungleichheit
Probiere >>> 2+3 != 23
Der Ausdruck 2+3 == 23 gibt True zurueck
- ▶ if sagt: Stimmt nicht, wird nicht ausgefuehrt
- ▶ Das else gehoert zum if, da es auf dieses Folgt, und in der selben Ebene ist

Das else-Statement

```
a = 23
```

```
if a != 23:
```

```
    print "a ist", a
```

```
else:
```

```
    print "Pech gehabt"
```

- ▶ != prueft auf Ungleichheit
Probiere >>> 2+3 != 23
Der Ausdruck 2+3 == 23 gibt True zurueck
- ▶ if sagt: Stimmt nicht, wird nicht ausgefuehrt
- ▶ Das else gehoert zum if, da es auf dieses Folgt, und in der selben Ebene ist
- ▶ Der else Zweig dieser Fallunterscheidung wird ausgefuehrt

Das elif-Statement

```
a = 5
if a == 23:
    print "a ist", a
elif a == 5:
    print "All Hail Eris"
else:
    print "Pech gehabt"
```

Das elif-Statement

```
a = 5
if a == 23:
    print "a ist", a
elif a == 5:
    print "All Hail Eris"
else:
    print "Pech gehabt"
▶ if sagt: Stimmt nit, wird nicht ausgefuehrt
```

Das elif-Statement

```
a = 5
if a == 23:
    print "a ist", a
elif a == 5:
    print "All Hail Eris"
else:
    print "Pech gehabt"
```

- ▶ if sagt: Stimmt nit, wird nicht ausgefuehrt
- ▶ Nun wird der elif Ausdruck geprueft

Das elif-Statement

```
a = 5
if a == 23:
    print "a ist", a
elif a == 5:
    print "All Hail Eris"
else:
    print "Pech gehabt"
```

- ▶ if sagt: Stimmt nit, wird nicht ausgefuehrt
- ▶ Nun wird der elif Ausdruck geprueft
- ▶ if - elif - else gehoeren Zusammen

Das elif-Statement

```
a = 5
if a == 23:
    print "a ist", a
elif a == 5:
    print "All Hail Eris"
else:
    print "Pech gehabt"
```

- ▶ if sagt: Stimmt nit, wird nicht ausgefuehrt
- ▶ Nun wird der elif Ausdruck geprueft
- ▶ if - elif - else gehoeren Zusammen
- ▶ es kann mehrere elif Ausdruecke geben