Python Grundlagen

Albert Weichselbraun

Inhalt

- Python – Grundlagen
- Datentypen und Operatoren
- Kontrollstrukturen
  Schleifen
  Verzweigungen
- Beispiele

Python – Typische Anwendungsgebiete

- Information Retrieval
- CGI- und Web-Anwendungen
- Automatisierung von administrativen Tätigkeiten
- Rapid Prototyping (Forschung, Entwicklung, ...)

Python – Installation testen

- Python Interpreter starten:
  Python2.4.c0 (#2, Jun 14 2006, 22:35:41)
  [GCC 4.1.2 20060613 (Debian 4.1.1-4)]
  Type „help“, „copyright“ or „licence“ ...
  >>>
- Hilfe bekommt man mit
  help(), dir(),
  mit Funktionsname.__doc__
  oder auf der Python Website:
  http://www.python.org

Python - Grundlagen

- Python ist **case-sensitive**
- Die Einrückung der Befehle determiniert den Programmfluss
- Kommentare beginnen mit #
- Alle Datentypen sind **Objekte**
- Verwendung der Sprache im Python Interpreter oder in Skripten

Python als Taschenrechner

- Umgang mit Zahlen:

  2+2, (50-5*6)/4
- Ganze Zahlen (Integer) contra Floats:

  7/3, 7/3.0
- Variablen:

  width=20, height=5*9, width*height

  x = y = z = 0
- Letzte Wert: _

  price = 100, price*1.2, _+200

Arithmetische Operationen

Beispiel	Operator	Ergebnis
4+wert	Addition	Summe von 4 und Wert
zahl1-zahl2	Subtraktion	Differenz von zahl1 und zahl2
nettoPreis*1.2	Multiplikation	Produkt von nettoPreis und 1.2
Kosten/Nutzen	Division	Kosten geteilt durch Nutzen
32%6	Modulo	Ganzzahliger Rest der Division
2**6	Exponent	2 hoch 6

- Beispiel: Kreisfläche

  flaeche = radius**2 * 3.14

Zuweisungsoperatoren

- weisen einer Variable Werte zu

  a=1 a,b = 12, 3
- Berechnung von Variablenwerten:

  a=a+12 a,b = b+3, a-1
- abgekürzte Form:

  a+=12 a*=2

  myString += ' (erledigt) '

Umgang mit Strings

Strings = Abfolge von alphanumerischen Zeichen

- **Eingabe** von Strings:
  'spam eggs', "doesn't", "'Contains " and '... "'
- **Operationen**
  len(word), a.capitalize(), a.upper()
- **Slicing**
  word = 'donaudampfschiffsfahrtsgesellschaft'
  word[0:5] = 'donau', a[-12:]='gesellschaft'

Umgang mit Strings - Slicing

```
 +---+---+---+---+---+
 | H | a | l | l | o |
 +---+---+---+---+---+
 0   1   2   3   4   5
-5  -4  -3  -2  -1
```

a='Hallo'
a[0:2] -> 'Ha', a[-3:5] = a[-3:] = 'llo'

Metazeichen

\n	Zeilenumbruch
\t	Tabulator
\b	Backspace
****	Backslash
\'	' innerhalb eines single-quoted Strings (' ')
\"	" innerhalb eines double-quoted Strings(" ")

Literale Interpretation von „\“: `r'mein Text mit \'`

Übungsaufgabe

- Setzen Sie Variablen mit folgendem Inhalt
  - Ihr Name
  - Ihre Adresse
  - Ihre Lieblingsfarbe
- Geben Sie diese Variable in einem Text mittels print aus.

Erste Schritte Richtung Programmierung

```
# Beispielprogramm

# Fibonacci Reihe

a, b = 0,1

while b < 10:
  print b
  a, b = b, a+b
```

Ausgabe in einer Zeile => Anhängen eines Kommas an den print-Befehl.

Mein erstes Python Programm

#!/usr/bin/env python

“““ Ein erstes Beispielprogramm “““

print “Hello World!“ # ein weiteres Kommentar

Python in Skripten

- Skripte werden wie folgt startfähig gemacht:
  in der **ersten Zeile** das Programm (als Kommentar) angeben

  #!/usr/bin/env python

  Das Skript mit Ausführungsrechten versehen:
  chmod a+x filename.py
  Starten mit ./filename.py

Benutzereingaben

- Benutzereingaben können mit dem Befehl

  ```
  wert_str = input('Prompt: ')
  ```

  eingelesen werden.
- Zahlenwerte müssen vor deren Verwendung konvertiert werden:

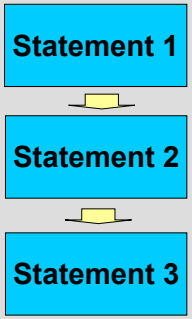
  ```
  radius = float(radius_str)
  anzahl = int(anzahl_str)
  ```

Übungsaufgaben

- Erstellen Sie ein Skript, das nach Eingabe des Bruttopreises einer Ware Nettopreis und MWSt. ausgibt.
- Schreiben Sie ein Programm, das den Benutzer nach Radius und Höhe eines Zylinders fragt und Volumen und Oberfläche des Zylinders berechnet.

Verzweigungen

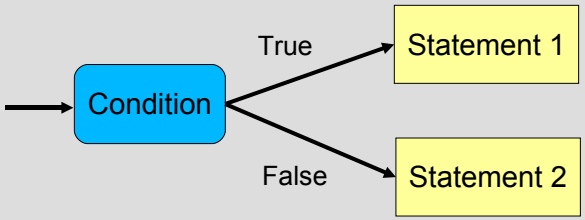
- bisher: Anweisungen sequentiell ausgeführt




```
radius = 2.15
flaeche = radius**2*3.14
print 'Fläche =', flaeche
```

Verzweigungen

- um auf unterschiedliche Situationen **reagieren** zu können, benötigt man **bedingte** Verzweigungen.



Vergleichsoperatoren

==	a == b	Gleichheit
!=	a != b	Ungleichheit
<	a < b	Kleiner als
>	a > b	Größer als
<=	a <= b	Kleiner gleich
>=	a >= b	Größer gleich

Logische Operatoren

- zur Verknüpfung von Vergleichen

Operator	Beispiel	Erklärung
and	a and b	Wahr, wenn beide wahr sind
or	a or b	Wahr, wenn mindestens eine Variable wahr ist
not	not a	Wahr, wenn a falsch ist.

- Beispiel:

```
if (alter>=10) and (alter<=19): print 'Teenager'
```

- **Unwahr:** False, '', 0, None

Übungsbeispiel

- Angabe:

  a,b,c = 3,0,0

- Bestimmen Sie die Wahrheitswerte:

  a and b

  a or b and not c

  a or (b and c) # Klammer optional

  (a or b) and c # Klammer obligatorisch

  (a or b) and not (a and b)

Verzweigungen

```
x = int(raw_input("Please enter an integer: "))
if x < 0:
    x = 0
    print 'Negative changed to zero'
elif x == 0:
    print 'Zero'
elif x == 1:
    print 'One'
else:
    print 'More'
```

Anmerkung: die zu einer Bedingung zugehörigen Befehle werden als **Block** bezeichnet.

Übungsaufgabe

Ein Programm setzt die folgenden Variablen:

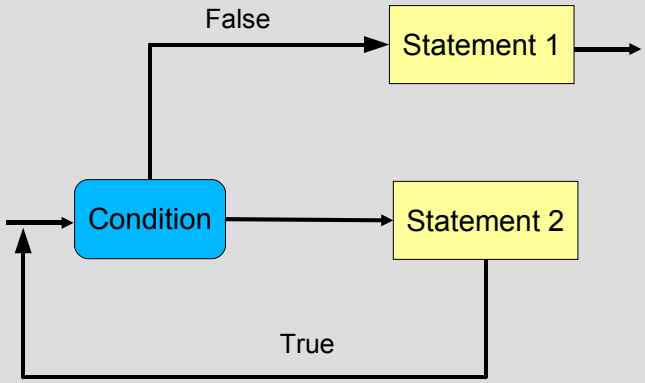
- Name
- Temperatur (in Grad Celsius)

Schreiben Sie Verzweigungen für den folgenden Sachverhalt.

- Temperatur **kleiner als** 20 Grad ist
  => Ausgabe: „Für *Name* ist es hier zu kalt.“
- Temperatur **höher als** 22 Grad
  => Ausgabe: „Für *Name* ist es zu heiß.“
- Liegt die Temperatur im Bereich **von 20 bis 22** Grad
  => Ausgabe:
  „Für *Name* ist die Temperatur von *Temperatur* Grad optimal.“

Schleifen

- ermöglichen es Programmteile zu wiederholen



Die „while“ Schleife

- Die Schleife wird ausgeführt, solange die Bedingung erfüllt ist.

```
counter = 0
while counter < 10:
   print 'Zähle noch ...', counter
   counter +=1

print 'Habe endlich %s erreicht!' % (counter)
```

Übungsaufgabe

- Schreiben Sie eine Programm, das in zweierschritten von 100 bis 80 zählt.
- Der Output soll folgendermaßen aussehen:

  1. Wert: 100
  2. Wert: 98
  3. Wert: 96
  ...
  11. Wert: 80

Übungsaufgabe - Lotto

- Schreiben Sie ein Programm, das 6 ganzzahlige Zufallszahlen zwischen 1 bis 45 ausgibt.
- Ausgabe:

  1. Zufallszahl:
  ...
  6. Zufallszahl:

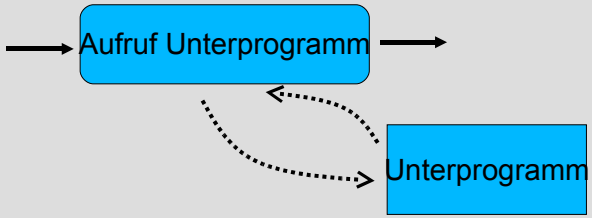
- **Hinweis:** Zufallszahlen werden in python wie folgt generiert:

  import random

  zufallszahl = random.randint(kleinsterWert, höchsterWert)

Funktionen

- Programme enthalten oft immer wiederkehrende Befehlsfolgen. Diese können in Subroutines = Unterprogramme ausgelagert werden.



Funktionen

- Subroutines können auch als mathematische Funktionen aufgefasst werden:

$$\underbrace{y}_{\text{Rückgabewert}} = f\underbrace{(x_1, x_2, \ldots)}_{\text{Argumente}}$$

Funktionen

- Definition von Funktionen:

```
def myFunc(argumente):
   ''' documentation '''
   ...
   return rückgabeWert
```

- Aufruf mittels Funktionsname()

```
myFunc(argumente) bzw. myFunc()
```

- Beispiel:

```
def zylinderVolumen(r,h):
   return r**2*3.14*h
```

Übungsaufgabe

- Schreiben Sie eine Subroutine mit Namen `aktienwert`, die den Kurs und die Anzahl der Aktien übernimmt und den derzeitigen Wert des Portfolios zurückgibt. Rufen Sie die Subroutine aus dem Hauptprogramm auf.

Übungsaufgabe*

- Schreiben Sie ein Programm, das den folgenden Output erzeugt (die erste Zeile hat 20 Zeichen):
- 01010101010101010101
  01010101010101010101
  01010101010101010101
  01010101010101010101
  ...
  01
  0
- **Hinweis:** Hier ist es nötig Schleifen zu verschachteln oder Subroutinen zu verwenden.

Kontrollstrukturen in Schleifen

- Mit der nächsten Iteration fortsetzen: continue
- Die Schleife abbrechen: break

```
for element in xrange(10):
  if element == 6:
     continue
  elif element == 8:
     print 'Acht ist das letzte Element...'
     break
  print element
```

Python - Datentypen

Gruppe	Typen	Beispiele	Kommentar
Skalar	Integer	i=1	
	Float	f=0.1	
Sequenzen	String	s="Hallo"	
	List	l=[1,2,3,"ana"]	# veränderbar
	Tuple	t=(1,2,3)	# unveränderbar
Dictionary	Dict	d={ 1: „a", 2: „b" }	

Skalar

- einfachste Form einer Variablen
- immer nur ein Wert
  21
  0.0023

Sequenzen

- sind Listen von Skalaren
- Listen
  meineListe = ['Salat', 'Eis', 'Ei']
- Tuples
  obst = ('apfel', 'birne', 'traube')
- Sets()
  wie Listen, jedoch ohne doppelte Einträge

Sequenzen - Listen

- Hinzufügen/Entfernen von Elementen:

```
meineListe.append( 'Dressing')
meineListe.remove( 'Ei' )
meineListe[0:1]=[] oder
del meineListe[0]
```

- weitere Befehle:

```
meineListe.reverse()
meineListe.sort()
```

Die „for“ Schleife

- Eine Sequenz von Elementen wird Schritt für Schritt abgearbeitet

```
for element in liste:
  print element
```

Übungsaufgabe

- Erzeugen Sie eine Liste mit mindestens fünf verschiedenen Sportarten.
- Geben Sie das erste Element aus.
- Geben Sie das letzte Element aus.
- Geben Sie die Anzahl der Elemente aus.
- Geben Sie die Sportarten wie folgt sortiert aus:

  1. Sportart: ...

  ...

  n. Sportart: ...

Strings als Sequenzen

- Sonderfall: Strings
  Listen von einzelnen Zeichen
  'Hans' = ['H', 'a', 'n', 's']
  immutable
- Im Computer: ASCII-Repräsentation der Zeichen
- Konvertierung mittels ord(), chr():

  Beispiel: ord('a') = 97, chr(97)='a'

String Manipulationen

- .strip(), .split()
- .find(), .replace()
- .upper(), .lower()
- Beispiele:

  „Vienna is the capital of austria“.split()

  [„Vienna“, „is“, „the“, „capital“, „of“, „austria“]
- Advanced Manipulations: Regular Expressions

Übungsbeispiel

- Schreiben Sie ein Programm, das eine Zeile als Input (`raw_input`) entgegennimmt, und dann die einzelnen Wörter alphabetisch sortiert, getrennt durch Beistriche ausgibt.
- **Beispiel**:

  Eingabe: Klaus Briggy Anna Martin

  Ausgabe: Anna, Briggy, Klaus, Martin

Übungsbeispiel

- Geben Sie einen String Element für Element aus

```
for element in 'meinString':
  print element
```

- Geben Sie die Zahlenwerte für die Zeichen eines Strings aus:

```
for element in 'meinString':
  print ord(element)
```

String Ausgabe

- Variablen können mittels Platzhalter in die Ausgabe eingefügt werden.
- Platzhalter: %s – Strings, %d – Zahlen,
  %f – Dezimalzahlen

```
name = 'Test'
for x in xrange(1,12):
   print 'Ausgabe Nr. %d von %s' % (name,x)
   print 'Kehrwert von %d: %2.2f' % (x, 1/x)
```

Übungsbeispiel*

Ermitteln Sie die Bedeutung dieses Textes, wenn folgendes gilt:

a --> c, b --> d, ... x --> z, y --> a, ...

g fmnc wms bgblr rpylqjyrc gr zw fylb. rfyrq ufyr amknsrcpq ypc dmp. bmgle gr gl zw fylb gq glcddgagclr ylb rfyr'q ufw rfgq rcvr gq qm jmle. sqgle qrpgle.kyicrpylq() gq pcamkkclbcb. lmu ynnjw ml rfc spj.

Dictionaries

- Aufbau: analog zu einem Wörterbuch

```
myDict={'Albert':5229,'Briggy':5228}
```

- Zugriff auf die Werte:

```
myDict['Briggy']
myDict['neuerEintrag'] = 5230
del myDict['zulöschenderEintrag']
```

- Überprüfen ob Werte im Dictionary enthalten sind:

```
'Briggy' in myDict
```

Dictionaries

- Ausgabe aller
  Schlüssel: myDict.keys()
  Werte: myDict.values()
  Schlüssel und Werte: myDict.items()
- Beispiel:

```
for name, durchwahl in myDict.items():
  print 'Ext: %s (%s)" %(durchwahl,name)
```

Übungsbeispiel

- Warenverzeichnis:
  Schreiben Sie ein Programm, das die Bezeichnung von Waren und deren Preis entgegennimmt und diese anschließend sortiert ausgibt.

Übungsbeispiel

- Ihr Warenverzeichnis enthält folgende Einträge:
  { 'Eiscreme': 2.20, 'Schokolade': 1.99,
  'Birnen': 1.50, 'Chips': 1.19 }
- Erstellen Sie eine Applikation, die vom Benutzer die Warenmengen abfragt und daraus den Gesamtpreis der Bestellung errechnet.

Umgang mit Dateien

- Dateien – entsprechendes File-Objekt:
  öffnen – Zugriff - schließen
- Öffnen einer Datei
  f=open('filename', 'r') ... read access
  f=open('filename', 'w') ... write access
- Zugriff:
  f.read(), f.readline(), f.readlines(), f.write()
- Schließen: f.close()

Beispiel

- Einlesen und Ausgabe von /etc/passwd

```
fileObject = open('/etc/passwd')
# Möglichkeit 1: Iterator
for line in fileObject:
   print line

fileObject.seek(0)
# Möglichkeit 2: Benutzen von readlines()
for line in fileObject.readlines():
   print line

# Ausgabe von nur einer Zeile
fileObject.seek(0)
print fileObject.readline()

fileObject.close()
```

Übungsbeispiel

- Schreiben Sie ein Programm, das eine Textdatei unter einen anderen Namen dubliziert (kopiert).
- Verändern Sie das Programm so, das bei der Zieldatei nun in jeder Zeile zuerst die Zeilennummer gefolgt von einem Doppelpunkt steht.

Umgang mit Dateien

- urllib
  stellt eine „Remote Resource“ als Datei zur Verfügung
- Zugriff mit den Standardmethoden für ein Fileobjekt (.readline(), .readlines(), etc.) möglich

Beispiel

Einlesen und Speichern einer Website:

```
import urllib
source=urllib.urlopen('http://www.heise.de')
destination=open('heise.html', 'w')

content = ''.join( source.readlines() )
destination.write(content)

source.close()
destination.close()
```

Beispiel

Zählen von Worten in einer Datei

```
d={}
for line in open('myfile'):
   for token in line.split():
      d[token] = d.get(token,0) + 1

print 'Tokens: ', d
```

CSV-Dateien

- CSV (Comma Separated Values)
- Datenaustausch zwischen Applikationen
- Aufbau von CSV-Files:

```
Toni,"Landstraße 22",5231
Petra,Landweg,5231
"Anita Grausam","Freihausstraße 22",5229
"Brigitte Rafael","Carminweg 6/6/3",5230
...
```

CSV-Dateien

- Ausgabe von Listen/Tuples als CSV-Files

```
import csv

values = [(1,'Anna',5229), (2,'Briggy',5230)]
fileObj = open('myFile.csv', 'w')
csvWriter = csv.writer( fileObj )
for row in values:
   csvWriter.writerow( row )
fileObj.close()
```

CSV-Dateien

- Einlesen von CSV-Dateien
- Die Werte werden als Liste übergeben

```
import csv

fileObj = open('myFile.csv')
csvReader = csv.reader( fileObj )
for row in csvReader:
   print row
fileObj.close()
```