Python - Datentypen

Albert Weichselbraun

Inhalt

- Scalar
- Listen, Tuples, Sets
- Dictionaries
- Klassen und Instanzen
- for - Schleife

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)
```

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.

Klassen

- Können als Datentypen interpretiert werden.
- Daten + Funktionalität werden zusammengefasst.

```
class Complex:
   ''' Eine einfache Beispielklasse '''
   def __init__(self, real, imag):
      self.r = real
      self.i = imag
```

Klassen

- **Instanzen** von einer Klasse werden durch **Instanzierung** erzeugt:

```
c = Complex(3.0,-4.5)
c.r, c.i
```

- Durch Instanzierung wird ein weiteres Objekt mit **gleichen Funktionen** aber **unterschiedlichen** Werten der **Instanzvariablen** erzeugt.
- Zugriff auf Instanzvariables mittels

```
self.variableName
```

Klassen

- Funktionalität wird durch entsprechende Methoden bereitgestellt.

```
def __add__(self, o):
   res=Complex( self.r+o.r, self.i+o.i )
   return res
```

- Vordefinierte Methoden
  - __str__: Stringrepräsentation des Objektes
  - __add__, __mul__, __div__, __sub__
  - ...

Übungsaufgabe

- Beispiel: Telefonbucheintrag

```
class TelefonBuchEintrag:
   def __init__(self, name, extension):
      self.name = name
      if not isinstance(extension, int):
        print 'Extension muss numerisch sein!'
      self.extension = extension

   def __str__(self):
      ''' gibt einen Eintrag aus '''
      return '%s (%s)' % (extension, name)
```

Iteratoren

- Klassen, welche die Methoden __iter__() und next() implementieren.
- __iter__() liefert eine Iterator Klasse zurück
- next() liefert den nächsten Wert des Iterators oder eine StopIteration Exception.

Die „for“ Schleife mit Iteratoren

- Iteration über ein Dictionary:

  for key in dictionary:

      print key

- Weitere Iteratoren:
  - dictionary.itervalues()
  - dictionary.iteritems()