Einführung in Python und PyS60

Bernhard Tittelbach

RealRaum — http://www.realraum.at — reality://innere_stadt.graz.at/sporgasse/16

15. Juli 2008

Outline

Python and PyS60

- interpretierte Sprache
- umfangreiche Library und zahlreiche Zusatzmodule
  `http://docs.python.org/lib/lib.html`
- Aktuelle Version: 2.5.2
- PyS60: Python 2.2 portiert auf Symbian S60 und UIQ, + UI Libraries
- Interpreter: `python`

Basic Syntax

- Einrücken statt Klammern
- Kein Zeilen Terminierungs Character
- Keine Klammer um Boolsche Statements
- Benannte Parameters

Example (Einrücken statt Klammern)

```
def find(f, seq):
  for item in seq:
    if f(item):
      return item

def unzip_helper(tuple,list1,list2):
  (a,b)=tuple
  list1+=[a]
  list2+=[b]
```

Datentypensystem von Python

stark es gibt keine implizite Konvertierung
`2 + "10"` wirft Exception

dynamisch Typendefinition passiert zur run-time nicht zur compile-time

- flexibler
- Fehler fällt erst auf bei Ausführung auf

implizit Der Typ einer Variable ergibt sich aus der Zuweisung.
Es gibt keine Typendeklaration.

Datentypen I

Liste
- Wichtigster Datentyp
- Mischen von enthaltenen Datentypen möglich

Tuple eigentliches Array in Python

Wörterbuch Hash mit log. Zugriffszeit

Example (Listen)

```
liste = [1,2.0,"3"]
liste_von_listen = [[1,2,3],[4,5,6]]
liste_von_listen[2][1]
> 4
liste[-1]
> "3"
```

Datentypen II

Example (Tuple)

```
tuple = ('a','b','c')
tuple[1]
> a
```

Example (Hash / Wörterbuch)

```
wb = {"Key" : "Value", "Key2": 2, "Corner": "Ecke"}
wb["Corner"]
> "Ecke"
```

Datentyp String and String Formating I

- Quotierung mit " oder ’
- Multiline String mit """
- Formatierung mit <format string> \% <parameter>
- Konkatenation mit +
- Internationale Unicode Strings beginnen mit u", u’ oder u"""

Datentyp String and String Formating II

Example (Bsp 1)

```
name = "Mr. Smith"
sender = "Mr. Pink"
anrede = "Dear %s" % name
text = """
Bla Bla Blahhhhh
Blablabla Bla Blaaaaahh Blah Bla
"""
b_ende = "With Regards\n"
msg = anrede + text + b_ende + sender
print msg
```

Datentyp String and String Formating III

Example (Bsp 2)

```
import math
name = u"Mr. Smith"
sender = u"Mr. Pink"
vorlage = u"""S.g. %s

The ratio of a circles diameter to its circumference is %.10f
See you in %d days.

Sincerely, %s"""
email = vorlage % (name, math.pi, 3, sender)
print email
```

Parameterübergabe

- Basistypen: „by Value “
- Listen,Tuple,Hashes,Klassen: „by Reference“
- d.h. ändert eine Funktion die ihr übergebene Liste ändert sich auch die Liste in der aufrufenden Funktion

Example (Bsp 2)

```
def inc(value,liste):
  value+=1
  liste+=[value]

a=1
liste=["element"]
inc(a,liste)
print a, liste
> 1    ["element",2]
```

Slices

```
a=range(1,10)
print [a[0]] + a[2:5] + a[-2:]
> [1, 4, 5, 8, 9]
```

- Negativer Index zählt vom List-Ende weg
- Achtung, `a[0]` ist kein slice, sondern gibt ein einzelnes Element zurück

List Comprehensions

```
vec1 = [2, 4, 6]
[3*x for x in vec1]
> [6, 12, 18]
[3*x for x in vec1 if x > 3]
> [12, 18]
vec2 = [4, 3, -9]
[x*y for x in vec1 for y in vec2]
> [8, 6, -18, 16, 12, -36, 24, 18, -54]
```

Lambda Form

```
def make_incrementor(n):
  return lambda x: x + n

f = make_incrementor(21)
f(21)
> 42
```

Spaß mit Listen I

Wichtigste Funktionen

map	eine Funktion für jedes Listenelement ausführen
filter	eine Liste mit einer Funktion filtern
zip	macht aus mehreren Listen eine Liste von Tuplen
reduce	berechnet einzelnes Ergebnis aus Liste (foldl)
del	löscht ein Listen Element

Spaß mit Listen II

Example (Map)

```
a = range(1,10)
map (make_incrementor(10),a)
> [11, 12, 13, 14, 15, 16, 17, 18, 19]
map (lambda x:chr(x+ord('a')-1), a)
> ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i']
```

Example (Filter)

```
filter(lambda x: x % 2 == 0, a)
> [2, 4, 6, 8]
```

Spaß mit Listen III

Example (Zip)

```
fragen = ['name', 'quest', 'favorite color']
antworten = ['lancelot', 'the holy grail', 'blue']
for q, a in zip(fragen,antworten):
  print ' Q:%s \n A:%s \n' % (q, a)
```

Example (Reduce)

```
a=range(1,10)
del(a[2])
reduce(lambda s,l: s+l, a)
```

For-Generatoren und Interatoren

```
a=list(x*x for x in range(10))
b=iter(x*x for x in range(10))
type(a)
> <type 'list'>
type(b)
> <type 'generator'>
print a
> [0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
b.next()
> 0
b.next()
> 1
b.next()
> 4
b.next()
> 9
```

Yield-Generatoren

```
def rev(data):
  for index in range(len(data)-1, -1, -1):
    yield data[index]

for char in rev('golf'):
  print char

i=rev('abcd')
i.next()
> 'd'
i.next()
> 'c'
```

- `yield` unterbricht Ausführung einer Funktion und returniert Ergebnis
- `rev` ist eine Funktion
- aber: `rev('golf')` ist ein Generator

Klasse Beispiel

- Alle Funktionen haben Referenz auf Klasse als erstes Argument: (`self`)
- Neue Klassentypen (nicht pys60, ¿ py2.4) sind von `object` abgeleitet
- Konstruktor mit `__init__`
- Parameter mit `parameter`

```
class Amplifier(object):
  valid_volumes=range(0,100)
  def __init__(self, inital_volume=10):
    self.Volume=inital_volume

  def __getVolume(self):
    print "Getting Volume"
    return self.__volume

  def __setVolume(self, val):
    print "Setting Volume"
    if val in self.valid_volumes:
      self.__volume=val
    elif val < min(self.valid_volumes):
      self.__volume=min(self.valid_volumes)
    else:
      self.__volume=max(self.valid_volumes)

  Volume = property(fget=__getVolume, fset=__setVolume)
```

Example (Verwendung von Amp)

```
amp = Amplifier(10)
print "Vol: %d" % amp.Volume

amp.Volume=-20
print "Vol=-20: %d" % amp.Volume

amp.Volume=200
print "Vol=200: %d" % amp.Volume
```

Abgeleitete Klassen

```
class Radio(Amplifier):
  def __init__(self, sender="fm4",volume=10):
    print "Ctr: Radio"
    self._sender=sender
    Amplifier.__init__(self,volume)

  def getSender(self):
    return self._sender
```

Example (Verwendung von Radio)

```
radio = Radio(volume=33,sender="oe1")
print "Sender: %s" % radio.getSender()
print "Volume: %d" % radio.Volume
```

Python For S60

- SourceForge:
  `http://sourceforge.net/projects/pys60`
- Wiki:
  `http://wiki.opensource.nokia.com/projects/PyS60_applications`
- PDF Dokumentation
- Module für GUI, DB und SysInfo