Einfuehrung in Python

the beauty of code & web application development with django
KunterBuntesSeminar WS10

Justus Winter

Universitaet Hamburg - Fachbereich Informatik

2009-11-05

1. get the facts
   - overview
   - background
   - history
   - philosophy
2. by examples
   - statements and control flow
3. concepts
   - duck typing
   - list comprehension
   - list operations
   - no information hiding
4. django
   - Let me introduce you to the magic...
5. References

python ...

- is a multi-paradigm language (object-oriented, imperative, functional)
- is strongly and dynamic typed, uses duck typing
- was released by Guido van Rossum in 1991
- comes with batteries included
- rocks ;)

- developed in the late 80ies for Amoeba at the CMI/Amsterdam
- first public version (0.9.0) released in 1991 to alt.sources
- successor of the abc programming language
- influenced by Modula-3, c and Lisp
- influenced Ruby and ECMAScript

version	release date	major new features
1.0	January 1994	lambda, map, filter and reduce
1.4		keyword arguments, complex numbers
2.0	2001-06-14	list comprehension, new garbage collector
2.2	2001-12-21	unification of c and python types, generators
2.4	2004-11-30	function decorators, generator expressions
2.5	2006-09-19	conditional expressions, with statement
2.6	2008-10-01	bridge to 3.0
2.7.1	2010-11-27	2.7 is the last major version in the 2.x series, includes backwards compatible features from 3.1
3.0	2008-12-03	all unicode, annotations, long $\rightarrow$ int, print function, new format strings
3.1.3	2010-11-27	latest version

Tim Toady

There is more than one way to do it.

- clear minimalistic language
- consistent and intuitive behavior
- lets the user create beautiful code
- no premature optimization (not even in cpython)
- extensible (c and c++ interface)

Zen of Python

There should be one, and preferably only one, obvious way to do it.

Tim Toady Bicarbonate

There is more than one way to do it, but sometimes consistency is not a bad thing either.

- clear minimalistic language
- consistent and intuitive behavior
- lets the user create beautiful code
- no premature optimization (not even in cpython)
- extensible (c and c++ interface)

Zen of Python

There should be one, and preferably only one, obvious way to do it.

variables

```
>>> a = 23
>>> type(a)
<type 'int'>
>>> a = 'fourtytwo'
>>> type(a)
<type 'str'>
>>> a + 23
TypeError: cannot concatenate 'str' and 'int' objects
```

if statement

```
>>> if 6 * 9 == 42:
  >     print('whoohoo')
>>>
```

if statement

```
>>> if 6 * 9 == 42:
...     print('whoohoo')
... else:
...     print('too bad')
too bad
>>>
```

if statement

```
>>> if 6 * 9 == 42:
...     print('whoohoo')
... elif 031 == 25:
...     print('xmas is halloween')
xmas is halloween
>>>
```

for statement

```
>>> for i in (0, 1, 2):
  >     print(i)
0
1
2
>>>
```

for statement

```
>>> for i in [0, 1, 2]:
  >     print(i)
0
1
2
>>>
```

for statement

```
>>> for i in range(23):
...     print(i)
0
1
⋮
22
>>>
```

def statement

```
>>> def quackAndWalk(who):
  >     print(who, 'quacks amused and walks away.')
>>> quackAndWalk('Donald')
Donald quacks amused and walks away.
>>> quackAndWalk('Tom')
Tom quacks amused and walks away.
>>> quackAndWalk(23)
23 quacks amused and walks away.
```

def statement (cont)

```
>>> def calculate(x):
  >     'Returns 2 * x^2 + x - 3'
  >     return 2 * x ** 2 + x - 3
>>> calculate
<function calculate at 0xb7c101ac>
>>> help(calculate)
Help on function calculate in module __main__:

calculate(x)
    Returns 2 * x^2 + x - 3
>>> calculate(0)
-3
>>> calculate(2)
7
```

class statement

```
>>> class Robot(object):
  >     'This is the root of the robot class hierarchy'
  >     def __init__(self, name):
  >         self.name = name
  >     def quack(self):
  >         print('''%s tries to quack: 'Kkkkwaaack'.'''
  >                                                 % (self.name))
>>> Robot
<class '__main__.Robot'>
>>> help(Robot)
...
>>> bot = Robot('Marvin')
>>> bot
<__main__.Robot object at 0xb7a9f22c>
>>> bot.quack()
Marvin tries to quack: 'Kkkkwaaack'.
```

Alex Martelli

If it walks like a duck and quacks like a duck, I would call it a duck.

- duck typing allows polymorphism without inheritance

```
>>> def calculate(a, b, c):
  >     return (a + b) * c
>>> print(calculate(1, 2, 3))
9
>>> print(calculate([1, 2, 3], [4, 5, 6], 2)
[1, 2, 3, 4, 5, 6, 1, 2, 3, 4, 5, 6]
>>> print(calculate('apples ', 'and oranges, ', 2))
apples and oranges, apples and oranges,
```

mathematical notation

$$a = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$$

$$b = \{2 * n | n \in a\}$$

```
>>> a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> [n for n in a]
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> [2 * n for n in a]
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18]
>>> [2 * n for n in a if n % 2 == 1]
[2, 6, 10, 14, 18]
```

```
>>> a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> a[2]
2
>>> a[-2]
8
>>> a[2:5]
[2, 3, 4]
>>> a[23]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: list index out of range
>>> 5 in a
True
```

```
>>> a = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
>>> a[2]
2
>>> a[-2]
8
>>> a[2:5]
(2, 3, 4)
>>> a[23]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: tuple index out of range
>>> 5 in a
True
```

```
>>> a = 'Hello cruel world.'
>>> a[2]
'l'
>>> a[-2]
'd'
>>> a[2:5]
'llo'
>>> a[23]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
IndexError: string index out of range
>>> 'o' in a
True
```

Quoting Wikipedia on Information Hiding

In object-oriented programming, information hiding reduces software development risk by shifting the code's dependency on an uncertain implementation (design decision) onto a well-defined interface. Clients of the interface perform operations purely through it so if the implementation changes, the clients do not have to change.

- python has no concept of private variables or functions


django

Introduction

django...

- ,,is a high-level Python Web framework that encourages rapid development and clean, pragmatic design."
- has a model-view-controller based design
- focuses on automating common tasks
- is highly modular and comes with batteries included
- adheres to the DRY[1] principle
- is easy to deploy (fcgi, mod_python, mod_wsgi, ...)
- has a built-in webserver that reloads code
- is BSD-licensed

[1] don't repeat yourself

django core

django core framework contains...

- an ORM[2] featuring many database backends and a lot of magic
- a template engine
- regular expression based URL dispatcher
- a standalone web server for development
- ...

[2] object relational mapper

django applications

django comes with a lot of bundled applications...

- an authentication framework
- the django admin interface
- {c,x}srf[3]-protection
- syndication tools (atom-feeds & co)
- filter for lightweight markup languages like markdown
- ...

[3] cross site request forgery

let's start hacking django!

installing the required packages

```
$ aptitude install python-django{,-doc} \
                   python-{docutils,markdown}
```

References

- http://python.org/
- http://en.wikipedia.org/wiki/Python_(programming_language)
- http://en.wikipedia.org/wiki/Duck_typing
- http://en.wikipedia.org/wiki/Information_hiding
- http://en.wikipedia.org/wiki/There%27s_more_than_one_way_to_do_it
- http://www.djangoproject.com/