python

Shellscripting

Sebastian Wiedenroth (wiedi@frubar.net)

Übersicht

- Einführung
  - Schnittstellen zum Betriebsystem
    - Files
    - Prozesse
    - Parameter parsen
- Demo
- Übungsaufgabe

Freitag, 4. Dezember 2009

Shellscripts

- zur Automatisierung von Aufgaben
- „kleine Helferchen“

- Shebang: `#!/usr/bin/env python`
- `chmod +x script.py`
- `./sciprt.py`

import sys

```
sys.argv            # Kommandozeilen argumente

sys.path            # searchpath für python module

sys.exit(arg)       # script beenden
```

import os

```
os.path.     # pfad manipulation
os.getcwd()  # current working dir
os.chmod()   # change mode of a file
os.mkdir()   # dir anlegen
os.rmdir()   # dir löschen
```

import os (2)

```
os.fork()     # fork a child process
os.kill()     # send signal to process
os.popen()    # run a child process
...
```

import shutil

- Highlevel modul für FS funktionen
  - shutil.copytree(src, dst)
  - shutil.rmtree(path)
  - shutil.move(src, dst)

Freitag, 4. Dezember 2009

Files

- „File Like Objects“
- Dateien, Prozesse, Sockets, ...
- open(name[, mode]) -> file object
- mode default = ‚r‘ für lesen
- read(), write(), readline(), readlines(), seek (), flush(), close(), ...

Files

```
f = open('bla.txt')
x = f.readlines()
f.close()

inhalt = open('blablabla.txt').readlines()

open('hui.txt', 'w').write('42')
```

Freitag, 4. Dezember 2009

import subprocess

- Highlevel modul für subprozesse
- subprocess.call(args)
- p = subprocess.Popen(args, ...)
- p.communicate()
- p.kill()
- p.stdin / p.stdout / p.stderr

import subprocess (2)

```
from subprocess import call, Popen, PIPE

retcode = call(['ping', '-c', '1', '127.0.0.1'])

p = Popen(['wall'], stdin=PIPE, stdout=PIPE)
p.stdin.write('party ')
out_data, err_data = p.communicate(input='time, excellent')

output = Popen(['uname', '-a'], stdout=PIPE).communicate()[0]
```

Freitag, 4. Dezember 2009

import optparse

- Parsing von command line Parametern

```
usage = "usage: %prog [options] arg1 arg2"
parser = OptionParser(usage=usage)

parser.add_option("-v", "--verbose",
                  action="store_true", dest="verbose", default=True,
                  help="make lots of noise [default]")

parser.add_option("-f", "--filename",
                  metavar="FILE", help="write output to FILE")

(options, args) = parser.parse_args()

if options.verbose:
   ...
```

import me please!

```
def foo():
    pass

def main():
    foo()

if __name__ == '__main__':
    main()
```

Freitag, 4. Dezember 2009