

J[P]ython - eine objektorientierte Skriptsprache in der gymnasialen Oberstufe

StD Dipl.-Inform. L. Humbert
Universität Dortmund, Didaktik der Informatik

Montag, 9. April 2001 - 16³⁰ - 17¹⁵ Uhr
Köln - MNU Jahrestagung - Informatik

Ziele des Informatikunterrichts

Leitlinien [Gesellschaft für Informatik 2000]

- Interaktion mit Informatiksystemen
- Wirkprinzipien von Informatiksystemen
- Informatische Modellierung
- Wechselwirkungen zwischen Informatiksystemen, Individuum und Gesellschaft

Informatikmodule

Informatikmodule sind Inhaltsbereiche, denen fachliche Elemente zugrunde liegen. Diese werden unter fachdidaktischen und sachlichen Gesichtspunkten so strukturiert, dass Unterrichtsreihen als Sequenzen zusammengestellt werden.

Informatikmodule erlauben, Inhaltsbereiche des Informatikunterrichts, die heute in der Sekundarstufe II unterrichtlich umgesetzt werden, zukünftig (teilweise) in der Sekundarstufe I in einem verpflichtenden Informatikunterricht einen Platz zu geben.

Konkretisierung von Modulen

1. schulisches Intranet, computergestützte Gruppenarbeit (**verpflichtendes** erstes Element)
 - informatische Fachkonzepte im Anwendungskontext
 - informatisches Modellieren
 - Objektorientiertes Modellieren
 - Automatentheoretische Modelle
 - Wissensbasiertes Modellieren

exemplarische Umsetzungen

Orientierung der Beispiele

- nicht primär Orientierung an der Programmierung
 - Modellierung umfasst Programmierung
- Voraussetzungen für die Umsetzung der Beispiele
 - eigene Klassen (Attribute, Methoden) -strukturen
 - Kontrollstrukturen (Zyklen, Verzweigungen)

Server-Klientenstrukturen

Klient

```
from UDF import UDFsocket
sendenAn=UDFsocket("Klient","haspe.homeip.net",8081)
sendenAn.datenaussenden("Holy Guido! It's working.")
```

Server-Klientenstrukturen

Server

```
from UDF import UDFsocketel
port=8081
serverSteckdose= UDFsocketel("Server", "haspe.homeip.net", port)
print "Server wartet auf Verbindung am Port:", port
i=1
while 1:
    eingangsDaten=serverSteckdose.datenempfang()
    print i, eingangsDaten, "von: Rechner", \
        serverSteckdose.getAbsender()[0], \
        "Port:", serverSteckdose.getAbsender()[1]
    i=i+1
```

Graphische Benutzungsoberflächen

```
from java import awt; from pawt import swing
labels = ['7','8','9','+', '4','5','6','-','1','2','3','*', '0','.','=','/']
keys = swing.JPanel(awt.GridLayout(4, 4)); display = swing.JTextField()
def push(event): display.replaceSelection(event.actionCommand)
def enter(event): display.text = str(eval(display.text)); display.selectAll()
for label in labels:
    key = swing.JButton(label)
    if label == '=': key.actionPerformed = enter
    else: key.actionPerformed = push
    keys.add(key)
panel = swing.JPanel(awt.BorderLayout())
panel.add("North", display)
panel.add("Center", keys)
swing.test(panel)
```

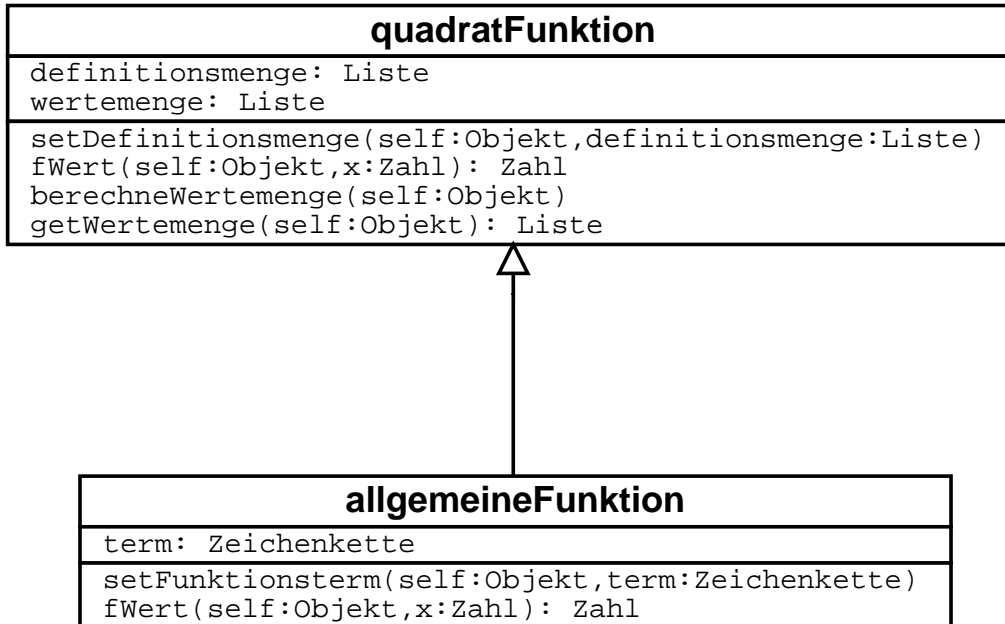
Die Quadratfunktion und mehr . . .

```
class quadratFunktion:
    def setDefinitionsmenge(self, definitionsmenge):
        self.definitionsmenge = definitionsmenge
    def getDefinitionsmenge(self):
        return self.definitionsmenge
    def fWert(self, x):
        return x*x
    def berechneWertemenge(self):
        self.wertemenge = map(self.fWert, self.definitionsmenge)
    def getWertemenge(self):
        return self.wertemenge
```

Testen der Klasse quadratFunktion

```
if __name__=="__main__":
    quadratfunktion= quadratFunktion()
    defmenge=[0.1,1,2,3,4,5,6,123]
    print "Definitionsmenge:", defmenge
    quadratfunktion.setDefinitionsmenge(defmenge)
    quadratfunktion.berechneWertemenge()
    wertemenge=quadratfunktion.getWertemenge()
    print "Werte:", wertemenge
    print " x | f(x) "
    print "____|_____"
    for index in range(len(defmenge)):
        print "(" ,defmenge[index],";", wertemenge[index], ")"
```

Erweiterung: UML-Diagramm



Erweiterung: Quelltext in Python

```
from quadratFunktion import quadratFunktion
class allgemeineFunktion(quadratFunktion):
    def setFunktionsterm(self, term):
        self.term= term
    def fWert(self, x):
        return eval(self.term)
```

Test der Klasse allgemeineFunktion

```
if __name__=="__main__":
    allgemeinefunktion= allgemeineFunktion()
    term= "x**3 + 2*x**2 +4*x"
    print "Funktion: f(x)=", term
    allgemeinefunktion.setFunktionsterm(term)
    defmenge=[0.1,1,2,3,4,5,6,123]
    print "Definitionsmenge:", defmenge
    allgemeinefunktion.setDefinitionsmenge(defmenge)
    allgemeinefunktion.berechneWertemenge()
    wertemenge=allgemeinefunktion.getWertemenge()
    print "Werte:", wertemenge
```

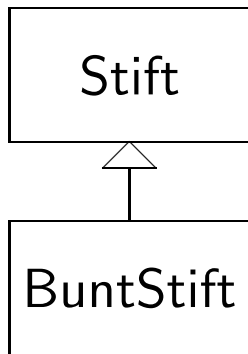
Erweiterung zum Zeichnen des Graphen

```
from allgemeineFunktion import allgemeineFunktion

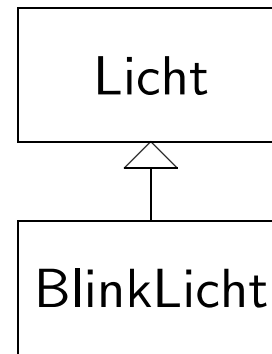
class fktMitGraf(allgemeineFunktion):
    def zeichne(self):
        from stiftUndCo import Bildschirm, BuntStift, Maus, Farbe
        zbreite=450; zhoehe=250
        zeichenflaeche= Bildschirm(zbreite, zhoehe, self.term)
    ...
```

Unterrichtsreihe zur Ampel - Vererbung Struktur

BuntStift (Stift)



BlinkLicht (Licht)



Unterrichtsreihe Ampel - Licht.py

Klassendefinition

```
from stiftUndCo import BuntStift, Farbe, ...
class Licht:
    def __init__(self, groesse=50):
        self.meinBuntStift = BuntStift()
        ...
    def setzeGroesse(self, groesse):
        self.groesse=groesse
        ...
```

```
if __name__=="__main__":
    from stiftUndCo import Bildschirm
    meinBildschirm = Bildschirm(640,480,"Licht")
    meinLicht= Licht()
    meinLicht.setzePosition(320,240)
    meinLicht.setzeFarbe(Farbe.GRUEN)
    meinLicht.an()
    ...
```

Skriptbestandteil: Testcode

Unterrichtsreihe Ampel - BlinkLicht.py

Fachkonzept Nebenläufigkeit

```
from thread import allocate_lock, \
    start_new_thread
class BlinkLicht(Licht):
    def __init__(self, groesse):
        Licht.__init__(self, groesse)
        self.Aus=1
        self.blink_lock=allocate_lock()

    def blinkAn(self):
        def blinkSchleife(self):
            while not self.aus():
                Licht.an(self)
                self.meineHilfe.warte(1000)
                Licht.aus(self)
                self.meineHilfe.warte(1000)

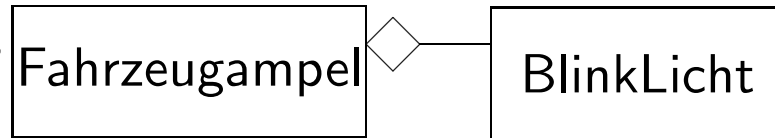
        self.Aus = 0
        start_new_thread(blinkSchleife,
            (self,))

    def blinkAus(self):
        self.Aus = 1

    def aus(self):
        return self.Aus
```

Ampelanlage - eine Analyse

Modell: Fahrzeugampel **hat** drei Lichter, davon ist eins als BlinkLicht zu realisieren



- alle Ampelphasen
- Zusammenschaltung zu einer Ampelanlage

Automatentheoretische Überlegungen für die Ampelsteuerung \implies Zustandsübergangdiagramm

. . . <http://www.python.org/> . . .

[Prechelt 2000] empirische Vergleichsstudien,

[Lefkowitz 2000] ein subjektiver Sprachvergleich: Java und Python,

[Hugunin 1997] Python und Java = JPython,

[Stajano 2000] Python in der Ausbildung,

[Elkner 2000] Einsatz von Python in der High-School,

[Downey u. a. 1999] Denken wie ein Informatiker,

[Figgins 2001] eXtreme Python,

[Liao 1999] das Pascal der 2000er,

[Yee 2000] Server-Klient-Anwendung: Chat-Server in Python,

[Mertz 2000] Automatentheorie in Python,

[van Rossum 1999] eine Vision

<http://in.hagen.de/humbert/> Zugang zu dem hier vorgestellten Material

. . . <http://www.jython.org/> . . .

Literatur

- [Downey u. a. 1999] DOWNEY, Allen B. ; ELKNER, Jeffrey ; ZADKA, Mosche: *How to Think Like a Computer Scientist – Python Version*. <http://www.ibiblio.org/obp/thinkCSpy/>. 1999. – geprüft 8. April 2001
- [Elkner 2000] ELKNER, Jeffrey. *Using Python in a High School Computer Science Program*. <http://www.python.org/workshops/2000-01/proceedings/papers/elkner/pyYHS.html>. 2000
- [Figgins 2001] FIGGINS, Stephen. *Extreme Python*. <http://www.oreillynet.com/lpt/a/719>. March 2001
- [Gesellschaft für Informatik 2000] GESELLSCHAFT FÜR INFORMATIK: Empfehlung der Gesellschaft für Informatik e.V. für ein Gesamtkonzept zur informatischen Bildung an allgemein bildenden Schulen. In: *Informatik Spektrum* 23 (2000), Dezember, Nr. 5, S. 378–382. – siehe auch: http://ddi.cs.uni-dortmund.de/ddi_bib/gi_empfehlung/gesamt2000/gesamtkonzept-26-9-2000.pdf
- [Hugunin 1997] HUGUNIN, Jim: Python and Java - The best of both worlds. In: *Proceedings of the 6th International Python Conference*. San Jose, Ca., October 1997. – <http://python.org/workshops/1997-10/proceedings/hugunin.ps>, S. 11–20
- [Lefkowitz 2000] LEFKOWITZ, Glyph. *A subjective analysis of two high-level, object-oriented languages*. <http://www.twistedmatrix.com/~glyph/rant/python-vs-java.html>. April 2000

- [Liao 1999] LIAO, Luby: *Python as the Pascal of 2000's*. ORA Open Source Convention at Monterey. August 1999. – <http://www.acusd.edu/~liao/python.pdf>
<http://conferences.oreilly.com/cd/python/presentations/liao/python.pdf>
- [Mertz 2000] MERTZ, David: State machines: Algorithms and programming approaches in Python. In: *IBM developerWorks* (2000), August. – Series: Charming Python 4/10 http://gnosis.cx/publish/tech_index.html <http://www-106.ibm.com/developerworks/library/python-state.html>
- [Prechelt 2000] PRECHELT, Lutz: An empirical comparison of C, C++, Java, Perl, Python, Rexx and Tcl for a search/string-processing program / Fakultät für Informatik, Universität Karlsruhe. Karlsruhe, March 2000 (2000-5). – Forschungsbericht. <http://wwwipd.ira.uka.de/~prechelt/Biblio/jccpprtTR.pdf> 34 Seiten
- [van Rossum 1999] ROSSUM, Guido van: *Computer Programming for Everybody (Revised Proposal) A Scouting Expedition for the Programmers of Tomorrow*. <http://www.python.org/doc/essays/cp4e.html>. July 1999. – Corporation for National Research Initiatives (CNRI)
- [Stajano 2000] STAJANO, Frank: Python in Education: Raising a Generation of Native Speakers. In: *Proceedings of 8th International Python Conference*, 2000. – <http://www.python.org/workshops/2000-01/proceedings/papers/stajano/stajano.html> <http://www.uk.research.att.com/~fms/http://www.uk.research.att.com/papers/tr.1999.10.html>

[Yee 2000] YEE, Ka-Ping. *[Edu-sig] writing an chat-server.*
<http://www.python.org/pipermail/edu-sig/2000-April/000320.html>. April 2000

- Folien http://bscw.gmd.de/pub/german.cgi/d25599195/9_April_Folien.pdf
- Quellcode <http://in.hagen.de/~humbert/vortraege/welcome.html>