

# L<sup>A</sup>T<sub>E</sub>X und Python

## Dynamische Dokumente professionell setzen

Uwe Ziegenhagen

26. August 2013

# Programm

L<sup>A</sup>T<sub>E</sub>X, Python

L<sup>A</sup>T<sub>E</sub>X Code erzeugen

Serienbriefe mittels Templates

Datenbanken abfragen

Webseiten scrapen mit BeautifulSoup4

Python aus L<sup>A</sup>T<sub>E</sub>X heraus

# Was ist L<sup>A</sup>T<sub>E</sub>X?

- ▶ L<sup>A</sup>T<sub>E</sub>X: Makrosammlung für Don Knuths T<sub>E</sub>X Textsatzsystem
- ▶ geht zurück auf das Jahr 1977, hat seither Maßstäbe für Computersatz gesetzt
- ▶ Verwendung heute vor allem
  - ▶ im wissenschaftlichen Bereich (Bachelor/Masterarbeiten, Dissertationen, Papers), insbesondere – aber nicht ausschließlich – für perfekten Formelsatz
  - ▶ dort, wo automatisiert große Mengen Text hochwertig gesetzt werden sollen ⇒ 1822 direktBank, „Persönlicher Fahrplan“ der Deutschen Bahn
  - ▶ dort, wo man Wert auf saubere Typographie legt
  - ▶ für alle meine Dokumente (auch diese Folien)
- ▶ mehr am Stand von Dante e. V. hier auf der FrOSCon

# L<sup>A</sup>T<sub>E</sub>X Beispiel

```
1 \documentclass{scrartcl}
2 \usepackage[paperwidth=10cm,%
3 paperheight=12cm,%
4 bottom=2.5cm]{geometry}
5 \usepackage[math]{iwona}
6 \begin{document}
7 \section{Einleitung}
8
9 Hallo, ich bin ein einfaches
10 \LaTeX-Beispiel.
11
12 \[E=mc^2\]
13
14 \end{document}
```

Listing 1: Einfaches L<sup>A</sup>T<sub>E</sub>X Beispiel

# L<sup>A</sup>T<sub>E</sub>X Beispiel

```
1 \documentclass{scrartcl}
2 \usepackage[paperwidth=10cm,%
3 paperheight=12cm,%
4 bottom=2.5cm]{geometry}
5 \usepackage[math]{iwona}
6 \begin{document}
7 \section{Einleitung}
8
9 Hallo, ich bin ein einfaches
10 \LaTeX-Beispiel.
11
12 \[E=mc^2\]
13
14 \end{document}
```

## 1 Einleitung

Hallo, ich bin ein einfaches L<sup>A</sup>T<sub>E</sub>X-Beispiel.

$$E = mc^2$$

# Python (Wikipedia)

- ▶ entwickelt Anfang der 1990er Jahre von Guido van Rossum
- ▶ universelle, üblicherweise interpretierte, höhere Programmiersprache
- ▶ unterstützt objektorientierte, aspektorientierte und funktionale Programmierung
- ▶ hat eine klare und übersichtliche Syntax
- ▶ Entwurfsphilosophie betont **Programmliesbarkeit**
- ▶ mein Zugang zu Python: save.tv Downloader<sup>1</sup>

---

<sup>1</sup><http://www.radekw.com/blog/2009/04/23/savetv-downloader/>

# L<sup>A</sup>T<sub>E</sub>X Code schreiben & übersetzen

- ▶ Beispiel „Spendenübersicht“
- ▶ CSV-Datei<sup>2</sup> mit Spalten für Name und Betrag
- ▶ Aufgabe: Tabelle der Spender mit L<sup>A</sup>T<sub>E</sub>X setzen

```
1 Spender;Summe
2 Donald Duck;10,00
3 Daisy Duck;20,00
4 Gustav Gans;50,00
5 Mickey Mouse;80,00
6 Dagobert Duck;00,10
```

Listing 2: Dateiaufbau Spender.csv

---

<sup>2</sup>„Character-Separated Values“

# Tabellen in L<sup>A</sup>T<sub>E</sub>X

```
1 \documentclass[12pt,ngerman]
2 {scrartcl}
3 \usepackage[paperwidth=10cm,%
4 paperheight=5cm,bottom=2.5cm]
5 {geometry}
6 \begin{document}
7
8 \begin{tabular}
9 {|l|r|c|p{1cm}|} \hline
10 erste & zweite & dritte & vierte Spalte
11 \\ \hline
12 12.34 & 56.78 & 90.12 & 34.56
13 \\ \hline
14 \end{tabular}
15
16 \end{document}
```

erste	zweite	dritte	vierte Spalte
12.34	56.78	90.12	34.56



## Beispiel Code

```
1 import csv # for CSV handling
2 import os # for sys calls
3
4 with open('spendertabelle.tex', 'wb') as texfile:
5     texfile.write("\\documentclass{scrartcl}\n");
6     texfile.write("\\usepackage[paperwidth=10cm,paperheight=8cm]
7     {geometry}\n");
8     texfile.write("\\usepackage{booktabs}\n");
9     texfile.write("\\begin{document}\n");
10    texfile.write("\\begin{tabular}{ll} \\toprule \n");
11    texfile.write("Name & Spende \\ \\ \\ \\ \\midrule\n");
12
13    with open('Spender.csv', 'rb') as csvfile:
14        spender = csv.DictReader(csvfile, delimiter=',')
15        for row in spender:
16            texfile.write(row["Spender"]+" & "+row["Summe"]+"\\ \\ \\ \\n")
17
18    texfile.write("\\bottomrule\n \\end{tabular}\n ");
19    texfile.write("\\end{document}");
20 texfile.close()
21
22 os.system("pdflatex spendertabelle.tex")
23 os.startfile("spendertabelle.pdf")
```

Listing 3: Tabellen schreiben

## Ergebnis

Name	Spende
Donald Duck	10,00
Daisy Duck	20,00
Gustav Gans	50,00
Mickey Mouse	80,00
Dagobert Duck	00,10

# Python Templates I

- ▶ „Template Strings“, seit Python 2.4 dabei
- ▶ dedizierte Template-Engines: Jinja2, Cheetah, Django Templating ⇒ zu Jinja2 später mehr
- ▶ Generelles Vorgehen:
  - ▶ Im Template werden Variablen definiert
  - ▶ Beim Verarbeiten werden diese Variable durch entsprechende Inhalte ersetzt
- ▶  $\LaTeX$  nutzt „\$“ für den mathematischen Modus, dies muss undefiniert werden.

```
1 from string import Template
2 s = Template('$who likes $what')
3 print(s.substitute(who='tim', what='kung pao'))
4 # tim likes kung pao
```

Listing 4: Python Template Beispiel

## Python Templates II

```
1 import string
2
3 class LaTeXTemplate(string.Template):
4     delimiter = "%%"
5
6 s = LaTeXTemplate('%%who likes %%what')
7 print(s.substitute(who='tim', what='kung pao'))
```

Listing 5: Python Template Beispiel

Nächste Schritte:

- ▶ Definition einer Briefvorlage
- ▶ Auslesen der Daten
- ▶ Befüllen und T<sub>E</sub>Xen der Briefe

# Erstellung der Briefvorlage

- ▶ basiert auf der KOMAScript `sclttr2` Klasse
- ▶ perfekt, um professionell aussehende Briefe zu erzeugen

```
1 \documentclass[ngerman,12pt]{sclttr2}
2 \usepackage[utf8]{inputenc}
3 \usepackage[paperheight=20cm]{geometry}
4 \usepackage[T1]{fontenc}
5 \usepackage{babel}
6
7 \setkomavar{fromname}{Tick Duck}
8 \setkomavar{fromaddress}{Erpelweg 1, Entenhausen}
9
10 \begin{document}
11 \begin{letter}{Donald Duck}
12 \opening{Sehr geehrte(r) Donald,}
13
14 wir danken für Ihre Spende in Höhe von 10,00 Euro
15
16 \closing{Mit freundlichen Grüßen}
17 \end{letter}
18
19 \end{document}
```

Tick Duck  
Erpelweg 1, Entenhausen

Tick Duck, Erpelweg 1, Entenhausen

Donald Duck

13. Juli 2013

Sehr geehrte(r) Donald,  
wir danken für Ihre Spende in Höhe von 10,00 Euro.  
Mit freundlichen Grüßen

Tick Duck

# Fertige L<sup>A</sup>T<sub>E</sub>X-Vorlage

```
1 \documentclass[ngerman,12pt]{scrlettr2}
2 \usepackage[utf8]{inputenc}
3 \usepackage[paperheight=20cm]{geometry}
4 \usepackage[T1]{fontenc}
5 \usepackage{babel}
6
7 \setkomavar{fromname}{Duck Foundation}
8 \setkomavar{fromaddress}{Erpelweg 1, Entenhausen}
9
10 \begin{document}
11 \begin{letter}{%%who}
12 \opening{Sehr geehrte(r) %%who,}
13
14 wir danken für Ihre Spende in Höhe von %%amount Euro.
15
16 \closing{Mit freundlichen Grüßen}
17 \end{letter}
18
19 \end{document}
```

Listing 6: Einfaches L<sup>A</sup>T<sub>E</sub>X Beispiel

## Python Code – Zwischenschritt

- ▶ Test, ob a) das Template aus der Datei geladen wird und b) die Substitution funktioniert.

```
1 import string
2 import os
3
4 class LaTeXTemplate(string.Template):
5     delimiter = "%%"
6
7     with open ("sclrttr2.tex", "r") as template:
8         data=template.read()
9
10        with open ("sclrttr2_final.tex", "w") as letter:
11            s = LaTeXTemplate(data)
12            letter.write(s.substitute(who='Mickey Mouse', amount='10,00'))
13            letter.close();
14
15    template.close()
16
17    os.system("pdflatex sclrttr2_final.tex")
18    os.startfile("sclrttr2_final.pdf")
```

Listing 7: Lauffähiger Code für einen einzelnen Brief

## Python Code – Finaler Code

```
1 import string, os, csv
2
3 class LaTeXTemplate(string.Template):
4     delimiter = "%%"
5
6 with open('spender.csv','rb') as csvfile:
7     spender = csv.DictReader(csvfile, delimiter=',')
8
9     with open ("sclttr2.tex", "r") as template:
10         data=template.read()
11         template.close()
12
13 for row in spender:
14     name = row["Spender"]
15     summe = row["Summe"]
16
17     with open (name.replace(" ", "") + ".tex", "w") as letter:
18         s = LaTeXTemplate(data)
19         letter.write(s.substitute(who=name, amount=summe))
20         letter.close();
21         os.system("pdflatex " + name.replace(" ", "") + ".tex")
22
23 csvfile.close()
```

Listing 8: Fertiger Code



# Ergebnis

Duck Foundation  
Erpelweg 1, Entenhausen

Duck Foundation, Erpelweg 1, Entenhausen

Mickey Mouse

—  
-

13. Juli 2013

Sehr geehrte(r) Mickey Mouse,  
wir danken für Ihre Spende in Höhe von 10,00 Euro.  
Mit freundlichen Grüßen

Duck Foundation

# Jinja2

- ▶ <http://jinja.pocoo.org/docs>
- ▶ Sandboxing
- ▶ Template Vererbung
- ▶ Leicht zu debuggen
- ▶ Konfigurierbare Syntax
- ▶ Eingebaute Makro-Sprache ⇐ interessant!
- ▶ nutzt `{{Variable}}` zur Expansion, muss umdefiniert werden.

```
1 from jinja2 import Template
2
3 template = Template('{{greeting}} {{ name }}!')
4 print(template.render(greeting='Hello',name='LaTeX'))
```

Listing 9: Jinja2 Beispiel

# Jinja2 & L<sup>A</sup>T<sub>E</sub>X I

```
1 # siehe http://e6h.de/post/11/
2 import jinja2
3 import os
4 from jinja2 import Template
5
6 latex_jinja_env = jinja2.Environment(
7     block_start_string = '\BLOCK{',
8     block_end_string = '}',
9     variable_start_string = '\VAR{',
10    variable_end_string = '}',
11    comment_start_string = '\#{',
12    comment_end_string = '}',
13    line_statement_prefix = '%-',
14    line_comment_prefix = '%#',
15    trim_blocks = True,
16    autoescape = False,
17    loader = jinja2.FileSystemLoader(os.path.abspath('.'))
18 )
19
20 template = latex_jinja_env.get_template('test.tex')
21 print(template.render(abc='Hello'))
```

Listing 10: Jinja2 Beispiel

# Jinja2 & L<sup>A</sup>T<sub>E</sub>X II

```
1 \section{\VAR{abc}}
```

Listing 11: L<sup>A</sup>T<sub>E</sub>X-Vorlage

```
1 \section{Hello}
```

Listing 12: Jinja2 Ergebniszeile

## Jinja2 & L<sup>A</sup>T<sub>E</sub>X III – Nutzung der Skriptsprache

- ▶ einfaches Beispiel
- ▶ nützlich für Tabellen, etc.

```
1 from jinja2 import Template
2
3 itemlist = ['first', 'second', 'third']
4
5 template = Template("\begin{itemize}\n"
6                     "% for item in liste %}"
7                     " \item {{item}}\n"
8                     "% endfor %}"
9                     "\end{itemize}")
10
11 print(template.render(liste=itemlist))
```

Listing 13: Jinja2 Minimalbeispiel für Schleifen

# Abfragen von Datenbanken

- ▶ Gängige Datenbanken
  - ▶ MySQL/MariaDB, Postgres
  - ▶ SQLite
  - ▶ ODBC
- ▶ als Beispiel jetzt SQLite

## SQLite

- ▶ Relationale Datenbank
- ▶ nur 350 KB C-Bibliothek
- ▶ unterstützt in-memory Datenbanken
- ▶ ist ACID-compliant<sup>3</sup>
- ▶ unterstützt Großteil des SQL Standards

---

<sup>3</sup>Atomicity, Consistency, Isolation, Durability

# SQLite: Erstellen der Datenbank

```
1 import sqlite3
2
3 conn = sqlite3.connect('Spenden.db') # oder ":memory:"
4 c = conn.cursor()
5
6 # Erstellen der Tabelle
7 c.execute("CREATE TABLE Spenden (Spender text, Summe real)")
8 c.execute("INSERT INTO Spenden VALUES ('Donald Duck',10.00)")
9 c.execute("INSERT INTO Spenden VALUES ('Daisy Duck',20.00)")
10 c.execute("INSERT INTO Spenden VALUES ('Gustav Gans',50.00)")
11 c.execute("INSERT INTO Spenden VALUES ('Mickey Mouse',80.00)")
12 c.execute("INSERT INTO Spenden VALUES ('Dagobert Duck',00.10)")
13
14 # Commit
15 conn.commit()
16
17 # Schliessen der Verbindung
18 conn.close()
```

Listing 14: Setup der Datenbank

# SQLite: Abfrage der Datenbank

```
1 import sqlite3
2
3 conn = sqlite3.connect('Spenden.db')
4 c = conn.cursor()
5
6 for row in c.execute('SELECT * from Spenden'):
7     print row[0] + " : " + str(row[1])
8
9 conn.close()
```

Listing 15: Abfrage der Datenbank



## SQLite: Zusammenbauen der Briefe

```
1 import string, os, csv, sqlite3
2
3 class LaTeXTemplate(string.Template):
4     delimiter = "%%"
5
6 with open ("scrlttr2.tex", "r") as template:
7     data=template.read()
8     template.close()
9
10 conn = sqlite3.connect('Spenden.db')
11 c = conn.cursor()
12
13 for row in c.execute('SELECT * from Spenden'):
14     with open (row[0].replace(" ", "") + ".tex", "w") as letter:
15         s = LaTeXTemplate(data)
16         letter.write(s.substitute(who=row[0], amount=row[1]))
17         letter.close();
18         os.system("pdflatex " + row[0].replace(" ", "") + ".tex")
19
20 conn.close()
```

Listing 16: Abfrage der Datenbank

# Beautiful Soup4

- ▶ Aufgabe: Webseiten in  $\text{\LaTeX}$ -Code verwandeln
- ▶ Beautiful Soup: HTML/XML Scraping Bibliothek
- ▶ übernimmt die hässliche Aufgabe des Parsens
- ▶ Einführung unter <http://www.pythonforbeginners.com/python-on-the-web/beautifulsoup-4-python/>

## Installation

```
1 pip install beautifulsoup4
2 easy_install beautifulsoup4
```

## BS4: Aufbau einer HTML Datei

```
1 <HTML>
2 <HEAD>
3 <TITLE>Eine einfache HTML-Datei</TITLE>
4 <meta name="description" content="A simple HTML page for BS4">
5 <meta name="author" content="Uwe Ziegenhagen">
6 <meta charset="UTF-8">
7 </HEAD>
8 <BODY>
9
10 <H1>Hallo Welt</H1>
11
12 <p>Ein kurzer Absatz mit ein wenig Text, der relativ nichtssagend ist.</p>
13
14 <H1>Nochmal Hallo Welt!</H1>
15
16 <p>Schon wieder ein kurzer Absatz mit ein wenig Text, der genauso
17 nichtssagend ist wie der Absatz zuvor.</p>
18 </BODY>
19 </HTML>
```

Listing 17: Abfrage der Datenbank

## BS4: Einführendes Beispiel

```
1 from bs4 import BeautifulSoup
2
3 with open ("simple.html", "r") as htmlsource:
4     html=htmlsource.read()
5
6 soup = BeautifulSoup(html)
7 # print soup.prettify()
8
9 print soup.title
10 print soup.title.string
11 print soup.p.string
12 print soup.a
```

Listing 18: Abfrage der Datenbank

<title>Eine einfache HTML-Datei</title>

Eine einfache HTML-Datei

Ein kurzer Absatz mit ein wenig Text, der relativ nichtssagend ist.

None

## BS4: Praxisbeispiel

- ▶ „alltägliche Missgeschicke“ unter [www.fmylife.com](http://www.fmylife.com)
- ▶ Ziel: auf dem iPad offline lesen

```
1 import urllib2
2 from bs4 import BeautifulSoup
3
4 # http://stackoverflow.com/questions/1752662/beautifulsoup-easy-way-to-
  # to-obtain-html-free-contents
5 def textOf(soup):
6     return u''.join(soup.findAll(text=True))
7
8 soup = BeautifulSoup(urllib2.urlopen('http://www.fmylife.com/').read())
9
10 for item in soup.findAll('div', attrs={'class': 'post article'}):
11     item = textOf(item)
12     print item[:item.find("FML#")]
```

Listing 19: Abfrage der Datenbank

Nach dem Schreiben und Sortieren in einer SQLite DB manuelle  
Bearbeitung des T<sub>E</sub>X-Codes: ⇒ 2500 Seiten A5-Dokument

## Python aus $\text{\LaTeX}$ heraus `\write18`

- ▶  $\text{\LaTeX}$  kennt den `\write18` Befehl
- ▶ spezieller Ausgabestream, ähnlich zu Pythons `os.system()`
- ▶ standardmäßig deaktiviert, da mögliche Sicherheitslücke
- ▶ muss mittels `--shell-escape` aktiviert werden
- ▶ Idee
  - ▶ Python-Code im Dokument speichern
  - ▶ diesen Code beim Übersetzen ausführen lassen
  - ▶ Ergebnisse in das  $\text{\LaTeX}$ -Dokument übernehmen
- ▶ erster Ansatz: was „Selbstgestricktes“
- ▶ zweiter Ansatz: `PythonTeX`

# 1. Ansatz: Selbstbau (T<sub>E</sub>X für Fortgeschrittene)

```
1 \makeatletter
2 \newenvironment{pycode}[1]% Speichere Inhalt von pycode
3   {\xdef\d@tn@me{#1}% komisches TeX-Zeug
4   \xdef\r@ncmd{python #1.py > #1.plog}% Aufruf von Python
5   \typeout{Writing file #1}\VerbatimOut{#1.py}%
6   }
7   {\endVerbatimOut %
8   \toks0{\immediate\write18}%
9   \expandafter\toks\expandafter1\expandafter{\r@ncmd}%
10  \edef\d@r@ncmd{\the\toks0{\the\toks1}}\d@r@ncmd %
11 \begin{columns}% zwei Spalten
12 \begin{column}{0.45\textwidth}% Spalte 1 Quellcode
13 \lstinputlisting{\d@tn@me.py}%
14 \end{column}% Ende 1. Spalte
15 \begin{column}{0.5\textwidth}%
16 \lstinputlisting{\d@tn@me.plog}% Spalte 2 Ergebnis
17 \end{column}% Ende 2. Spalte
18 \end{columns}}
19 \makeatother
```

Listing 20: Lösung für Beamer-Folien

# Einfaches Beispiel

```
1 \begin{pycode}{abc}
2 import datetime
3 now = datetime.datetime.now()
4 print str(now)
5
6 print('Hello World')
7 print(1+2)
8 \end{pycode}
```

```
1 import datetime
2 now = datetime.datetime.now()
3 print str(now)
4
5 print('Hello World')
6 print(1+2)
```

```
1 2013-08-26 04:07:32.914000
2 Hello World
3 3
```



## 2. Ansatz: Python $\text{\TeX}$

- ▶ von Geoffrey Poore, letzte Version 0.12 beta vom 24.06.2013
- ▶ <https://github.com/gpoore/pythontex>
- ▶ Installations-Skript<sup>4</sup> für  $\text{\TeX}$  Live<sup>5</sup>
- ▶ Befehle
  - ▶ `\py <Code>` setzt das Ergebnis
  - ▶ `\pyc <Code>` führt den Code aus
  - ▶ `\pyb <Code>` führt aus und setzt Ergebnis
  - ▶ `\pyv <Code>` setzt nur den Code
- ▶ Umgebungen
  - ▶ `pycode` wie `\pyc`
  - ▶ `pyblock` wie `\pyb`
  - ▶ `pyverbatim` wie `\pyv`
  - ▶ `pyconsole` emuliert eine Python-Konsole

---

<sup>4</sup>`python pythontex_install_texlive.py`

<sup>5</sup>mit Mik $\text{\TeX}$  die gängigste Distribution

# Python<sub>T</sub>E<sub>X</sub> Beispiel

## Workflow

- ▶ `pdflatex dateiname.tex`
- ▶ `pythontex dateiname`
- ▶ `pdflatex dateiname.tex`

```
1 \documentclass[12pt]{scrartcl}
2 \usepackage[T1]{fontenc}
3 \usepackage[utf8]{inputenc}
4 \usepackage{pythontex}
5
6 \begin{document}
7
8 \py{2+5}
9
10 \end{document}
```

Listing 21: Python<sub>T</sub>E<sub>X</sub> Beispiel

# Fazit

- ▶ Automatisierung
  - ▶ vermindert „Operational Risk“
  - ▶ erspart langweilige, manuelle Arbeit
  - ▶ kann unglaublich viel Zeit sparen
- ▶  $\LaTeX$  + Python mächtige Kombination
- ▶ CSV, Datenbank, WWW, etc.  $\Rightarrow$  kein Problem
- ▶ Interessante Gebiete
  - ▶ JSON2 $\LaTeX$
  - ▶ NumPy, SciPi
  - ▶ Visualisierung
  - ▶ LaTeX und Python in Emacs Org Mode