Combining LaTeX with Python

Uwe Ziegenhagen

August 9, 2019

Dante e. V. Heidelberg

About me

- Uwe Ziegenhagen, from Cologne, Germany
- In-house analyst in banking and private equity
- Responsible for developing and maintaining individual software applications
- Teacher for IT-related subjects at a private University of Applied Sciences

What’s this talk about?

- LaTeX-files are pure text files, so pretty much any programming language can be used to create them
- Python has been my favourite programming language
- Python is sufficiently fast, easy to learn and has a huge set of libraries
- This talk is about Python and the way we can utilize it with LaTeX

Today's Topics

- Introducing Python
- Creating LaTeX files with Python
- Running Python from within LaTeX

Python

- Is a general purpose scripting language
- Comes with a rich standard library $\Rightarrow$ “batteries included”
- Was invented 1991 by Guido van Rossum at the Centrum Wiskunde & Informatica in the Netherlands, Version 1.0 in 1994
- Comes in version 2.7 and 3.x $\Rightarrow$ use version 3!

Python Design Philosophy

- Open source
- Simple, intuitive, but incredibly powerful
- Source code as readable as plain English
- Is suitable for everyday jobs as well as for machine learning
- Offers short development cycles
- Uses indentation, not brackets

Some basic Python

The usual “Hello World!”

```python
print('Hello' + ' ' + 'World')
```

Some function definition

```python
def addTwo(a, b):
    return a+b

print(addTwo(5,3))
print(addTwo('U','S'))
```

Interation over lists, arrays, etc.

```python
some_string = 'Hello TUG!'
for i in some_string:
    print(i)
```

Some functional Python

Mapping a function on a list

```
some_list = [1, 2, 3, 4, 5, 6]
g = lambda x : x**2
print(list(map(g,some_list)))
```

Filtering even values from a list

```
some_list = [1, 2, 3, 4, 5, 6, 7, 8]
result = filter(lambda x: x % 2 == 0, some_list)
print(list(result))
```

Some object-oriented Python

Classes and objects

```python
class Person:

    def __init__(self, name, age):
        self.name = name
        self.age = age

    def print_age(self):
        print(self.name + ' is ' + str(self.age))

john = Person('John', 50)
john.print_age()
```

Today’s Topics

- Introducing Python ✓
- Creating LaTeX files with Python
- Doing Python within LaTeX

Creating Files

Writing LaTeX-Files I

- Context manager `with`
- takes care of errors and closes the file handle afterwards
- Backslashes need to be escaped[1]

```
with open('sometexfile.tex','w') as file:
    file.write('\\documentclass{article}\n')
    file.write('\\begin{document}\n')
    file.write('Hello Palo Alto!\n')
    file.write('\\end{document}\n')
```

Listing 1: Writing TeX-files

[1] There are “raw” strings `r'hello'` as well…

Writing LaTeX-Files II

```
import subprocess, os

with open('sometexfile.tex','w') as file:
    file.write('\\documentclass{article}\n')
    file.write('\\begin{document}\n')
    file.write('Hello Palo Alto!\n')
    file.write('\\end{document}\n')

x = subprocess.call('pdflatex sometexfile.tex')
if x != 0:
    print('Exit-code not 0, check result!')
else:
    os.system('start sometexfile.pdf')
```

Listing 2: Writing & Processing TeX-files

Dynamic Text Replacements I

- Define variable `place`
- Read template file with `$variable$` inside
- Replace `$SomePlace$` with variable
- Write new file

```python
place = 'Palo Alto'

with open('someplace.tex','r') as myfile:
    text = myfile.read()
    text_new = text.replace('$SomePlace$', place)

    with open('someplace_new.tex', 'w') as output:
        output.write(text_new)
```

Listing 3: Replacing text

Dynamic Text Replacements II

- Approach can be easily extended
- `kv` is a key-value dictionary

```
kv = {'place':'Palo Alto', 'month':'August'}

with open('sometemplate.tex','r') as myfile:
    text = myfile.read()

    for key, value in kv.items():
        text = text.replace('$'+key+'$', value)

    with open('someplace_new2.tex', 'w') as output:
        output.write(text)
```

Listing 4: Replacing text with dictionaries

Python’s Jinja2 Template System

- Approach works, but it’s like “re-inventing the wheel”
- Python offers a variety of template engines[2]
- Some template engines even allow templates to be mixed with logic
- I have worked with Jinja2[3]: full Unicode support, sandboxed execution, template inheritance, etc.
- “For instance you can reconfigure Jinja2 to better fit output formats such as LaTeX or JavaScript.”

[2] See https://wiki.python.org/moin/Templating
[3] http://jinja.pocoo.org/docs/2.10/

Jinja2 – A Basic (non-TEX) Example

```python
from jinja2 import Template

mytemplate = Template("Hello {{place}}!")
print(mytemplate.render(place="Palo Alto"))

mytemplate = Template("Some numbers: {% for n in range
    (1,10) %}{{n}}{% endfor %}")
print(mytemplate.render())
```

Listing 5: A Jinja2 example

What can we learn from this example:

1. Syntax is (easily) understandable
2. Jinja2 brings its own notation for looping, etc.
3. Extensive use of “{”, “%”, “}”

Jinja2 for LaTeX

```python
import os
import jinja2

latex_jinja_env = jinja2.Environment(
    block_start_string = '\BLOCK{',
    block_end_string = '}',
    variable_start_string = '\VAR{',
    variable_end_string = '}',
    comment_start_string = '\#{',
    comment_end_string = '}',
    line_statement_prefix = '%-',
    line_comment_prefix = '%#',
    trim_blocks = True,
    autoescape = False,
    loader = jinja2.FileSystemLoader(os.path.abspath('.'))
    )
```

Jinja2 for LaTeX- Some Explanation

- based on `https://web.archive.org/web/20121024021221/http://e6h.de/post/11/`
- allows to load templates from the file system
- redefines the template structure:

  single variables instead of “{{ }}” we use `\VAR{}`
  logic blocks instead of `\{% %\}` we use `\BLOCK{}`
- both commands will be defined in the document as empty commands via `\newcommand` (so the template can be compiled as well)

Jinja Example generating LaTeX I

```latex
\documentclass[12pt,english]{article}
\usepackage[T1]{fontenc}
\usepackage{babel}

\newcommand{\VAR}[1]{}
\newcommand{\BLOCK}[1]{}

\begin{document}

Hello \VAR{place}!

\end{document}
```

Listing 6: LaTeX Template for Jinja2

Jinja Example generating LaTeX II

- Excerpt from the complete code
- Running the Python Code replaces the placeholders with content

```python
# load template from file
template = latex_jinja_env.get_template('jinja-01.tex')
# combine template and variables
document = template.render(place='Palo Alto')
#write document
with open('final-02.tex','w') as output:
    output.write(document)
```

Listing 7: Rendering the document

Jinja Example generating LaTeX II: Output

```latex
\documentclass[12pt,english]{article}
\usepackage[T1]{fontenc}
\usepackage{babel}

\newcommand{\VAR}[1]{}
\newcommand{\BLOCK}[1]{}

\begin{document}

Hello Palo Alto!

\end{document}
```

Listing 8: The generated document

Extending the Python Code

- Excerpt from full code, uses a list of cities
- Save each file under <cityname>.tex, replaces spaces in filename
- Could be extended to run in parallel threads: https://www.uweziegenhagen.de/?p=3501

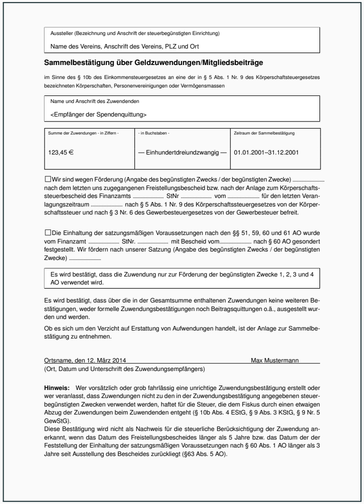
```python
template = latex_jinja_env.get_template('jinja-01.tex')
list_of_towns = ['Berlin', 'New York', 'Tokyo']

for town in list_of_towns:
    document = template.render(place=town)
    with open(town.replace(' ','') + '.tex','w') as output:
        output.write(document)
    x = subprocess.call('pdflatex '+ town.replace(' ','') +'.tex')
    if x != 0:
        print('Exit-code not 0 for ' + town + ', check Code!')
```

Jinja 2 - A complex example

- For several years I had been treasurer for a charitable makerspace in Cologne
- Donations were tax deductable, receipts had to be made following some strict template
- Relies heavily on pandas library, which offers R-like "DataFrames"
- Uses Excel files for the storage of data
- See `https://www.uweziegenhagen.de/?p=3359` for the code

The LaTeX Template for the Tax Receipts



Aussteller (Bezeichnung und Anschrift der steuerbegünstigten Einrichtung)
Name des Vereins, Anschrift des Vereins, PLZ und Ort

Sammelbestätigung über Geldzuwendungen/Mitgliedsbeiträge

im Sinne des § 10b des Einkommensteuergesetzes an eine der in § 5 Abs. 1 Nr. 9 des Körperschaftsteuergesetzes bezeichneten Körperschaften, Personenvereinigungen oder Vermögensmassen

Name und Anschrift des Zuwendenden
<Empfänger der Spendenquittung>

Summe der Zuwendungen - in Ziffern -	- in Buchstaben -	Zeitraum der Sammelbestätigung
123,45 €	— Einhundertdreiundzwangig —	01.01.2001–31.12.2001

☐Wir sind wegen Förderung (Angabe des begünstigten Zwecks / der begünstigten Zwecke) .... nach dem letzten uns zugegangenen Freistellungsbescheid bzw. nach der Anlage zum Körperschaftssteuerbescheid des Finanzamts .... StNr .... vom .... für den letzten Veranlagungszeitraum .... nach § 5 Abs. 1 Nr. 9 des Körperschaftssteuergesetzes von der Körperschaftssteuer und nach § 3 Nr. 6 des Gewerbesteuergesetzes von der Gewerbesteuer befreit.

☐Die Einhaltung der satzungsmäßigen Voraussetzungen nach den §§ 51, 59, 60 und 61 AO wurde vom Finanzamt .... StNr. .... mit Bescheid vom.................... nach § 60 AO gesondert festgestellt. Wir fördern nach unserer Satzung (Angabe des begünstigten Zwecks / der begünstigten Zwecke) ....

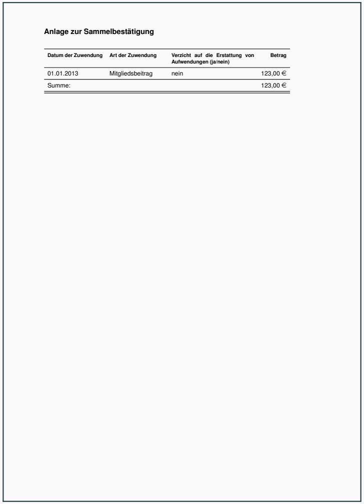
Es wird bestätigt, dass die Zuwendung nur zur Förderung der begünstigten Zwecke 1, 2, 3 und 4 AO verwendet wird.

Es wird bestätigt, dass über die in der Gesamtsumme enthaltenen Zuwendungen keine weiteren Bestätigungen, weder formelle Zuwendungsbestätigungen noch Beitragsquittungen o.ä., ausgestellt wurden und werden.

Ob es sich um den Verzicht auf Erstattung von Aufwendungen handelt, ist der Anlage zur Sammelbestätigung zu entnehmen.

Ortsname, den 12. März 2014 Max Mustermann
(Ort, Datum und Unterschrift des Zuwendungsempfängers)

Hinweis: Wer vorsätzlich oder grob fahrlässig eine unrichtige Zuwendungsbestätigung erstellt oder wer veranlasst, dass Zuwendungen nicht zu den in der Zuwendungsbestätigung angegebenen steuerbegünstigten Zwecken verwendet werden, haftet für die Steuer, die dem Fiskus durch einen etwaigen Abzug der Zuwendungen beim Zuwendenden entgeht (§ 10b Abs. 4 EStG, § 9 Abs. 3 KStG, § 9 Nr. 5 GewStG).
Diese Bestätigung wird nicht als Nachweis für die steuerliche Berücksichtigung der Zuwendung anerkannt, wenn das Datum des Freistellungsbescheides länger als 5 Jahre bzw. das Datum der der Feststellung der Einhaltung der satzungsmäßigen Voraussetzungen nach § 60 Abs. 1 AO länger als 3 Jahre seit Ausstellung des Bescheides zurückliegt (§63 Abs. 5 AO).



Anlage zur Sammelbestätigung

Datum der Zuwendung	Art der Zuwendung	Verzicht auf die Erstattung von Aufwendungen (ja/nein)	Betrag
01.01.2013	Mitgliedsbeitrag	nein	123,00 €
Summe:			123,00 €

Today's Topics

- Introducing Python ✓
- Creating LaTeX files with Python ✓
- Using Python from LaTeX

Using Python from LaTeX

Several approaches

- Similar projects like Sweave and knitR exist for Python as well:
    - knitpy (`https://pypi.org/project/knitpy/`)
    - pyLit and PyReport for literate programming
- I want to show two other approaches
    - Plain-vanilla code execution
    - PythonTeX

My “own” Approach (Thank you, Google and TSX)

- Basic idea:
    - Use special LaTeX environment
    - During compilation, write its content to external file
    - Run external program on this file, create output file (requires `--shell-escape` enabled)
    - Include this output in the LaTeX output, here with syntax highlighting by the `minted` package
- Advantage: Needs only a single LaTeX-run, can be adapted for other languages
- Disadvantage: Needs adjustments if non-text output is to be included, always writes and executes with each LaTeX-run

Example

```latex
\usepackage{fancyvrb}
\makeatletter
\newenvironment{pycode}[1]%
  {\xdef\d@tn@me{#1}\xdef\r@ncmd{python #1.py > #1.plog}%
  \typeout{Writing file #1}\VerbatimOut{#1.py}%
  }
  {\endVerbatimOut %
 \toks0{\immediate\write18}%
 \expandafter\toks\expandafter1\expandafter{\r@ncmd}%
 \edef\d@r@ncmd{\the\toks0{\the\toks1}}\d@r@ncmd %
 \noindent Input
 \inputminted{python}{\d@tn@me.py}%
 \noindent Output
 \inputminted{text}{\d@tn@me.plog}%
}
\makeatother
```

Listing 9: Write ext. file and execute

Example

Used in the document as following:

```
\begin{document}

\begin{pycode}{abc}
import pandas as pd
print(pd.__version__);
print(1+123424)
\end{pycode}


\end{document}
```

Listing 10: Write ext. file and execute

Result

Input

```python
import pandas as pd
print(pd.__version__);
print(1+123424)
```

Output

```
0.24.2
123425
```

PythonTEX

- Different approach: `PythonTEX` package by Geoffrey Poore[4], also author of the `minted` package for syntax highlighting
- Workflow:
  - embed Python-code in LATEX documents
  - run LATEX on the document
  - run `pythontex` on the file
  - run LATEX on the document
- Python-code is only rerun if it has been modified
- Supports parallelization and non-text output

[4]`https://github.com/gpoore/pythontex`

A simple PythonTeX example [5]

```latex
%!TEX TS-program = Arara
% arara: pdflatex: {shell: yes}
% arara: pythontex
% arara: pdflatex: {shell: yes}
\documentclass[12pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{pythontex} % <--
\begin{document}

\py{2+2}

\end{document}
```

Listing 11: A simple PythonTeX example

[5] Using a custom Arara rule, see `https://tex.stackexchange.com/questions/357881/arara-rule-for-pythontex`

PythonTeX commands and environments I

PythonTeX offers various inline commands:

- `\py{<expression>}` prints value of expression
- `\pyc{<code>}` executes `code`, output goes to STDOUT
- `\pys{<code>}` supports variable and expression substitution
- `\pyb{<code>}` execute code and prettyprint result
- `\pyv{<code>}` prettyprint code

PythonTeX commands and environments II

PythonTeX also offers various environments:

- `pycode` executed, but not typeset
- `pysub` variable and expression substitution
- `pyverbatim` typeset, but not executed
- `pyblock` executed and typeset
- `pyconsole` simulates an interactive Python-console

Getting stockquotes

```
\documentclass[12pt]{article}
\usepackage[utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{pythontex}
\usepackage{booktabs}
\begin{document}

\pyc{from yahoo_fin import stock_info as si}

\begin{tabular}{lr} \toprule
Company & Latest quote \\ \midrule
Apple & \py{round(si.get_live_price("aapl"),2)} \\
Amazon & \py{round(si.get_live_price("amzn"),2)} \\
Facebook & \py{round(si.get_live_price("fb"),2)} \\ \bottomrule
\end{tabular}

\end{document}
```

Listing 12: Write ext. file and execute

Resulting document

Document was compiled using:

1. `pdflatex`
2. `pythontex`
3. `pdflatex`

Company	Latest quote
Apple	203.43
Amazon	1832.89
Facebook	190.16

Summary

Summary

- Python is easy to learn and powerful ✓
- Creating LaTeX files is simple ✓
- We can (easily) control Python from LaTeX ✓
- For questions and comments please contact me

ziegenhagen@gmail.com

This presentation

- Clicking 📄 and 📄 opens the example files (at least in Adobe Reader)
- LaTeX-source 📄
- Document class: Beamer
- Document theme: Metropolis
- Font: Source Sans Pro