

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

- $\text{\LaTeX}$ -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  $\text{\LaTeX}$

# Today's Topics


- Introducing Python
- Creating  $\text{\LaTeX}$  files with Python
- Running Python from within  $\text{\LaTeX}$

- 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!” 

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

Some function definition 

```
1 def addTwo(a, b):  
2     return a+b  
3  
4 print(addTwo(5,3))  
5 print(addTwo('U','S'))
```


Iteration over lists, arrays, etc. 

```
1 some_string = 'Hello TUG!'  
2 for i in some_string:  
3     print(i)
```

# Some functional Python

Mapping a function on a list 

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

Filtering even values from a list 

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



# Some object-oriented Python

## Classes and objects

```
1 class Person:
2
3     def __init__(self, name, age):
4         self.name = name
5         self.age = age
6
7     def print_age(self):
8         print(self.name + ' is ' + str(self.age))
9
10 john = Person('John', 50)
11 john.print_age()
```

# Today's Topics

- Introducing Python ✓
- Creating  $\text{\LaTeX}$  files with Python
- Doing Python within  $\text{\LaTeX}$

# Creating Files

---

# Writing L<sup>A</sup>T<sub>E</sub>X-Files I

- Context manager with
- takes care of errors and closes the file handle afterwards
- Backslashes need to be escaped<sup>1</sup>

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

Listing 1: Writing T<sub>E</sub>X-files 

---

<sup>1</sup>There are “raw” strings `r'hello'` as well...

# Writing L<sup>A</sup>T<sub>E</sub>X-Files II

```
1 import subprocess, os
2
3 with open('sometexfile.tex','w') as file:
4     file.write('\\documentclass{article}\\n')
5     file.write('\\begin{document}\\n')
6     file.write('Hello Palo Alto!\\n')
7     file.write('\\end{document}\\n')
8
9 x = subprocess.call('pdflatex sometexfile.tex')
10 if x != 0:
11     print('Exit-code not 0, check result!')
12 else:
13     os.system('start sometexfile.pdf')
```

Listing 2: Writing & Processing T<sub>E</sub>X-files 

# Dynamic Text Replacements I

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

```
1 place = 'Palo Alto'
2
3 with open('someplace.tex','r') as myfile:
4     text = myfile.read()
5     text_new = text.replace('$SomePlace$', place)
6
7     with open('someplace_new.tex', 'w') as output:
8         output.write(text_new)
```

Listing 3: Replacing text 

# Dynamic Text Replacements II

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

```
1 kv = {'place':'Palo Alto', 'month':'August'}
2
3 with open('somemplate.tex','r') as myfile:
4     text = myfile.read()
5
6     for key, value in kv.items():
7         text = text.replace('$'+key+'$', value)
8
9     with open('someplace_new2.tex', 'w') as output:
10         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<sup>2</sup>
- Some template engines even allow templates to be mixed with logic
- I have worked with Jinja2<sup>3</sup>: full Unicode support, sandboxed execution, template inheritance, etc.
- “For instance you can reconfigure Jinja2 to better fit output formats such as LaTeX or JavaScript.”

---

<sup>2</sup>See <https://wiki.python.org/moin/Templating>

<sup>3</sup><http://jinja.pocoo.org/docs/2.10/>



# Jinja2 – A Basic (non-TeX) Example

```
1 from jinja2 import Template
2
3 mytemplate = Template("Hello {{place}}!")
4 print(mytemplate.render(place="Palo Alto"))
5
6 mytemplate = Template("Some numbers: {% for n in range
7                        (1,10) %}{{n}}{% endfor %}")
8 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 “{”, “%”, “}”


```
1 import os
2 import jinja2
3
4 latex_jinja_env = jinja2.Environment(
5     block_start_string = '\BLOCK{',
6     block_end_string = '}',
7     variable_start_string = '\VAR{',
8     variable_end_string = '}',
9     comment_start_string = '\#{',
10    comment_end_string = '}',
11    line_statement_prefix = '%-',
12    line_comment_prefix = '%#',
13    trim_blocks = True,
14    autoescape = False,
15    loader = jinja2.FileSystemLoader(os.path.abspath('.'))
16 )
```

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

```
1 \documentclass[12pt,english]{article}
2 \usepackage[T1]{fontenc}
3 \usepackage{babel}
4
5 \newcommand{\VAR}[1]{}
6 \newcommand{\BLOCK}[1]{}
7
8 \begin{document}
9
10 Hello \VAR{place}!
11
12 \end{document}
```

Listing 6: LaTeX Template for Jinja2 

# Jinja Example generating L<sup>A</sup>T<sub>E</sub>X II

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

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

Listing 7: Rendering the document 

# Jinja Example generating L<sup>A</sup>T<sub>E</sub>X II: Output

```
1 \documentclass[12pt,english]{article}
2 \usepackage[T1]{fontenc}
3 \usepackage{babel}
4
5 \newcommand{\VAR}[1]{}
6 \newcommand{\BLOCK}[1]{}
7
8 \begin{document}
9
10 Hello Palo Alto!
11
12 \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>

```
1 template = latex_jinja_env.get_template('jinja-01.tex')
2 list_of_towns = ['Berlin', 'New York', 'Tokyo']
3
4 for town in list_of_towns:
5     document = template.render(place=town)
6     with open(town.replace(' ', '') + '.tex', 'w') as output:
7         output.write(document)
8     x = subprocess.call('pdflatex ' + town.replace(' ', '') + '.tex')
9     if x != 0:
10         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 deductible, 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 L<sup>A</sup>T<sub>E</sub>X 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, Personengesellschaften oder Vermögensmassen

Name und Anschrift des Zuwendenden

<Empfänger der Spendengültung>

Summe der Zuwendungen in Zahlen

123,45 €

in Buchstaben

— Einhundertdreißig und zwanzig —

Datum der Sammelbestätigung

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 Körperschaftsteuerbescheid des Finanzamts ..... SINr. .... vom ..... für den letzten Veranlagungszeitraum ..... nach § 5 Abs. 1 Nr. 9 des Körperschaftsteuergesetzes von der Körperschaftsteuer 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 ..... SINr. .... 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.

Ort, Datum, 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  $\text{\LaTeX}$  files with Python ✓
- Using Python from  $\text{\LaTeX}$

# 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
  - Python $\text{\TeX}$

# 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

```
1 \usepackage{fancyvrb}
2 \makeatletter
3 \newenvironment{pycode}[1]%
4   {\xdef\d@tn@me{#1}\xdef\r@ncmd{python #1.py > #1.plog}%
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 \noindent Input
12 \inputminted{python}{\d@tn@me.py}%
13 \noindent Output
14 \inputminted{text}{\d@tn@me.plog}%
15 }
16 \makeatother
```

Listing 9: Write ext. file and execute 

# Example

Used in the document as following:

```
1 \begin{document}
2
3 \begin{pycode}{abc}
4 import pandas as pd
5 print(pd.__version__);
6 print(1+123424)
7 \end{pycode}
8
9
10 \end{document}
```

Listing 10: Write ext. file and execute 

# Result

Input

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

Output

```
1 0.24.2
2 123425
```





- Different approach: PythonT<sub>E</sub>X package by Geoffrey Poore<sup>4</sup>, also author of the `minted` package for syntax highlighting
- Workflow:
  - embed Python-code in L<sup>A</sup>T<sub>E</sub>X documents
  - run L<sup>A</sup>T<sub>E</sub>X on the document
  - run `pythontex` on the file
  - run L<sup>A</sup>T<sub>E</sub>X on the document
- Python-code is only rerun if it has been modified
- Supports parallelization and non-text output

---

<sup>4</sup><https://github.com/gpoore/pythontex>

# A simple PythonTeX example <sup>5</sup>

```
1 %!TEX TS-program = Arara
2 % arara: pdflatex: {shell: yes}
3 % arara: pythontex
4 % arara: pdflatex: {shell: yes}
5 \documentclass[12pt]{article}
6 \usepackage[utf8]{inputenc}
7 \usepackage[T1]{fontenc}
8 \usepackage{pythontex} % <--
9 \begin{document}
10
11 \py{2+2}
12
13 \end{document}
```

Listing 11: A simple PythonTeX example  

---

<sup>5</sup>Using a custom Arara rule, see <https://tex.stackexchange.com/questions/357881/arara-rule-for-pythontex>

PythonT<sub>E</sub>X 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

PythonT<sub>E</sub>X 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

```
1 \documentclass[12pt]{article}
2 \usepackage[utf8]{inputenc}
3 \usepackage[T1]{fontenc}
4 \usepackage{pythontex}
5 \usepackage{booktabs}
6 \begin{document}
7
8 \pyc{from yahoo_fin import stock_info as si}
9
10 \begin{tabular}{lr} \toprule
11 Company & Latest quote \\ \midrule
12 Apple & \py{round(si.get_live_price("aapl"),2)} \\
13 Amazon & \py{round(si.get_live_price("amzn"),2)} \\
14 Facebook & \py{round(si.get_live_price("fb"),2)} \\ \bottomrule
15 \end{tabular}
16
17 \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  $\text{\LaTeX}$  files is simple ✓
- We can (easily) control Python from  $\text{\LaTeX}$  ✓
- For questions and comments please contact me

ziegenhagen@gmail.com



# This presentation

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