PythonTEX
Zusammenspiel von Python und LATEX

Günter Partosch

Justus-Liebig-Universität Gießen, Hochschulrechenzentrum (HRZ)

Version 1.2.2, 8. April 2018

Inhalt des Vortrags

Zusammenfassung

Python hat sich in den letzten Jahren zu einer der wichtigsten Programmiersprache entwickelt. Das ist einmal begründet in der einfachen Syntax als auch in der großen Flexibilität. Zahlreiche Python-Module erweitern den Funktionsumfang und ermöglichen interessante Anwendungen.

Im Vortrag wird gezeigt, wie Sie mit Hilfe von PythonTEX diese schönen Eigenschaften in LATEX nutzen können. Den Abschluss bildet die detaillierte Darstellung zweier Fallbeispiele aus dem Oberstufen-Mathematik-Unterricht.

Links
Die jeweils neueste Version des Vortrags finden Sie unter http://www.staff.uni-giessen.de/partosch/unterlagen/pythontex.pdf, die Beispiele unter http://www.staff.uni-giessen.de/partosch/unterlagen/pythontex-beispiele.zip.

Python

Python

- entwickelt ab 1991 von Guido van Rossum; später von der Python Software Foundation; aktuelle Version 3.6.5
- ein **Entwicklungsziel**: gut lesbarer, knapper Programmierstil $\Longrightarrow$ Unterrichtssprache an Schulen und Hochschulen
  - untergeordnete Blöcke werden eingerückt – nicht geklammert
  - ganz zeilenomrientiert (mit wenigen Ausnahmen) – 1 Anweisung / Zeile
  - nur noch wenige Schlüsselwörter
- Python unterstützt u. a. **objektorientierte, strukturierte und funktionale Programmierung**
- **dynamische Datentypisierung**: Wert bestimmt den jeweiligen Typ; keine statische Typüberprüfung
- gut ausgebaute und gut gepflegte **Standardbibliothek** als Basis [van Rossum et al. 2018, `reference.pdf`]; Erweiterungen über zahlreiche Module [van Rossum et al. 2018, `library.pdf`]
- **Tutorium** [van Rossum et al. 2018, `tutorial.pdf`]
- **FAQs** [van Rossum et al. 2018, `faq.pdf`]
- **HowTos** zu verschiedenen Themen, z. B. Sortierung, reguläre Ausdrücke, funktionale Programmierung

Python – Datentypen (1)
einfache Typen (1)

- ganzzahlige Werte (`int`)
- Fließkomma-Werte (`float`)
- logische Werte (`bool`)
- komplexe Werte (`complex`)

Ausflug: Zuweisungen

```
a = 23                    # ganzzahliger Wert
a = 1.2; b = 2.34e-20     # Fließkomma-Werte; 2 Anw. in Zeile
a, b = 1, 20              # a = 1; b = 20
a = b = 5                 # b = 5; a = b; Mehrfachzuweisung
a += 25                   # a = a + 25
c1 = 2.3 + 4.5j           # komplexe Zahl
```

zulässige Operatoren

Typ	<	>	==	!=	>=	<=	+	-	*	**	/	//	%
ganzzahlig	x	x	x	x	x	x	x	x	x	x	x	x	x
Fließkomma	x	x	x	x	x	x	x	x	x	x	x		
logisch			x	x									
komplex			x	x			x	x	x	x	x		

bei `bool`: `and`, `or`, `not`

Python – Datentypen (2)
einfache Typen (2)

einige Funktionen für diese Datentypen

Funktion	Bedeutung
`abs(`*zahl*`)`	Absolutbetrag von *zahl*
`bin(`*zahl*`)`	binäre Darstellung von *zahl*
`divmod(`*zahl1*`, `*zahl2*`)`	liefert Tupel (*zahl1* // *zahl2*, *zahl1* % *zahl2*)
`hex(`*zahl*`)`	hexadezimale Darstellung von *zahl*
`oct(`*zahl*`)`	oktale Darstellung von *zahl*
`chr(`*zahl*`)`	liefert das Zeichen mit der Codierung *zahl*
`str(`*ausdruck*`)`	liefert zugehörige Zeichenketten-Repräsentierung

Python – Datentypen (3)
zusammengesetzte Typen (1)

Listen

```
werte = [2, "a", 1.3]
werte.append("abc")
werte.insert(1, 3.4)
```

Tupel

```
werte = (2, "a", 1.3)
#werte.append("abc")
#werte.insert(1, 3.4)
```

einige gemeinsame Konstrukte

```
l1 = [1, 2]; l2 = [3, 4]; l3 = l1 + l2 # Verkettung zweier Listen
t1 = (2, 3) + (3, 4, 5)# Verkettung zweier Tupel
i1 = len(l3)             # Länge
l10 = t1[0]              # 1. Element
i2 = l3[i1 - 1]          # letztes Element
l4 = l3[1:4]             # 1. bis 3. Element
```

Python – Datentypen (4)
zusammengesetzte Typen (2)

und dann gibt es noch Mengen

```
s1 = set('abc')
s2 = set('bcd')
print(s1 - s2)   # Differenzmenge
print(s1 | s2)   # Vereinigungsmenge
print(s1 & s2)   # Schnittmenge
print(s1 ^ s2)   # Symmetrische Differenz
print('a' in s1) # Element enthalten in
print(len(s1))   # Mächtigkeit einer Menge
```

und Wörterbücher

```
staedte ={'Gießen' : 75000, 'Marburg' : 60000}
for s in staedte:
    print(s, "mit", staedte[s], "Einwohner")
```

Python – Datentypen (5)
Zeichenketten

- Zeichenkette (`string`) in Python: Folge von Zeichen
- eine Python-Zeichenkette ist unveränderlich nachdem sie generiert wurde

Zeichenketten-Darstellungen

```
z1 = 'eine Zeichenkette mit Gänsefüßchen (") aber ohne Apostroph\n'
z2 = "eine Zeichenkette ohne Gänsefüßchen (') aber mit Apostroph"
z3 = """mit eingebetteten
      Zeilenenden"""
z4 = r"\section{Eine Überschrift in \LaTeX}\n"
```

einige String-Methoden/Funktionen

Funktion/Methode	Bedeutung
string`.capitalize()`	liefert Zeichenkette mit großem Anfangsbuchstab
string`.upper()`	liefert Zeichenkette nur aus Großbuchstaben
`len(`*string*`)`	liefert Länge von *string*
string`.find(`*string1*`)`	findet Position von *string1* in *string*
string`.replace(`*string1*`,` *string2*`)`	ersetzt *string1* in *string* durch *string2*
string`.split(`*trenner*`)`	trennt *string* bei *trenner* auf; liefert Liste
string`.isdigit()`	liefert `True`, falls *string* nur Ziffern enthält

Python – Schlüsselwörter


```
and       def       finally   in        or        while
as        del       for       is        pass      with
assert    elif      from      lambda    raise     yield
break     else      global    None      return
class     except    if        nonlocal  True
continue  False     import    not       try
```

Schlüsselwörter sind Bestandteile der Sprache Python $\Longrightarrow$ keine solche eigenen Bezeichner

Python – Kontrollstrukturen (1)

if-Anweisung

```
if a <= 2:
    print("Der Wert ist kleiner 2.")
```

if-Anweisung mit einer Alternative

```
if a <= 2:
    print("Der Wert ist kleiner gleich 2.")
else:
    print("Der Wert ist größer 2.")
```

if-Anweisung mit mehreren Alternativen

```
if auswahl in ["Ja", "j"]:
    print("Ja")
elif auswahl in ["Nein", "n"]:
    print("Nein")
else:
    print("keine richtige Antwort")
```

Python – Kontrollstrukturen (2)

for-Anweisung

```
for i in ["Günter Partosch", "Emil Mayer"]:
    print('Der Teilnehmer heißt:', i)
```

for-Anweisung mit range()

```
summe = 0
for i in range(1, 21):
    summe = summe + i
```

while-Anweisung

```
summe = 0; i = 1
while i <= 20: summe = summe + i; i += 1
```

try-except-Konstrukt

```
try:
    # Initialisierung für Programm-Parameter und Variablen einlesen
    from zaehlen2_ini import *
except ImportError:
    # lokal Programm-Parameter und Variablen initialisieren
    print("---Warnung: zaehlen2_ini.py nicht gefunden")
```

Python – Module

- erweitern den Funktionsumfang von Python
- oder nehmen Einstellungen vor
- Übersicht und Beschreibungen in [van Rossum et al. 2018, `library.pdf`]

Import von Modulen

`import` *quelle*
`import` *quelle* `as` *name*
`from` *quelle* `import *` | *name(n)*

kleine Auswahl

`csv`	CSV-Daten verarbeiten	`randassign`	Zufallszahlen
`math`	mathematische Funktionen	`random`	Zufallszahlen
`matplotlib`	2D-Grafiken erstellen	`re`	reguläre Ausdrücke
`numpy`	numerische Mathematik	`subprocess`	Sub-Prozesse aufrufen
`pickle`	Python-Objekte serialisieren	`sympy`	symbolische Mathematik
`pyx`	PS/PDF-Grafiken erstellen	`time`	Zeitmessungen

PythonTEX

PythonTEX

- PythonTEX [Poore 2017];
  einführende Texte in [Poore 2015, Mertz et al. 2013, Gosling 2016]
  aktuelle Version: 0.16
- Python-Code in ein LATEX-Dokument einbetten

PythonTEX

- PythonTEX [Poore 2017];
  einführende Texte in [Poore 2015, Mertz et al. 2013, Gosling 2016]
  aktuelle Version: 0.16
- Python-Code in ein LATEX-Dokument einbetten
  - an Python übergeben, übersetzen und ausführen

PythonTEX

- PythonTEX [Poore 2017];
  einführende Texte in [Poore 2015, Mertz et al. 2013, Gosling 2016]
  aktuelle Version: 0.16
- Python-Code in ein LATEX-Dokument einbetten
  - an Python übergeben, übersetzen und ausführen
  - betreffende Ausgabe im LATEX-Dokument einfügbar

PythonTEX

- PythonTEX [Poore 2017];
  einführende Texte in [Poore 2015, Mertz et al. 2013, Gosling 2016]
  aktuelle Version: 0.16
- Python-Code in ein LATEX-Dokument einbetten
  - an Python übergeben, übersetzen und ausführen
  - betreffende Ausgabe im LATEX-Dokument einfügbar
  - Python-Code wird nur dann erneut übersetzt, wenn er geändert wurde

PythonTEX

- PythonTEX [Poore 2017];
  einführende Texte in [Poore 2015, Mertz et al. 2013, Gosling 2016]
  aktuelle Version: 0.16
- Python-Code in ein LATEX-Dokument einbetten
  - an Python übergeben, übersetzen und ausführen
  - betreffende Ausgabe im LATEX-Dokument einfügbar
  - Python-Code wird nur dann erneut übersetzt, wenn er geändert wurde
  - Python-Code ggf. in Sessions aufteilbar, die parallel abgearbeitet werden

PythonTEX

- PythonTEX [Poore 2017];
  einführende Texte in [Poore 2015, Mertz et al. 2013, Gosling 2016]
  aktuelle Version: 0.16
- Python-Code in ein LATEX-Dokument einbetten
  - an Python übergeben, übersetzen und ausführen
  - betreffende Ausgabe im LATEX-Dokument einfügbar
  - Python-Code wird nur dann erneut übersetzt, wenn er geändert wurde
  - Python-Code ggf. in Sessions aufteilbar, die parallel abgearbeitet werden
- Python-Code kann aufbereitet im LATEX-Dokument ausgegeben werden (prettyprinting)

PythonTEX

- PythonTEX [Poore 2017];
  einführende Texte in [Poore 2015, Mertz et al. 2013, Gosling 2016]
  aktuelle Version: 0.16
- Python-Code in ein LATEX-Dokument einbetten
  - an Python übergeben, übersetzen und ausführen
  - betreffende Ausgabe im LATEX-Dokument einfügbar
  - Python-Code wird nur dann erneut übersetzt, wenn er geändert wurde
  - Python-Code ggf. in Sessions aufteilbar, die parallel abgearbeitet werden
- Python-Code kann aufbereitet im LATEX-Dokument ausgegeben werden (prettyprinting)
- PythonTEX unterstützt derzeit direkt die Module `pylab` (aus `matplotlib` und `numpy`) und `sympy`

PythonTEX

- PythonTEX [Poore 2017];
  einführende Texte in [Poore 2015, Mertz et al. 2013, Gosling 2016]
  aktuelle Version: 0.16
- Python-Code in ein LATEX-Dokument einbetten
  - an Python übergeben, übersetzen und ausführen
  - betreffende Ausgabe im LATEX-Dokument einfügbar
  - Python-Code wird nur dann erneut übersetzt, wenn er geändert wurde
  - Python-Code ggf. in Sessions aufteilbar, die parallel abgearbeitet werden
- Python-Code kann aufbereitet im LATEX-Dokument ausgegeben werden (prettyprinting)
- PythonTEX unterstützt derzeit direkt die Module `pylab` (aus `matplotlib` und `numpy`) und `sympy`
- Unterstützung für Ruby, Julia und Octave schon eingebaut; lässt sich aber auch auf andere Sprachen ausdehnen

PythonTEX

- PythonTEX [Poore 2017];
  einführende Texte in [Poore 2015, Mertz et al. 2013, Gosling 2016]
  aktuelle Version: 0.16
- Python-Code in ein LATEX-Dokument einbetten
  - an Python übergeben, übersetzen und ausführen
  - betreffende Ausgabe im LATEX-Dokument einfügbar
  - Python-Code wird nur dann erneut übersetzt, wenn er geändert wurde
  - Python-Code ggf. in Sessions aufteilbar, die parallel abgearbeitet werden
- Python-Code kann aufbereitet im LATEX-Dokument ausgegeben werden (prettyprinting)
- PythonTEX unterstützt derzeit direkt die Module `pylab` (aus `matplotlib` und `numpy`) und `sympy`
- Unterstützung für Ruby, Julia und Octave schon eingebaut; lässt sich aber auch auf andere Sprachen ausdehnen

PythonTEX – Installation

Anforderungen/Voraussetzungen

- aktuelle TEX/LATEX-Installation (MiKTEX/TEXLive)
- oder Download von https://github.com/gpoore/pythontex
- LATEX-Pakete: `fancyvrb`, `fvextra`, `etoolbox`, `xstring`, `pgfopts`, `newfloat`, `currfile`, `color/xcolor`
- ggf. `graphicx` – `mdframed` oder `tcolorbox` oder `framed`
- aktuelle Python-Installation (aktuell 3.6.5)
- ggf. Python-Module: `pygments` – `numpy`, `scipy`, `matplotlib`, `sympy`
- Editoren/Entwicklungsumgebungen für LATEX und Python

Dateien nach der Installation

- `pythontex.sty` in `C:/texlive/2017/texmf-dist/tex/latex/pythontex`
- `syncpdb.py` in `C:/texlive/2017/texmf-dist/doc/latex/pythontex`
- sonst bei mir alle in `C:/texlive/2017/texmf-dist/scripts/pythontex`
- `pythontex.py`
- `pythontex_engines.py`
- `pythontex_utils.py`
- `depythontex.py`
- `pythontex_install.py`

PythonTEX – Optionen des Pakets pythontex

Optionen des LaTeX-Pakets pythontex (Auswahl) [Poore 2017, 4.1]

debug	Debugging ermöglichen
depythontex= true | false	zusätzliche Datei für den Aufruf von depythontex erzeugen
hashdependencies= true | false	auf geänderte externe Dateien überprüfen
prettyprinter= pygments | fancyvrb	Paket/Programm für Prettyprinting
prettyprintinline= true | false	Inline-Prettyprinting erlauben
pyginline= true | false	eingebettete Pygments-Ausgabe erlauben
pygments= true | false	Pygments-Ausgabe erlauben
rerun= never | modified | errors | warnings | always	bei bestimmten Bedingungen erneut übersetzen
runall= true | false	auch wenn Python-Anteile nicht geändert wurden, erneut übersetzen
upquote= true | false	aufrechte Anführungszeichen

Üblicherweise werden die Optionen als Parameter an das Programm pythontex weiter geleitet.

PythonTEX – Parameter des Programms pythontex

Aufruf des Programms pythontex

pythontex [*parameter*] *datei*[.pytxcode]

Optionale Parameter des Programms pythontex (Auswahl) [Poore 2017, 3.2]

`--error-exit-code true | false`	liefert auch bei Fehler den Fehlercode 1
`--runall [true | false]`	wie die Paket-Option `runall`
`--rerun never | modified | errors | warnings | always`	wie die Paket-Option `rerun`
`--hashdependencies [true | false]`	wie die Paket-Option `hashdependencies`
`--jobs` n	erlaubt n gleichzeitige Jobs (Voreinstellung: `cpu_count()`)
`--verbose`	»geschwätzige« Ausgabe

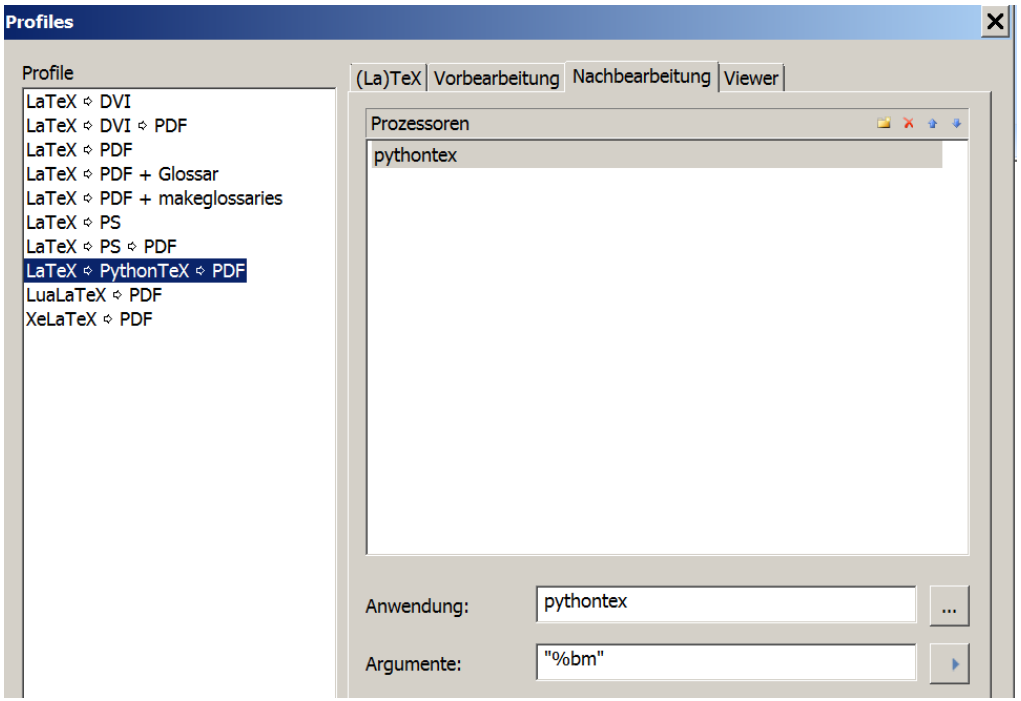
PythonTEX – Workflow (1)

im einfachsten Fall auf der Kommandozeile

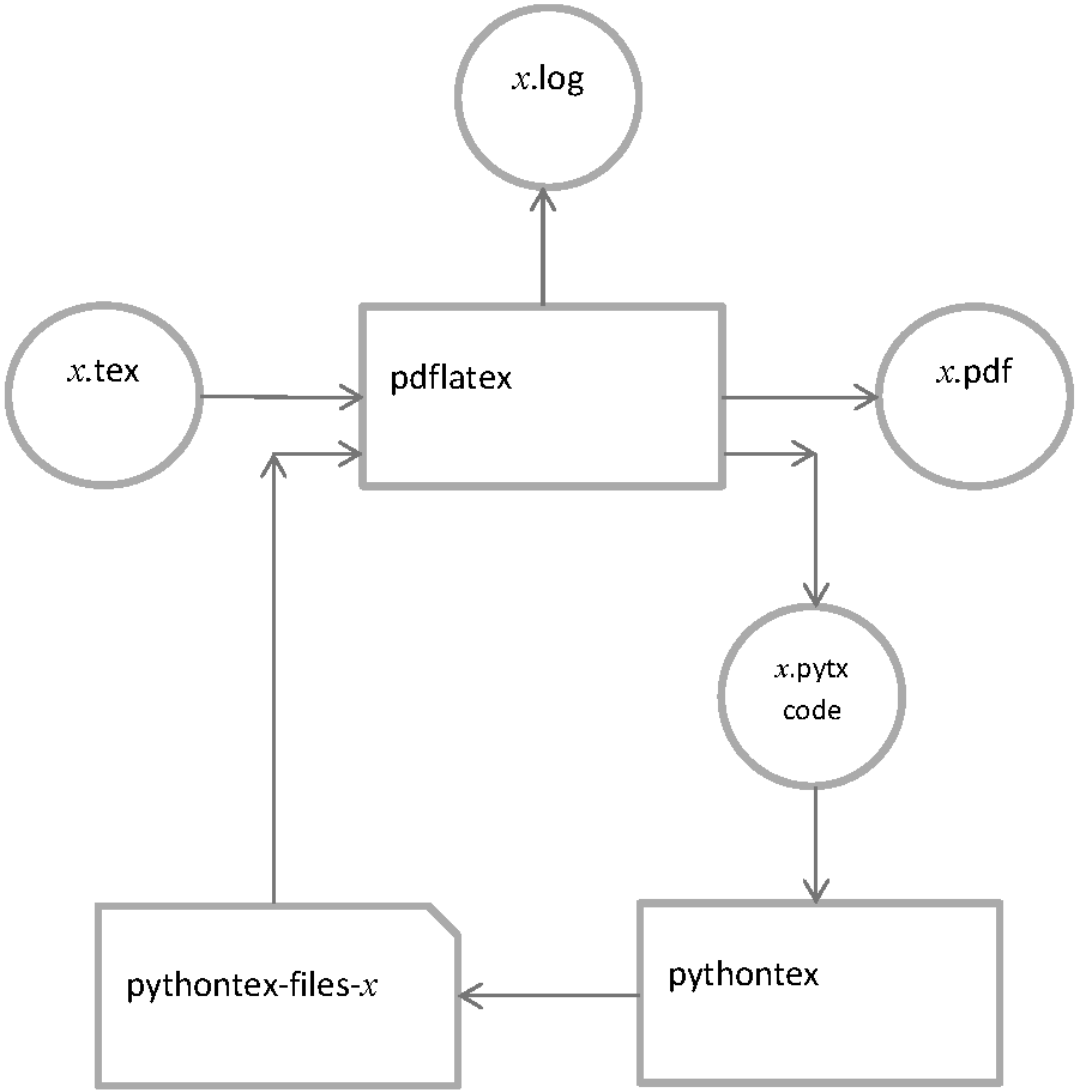
PythonTEX – Workflow (1)

pdflatex *datei*[.tex]
pythontex [*parameter*] *datei*[.pytxcode]
pdflatex *datei*[.tex]
...

beim Einsatz einer LATEX-Benutzeroberfläche, beispielsweise im TEXnicCenter



PythonTEX – Workflow (2)



PythonTEX – Anweisungen

\py und verwandte Anweisungen und Umgebungen (1)

\py und verwandte Anweisungen und Umgebungen (1) [Poore 2017]

Python		Python + Modul pylab		
Anweisung	Umgebung	Anweisung	Umgebung	
\py{*ausdruck*}		\pylab{*ausdruck*}		W
\pyc{*code*}	pycode	\pylabc{*code*}	pylabcode	C + A
\pyb{*code*}	pyblock	\pylabb{*code*}	pylabblock	C + P
\pyv{*code*}	pyverbatim	\pylabv{*code*}	pylabverbatim	P

\py und verwandte Anweisungen und Umgebungen (2) [Poore 2017]

Python + Modul sympy		
Anweisung	Umgebung	
\sympylab{*ausdruck*}		W
\sympyc{*code*}	sympycode	C + A
\sympyb{*code*}	sympyblock	C + P
\sympyv{*code*}	sympyverbatim	P

W: Ausgabe des Werts von *ausdruck*
C + A: Ausführung von *code* und lediglich Ausgabe auf die Standardausgabe
C + P: Ausführung von *code* und Prettyprinting mittels \printpythontex
P: nur Prettyprinting

Anmerkungen und Ergänzungen

- alle Anweisungen und Umgebungen ggf. auch mit dem optionalen Parameter `[`*session*`]`, z. B. `\py[calc]{x**2}`
- für die Simulation einer interaktiven Sitzung im Editor gibt es noch die
  - Anweisungen
    `\pycon`, `\pyconc`, `\pyconv`
    `\pylabcon`, `\pylabconc`, `\pylabconv`
    `\sympycon`, `\sympyconc`, `\sympyconv`
  - und die Umgebungen
    `pyconsole`, `pyconcode`, `pyconverbatim`
    `pylabconsole`, `pylabconcode`, `pylabconverbatim`
    `sympyconsole`, `sympyconcode`, `sympyconverbatim`
- `\py{`*ausdruck*`}` vs. `\pyc{`*code*`}`:
  - `\py{2**10}` → 1024
  - `\pyc{print(2**10)}` → 1024

PythonTEX – Anweisungen

eigene LATEX-Anweisungen, die Python nutzen

Vereinbarungen — nach [Poore 2017, 6.1]

```
% Potenz ausgeben
  \newcommand{\hoch}[2]{\py{#1**#2}}
% Zeichenkette umkehren
  \newcommand{\umgekehrt}[1]{\py{"#1"[::-1]}}
% 1. und letztes Zeichen einer Zeichenkette vertauschen
  \newcommand{\swapfirstlast}[1]{%
    \pyc{s = "#1"} \py{s[-1] + s[1:-1] + s[0]}}
```

Aufrufe

```
3 hoch 20: \hoch{3}{20}\\
Zeichenkette umkehren: \umgekehrt{Das ist ein Text!}\\
tausche erstes und letztes Zeichen: \swapfirstlast{0123456789abcdefghijk}
```

Quelle `pytex35.tex` zeigen → Ergebnis

PythonTEX – Beispiele (1)

Präambel für die folgenden kleinen Beispiele

```
\documentclass[parskip=half,fontsize=11,paper=a4]{scrartcl}
\usepackage{pythontex}
\usepackage[T1]{fontenc}
\usepackage[utf8]{inputenc}
\usepackage[intlimits]{amsmath}
```

Beispiel pytex1.tex [Mertz et al. 2013]

```
% aus Mertz, Slough 2013 - A Gentle Introduction to PythonTeX
\section*{PythonTeX: py}
% eingebetteter Python-Aufruf
Wissen Sie, dass $2^{65} = \py{2**65}$?
```

Quelle pytex1.tex zeigen $\Longrightarrow$ Ergebnis

PythonTEX – Beispiele (2)

Beispiel pytex4.tex

```
\section*{PythonTeX: pycode/pyblock-Umgebung, printpythontex, ...}

\begin{pyblock}
# Aufbau einer tabular-Umgebung in einer Schleife
# Python-Code wird ausgegeben
anfang, ende = 1, 30
print(r"\begin{tabular}{r|r}")
print(r"$m$ & $2^m$ \\ \hline")
for m in range(anfang, ende + 1):
    print(m, "&", 2**m, r"\\")
print(r"\end{tabular}")
\end{pyblock}

\printpythontex % Ausgabe des Blocks
```

Quelle pytex4.tex zeigen $\Longrightarrow$ Ergebnis

PythonTEX – Beispiele (3)

Beispiel pytex11.tex nach [Mertz et al. 2013]

```
% aus Mertz, Slough 2013 - A Gentle Introduction to PythonTeX
\section*{PythonTeX: pythontexcustomcode, sympy, def, Schleife, Primzahl}

\begin{pythontexcustomcode}{py}
from sympy import prime                 # symb. Mathematik, hier Primzahlen

def Primzahlen(n):                      # Definition einer Python-Funktion
    for i in range(1, n):               # Annahme n >= 3
        print(prime(i), " ")            # nächste Primzahl
    print("und ", prime(n))             # letzte Primzahl
\end{pythontexcustomcode}

Die ersten 1000 Primzahlen sind \pyc{Primzahlen(1000)}.
```

Quelle pytex11.tex zeigen $\Longrightarrow$ Ergebnis

PythonTEX – Beispiele (4)

Beispiel pytex12.tex [Mertz et al. 2013]

```
% aus Mertz, Slough 2013 - A Gentle Introduction to PythonTeX
\section*{PythonTeX: pyblock, printpythontex, sympy, Binome, ...}

\begin{sympyblock}
#from sympy import *            # symbolische Mathematik
var("a, b")                     # sympy-Variablen
Binome = []                     # Liste für Binomi-Ausdrücke vorbesetzt

for m in range(1, 10):
    Binome.append((a + b)**m) # Binomi-Ausdrücke erzeugen

print(r"\begin{align*}")        # Tabelle mit align*-Umgebung
for expr in Binome:             # SChleife über alle Binome
    print(latex(expr), "&=", latex(expand(expr)), r"\\")
print(r"\end{align*}")
\end{sympyblock}

\printpythontex
```

Quelle pytex12.tex zeigen $\Longrightarrow$ Ergebnis

PythonTEX – Beispiele (5)

Beispiel pytex24.tex

```
\section*{PythonTeX: pyblock, sympy, Gleichungssystem}

\begin{pyblock}
import sympy as sy                          # symbolische Mathematik
h, z, e = sy.symbols('h z e')               # sympy-Variablen initiieren

gls = [                                     # Gleichungssystem formulieren
sy.Eq(z + h + e, 18),
sy.Eq(h - 6    , 2 * z),
sy.Eq(e - 6    , 3 * z),
]

ergebnis = sy.solve(gls)                    # Gleichungssystem lösen
for f in ergebnis:                          # Lösung ausgeben
    print(f, ":", ergebnis[f], r"\\")
\end{pyblock}
\printpythontex                             % letzten pyblock ausgeben
```

Quelle pytex24.tex zeigen $\Longrightarrow$ Ergebnis

PythonTeX – Beispiele (6)

Beispiel pytex43.tex [Poore 2013]

```
% Poore 2013 - PythonTeX: Reproducible Documents with PythonTeX
\section*{PythonTeX: sympy, sympyblock, printpythontex, Ableitung, ...}

\begin{sympyblock}
#from sympy import *
x = symbols('x')                                  # sympy-Variable

print(r'\begin{align*}')
for funk in [sin(x), sinh(x), csc(x)]:            # zu untersuchende Funktionen
    links  = Derivative(funk, x)                  # Ableitung, formal
    rechts = Derivative(funk, x).doit()           # Ableitung ausführen
    gl     = latex(links) + '&=' + latex(rechts) + r'\\'
    print(gl.replace('d', r'\mathrm{d} '))  # d austauschen
print(r'\end{align*}')
\end{sympyblock}

\printpythontex
```

Quelle pytex43.tex zeigen $\Longrightarrow$ Ergebnis

PythonTEX – Beispiele (7)

Fallbeispiel I

gegeben

Liste mathematischer Funktionen:
$\operatorname{acos}(x)$, $\operatorname{acosh}(x)$, $\operatorname{acot}(x)$, $\operatorname{acoth}(x)$, $\operatorname{asin}(x)$, $\operatorname{asinh}(x)$, $\operatorname{atan}(x)$, $\operatorname{atanh}(x)$, $\cos^2(x)$, $\cos(x)$, $\cosh(x)$, $\cot^2(x)$, $\cot(x)$, $\coth(x)$, $\operatorname{erf}(x)$, $\operatorname{erfc}(x)$, e^x, $\Gamma(x)$, $\log(x)$, $\sin^2(x)$, $\sin(x)$, $\sinh(x)$, $\sqrt{x}$, $\tan^2(x)$, $\tan(x)$, $\tanh(x)$, $\csc(x)$

gesucht

Liste mit zugehörigen Ableitungen und Integrale

Werkzeuge

Import des Python-Moduls `sympy`:

```
from sympy import *
```

mit den Funktionen/Methoden:
`latex`, `eval`, `Derivative`, `Integral`, `doit`

Quelle `pytex23.tex` zeigen $\Longrightarrow$ Ergebnis

PythonTEX – Beispiele (8)
Fallbeispiel II (a)

gegeben

ein Polynom »beliebigen« Grades:

$$\sum_{i=0}^{n} a_i x^i$$

gesucht (Kurvendiskussion)

- alle reellwertigen Nullstellen mit beliebiger, vorgebbarer Genauigkeit
- Ableitungen ($i = 1, \ldots, n$)
- alle reellwertigen Nullstellen der Ableitungen
- falls vorhanden: alle Extremstellen (Minima, Maxima)
- falls vorhanden: alle Wendestellen
- falls vorhanden: Symmetriepunkte bzw. Symmetrieachsen
- Achsendurchgang für $x = 0$
- Graphen der Funktion und aller ihrer Ableitungen
- vollwertiges, vollständiges LaTeX-Dokument

PythonTEX – Beispiele (9)
Fallbeispiel II (b)

Benötigte Python-Module

- `sympy`: symbolische Mathematik [Meurer et al. 2016]
- `pyx`: Grafik [Lehmann et al. 2015]

Python-Konstrukte

- Listen, auch mehrdimensional
- `if`-Abfrage
- `for`-Schleife
- `if-except`-Konstrukt
- `len(`*liste*`)`
- `range(`*parameter*`)`
- *liste*`.append`
- `round(`*ausdruck*`)`
- `str(`*ausdruck*`)`
- `eval(`*string*`)`

PythonTEX – Beispiele (10)
Fallbeispiel II (c)

sympy-Methoden [Meurer et al. 2016]

- `symbols(`*string*`)`
- `sympify(`*string*`)`
- `degree(`*sympy-ausdruck*`)`
- `latex(`*sympy-ausdruck*`)`
- `nroots(`*sympy-ausdruck*`)`
- `solve(`*sympy-ausdruck*`)`
- `diff(`*sympy-ausdruck*`)`
- *sympy-objekt*`.subs(`*parameter*`)`

PythonTEX-Anweisungen und -Umgebungen

- `\printpythontex`
- `\setpythontexcontext`
- `pyblock`-Umgebung
- `pycode`-Umgebung

Quelle `pytex31.tex` zeigen $\Longrightarrow$ Ergebnis

Literatur

Literatur/Links (1)

Python

van Rossum, Guido; Python development team: *Functional Programming HOWTO – Release 3.6.5*; 2018; als `howto-functional.pdf` in https://docs.python.org/3/archives/python-3.6.5-docs-pdf-a4.zip; hrsg. von Python Software Foundation; zuletzt besucht am 2018-03-30

van Rossum, Guido; Python development team: *Python Frequently Asked Questions – Release 3.6.5*; 2018; als `faq.pdf` in https://docs.python.org/3/archives/python-3.6.5-docs-pdf-a4.zip; hrsg. von Python Software Foundation; zuletzt besucht am 2018-03-30

van Rossum, Guido; Python development team: *Python Tutorial – Release 3.6.5*; 2018; als `tutorial.pdf` in https://docs.python.org/3/archives/python-3.6.5-docs-pdf-a4.zip; hrsg. von Python Software Foundation; zuletzt besucht am 2018-03-30

van Rossum, Guido; Python development team: *Regular Expression HOWTO – Release 3.6.5*; 2018; als `howto-regex.pdf` in https://docs.python.org/3/archives/python-3.6.5-docs-pdf-a4.zip; hrsg. von Python Software Foundation; zuletzt besucht am 2018-03-30

van Rossum, Guido; Python development team: *Sorting HOW TO – Release 3.6.5*; 2018; als `howto-sorting.pdf` in https://docs.python.org/3/archives/python-3.6.5-docs-pdf-a4.zip; hrsg. von Python Software Foundation; zuletzt besucht am 2018-03-30

Literatur/Links (2)

van Rossum, Guido; Python development team: *The Python Language Reference – Release 3.6.5*; 2018; als `reference.pdf` in https://docs.python.org/3/archives/python-3.6.5-docs-pdf-a4.zip; hrsg. von Python Software Foundation; zuletzt besucht am 2018-03-30

van Rossum, Guido; Python development team: *The Python Library Reference – Release 3.6.5*; 2018; als `library.pdf` in https://docs.python.org/3/archives/python-3.6.5-docs-pdf-a4.zip; hrsg. von Python Software Foundation; zuletzt besucht am 2018-03-30

PythonTeX

Abbasi, Nasser M.: *A very simple introduction to using Python in Latex*;; 2015;; http://www.12000.org/my_notes/python_in_latex/index.pdf;; zuletzt besucht am 2018-03-21

Dautermann, Wolfgang: *Programmierung mit LaTeX – . . . und anderen Programmiersprachen*; 2014; hrsg. von FH Johanneum; http://wolfgang.dautermann.at/vortraege/Linuxday-2014-Programmierung-mit-Latex.pdf; zuletzt besucht am 2018-03-21

Giacomelli, Roberto; Pignalberi, Gianluca: *Typesetting and highlighting Unicode source code with LaTeX – a package comparison*; 2014; in *Ars TeXnica* (18), S. 39–44; http://www.guitex.org/home/images/ArsTeXnica/AT018/UnicodeSC.pdf; zuletzt besucht am 2018-03-21

Literatur/Links (3)


Gosling, P. E.: *PythonTeX Quickstart*; 2016; http://mirrors.ctan.org/macros/latex/contrib/pythontex/pythontex_quickstart.pdf; zuletzt besucht am 2018-03-21


Hilpisch, Yves: *Wissenschaftliches Publizieren mit Python*; 2013;; http://www.hilpisch.com/CAE_Pycon_DE_Scientific_Publishing.pdf; zuletzt besucht am 2018-03-21


Mertz, Andrew; Slough, William: *A Gentle Introduction to PythonTeX*; 2013; hrsg. von Eastern Illinois University; https://tug.org/tug2013/slides/Mertz-A_Gentle_Introduction_to_PythonTeX.pdf; zuletzt besucht am 2018-03-21


Nettles, Bill: *nucleardata – provides nuclide information*; 2016; http://mirrors.ctan.org/macros/latex/contrib/nucleardata/nucleardata.pdf; zuletzt besucht am 2018-03-21


Nettles, Bill; Poore, Geoffrey M.: *Using Python and pdfLaTeX to Generate Customized Physics Problems*; 2016; hrsg. von Union University; https://www.aapt.org/docdirectory/meetingpresentations/WM16/AAPTpaper_CI07_Nettles.pdf; zuletzt besucht am 2018-03-21


Poore, Geoffrey M.: *Reproducible Documents with PythonTeX*; 2013; http://conference.scipy.org/proceedings/scipy2013/pdfs/poore.pdf; zuletzt besucht am 2018-03-21

Literatur/Links (4)

Poore, Geoffrey M.: *PythonTeX – Reproducible documents with LaTeX, Python, and more*; 2015; in *Computational Science & Discovery* 8 (1), S. 1–20; http://iopscience.iop.org/article/10.1088/1749-4699/8/1/014010/pdf; zuletzt besucht am 2018-03-21

Poore, Geoffrey M.: *PythonTeX – Fast Access to Python from within LaTeX*; 2015; http://conference.scipy.org/proceedings/scipy2012/pdfs/geoffrey_poore.pdf; zuletzt besucht am 2018-03-21

Poore, Geoffrey M.: *PythonTeX Gallery*; 2017; http://tug.ctan.org/macros/latex/contrib/pythontex/pythontex_gallery.pdf; zuletzt besucht am 2018-03-21

Poore, Geoffrey M.: *The PythonTeX package – v0.16*; 2017; http://mirrors.ctan.org/macros/latex/contrib/pythontex/pythontex.pdf; zuletzt besucht am 2018-03-21

Benutzte Python-Module

Lehmann, Jörg; Schindler, Michael; Wobst, André: *PyX Manual – Release 0.14.1*; 2015; http://pyx.sourceforge.net/manual.pdf; zuletzt besucht am 2018-04-05

Meurer, Aaron; Čertík, Ondřej; Kumar, Amit; Moore, Jason; Singh, Sartaj; Gupta, Harsh: *SymPy Tutorial*; 2016; http://www.sympy.org/scipy-2016-tutorial/intro.pdf; zuletzt besucht am 2018-04-05