PythonTEX
Zusammenspiel von Python und LATEX

Günter Partosch

Justus-Liebig-Universität Gießen, Hochschulrechenzentrum (HRZ)

Version 1.2.3, 2. September 2019

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 3.7.4 (8.7.2019)
- ein **Entwicklungsziel**: gut lesbarer, knapper Programmierstil $\Longrightarrow$ Unterrichtssprache an Schulen und Hochschulen
  - untergeordnete Blöcke werden eingerückt $\rightarrow$ nicht geklammert
  - ganz zeilenorientiert (mit wenigen Ausnahmen) $\rightarrow$ 1 Anweisung pro 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. 2019, `reference.pdf`]; Erweiterungen über zahlreiche Module [van Rossum et al. 2019, `library.pdf`]
- **Tutorium** [van Rossum et al. 2019, `tutorial.pdf`]
- **FAQs** [van Rossum et al. 2019, `faq.pdf`]
- **HowTo**s 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              # bedeutet: a = 1; b = 20
a = b = 5                 # bedeutet: 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]  # einzelne Elemente durch Aufzählung
werte.append("abc")    # neues Element am Ende
werte.insert(1, 3.4)   # neues Element an der Position 1 (2. Element)
```

Tupel

```
werte = (2, "a", 1.3)  # einzelne Elemente durch Aufzählung
#werte.append("abc")   # Anhängen nicht möglich
#werte.insert(1, 3.4)  # Einfügen nicht möglich
```

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]            # 2. bis 5. Element
```

Python – Datentypen (4)

zusammengesetzte Typen (2)

und dann gibt es noch Mengen (set)

```
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 (dictionary)

```
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 nach der Vereinbarung

Zeichenketten-Darstellungen

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

einige String-Methoden/Funktionen

Funktion/Methode	Bedeutung
string`.capitalize()`	liefert Zeichenkette mit gr. Anfangsbuchstaben
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 t in ["Günter Partosch", "Emil Mayer"]:
    print('Der Teilnehmer heißt:', t)
```

for-Anweisung mit range()

```
summe = 0
for i in range(1, 21):  # \sum_i=1^20 i
    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. 2019, `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

Python

PythonTEX
- Installation
- Parameter, Optionen
- Workflow
- Anweisungen
- Beispiele

Literatur

PythonTEX

- PythonTEX [Poore 2019];
  einführende Texte in [Poore 2015, Mertz et al. 2013, Gosling 2016]
  aktuelle Version: 0.17
- Python-Code in ein LATEX-Dokument eingebettet

PythonTEX

- PythonTEX [Poore 2019];
  einführende Texte in [Poore 2015, Mertz et al. 2013, Gosling 2016]
  aktuelle Version: 0.17
- Python-Code in ein LATEX-Dokument eingebettet
  - an Python übergeben → übersetzt → ausgeführt

PythonTEX

- PythonTEX [Poore 2019];
  einführende Texte in [Poore 2015, Mertz et al. 2013, Gosling 2016]
  aktuelle Version: 0.17
- Python-Code in ein LATEX-Dokument eingebettet
  - an Python übergeben → übersetzt → ausgeführt
  - betreffende Ausgabe ins LATEX-Dokument einfügbar

PythonTEX

- PythonTEX [Poore 2019];
  einführende Texte in [Poore 2015, Mertz et al. 2013, Gosling 2016]
  aktuelle Version: 0.17
- Python-Code in ein LATEX-Dokument eingebettet
  - an Python übergeben → übersetzt → ausgeführt
  - betreffende Ausgabe ins LATEX-Dokument einfügbar
  - Python-Code wird nur dann erneut übersetzt, wenn er geändert wurde

PythonTEX

- PythonTEX [Poore 2019];
  einführende Texte in [Poore 2015, Mertz et al. 2013, Gosling 2016]
  aktuelle Version: 0.17
- Python-Code in ein LATEX-Dokument eingebettet
  - an Python übergeben → übersetzt → ausgeführt
  - betreffende Ausgabe ins 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 2019];
  einführende Texte in [Poore 2015, Mertz et al. 2013, Gosling 2016]
  aktuelle Version: 0.17
- Python-Code in ein LATEX-Dokument eingebettet
  - an Python übergeben → übersetzt → ausgeführt
  - betreffende Ausgabe ins 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 2019];
  einführende Texte in [Poore 2015, Mertz et al. 2013, Gosling 2016]
  aktuelle Version: 0.17
- Python-Code in ein LATEX-Dokument eingebettet
  - an Python übergeben → übersetzt → ausgeführt
  - betreffende Ausgabe ins 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 2019];
  einführende Texte in [Poore 2015, Mertz et al. 2013, Gosling 2016]
  aktuelle Version: 0.17
- Python-Code in ein LATEX-Dokument eingebettet
  - an Python übergeben → übersetzt → ausgeführt
  - betreffende Ausgabe ins 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 2019];
  einführende Texte in [Poore 2015, Mertz et al. 2013, Gosling 2016]
  aktuelle Version: 0.17
- Python-Code in ein LATEX-Dokument eingebettet
  - an Python übergeben → übersetzt → ausgeführt
  - betreffende Ausgabe ins 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 3.7.4)
- ggf. Python-Module: `pygments` – `numpy`, `scipy`, `matplotlib`, `sympy`
- Editoren/Entwicklungsumgebungen für LATEX und Python

Dateien nach der Installation

- `pythontex.sty` in `C:/texlive/2019/texmf-dist/tex/latex/pythontex`
- `syncpdb.py` in `C:/texlive/2019/texmf-dist/doc/latex/pythontex`
- bei mir sonst alle in `C:/texlive/2019/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 2019, 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 2019, 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

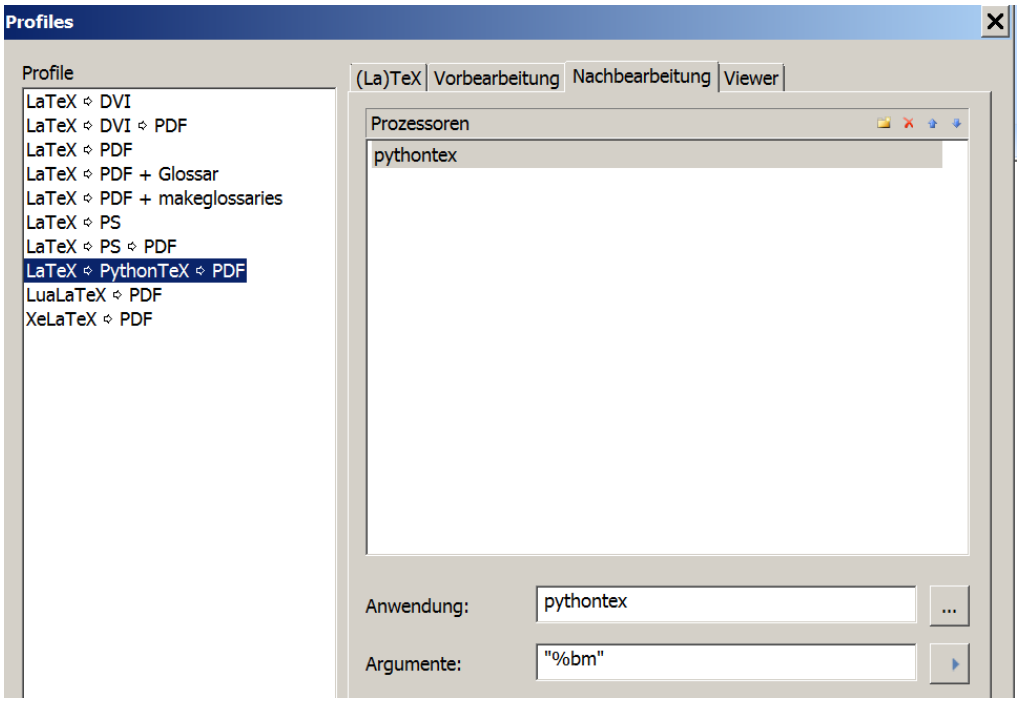
PythonTEX – Workflow (1)

im einfachsten Fall auf der Kommandozeile

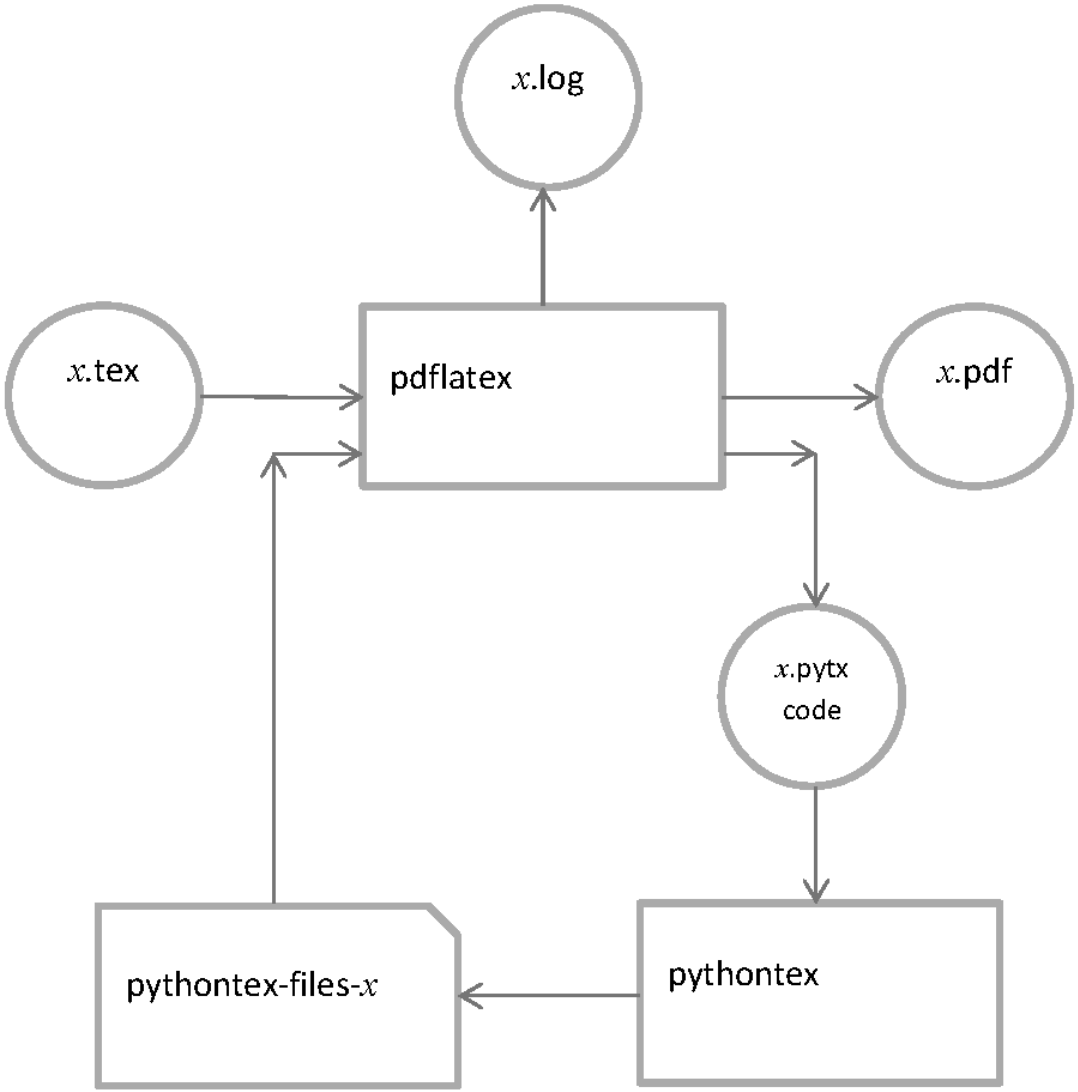
PythonTEX – Workflow (1)

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

beim Einsatz einer LATEX-Benutzeroberfläche (beispielsweise TEXnicCenter)



PythonTEX – Workflow (2)



PythonTEX – Anweisungen

\py und verwandte Anweisungen und Umgebungen (1)

\py und verwandte Anweisungen und Umgebungen (1) [Poore 2019]

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 2019]

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

PythonTEX – Anweisungen

\py und verwandte Anweisungen und Umgebungen (2)

Anmerkungen und Ergänzungen

- alle Anweisungen und Umgebungen ggf. auch mit einem 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 2019, 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. 2019]

Python-Konstrukte

- Listen, auch mehrdimensional
- `if`-Abfrage
- `for`-Schleife
- `try-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 3.7.4*; 2019; als `howto-functional.pdf` in https://docs.python.org/3/archives/python-3.7.4-docs-pdf-a4.zip; hrsg. von Python Software Foundation; zuletzt besucht am 2019-10-02

van Rossum, Guido; Python development team: *Python Frequently Asked Questions – Release 3.7.4*; 2019; als `faq.pdf` in https://docs.python.org/3/archives/python-3.7.4-docs-pdf-a4.zip; hrsg. von Python Software Foundation; zuletzt besucht am 2019-10-02

van Rossum, Guido; Python development team: *Python Tutorial – Release 3.7.4*; 2019; als `tutorial.pdf` in https://docs.python.org/3/archives/python-3.7.4-docs-pdf-a4.zip; hrsg. von Python Software Foundation; zuletzt besucht am 2019-10-02

van Rossum, Guido; Python development team: *Regular Expression HOWTO – Release 3.7.4*; 2019; als `howto-regex.pdf` in https://docs.python.org/3/archives/python-3.7.4-docs-pdf-a4.zip; hrsg. von Python Software Foundation; zuletzt besucht am 2019-10-02

van Rossum, Guido; Python development team: *Sorting HOW TO – Release 3.7.4*; 2019; als `howto-sorting.pdf` in https://docs.python.org/3/archives/python-3.7.4-docs-pdf-a4.zip; hrsg. von Python Software Foundation; zuletzt besucht am 2019-10-02

Literatur/Links (2)

van Rossum, Guido; Python development team: *The Python Language Reference – Release 3.7.4*; 2019; als `reference.pdf` in https://docs.python.org/3/archives/python-3.7.4-docs-pdf-a4.zip; hrsg. von Python Software Foundation; zuletzt besucht am 2019-10-02

van Rossum, Guido; Python development team: *The Python Library Reference – Release 3.7.4*; 2019; als `library.pdf` in https://docs.python.org/3/archives/python-3.7.4-docs-pdf-a4.zip; hrsg. von Python Software Foundation; zuletzt besucht am 2019-10-02

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 2019-10-02

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 2019-10-02

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 2019-10-02

Literatur/Links (3)


Gosling, P. E.: *PythonTeX Quickstart*; 2016; http://mirrors.ctan.org/macros/latex/contrib/pythontex/pythontex_quickstart.pdf; zuletzt besucht am 2019-10-02


Hilpisch, Yves: *Wissenschaftliches Publizieren mit Python*; 2013;; http://www.hilpisch.com/CAE_Pycon_DE_Scientific_Publishing.pdf; zuletzt besucht am 2019-10-02


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 2019-10-02


Nettles, Bill: *nucleardata – provides nuclide information*; 2018; http://mirrors.ctan.org/macros/latex/contrib/nucleardata/nucleardata.pdf; zuletzt besucht am 2019-10-02


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 2019-10-02


Poore, Geoffrey M.: *Reproducible Documents with PythonTeX*; 2013; http://conference.scipy.org/proceedings/scipy2013/pdfs/poore.pdf; zuletzt besucht am 2019-10-02

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 2019-10-02

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 2019-10-02

Poore, Geoffrey M.: *PythonTeX Gallery*; 2017; http://tug.ctan.org/macros/latex/contrib/pythontex/pythontex_gallery.pdf; zuletzt besucht am 2019-10-02

Poore, Geoffrey M.: *The PythonTeX package – v0.17*; 2019; http://mirrors.ctan.org/macros/latex/contrib/pythontex/pythontex.pdf; zuletzt besucht am 2019-10-02

Benutzte Python-Module

Lehmann, Jörg; Schindler, Michael; Wobst, André: *PyX Manual – Release 0.15.1*; 2019; https://pyx-project.org/; zuletzt besucht am 2019-10-02

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 2019-10-02

PythonTeX: py

Wissen Sie, dass $2^{65} = 36893488147419103232$?

PythonTeX: pythontexcustomcode, sympy, def, Schleife, Primzahl

Die ersten 1002 Primzahlen sind 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179 181 191 193 197 199 211 223 227 229 233 239 241 251 257 263 269 271 277 281 283 293 307 311 313 317 331 337 347 349 353 359 367 373 379 383 389 397 401 409 419 421 431 433 439 443 449 457 461 463 467 479 487 491 499 503 509 521 523 541 547 557 563 569 571 577 587 593 599 601 607 613 617 619 631 641 643 647 653 659 661 673 677 683 691 701 709 719 727 733 739 743 751 757 761 769 773 787 797 809 811 821 823 827 829 839 853 857 859 863 877 881 883 887 907 911 919 929 937 941 947 953 967 971 977 983 991 997 1009 1013 1019 1021 1031 1033 1039 1049 1051 1061 1063 1069 1087 1091 1093 1097 1103 1109 1117 1123 1129 1151 1153 1163 1171 1181 1187 1193 1201 1213 1217 1223 1229 1231 1237 1249 1259 1277 1279 1283 1289 1291 1297 1301 1303 1307 1319 1321 1327 1361 1367 1373 1381 1399 1409 1423 1427 1429 1433 1439 1447 1451 1453 1459 1471 1481 1483 1487 1489 1493 1499 1511 1523 1531 1543 1549 1553 1559 1567 1571 1579 1583 1597 1601 1607 1609 1613 1619 1621 1627 1637 1657 1663 1667 1669 1693 1697 1699 1709 1721 1723 1733 1741 1747 1753 1759 1777 1783 1787 1789 1801 1811 1823 1831 1847 1861 1867 1871 1873 1877 1879 1889 1901 1907 1913 1931 1933 1949 1951 1973 1979 1987 1993 1997 1999 2003 2011 2017 2027 2029 2039 2053 2063 2069 2081 2083 2087 2089 2099 2111 2113 2129 2131 2137 2141 2143 2153 2161 2179 2203 2207 2213 2221 2237 2239 2243 2251 2267 2269 2273 2281 2287 2293 2297 2309 2311 2333 2339 2341 2347 2351 2357 2371 2377 2381 2383 2389 2393 2399 2411 2417 2423 2437 2441 2447 2459 2467 2473 2477 2503 2521 2531 2539 2543 2549 2551 2557 2579 2591 2593 2609 2617 2621 2633 2647 2657 2659 2663 2671 2677 2683 2687 2689 2693 2699 2707 2711 2713 2719 2729 2731 2741 2749 2753 2767 2777 2789 2791 2797 2801 2803 2819 2833 2837 2843 2851 2857 2861 2879 2887 2897 2903 2909 2917 2927 2939 2953 2957 2963 2969 2971 2999 3001 3011 3019 3023 3037 3041 3049 3061 3067 3079 3083 3089 3109 3119 3121 3137 3163 3167 3169 3181 3187 3191 3203 3209 3217 3221 3229 3251 3253 3257 3259 3271 3299 3301 3307 3313 3319 3323 3329 3331 3343 3347 3359 3361 3371 3373 3389 3391 3407 3413 3433 3449 3457 3461 3463 3467 3469 3491 3499 3511 3517 3527 3529 3533 3539 3541 3547 3557 3559 3571 3581 3583 3593 3607 3613 3617 3623 3631 3637 3643 3659 3671 3673 3677 3691 3697 3701 3709 3719 3727 3733 3739 3761 3767 3769 3779 3793 3797 3803 3821 3823 3833 3847 3851 3853 3863 3877 3881 3889 3907 3911 3917 3919 3923 3929 3931 3943 3947 3967 3989 4001 4003 4007 4013 4019 4021 4027 4049 4051 4057 4073 4079 4091 4093 4099 4111 4127 4129 4133 4139 4153 4157 4159 4177 4201 4211 4217 4219 4229 4231 4241 4243 4253 4259 4261 4271 4273 4283 4289 4297 4327 4337 4339 4349 4357 4363 4373 4391 4397 4409 4421 4423 4441 4447 4451 4457 4463 4481 4483 4493 4507 4513 4517 4519 4523 4547 4549 4561 4567 4583 4591 4597 4603 4621 4637 4639 4643 4649 4651 4657 4663 4673 4679 4691 4703 4721 4723 4729 4733 4751 4759 4783 4787 4789 4793 4799 4801 4813 4817 4831 4861 4871 4877 4889 4903 4909 4919 4931 4933 4937 4943 4951 4957 4967 4969 4973 4987 4993 4999 5003 5009 5011 5021 5023 5039 5051 5059 5077 5081 5087 5099 5101 5107 5113 5119 5147 5153 5167 5171 5179 5189 5197 5209 5227 5231 5233 5237 5261 5273 5279 5281 5297 5303 5309 5323 5333 5347 5351 5381 5387 5393

5399 5407 5413 5417 5419 5431 5437 5441 5443 5449 5471 5477 5479 5483 5501 5503 5507
5519 5521 5527 5531 5557 5563 5569 5573 5581 5591 5623 5639 5641 5647 5651 5653 5657
5659 5669 5683 5689 5693 5701 5711 5717 5737 5741 5743 5749 5779 5783 5791 5801 5807
5813 5821 5827 5839 5843 5849 5851 5857 5861 5867 5869 5879 5881 5897 5903 5923 5927
5939 5953 5981 5987 6007 6011 6029 6037 6043 6047 6053 6067 6073 6079 6089 6091 6101
6113 6121 6131 6133 6143 6151 6163 6173 6197 6199 6203 6211 6217 6221 6229 6247 6257
6263 6269 6271 6277 6287 6299 6301 6311 6317 6323 6329 6337 6343 6353 6359 6361 6367
6373 6379 6389 6397 6421 6427 6449 6451 6469 6473 6481 6491 6521 6529 6547 6551 6553
6563 6569 6571 6577 6581 6599 6607 6619 6637 6653 6659 6661 6673 6679 6689 6691 6701
6703 6709 6719 6733 6737 6761 6763 6779 6781 6791 6793 6803 6823 6827 6829 6833 6841
6857 6863 6869 6871 6883 6899 6907 6911 6917 6947 6949 6959 6961 6967 6971 6977 6983
6991 6997 7001 7013 7019 7027 7039 7043 7057 7069 7079 7103 7109 7121 7127 7129 7151
7159 7177 7187 7193 7207 7211 7213 7219 7229 7237 7243 7247 7253 7283 7297 7307 7309
7321 7331 7333 7349 7351 7369 7393 7411 7417 7433 7451 7457 7459 7477 7481 7487 7489
7499 7507 7517 7523 7529 7537 7541 7547 7549 7559 7561 7573 7577 7583 7589 7591 7603
7607 7621 7639 7643 7649 7669 7673 7681 7687 7691 7699 7703 7717 7723 7727 7741 7753
7757 7759 7789 7793 7817 7823 7829 7841 7853 7867 7873 7877 7879 7883 7901 7907 7919
7927 und 7933 .

PythonTeX: pyblock, printpythontex, sympy, Binome, Schleife, Tabelle

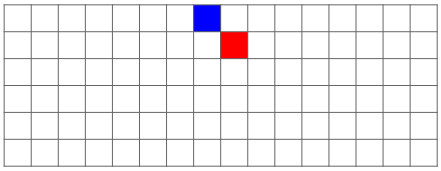
```
#from sympy import *             # symbolische Mathematik
var("a, b")                      # sympy-Variablen
Binome = []                      # Liste für Binomi-Ausdrücke vorbesetzt

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*}")
```

$$
\begin{aligned}
(a+b)^1 &= a+b \\
(a+b)^2 &= a^2+2ab+b^2 \\
(a+b)^3 &= a^3+3a^2b+3ab^2+b^3 \\
(a+b)^4 &= a^4+4a^3b+6a^2b^2+4ab^3+b^4 \\
(a+b)^5 &= a^5+5a^4b+10a^3b^2+10a^2b^3+5ab^4+b^5 \\
(a+b)^6 &= a^6+6a^5b+15a^4b^2+20a^3b^3+15a^2b^4+6ab^5+b^6 \\
(a+b)^7 &= a^7+7a^6b+21a^5b^2+35a^4b^3+35a^3b^4+21a^2b^5+7ab^6+b^7 \\
(a+b)^8 &= a^8+8a^7b+28a^6b^2+56a^5b^3+70a^4b^4+56a^3b^5+28a^2b^6+8ab^7+b^8 \\
(a+b)^9 &= a^9+9a^8b+36a^7b^2+84a^6b^3+126a^5b^4+126a^4b^5+84a^3b^6+36a^2b^7+9ab^8+b^9
\end{aligned}
$$

PythonTeX: pylab, pylabcode, fill, plot, Graph einer Funktion



$16 \times 6 = 96$; Gitternetz
roter Block bei $(8, 4)$
blauer Block bei $(7, 5)$.

PythonTeX: pyblock, sympy, Funktionen, Ableitungen, Integrale

```
#from re import sub                                  # reguläre Ausdrücke, hier sub
#from sympy import *                                 # symbolische Mathematik
var('x')                                            # symbolische Variable x

# Liste von Funktionen, die in der Tabelle aufgeführt werden sollen
funktionen = ['acos(x)', 'acosh(x)', 'acot(x)', 'acoth(x)', 'asin(x)', 'asinh(x)',
              'atan(x)', 'atanh(x)', 'cos(x)**2', 'cos(x)', 'cosh(x)', 'cot(x)**2',
              'cot(x)', 'coth(x)', 'erf(x)', 'erfc(x)', 'exp(x)', 'gamma(x)','ln(x)',
              'sin(x)**2', 'sin(x)', 'sinh(x)', 'sqrt(x)', 'tan(x)**2', 'tan(x)',
              'tanh(x)', 'csc(x)']

f          = "f(x) "
fableit    = "f'(x) "
fint       = "\int\!\!f(x)\,dx "

print(r'\begin{align*}')                            # Tabelle
for funk in funktionen:                             # Schleife
    if funk in ['sin(x)**2']:                       # ggf. neue Seite
        print(r"\displaybreak")
    meineableit = 'Derivative(' + funk + ', x)'     # Ableitung
    meinint     = 'Integral(' + funk + ', x)'       # Integral
    # f(x) = latex(f)  f'(x) = latex(f`(x))  int(x) = latex(int(x))\\
    print(latex(f),       r"&=", latex(eval(funk)),                    r'&')
    print(latex(fableit), r"&=", latex(eval(meineableit + '.doit()')), r"&")
    print(latex(fint),    r"&=", latex(eval(meinint + '.doit()')),     r'\\')
print(r'\end{align*}')
```

$f(x) = \operatorname{acos}(x)$	$f'(x) = -\frac{1}{\sqrt{-x^2+1}}$	$\int f(x)\,dx = x\operatorname{acos}(x) - \sqrt{-x^2+1}$
$f(x) = \operatorname{acosh}(x)$	$f'(x) = \frac{1}{\sqrt{x^2-1}}$	$\int f(x)\,dx = x\operatorname{acosh}(x) - \sqrt{x^2-1}$
$f(x) = \operatorname{acot}(x)$	$f'(x) = -\frac{1}{x^2+1}$	$\int f(x)\,dx = x\operatorname{acot}(x) + \frac{1}{2}\log\left(x^2+1\right)$
$f(x) = \operatorname{acoth}(x)$	$f'(x) = \frac{1}{-x^2+1}$	$\int f(x)\,dx = x\operatorname{acoth}(x) + \log(x+1) - \operatorname{acoth}(x)$
$f(x) = \operatorname{asin}(x)$	$f'(x) = \frac{1}{\sqrt{-x^2+1}}$	$\int f(x)\,dx = x\operatorname{asin}(x) + \sqrt{-x^2+1}$
$f(x) = \operatorname{asinh}(x)$	$f'(x) = \frac{1}{\sqrt{x^2+1}}$	$\int f(x)\,dx = x\operatorname{asinh}(x) - \sqrt{x^2+1}$
$f(x) = \operatorname{atan}(x)$	$f'(x) = \frac{1}{x^2+1}$	$\int f(x)\,dx = x\operatorname{atan}(x) - \frac{1}{2}\log\left(x^2+1\right)$
$f(x) = \operatorname{atanh}(x)$	$f'(x) = \frac{1}{-x^2+1}$	$\int f(x)\,dx = x\operatorname{atanh}(x) + \log(x+1) - \operatorname{atanh}(x)$
$f(x) = \cos^2(x)$	$f'(x) = -2\sin(x)\cos(x)$	$\int f(x)\,dx = \frac{x}{2} + \frac{1}{2}\sin(x)\cos(x)$
$f(x) = \cos(x)$	$f'(x) = -\sin(x)$	$\int f(x)\,dx = \sin(x)$
$f(x) = \cosh(x)$	$f'(x) = \sinh(x)$	$\int f(x)\,dx = \sinh(x)$
$f(x) = \cot^2(x)$	$f'(x) = \left(-2\cot^2(x) - 2\right)\cot(x)$	$\int f(x)\,dx = -x - \frac{\cos(x)}{\sin(x)}$
$f(x) = \cot(x)$	$f'(x) = -\cot^2(x) - 1$	$\int f(x)\,dx = \frac{1}{2}\log\left(\cos^2(x) - 1\right)$
$f(x) = \coth(x)$	$f'(x) = -\frac{1}{\sinh^2(x)}$	$\int f(x)\,dx = x - \log(\tanh(x)+1) + \log(\tanh(x))$
$f(x) = \operatorname{erf}(x)$	$f'(x) = \frac{2}{\sqrt{\pi}}e^{-x^2}$	$\int f(x)\,dx = x\operatorname{erf}(x) + \frac{e^{-x^2}}{\sqrt{\pi}}$
$f(x) = \operatorname{erfc}(x)$	$f'(x) = -\frac{2}{\sqrt{\pi}}e^{-x^2}$	$\int f(x)\,dx = x\operatorname{erfc}(x) - \frac{e^{-x^2}}{\sqrt{\pi}}$
$f(x) = e^x$	$f'(x) = e^x$	$\int f(x)\,dx = e^x$
$f(x) = \Gamma(x)$	$f'(x) = \Gamma(x)\operatorname{polygamma}(0,x)$	$\int f(x)\,dx = \int \Gamma(x)\,dx$
$f(x) = \log(x)$	$f'(x) = \frac{1}{x}$	$\int f(x)\,dx = x\log(x) - x$
$f(x) = \sin^2(x)$	$f'(x) = 2\sin(x)\cos(x)$	$\int f(x)\,dx = \frac{x}{2} - \frac{1}{2}\sin(x)\cos(x)$

$f(x) = \sin(x)$	$f'(x) = \cos(x)$	$\int f(x)\,dx = -\cos(x)$
$f(x) = \sinh(x)$	$f'(x) = \cosh(x)$	$\int f(x)\,dx = \cosh(x)$
$f(x) = \sqrt{x}$	$f'(x) = \dfrac{1}{2\sqrt{x}}$	$\int f(x)\,dx = \dfrac{2x^{\frac{3}{2}}}{3}$
$f(x) = \tan^2(x)$	$f'(x) = \left(2\tan^2(x) + 2\right)\tan(x)$	$\int f(x)\,dx = -x + \dfrac{\sin(x)}{\cos(x)}$
$f(x) = \tan(x)$	$f'(x) = \tan^2(x) + 1$	$\int f(x)\,dx = -\dfrac{1}{2}\log\left(\sin^2(x) - 1\right)$
$f(x) = \tanh(x)$	$f'(x) = -\tanh^2(x) + 1$	$\int f(x)\,dx = x - \log(\tanh(x) + 1)$
$f(x) = \csc(x)$	$f'(x) = -\cot(x)\csc(x)$	$\int f(x)\,dx = \dfrac{1}{2}\log(\cos(x) - 1) - \dfrac{1}{2}\log(\cos(x) + 1)$

PythonTeX: pyblock, sympy, Gleichungssystem

```
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"\\")
```

e : 9
h : 8
z : 1

Kurvendiskussion ganzrationaler Funktionen

mit LaTeX, Python (SymPy und PyX) und PythonTeX, Version 2.2.3

Günter Partosch*

2018-04-04

*Guenter.Partosch@hrz.uni-giessen.de

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Funktion

Die folgende ganz-rationale Funktion wird untersucht:

$$
\begin{aligned}
f(x) &= (x-4)\,(x-2)\,(x-1)\,(x+2)\,(x+4)\,(x+7) \\
\text{expandiert: } f(x) &= x^6 + 6x^5 - 27x^4 - 120x^3 + 204x^2 + 384x - 448 \\
\text{Grad} &= 6 \\
f(0) &= -448.0
\end{aligned}
$$

2 Ableitungen

Zur weiteren Untersuchung werden die Ableitungen der Funktion benötigt:

$$
\begin{aligned}
f(x) &= x^6 + 6x^5 - 27x^4 - 120x^3 + 204x^2 + 384x - 448 \\
f'(x) &= 6x^5 + 30x^4 - 108x^3 - 360x^2 + 408x + 384 \\
f''(x) &= 30x^4 + 120x^3 - 324x^2 - 720x + 408 \\
f'''(x) &= 120x^3 + 360x^2 - 648x - 720 \\
f''''(x) &= 360x^2 + 720x - 648 \\
f'''''(x) &= 720x + 720 \\
f''''''(x) &= 720
\end{aligned}
$$

3 Nullstellen

Zur Bestimmung von Minima, Maxima, Sattelpunkten (Abschnitt 4.1 auf Seite 7 und Abschnitt 4.2 auf Seite 8) oder Wendestellen (Abschnitt 4.2 auf Seite 8) werden die Nullstellen der Ableitungen (Abschnitt 2 auf Seite 5) benötigt:

$f(x)$: Nullstelle 0 : -7.0
$f(x)$: Nullstelle 1 : -4.0
$f(x)$: Nullstelle 2 : -2.0
$f(x)$: Nullstelle 3 : 1.0
$f(x)$: Nullstelle 4 : 2.0
$f(x)$: Nullstelle 5 : 4.0

$f'(x)$: Nullstelle 0 : -6.082
$f'(x)$: Nullstelle 1 : -3.154
$f'(x)$: Nullstelle 2 : -0.651
$f'(x)$: Nullstelle 3 : 1.522
$f'(x)$: Nullstelle 4 : 3.365

$f''(x)$: Nullstelle 0 : -5.092
$f''(x)$: Nullstelle 1 : -2.067
$f''(x)$: Nullstelle 2 : 0.483
$f''(x)$: Nullstelle 3 : 2.676

$f'''(x)$: Nullstelle 0 : -3.978
$f'''(x)$: Nullstelle 1 : -0.833
$f'''(x)$: Nullstelle 2 : 1.811

$f''''(x)$: Nullstelle 0 : -2.673
$f''''(x)$: Nullstelle 1 : 0.673

$f'''''(x)$: Nullstelle 0 : -1.0

4 Weitere Untersuchungen der Extremstellen

4.1 Untersuchung der Nullstellen der 1. Ableitung

Bedingungen für das Vorliegen eines Extremums bei x_1:

- notwendige Voraussetzung: $f'(x_1) = 0$
- hinreichende: $f''(x_1) \neq 0$ (> 0: Minimum; < 0: Maximum)

Bedingungen für das Vorliegen eines Sattelpunkts bei x_1:

- notwendige Voraussetzung: $f'(x_1) = 0$ und $f''(x_1) = 0$
- hinreichende: $f'''(x_1) \neq 0$

$f'(x)$: Nullstelle 0 bei x = -6.082 $\Longrightarrow$ mögliche Extremstelle

- in 2. Ableitung $(30x^4 + 120x^3 - 324x^2 - 720x + 408) \Longrightarrow$ y = 6854.71 $> 0 \Longrightarrow$ mögliche Minimumstelle
- in 3. Ableitung $(120x^3 + 360x^2 - 648x - 720) \Longrightarrow$ y = -10459.961 < 0
- in 4. Ableitung $(360x^2 + 720x - 648) \Longrightarrow$ y = 8289.818 > 0
- in 5. Ableitung $(720x + 720) \Longrightarrow$ y = -3659.079 < 0
- in 6. Ableitung $(720) \Longrightarrow$ y = 720 > 0

$f'(x)$: Nullstelle 1 bei x = -3.154 $\Longrightarrow$ mögliche Extremstelle

- in 2. Ableitung $(30x^4 + 120x^3 - 324x^2 - 720x + 408) \Longrightarrow$ y = -1340.013 $< 0 \Longrightarrow$ mögliche Maximumstelle
- in 3. Ableitung $(120x^3 + 360x^2 - 648x - 720) \Longrightarrow$ y = 1140.229 > 0
- in 4. Ableitung $(360x^2 + 720x - 648) \Longrightarrow$ y = 661.663 > 0
- in 5. Ableitung $(720x + 720) \Longrightarrow$ y = -1550.585 < 0
- in 6. Ableitung $(720) \Longrightarrow$ y = 720 > 0

$f'(x)$: Nullstelle 2 bei x = -0.651 $\Longrightarrow$ mögliche Extremstelle

- in 2. Ableitung $(30x^4 + 120x^3 - 324x^2 - 720x + 408) \Longrightarrow$ y = 711.766 $> 0 \Longrightarrow$ mögliche Minimumstelle
- in 3. Ableitung $(120x^3 + 360x^2 - 648x - 720) \Longrightarrow$ y = -178.275 < 0
- in 4. Ableitung $(360x^2 + 720x - 648) \Longrightarrow$ y = -964.26 < 0

- in 5. Ableitung $(720x + 720) \Longrightarrow$ y = 250.97 > 0
- in 6. Ableitung $(720) \Longrightarrow$ y = 720 > 0

$f'(x)$: Nullstelle 3 bei x = 1.522 $\Longrightarrow$ mögliche Extremstelle

- in 2. Ableitung $(30x^4 + 120x^3 - 324x^2 - 720x + 408) \Longrightarrow$ y = -854.448 < 0 $\Longrightarrow$ mögliche Maximumstelle
- in 3. Ableitung $(120x^3 + 360x^2 - 648x - 720) \Longrightarrow$ y = -448.859 < 0
- in 4. Ableitung $(360x^2 + 720x - 648) \Longrightarrow$ y = 1282.313 > 0
- in 5. Ableitung $(720x + 720) \Longrightarrow$ y = 1816.054 > 0
- in 6. Ableitung $(720) \Longrightarrow$ y = 720 > 0

$f'(x)$: Nullstelle 4 bei x = 3.365 $\Longrightarrow$ mögliche Extremstelle

- in 2. Ableitung $(30x^4 + 120x^3 - 324x^2 - 720x + 408) \Longrightarrow$ y = 2733.985 > 0 $\Longrightarrow$ mögliche Minimumstelle
- in 3. Ableitung $(120x^3 + 360x^2 - 648x - 720) \Longrightarrow$ y = 5746.865 > 0
- in 4. Ableitung $(360x^2 + 720x - 648) \Longrightarrow$ y = 5850.466 > 0
- in 5. Ableitung $(720x + 720) \Longrightarrow$ y = 3142.641 > 0
- in 6. Ableitung $(720) \Longrightarrow$ y = 720 > 0

4.2 Untersuchung der Nullstellen der 2. Ableitung

Bedingungen für das Vorliegen einer Wendestelle bei x_1:

- notwendige Voraussetzung: $f''(x_1) = 0$
- hinreichende: $f'''(x_1) \neq 0$

$f''(x)$: Nullstelle 0 bei x = -5.092 $\Longrightarrow$ mögliche Wendestelle

- in 3 . Ableitung $(120x^3 + 360x^2 - 648x - 720) \Longrightarrow$ y = -3931.195 < 0 $\Longrightarrow$ mögliche Wendestelle
- in 4 . Ableitung $(360x^2 + 720x - 648) \Longrightarrow$ y = 5021.022 > 0
- in 5 . Ableitung $(720x + 720) \Longrightarrow$ y = -2946.488 < 0
- in 6 . Ableitung $(720) \Longrightarrow$ y = 720 > 0

$f''(x)$: Nullstelle 1 bei x = -2.067 $\Longrightarrow$ mögliche Wendestelle

- in 3 . Ableitung $(120x^3 + 360x^2 - 648x - 720) \Longrightarrow$ y = 1097.671 > 0 $\Longrightarrow$ mögliche Wendestelle
- in 4 . Ableitung $(360x^2 + 720x - 648) \Longrightarrow$ y = -598.263 < 0
- in 5 . Ableitung $(720x + 720) \Longrightarrow$ y = -768.129 < 0

- in 6 . Ableitung (720) $\Longrightarrow$ y = 720 > 0

$f''(x)$: Nullstelle 2 bei x = 0.483 $\Longrightarrow$ mögliche Wendestelle

- in 3 . Ableitung ($120x^3 + 360x^2 - 648x - 720$) $\Longrightarrow$ y = -935.434 < 0 $\Longrightarrow$ mögliche Wendestelle
- in 4 . Ableitung ($360x^2 + 720x - 648$) $\Longrightarrow$ y = -216.475 < 0
- in 5 . Ableitung ($720x + 720$) $\Longrightarrow$ y = 1067.613 > 0
- in 6 . Ableitung (720) $\Longrightarrow$ y = 720 > 0

$f''(x)$: Nullstelle 3 bei x = 2.676 $\Longrightarrow$ mögliche Wendestelle

- in 3 . Ableitung ($120x^3 + 360x^2 - 648x - 720$) $\Longrightarrow$ y = 2424.958 > 0 $\Longrightarrow$ mögliche Wendestelle
- in 4 . Ableitung ($360x^2 + 720x - 648$) $\Longrightarrow$ y = 3857.715 > 0
- in 5 . Ableitung ($720x + 720$) $\Longrightarrow$ y = 2647.004 > 0
- in 6 . Ableitung (720) $\Longrightarrow$ y = 720 > 0

5 Symmetrieeigenschaften

Bedingungen für das Vorliegen von Symmetrien:

- Funktion ist gerade: $f(x_0-h) = f(x_0+h), \forall h \in \mathbf{R} \Longrightarrow$ Symmetrie bzgl. der Geraden $x = x_0$
- Funktion ist ungerade: $f(x_0-h) + f(x_0+h) = 2f(x_0), \forall h \in \mathbf{R} \Longrightarrow$ Symmetrie bzgl. des Punktes $(x_0 \mid f(x_0))$

keine Symmetriegerade ermittelbar

6 Zusammenfassung

6.1 Nullstellen

Die folgenden Nullstellen wurden gefunden:

$$f(x) : \text{Nullstelle } 0 : \text{ -7.0}$$
$$f(x) : \text{Nullstelle } 1 : \text{ -4.0}$$
$$f(x) : \text{Nullstelle } 2 : \text{ -2.0}$$
$$f(x) : \text{Nullstelle } 3 : \text{ 1.0}$$
$$f(x) : \text{Nullstelle } 4 : \text{ 2.0}$$
$$f(x) : \text{Nullstelle } 5 : \text{ 4.0}$$

6.2 Minima

Die folgenden reellwertigen Minimumpunkte wurden ermittelt:

(-6.082 | -4502.128)
(-0.651 | -583.896)
(3.365 | -839.555)

Siehe dazu auch Abschnitt 4.1 auf Seite 7.

6.3 Maxima

Die folgenden reellwertigen Maximumpunkte wurden ermittelt:

(-3.154 | 575.103)
(1.522 | 102.477)

Siehe dazu auch Abschnitt 4.1 auf Seite 7.

6.4 Wendepunkte

Die folgenden reellwertigen Wendepunkte wurden ermittelt:

(-5.092 | -2531.614)
(-2.067 | 48.236)
(0.483 | -229.857)
(2.676 | -453.421)

Siehe dazu auch Abschnitt 4.2 auf Seite 8.

6.5 Sattelpunkte

Es wurden keine reellwertigen Sattelpunkte ermittelt. Siehe dazu Abschnitt 4.1 auf Seite 7.

6.6 Symmetrien

Es wurde keine Symmetriegerade/kein Symmetriepunkt ermittelt. Siehe dazu Abschnitt 5 auf Seite 10.

7 Graphische Darstellungen

- Falls darstellbar, werden Nullstellen (Abschnitt 6.1 auf Seite 11) durch kleine Quadrate gekennzeichnet.
- Falls vorhanden, werden „besondere“ Punkte [Minima (Abschnitt 6.2 auf Seite 11), Maxima (Abschnitt 6.3 auf Seite 11), Sattelpunkte (Abschnitt 6.5 auf Seite 12), Wendepunkte (Abschnitt 6.4 auf Seite 12)] durch kleine Kreise gekennzeichnet.

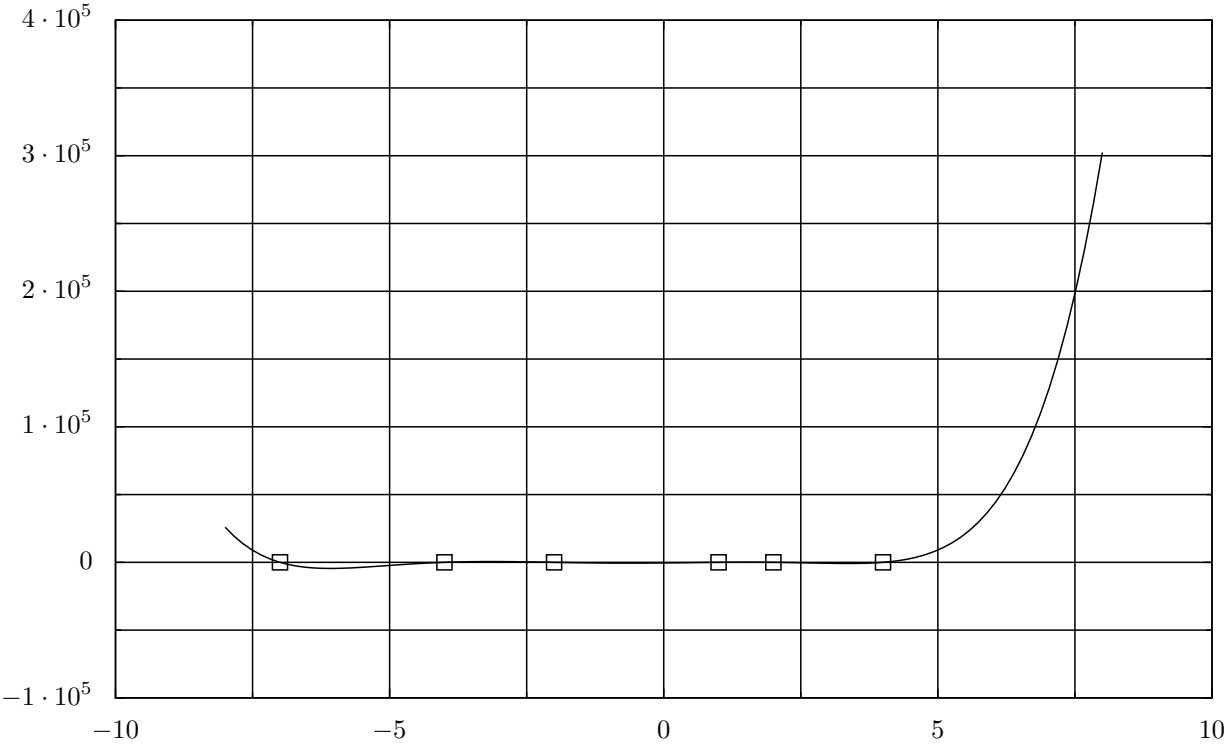


Abbildung 7.1: $f(x) = x^6 + 6x^5 - 27x^4 - 120x^3 + 204x^2 + 384x - 448$ (Übersicht)

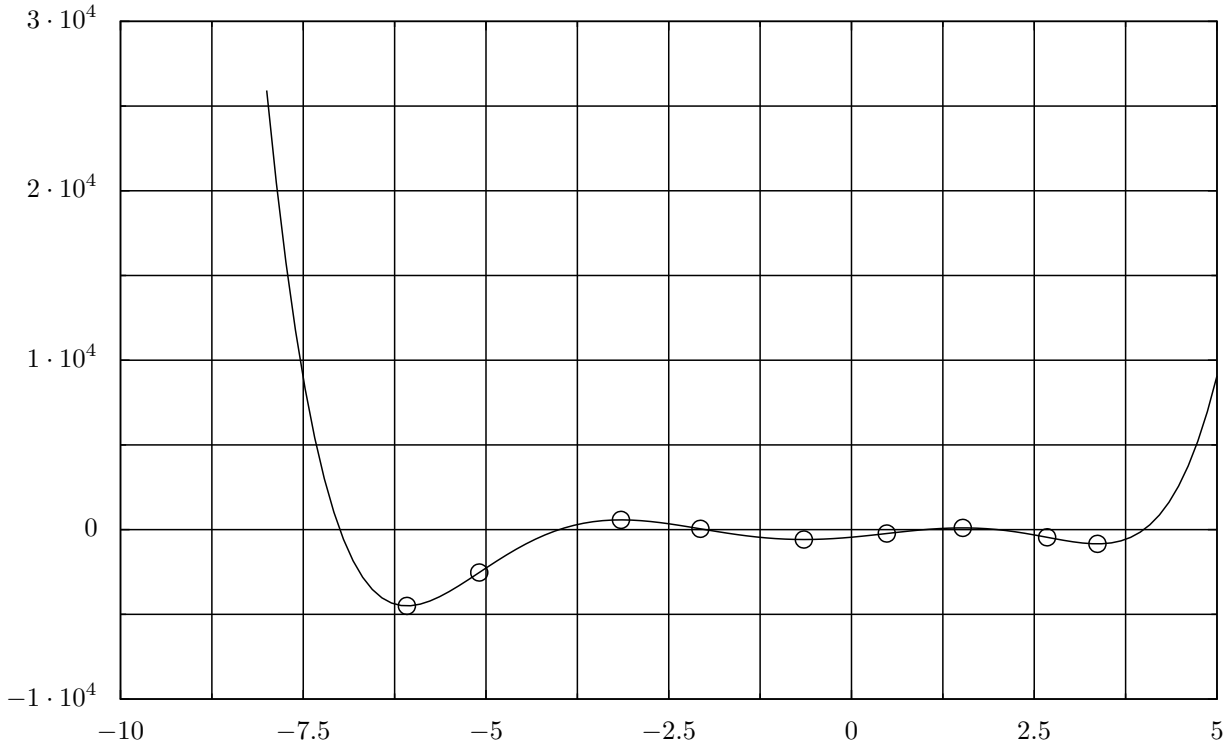


Abbildung 7.2: $f(x) = x^6 + 6x^5 - 27x^4 - 120x^3 + 204x^2 + 384x - 448$

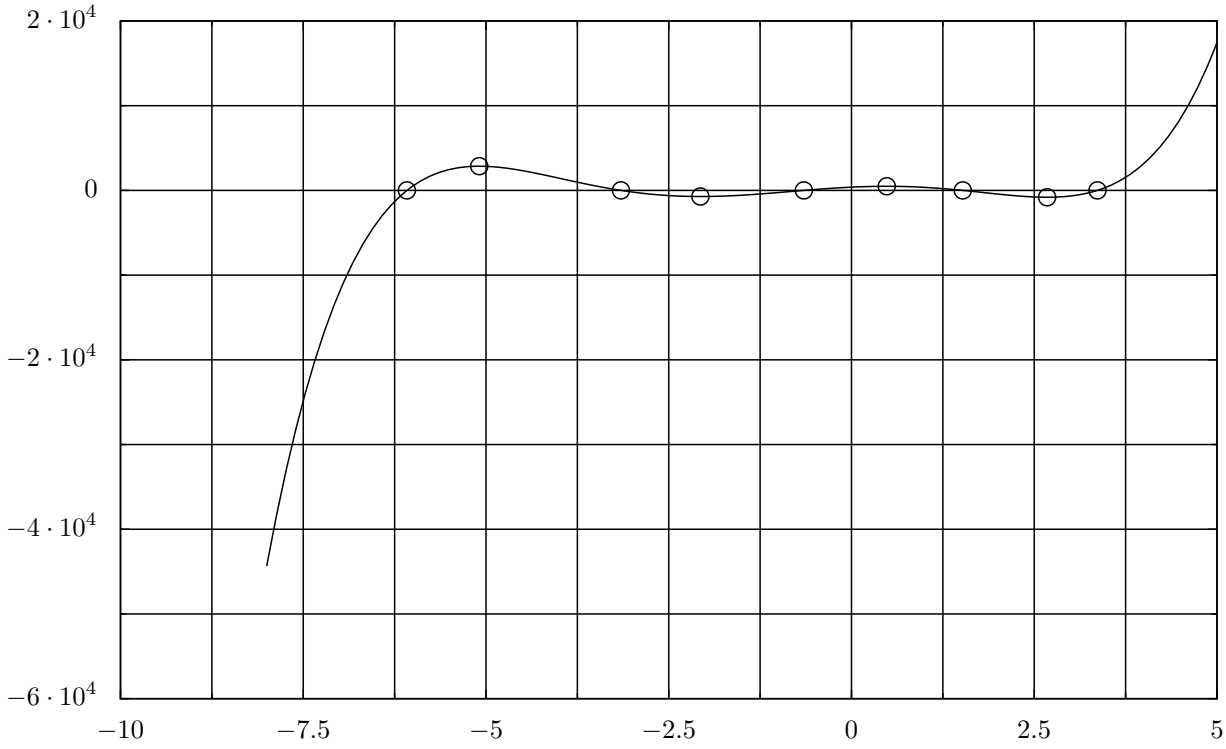


Abbildung 7.3: $f'(x) = 6x^5 + 30x^4 - 108x^3 - 360x^2 + 408x + 384$

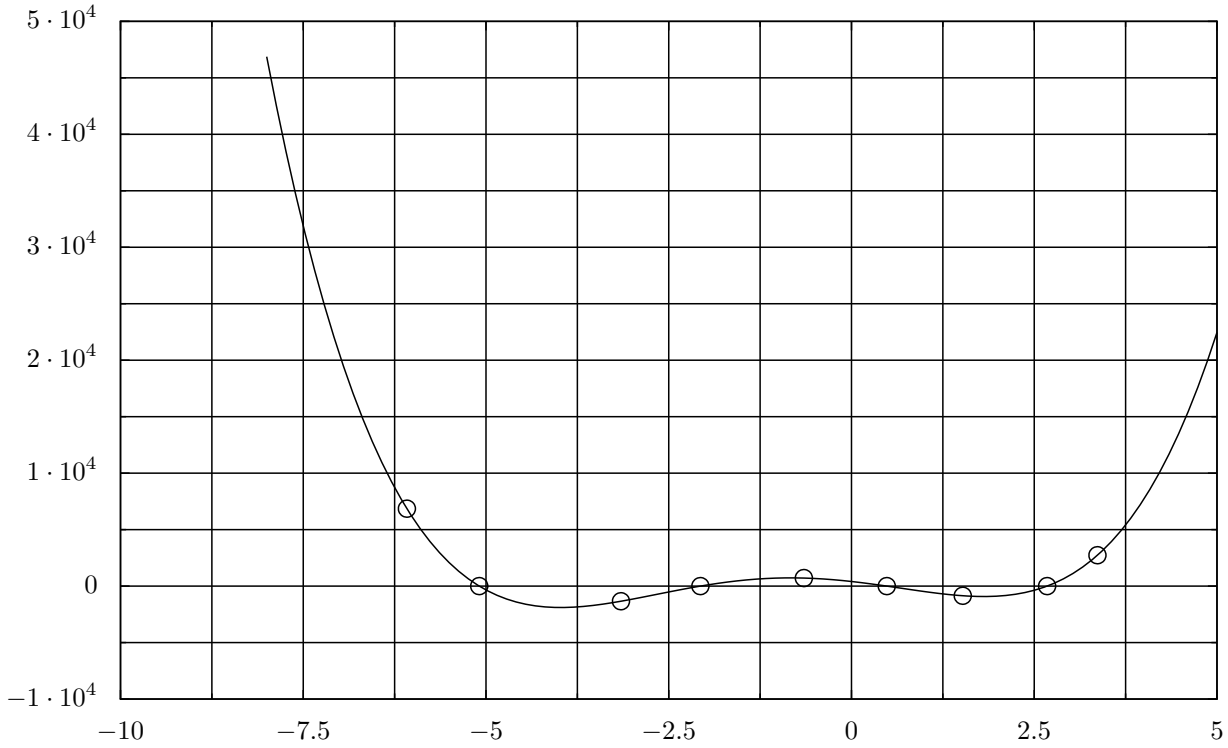


Abbildung 7.4: $f''(x) = 30x^4 + 120x^3 - 324x^2 - 720x + 408$

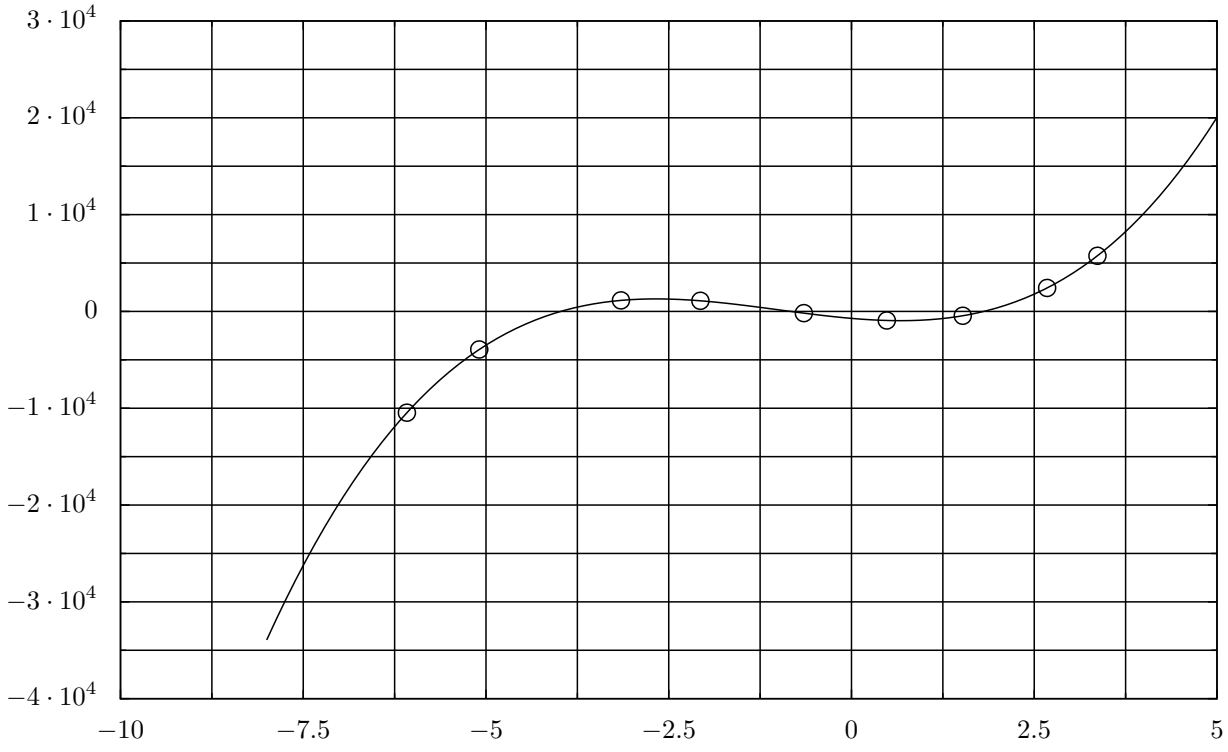


Abbildung 7.5: $f'''(x) = 120x^3 + 360x^2 - 648x - 720$

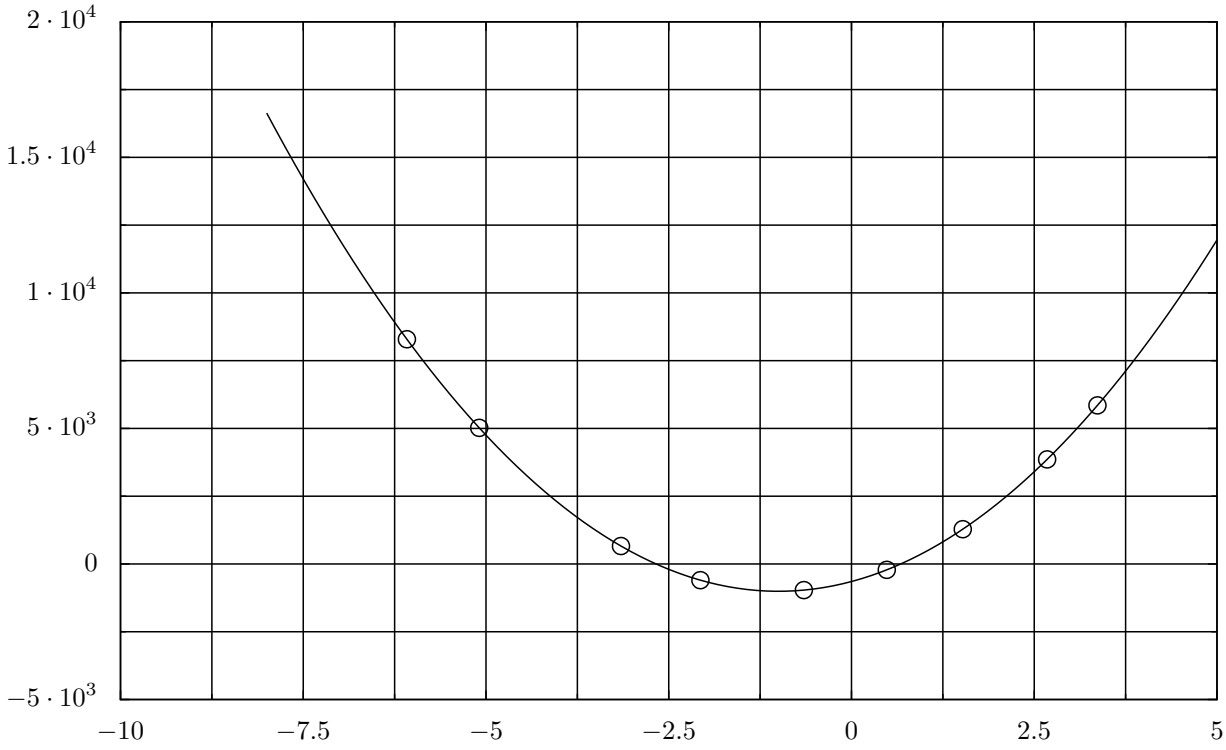


Abbildung 7.6: $f''''(x) = 360x^2 + 720x - 648$

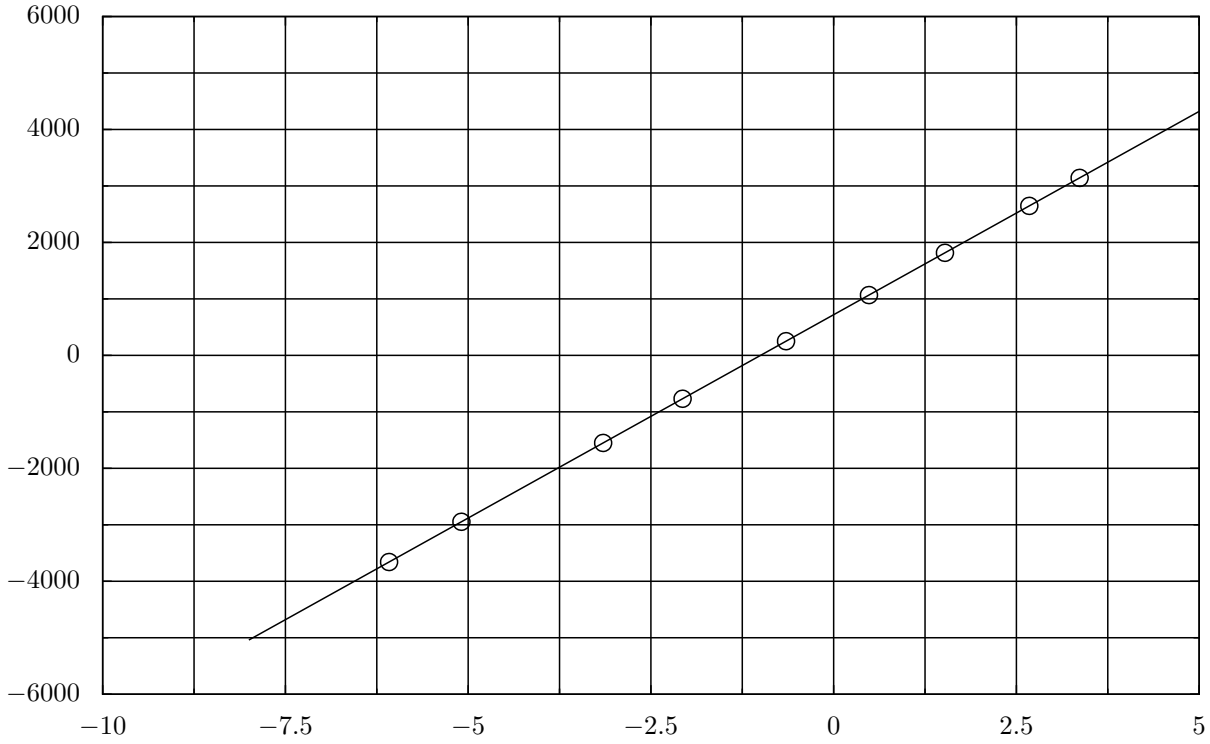


Abbildung 7.7: $f'''''(x) = 720x + 720$

PythonTeX: sympyblock, sympy.stats, LaTeX-Anweisungen mit Python

```
from sympy.stats import DiscreteUniform, sample      # sympy.stats (Statistik),
                                                     # hier Verteilungen/Beispieldaten
x = Symbol('x')
a = DiscreteUniform('a', range(-10, 11))
b = DiscreteUniform('b', range(-10, 11))
c = DiscreteUniform('c', range(-10, 11))
d = DiscreteUniform('d', range(-10, 11))

def zufallsquadratisch():                            # Zufallspolynom 2. Grades
    return Eq(sample(a)*x**2 + sample(b)*x + sample(c))

def zufallskubisch():                                # Zufallspolynom 3. Grades
    return Eq(sample(a)*x**3 + sample(b)*x*x + sample(c)*x + sample(d))
```

Zufallspolynom: `$\zufallsquadratisch$`: $-x^2 - 10x + 2 = 0$
Zufallspolynom: `$\zufallskubisch$`: $-10x^3 + 7x^2 + 4x - 9 = 0$
3 hoch 20: 3486784401
Zeichenkette umkehren: !txeT nie tsi saD
tausche erstes und letztes Zeichen: k123456789abcdefghij0

PythonTeX: pycode/pyblock-Umgebung, printpythontex, Schleife, Tabelle mit Zweierpotenzen

```
# 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}")
```

m	2^m
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
9	512
10	1024
11	2048
12	4096
13	8192
14	16384
15	32768
16	65536
17	131072
18	262144
19	524288
20	1048576
21	2097152
22	4194304
23	8388608
24	16777216
25	33554432
26	67108864
27	134217728
28	268435456
29	536870912
30	1073741824

PythonTeX: sympy, sympyblock, printpythontex, Ableitung, eq.replace

```
#from sympy import *
x = symbols('x')                                  # sympy-Variable

print(r'\begin{align*}')
for funk in [sin(x), sinh(x), csc(x)]:
    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*}')
```

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}x} \sin\left(x\right) &= \cos\left(x\right)\\
\frac{\mathrm{d}}{\mathrm{d}x} \sinh\left(x\right) &= \cosh\left(x\right)\\
\frac{\mathrm{d}}{\mathrm{d}x} \csc\left(x\right) &= - \cot\left(x\right) \csc\left(x\right)
\end{aligned}$$