Diejenigen, die aktiv mitmachen wollen, setzen sich bitte nach vorn.

Python & LaTeX

Dante e. V. Frühjahrstagung 2017

Dr. Uwe Ziegenhagen

22. März 2017

Überblick

Was machen wir heute?

- Python Grundlagen
- Python in LaTeX Dokumenten
- Erzeugung von LaTeX Dokumenten

Voraussetzungen

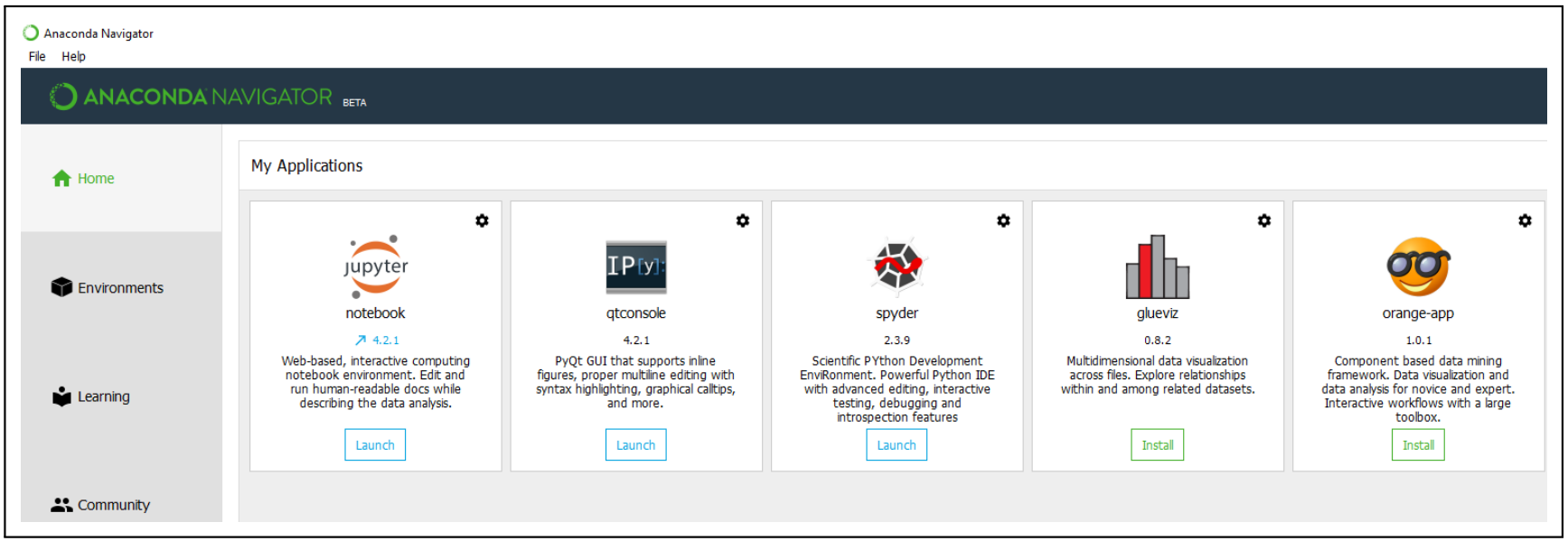
Was wird benötigt

- Aktuelle TEX Installation
- Python3 Installation, vorzugsweise Anaconda/Winpython
- Pakete:
  - numpy
  - jinja2
  - pandas
- Diese Folien und Code-Beispiele unter:

`https://github.com/UweZiegenhagen/PythonAndLaTeX`

„Scientific Python“ Distributionen

- Linux/MacOS X kommen mit Python, aber nicht SciPy
- Manuell nachinstallieren oder „echte“ Installation
- Meine Empfehlung: Anaconda
  - Anaconda (`https://www.continuum.io/downloads`)
  - WinPython (`https://winpython.github.io`)



Das SciPy Framework

Neben `pandas` ist enthalten:

NumPy	matrices, vectors, algorithms
IPython	Matlab/Mathematica-like environment
Matplotlib	scientific plotting, basis for `seaborn` library
SymPy	symbolic mathematics
...	etc, etc.

Testen der Installation

- Funktionieren die folgenden Befehle?

```
1 import jinja2
2 import pandas
3 import numpy
```

Python

- Erfunden von Guido van Rossum (Niederlande)
- Fokus auf lesbaren und verständlichen Code
- „batteries included“ $\Rightarrow$ umfangreiche Standardbibliothek
- Mein erster Kontakt mit Python: Downloadskript für den SaveTV Online-VCR
- Python2 versus Python3 $\Rightarrow$ Python3
- Editor? Ich nutze Spyder3, auch empfehlenswert: Geany
- Python selbst liefert IDLE mit

Python „Hello World“

```
1  print('Hello Python')
2  # Kommentar
3  a = 123.4
4  a+=2
5  print(a+2)
6
7  def myFunction(a):
8      b = a + a
9      return b
10
11 print(myFunction(2)) # 4
12 print(myFunction('a')) # 'aa'
```

Listing 1: Hello World in Python 3.x, sources/helloWorld.py

Strings, Listen und Tupel

Strings

```
1  a = 'Hallo'
2  b = 'Welt'
3
4  c = a + ' ' + b
5  'W' in c # True
6  print(c[0]) # 'H'
7  print(c[-1]) # 't'
8  print(c[1:3]) # 'al'
9  print(c[1:4]) # 'all'
10 print(c[1:-1]) # 'allo Wel'
11 print(c[1:]) # 'allo Welt'
12
13 for i in c:
14   print(i)
```

Listing 2: Strings, sources/Strings.py

Strings, Listen und Tupel

Stringfunktionen

```
1 meinString = 'Hallo Welt'
2
3 meinString.upper()
4 meinString.find('Welt')
5 meinString.split(' ')
6 meinString.replace('Welt', 'World')
```

Listing 3: Strings, sources/Strings2.py

Strings, Listen und Tupel

Listen

- Index von 0 bis $n - 1$
- veränderbar

```
1 beatles = ['John', 'Paul', 'Ringo', 'George']
2 print(len(beatles))
3 print(beatles[3])
4 beatles.append('Yoko Ono')
5 print(beatles.index('John'))
```

Listing 4: Listen, sources/Listen.py

Strings, Listen und Tupel

Tupel

- ähnlich wie Listen
- Index von 0 bis $n - 1$
- nicht veränderbar, also „schreibgeschützt“

```
1 monate=('Jan', 'Feb', 'Mar', 'Apr', 'Mai')
2 print(monate[1])
3 print(monate[1:3])
```

Listing 5: Tupel, sources/Tupel.py

Dictionaries

Key-Value Paare

- nicht veränderbar

```
1 lookup={'EUR':'Euro', 'GBP':'Pound', 'USD':'US-Dollar'}
2 print(lookup['EUR'])
```

Listing 6: Tupel, sources/Dictionaries.py

Flusssteuerung

if/then

```
1  a = 1
2
3  def doThis():
4      print('Done')
5
6  if a==1:
7    doThis()
8  else:
9    pass # pass = "Do nothing"
```

Listing 7: if-then, sources/ifthen.py

"condition" kann ein üblicher Boolean Ausdruck sein.

Flusssteuerung

for

```
1 myString = 'Python'
2 for c in myString:
3     print(c)
```

Listing 8: for-Schleifen, sources/for.py

Flusssteuerung

while

```
1 n = 4
2 while n>0:
3   print(n)
4   n-=1
```

Listing 9: while-Schleifen, sources/while.py

Funktionen

```
1 def add(a,b):
2   return a+b
3
4 def multiply3(a,b,c=1):
5   return a*b*c
```

Listing 10: Definition von Funktionen, sources/Funktionen.py

Ein- und Ausgabe

Kommandozeile

```
1 a = input('Erstes Wort')
2 b = input('Zweites Wort')
3
4 print(a, b)
5 print(a, b, sep='')
6 print(a, b, sep=':')
```

Listing 11: Ein- und Ausgabe: Kommandozeile, sources/io.py

Ein- und Ausgabe

Dateien lesen

```
1 dateiname = 'exampleFile.txt'
2
3 filepointer = open(dateiname,'r')
4 # 'r' (read) oder 'a' (append)
5 for zeile in filepointer:
6         print(zeile)
7
8 filepointer.close()
```

Listing 12: Ein- und Ausgabe: Dateien,sources/readWriteFile.py

- Um Excel, CSV und ähnliches zu lesen $\Rightarrow$ pandas sehr empfehlenswert

Ein- und Ausgabe

UTF8-Dateien lesen und schreiben

```
1  import io
2  import datetime
3
4  jetzt = datetime.datetime.now()
5  dateiname = 'example.txt'
6
7  with io.open(dateiname,'w',encoding='utf8') as datei:
8      datei.write('äüöß ' + jetzt.isoformat())
9
10 with io.open(dateiname,'r',encoding='utf8') as datei:
11     text = datei.read()
12     print(text)
```

Listing 13: UTF8, sources/readWriteUTF8.py

Andere wichtige Befehle

Das os Modul

- `os.system(<Befehl>)` Führt externen Befehl aus
- `os.start(<Datei>)` Öffnet Datei mit der Standard-Applikation (nur unter Windows)

Aufgabe 1

- Wir erzeugen Rechenaufgaben für Kinder
- links die Aufgabe, rechts die Lösung
- Aufgaben randomisiert (`random` Modul)
- Datei automatisch übersetzen und ausführen
- Tipp für Font: TeX Gyre Schola, `\usepackage{tgschola}`

Python und LaTeX verbinden

Q: Wie kann man LaTeX und Python in nur einem Dokument verwalten?
A: Literate programming[1]

- etwas „Selbstgestricktes“
- PythonTeX von Geoffrey M. Poore
  `https://www.ctan.org/pkg/pythontex`

[1] WEB, CWEB, NOWEB, SWEAVE, knitR

Python im LaTeX-Lauf

```
\makeatletter
\newenvironment{pycode}[1]%
  {\xdef\d@tn@me{#1}\xdef\r@ncmd{python #1.py > #1.plog}%
  \typeout{Writing file #1}\VerbatimOut{#1.py}%
  }
  {\endVerbatimOut %
 \toks0{\immediate\write18}%
 \expandafter\toks\expandafter1\expandafter{\r@ncmd}%
 \edef\d@r@ncmd{\the\toks0{\the\toks1}}\d@r@ncmd %
 \lstinputlisting[language={Python},label=listing:\d@tn@me
     ,basicstyle={\ttfamily\footnotesize}]{\d@tn@me.py}%
 \lstinputlisting[basicstyle={\ttfamily\footnotesize}]{\
     d@tn@me.plog}%
}
\makeatother
```

- Beispiele unter /sources, leicht anpassbar

PythonTEX

- „PythonTeX“ Paket von Geoffrey M. Poore
- Paket in Version 0.15 auf CTAN, in TEX Live enthalten
- unter Windows `pythontex.exe`
- stellt mehrere Befehle und Umgebungen bereit

PythonTEX

Befehle

- `\py{}` für Code, der einen String zurückliefert
- `\pyc{}` für Code, der nur ausgeführt wird
- `\pys{}` mit Substitution
- `\pyv{}` für Code, der nur gesetzt wird
- `\pyb{}` für Ausführung und Satz der Ergebnisse

Umgebungen

pycode	wie `pyc`
pysub	wie `pys`
pyverbatim	wie `pyv`
pyblock	wie `pyb`
pyconsole	simuliert eine Python-Konsole

PythonTEX

Arbeitsweise

Arara-Regel

- Was ist Arara?
- texwelt.de/wissen/fragen/8764/was-ist-arara

```
!config
# Nomentbl rule for arara
# author: Uwe Ziegenhagen
# requires arara 3.0+
identifier: pythontex
name: pythontex
command: <arara> pythontex @{options} "@{getBasename(file)}.pytxcode"
arguments:
- identifier: style
  flag: <arara> @{parameters.style}
  default: pythontex
- identifier: options
  flag: <arara> @{parameters.options}
```

Listing 14: UTF8, sources/pythontex.yaml

Daten verarbeiten

Die wunderbare Welt von pandas

- mein „täglich Brot“: Datenbestände zwischen Banksystemen abgleichen
- „pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.“[2]
- Initiiert 2008 durch Wes McKinney von AQR Capital Management für hoch-performante quantitative Analyse
- Wesentliche Teile in C/Cython implementiert
- Current version is 0.19

[2] Source: pandas.pydata.org

Series und DataFrames

- central data structures in `pandas`

Daten lesen

Command	Description
read_pickle	read Pickle objects
read_table	for general table-like formats
read_csv	Comma-Separated Values
read_fwf	for weird fixed-width formats
read_clipboard	read from clipboard
read_excel	read Excel files

other commands for HTML, JSON, HDF5, ...

Reading CSV

- CSV can be pretty „messed up“:
  - column separator
  - decimal separator
  - text encoding
- see the specification:
  `http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_csv.html`

sep	specify the column separator
thousands	Thousands separator
encoding	hopefully UTF-8 (BOM!)
decimal	decimal separator
converters	converters={'A': str} for explicit conversion

Reading Excel

- Use `pd.read_excel()` to read XLSX files (which are just zipped XMLs)
- see the documentation: `http://pandas.pydata.org/pandas-docs/stable/generated/pandas.read_excel.html`
- to export to Excel use `pd.to_excel()` function
- remarks:
  - Excel export is much slower than CSV
  - Exporting „styled“ Excel requires additional effort
  - Excel can be well controlled via COM (Common Object Model)

Querying DataFrames

Getting basic information

- For the next slides load `Northwind` data
- First task after loading data: do some sanity checks

```
1 customers = pd.read_excel('Northwind.xlsx',
2                           sheetname = 'Customers')
3
4 print(len(customers))   # Zeilenzahl
5 print(customers.head()) # die ersten 5 Zeilen
6 print(customers.tail()) # die letzten 5 Zeilen
7 print(list(customers)) # die Spaltennamen
```

Pandas Dataframe Operations

Selection and Filtering I

- pandas provides sophisticated methods to select, filter and transform rows and columns
- Select only certain columns

```
df = df[['colA', 'colB']]
```

- Select only first two rows (hint: index starts at 0)

```
df.iloc[:1]
```

- Select only rows where column value is greater

```
df[df['colA'] > 50]
```

Pandas Dataframe Operations

Selection and Filtering II

- Select only rows where column value is greater than 50 and smaller than 500

```
df[(df['colA'] > 50)| (df['colA'] < 500)]
```

- Select only rows where column value is not

```
df[~(df['colA'] == 'HelloWorld')]
```

- Select those rows, where column b is 'A' or 'B'

```
df = df[ (df['b'] == 'A')| (df['b'] == 'I')]
```

- more readable alternative via `isin()`

```
df = df[df['b'].isin(['A','I'])]
```

- or the opposite

```
df = df[~df['b'].isin(['A','I'])]
```

Pandas Dataframe Operations

Merging and Joining

- `merge()` provides SQL-like merging
- very handy to combine different datasets
- Supported are the followin join-types:
  - Left
  - Right
  - Inner
  - Full Outer
- `join()` is special alias for `merge()`, works on index, not columns (the default for merge)

Pandas Dataframe Operations

Merging and Joining

- Default-command for `merge()`

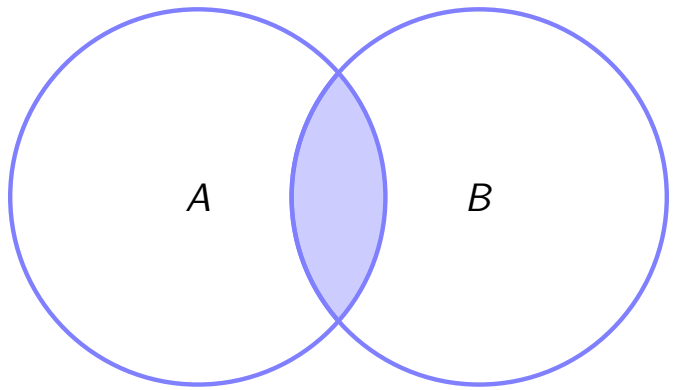
```
leftDataFrame.merge(rightDataFrame, how='inner',
on=None, left_on=None, right_on=None, left_index=False,
right_index=False, sort=False, suffixes=('_x', '_y'),
copy=True, indicator=False)
```

1. define the other DataFrame to merge
2. define how to merge
3. define the keys to use for the merge

Merging

Inner Join

- Select all data which is in A and B



left

	A	Key
0	A0	K0
1	A1	K1
2	A2	K2
3	A3	K4

right

	B	Key
0	B0	K0
1	B1	K1
2	B2	K2
3	C3	K5

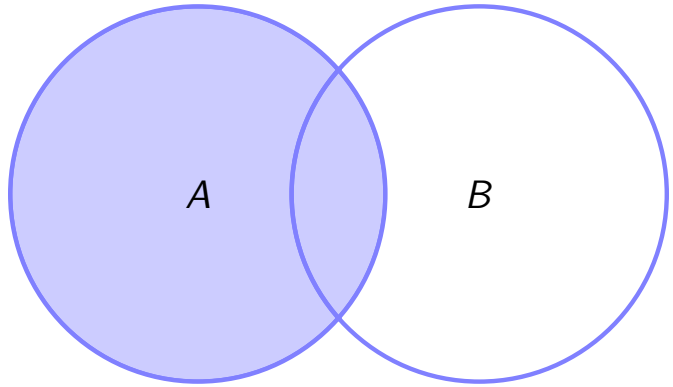
merged

	A	B	Key
0	A0	B0	K0
1	A1	B1	K1
2	A2	B2	K2

Merging

Left Join

- Select all data which is in A and B



left

	A	Key
0	A0	K0
1	A1	K1
2	A2	K2
3	A3	K4

right

	B	Key
0	B0	K0
1	B1	K1
2	B2	K2
3	C3	K5

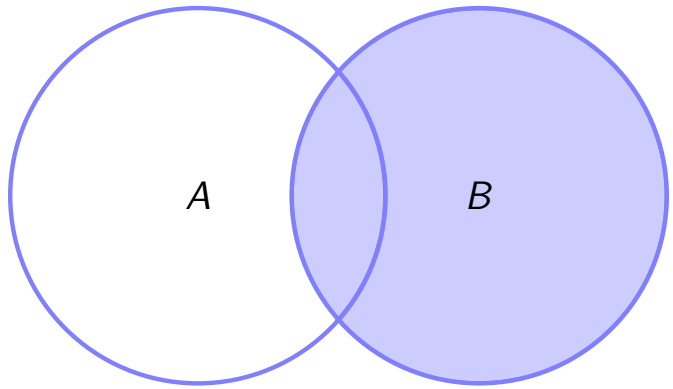
merged

	A	B	Key
0	A0	B0	K0
1	A1	B1	K1
2	A2	B2	K2
3	A3	NaN	K4

Merging

Right Join

- Select all data which is in B



left

	A	Key
0	A0	K0
1	A1	K1
2	A2	K2
3	A3	K4

right

	B	Key
0	B0	K0
1	B1	K1
2	B2	K2
3	C3	K5

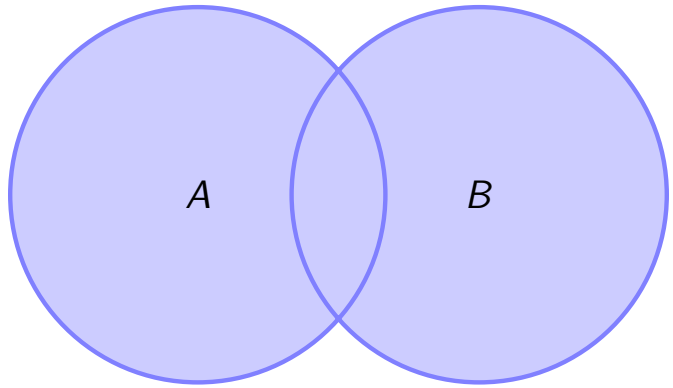
merged

	A	B	Key
0	A0	B0	K0
1	A1	B1	K1
2	A2	B2	K2
3	NaN	B3	K5

Merging

Full Outer Join

- Select all data which is in A or B



left

	A	Key
0	A0	K0
1	A1	K1
2	A2	K2
3	A3	K4

right

	B	Key
0	B0	K0
1	B1	K1
2	B2	K2
3	C3	K5

merged

	A	B	Key
0	A0	B0	K0
1	A1	B1	K1
2	A2	B2	K2
3	A3	NaN	K4
4	NaN	B3	K5

Merging

Join Examples

Exercise

1. Get `Northwind.xlsx`
2. Load the sheets and into dataframes
3. Merge both frames using different join types

Example: Merging Rows into Columns

Spalte	Wert
ColA	Andi
ColB	Berni
ColC	Cesar
ColA	Dorian
ColB	Ernst
ColC	Frank

```
1 import pandas as pd
2 daten = pd.read_excel('combine.xlsx')
3 result = pd.DataFrame(columns=['ColA', 'ColB', 'ColC'])
4 for i, row in daten.iterrows():
5     result.loc[i // 3, row['Spalte']] = row['Wert']
6
7 print(result)
```

Example: Creating Tax Donation Receipts

- Donations to Dingfabrik are tax-deductible
- Manual creation error-prone and labor-intensive
- Last year: complicated mix (Python, MySQL, LaTeX)
- This year: pandas, much easier
- Interested? `http://uweziegenhagen.de/?p=3359`