

# DATEN FÖRDERN UND VEREDELN

Bring deinen Datenschatz zu Tage – 14. November 2023



Tag der Forschungsdaten 2023

## Einführung in die Programmierung mit Python

Dr. Bernd Zey

Abteilung Forschungsdatenmanagement

[fdm@tu-dortmund.de](mailto:fdm@tu-dortmund.de)

<https://mailman.tu-dortmund.de/mailman/listinfo/fdm>



- Worum geht es heute hier?
  - Kurz-Einstieg in Python
  - Anwendungsfall
    - Tabellarische Daten laden und auswerten
    - Ergebnisse grafisch darstellen
    - Ergebnisse als Datei speichern
  - Keine Python-/Programmierkenntnisse erforderlich
- Motivation
  - Warum Python?
  - Entwicklungsumgebungen (IDE)
- Ausblick, Take-Home
- Live-Coding
  - JupyterLab

# Warum Python?



- Standard in Data Science
- Zahlreiche Module für die statistische Auswertung und Visualisierung (Open Source)
  - Pandas, NumPy, ...
  - Matplotlib, Seaborn, ...
- Einsteigerfreundliche Programmiersprache
  - Allgemeine Programmiersprache, Plattform-unabhängig
  - Viele (wissenschaftliche) Open Source-Projekte
  - Einfache Syntax
  - Objektorientierung, Prozedural, ...
  - Schnell umfangreiche Programme möglich
  - Einfaches Datei-Handling
  - Viele Module, u.a. für Machine Learning (TensorFlow, Keras, ...)
  - Online-Hilfe





# Warum Python? Was ist mit Tool XY?

## Alternativen

- KNIME

The screenshot displays the KNIME Analytics Platform interface. On the left, the 'Workflow Coach' panel lists recommended nodes with their community usage percentages:

Recommended Nodes	Community
Excel Reader	37%
CSV Reader	22%
Table Creator	9%
File Reader	7%
DB Reader	4%
DB Query Reader	3%
DB Table Selector	3%
Table Reader	2%
List Files/Folders	2%
File Reader (Complex Format)	2%
Create Date&Time Range	1%
Variable Creator	<1%
Microsoft SQL Server Connector	<1%
DB Connector	<1%
Microsoft Authentication	<1%
Read Excel Sheet Names	<1%
DB Table Creator	<1%
SQLite Connector	<1%
MySQL Connector	<1%
PostgreSQL Connector	<1%

The main workspace shows a workflow diagram with the following nodes:

- Node 1: CSV Reader
- Node 2: Box Plot
- Node 3: Image Writer (Port)
- Node 4: Line Plot
- Node 5: Image Writer (Port)
- Node 6: Conditional Box Plot
- Node 7: Image Writer (Port)
- Node 8: Statistics
- Node 9: CSV Writer

# Warum Py

## Alternativen

- KNIME
- R

The screenshot displays the RStudio interface. The top-left pane shows the R script code for loading the 'nycflights13' dataset, creating a 'daily' data frame, and generating a boxplot of flights per weekday. The top-right pane shows the environment with the 'daily' data frame containing 365 observations. The bottom-left pane shows the console output, including a preview of the 'daily' data frame and the execution of the boxplot code. The bottom-right pane shows the resulting boxplot titled 'Number of 2013 New York Flights Each Weekday', with the y-axis labeled 'Flights' and the x-axis labeled 'Weekday'.

```
1 library(nycflights13) ## package containing flights dataset
2 library(lubridate)
3 library(dplyr)
4 library(ggplot2)
5
6 head(flights, n = 3)
7 daily <- flights %>%
8   mutate(date = make_date(year, month, day)) %>%
9   count(date) %>%
10  mutate(wday = wday(date, label = TRUE))
11 head(daily, n = 3)
12 ggplot(daily, aes(wday, n)) +
13   geom_boxplot(outlier.colour = "hotpink") +
14   labs(x = "Weekday", y = "Flights",
15        subtitle = "Number of 2013 New York Flights Each Weekday")
16
17:1 | (Top Level) | R Script
```

```
~/Documents/Flights/
# A tibble: 3 x 19
  year month day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay carrier
<int> <int> <int> <int> <dbl> <int> <dbl> <chr> <int> <dbl> <chr>
1 2013 1 1 517 515 2 830 819 11 UA
2 2013 1 1 533 529 4 850 830 20 AA
3 2013 1 1 542 540 2 923 850 33 AA
# ... with 9 more variables: flight <int>, tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>,
# distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dtm>
> daily <- flights %>%
+   mutate(date = make_date(year, month, day)) %>%
+   count(date) %>%
+   mutate(wday = wday(date, label = TRUE))
> head(daily, n = 3)
# A tibble: 3 x 3
  date n wday
<date> <int> <ord>
1 2013-01-01 842 Tue
2 2013-01-02 943 Wed
3 2013-01-03 914 Thu
> ggplot(daily, aes(wday, n)) +
+   geom_boxplot(outlier.colour = "hotpink") +
+   labs(x = "Weekday", y = "Flights",
+        subtitle = "Number of 2013 New York Flights Each Weekday")
>
```

Number of 2013 New York Flights Each Weekday

Weekday	Flights (approx. median)
Sun	900
Mon	950
Tue	950
Wed	950
Thu	950
Fri	950
Sat	750

Screenshot von commons.wikimedia.org/



# Warum Python? Was ist mit Tool XY?

## Alternativen

- KNIME
- R
- Matlab (Octave)

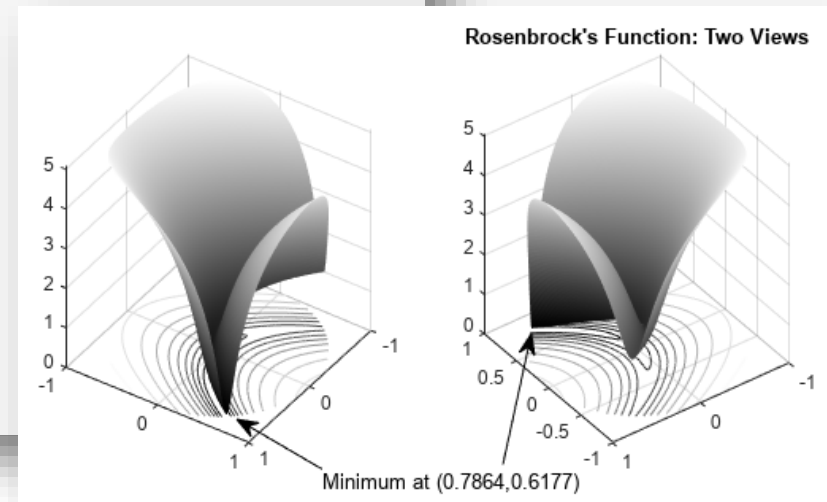
```
rosenbrock = @(x)100*(x(:,2) - x(:,1).^2).^2 + (1 - x(:,1)).^2; % Vectorized function

figure1 = figure('Position',[1 200 600 300]);
colormap('gray');
axis square;
R = 0:.002:1;
TH = 2*pi*(0:.002:1);
X = R'*cos(TH);
Y = R'*sin(TH);
Z = log(1 + rosenbrock([X(:),Y(:)]));
Z = reshape(Z,size(X));

% Create subplot
subplot1 = subplot(1,2,1,'Parent',figure1);
view([124 34]);
grid('on');
hold on;

% Create surface
surf(X,Y,Z,'Parent',subplot1,'LineStyle','none');

% Create contour
contour(X,Y,Z,'Parent',subplot1);
```



Screenshots von mathworks.com

# Warum Python? Was ist mit Tool XY?



## Alternativen

- KNIME
- R
- Matlab (Octave)
- Excel
- SPSS (PSPP)
- Stata
- Andere Programmiersprachen (C++, Java, ...)
- ...

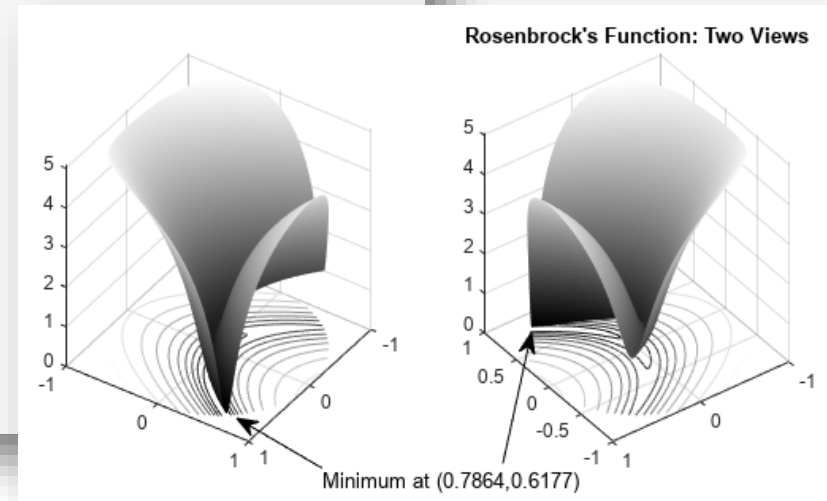
```
rosenbrock = @(x)100*(x(:,2) - x(:,1).^2).^2 + (1 - x(:,1)).^2; % Vectorized function

figure1 = figure('Position',[1 200 600 300]);
colormap('gray');
axis square;
R = 0:.002:1;
TH = 2*pi*(0:.002:1);
X = R'*cos(TH);
Y = R'*sin(TH);
Z = log(1 + rosenbrock([X(:),Y(:)]));
Z = reshape(Z,size(X));

% Create subplot
subplot1 = subplot(1,2,1,'Parent',figure1);
view([124 34]);
grid('on');
hold on;

% Create surface
surf(X,Y,Z,'Parent',subplot1,'LineStyle','none');

% Create contour
contour(X,Y,Z,'Parent',subplot1);
```



Screenshots von mathworks.com

- Entwick
- Jup

2023-11-pyth... - JupyterLab

conda: anaconda3

File Edit View Run Kernel Tabs Settings Help

Filter files by name

Name Last Modified

data 21 hours ago

2023-11-python-works... a minute ago

### Daten laden

- mit `read_csv()`
- wichtige Parameter der Funktion: `sep` und `decimal`

```
[187]: # Wichtig: für sep und decimal: vorher Datei checken, wie Zahlen formatiert und getrennt werden (mit Text-Editor öffnen)
data = pd.read_csv(filename, sep=";", decimal=",")
```

```
[188]: # data anzeigen
data
```

	wavelength	PTA	PTAAC	PTAAC after 5 Rec Runs	PTAAC after 20 Rec Runs
0	399.26491	61.88036	96.04368	98.79301	97.37103
1	401.19373	65.91014	94.78482	98.45279	95.97457
2	403.12254	66.66432	93.81900	98.18493	94.53716
3	405.05136	66.87038	92.93745	98.22182	93.39952
4	406.98018	66.34057	92.17670	98.40700	92.81974
...	...	...	...	...	...
1864	3992.64912	83.78279	100.01460	99.99696	100.06051
1865	3994.57794	83.83957	99.97816	100.00728	100.05586
1866	3996.50675	83.84588	99.92410	99.98987	100.04717
1867	3998.43557	83.83170	99.91610	99.98242	100.03501
1868	4000.36438	83.78698	99.94827	99.98046	100.02981

1869 rows × 5 columns

```
[ ]: # Datentypen von data
data.dtypes
```

```
[ ]: # Ausgabe in anderer Python-IDE mit print
print(data)
```

```
[ ]: # Excel-Datei (.xlsx) Laden und in data_xls speichern
filename_xls = os.path.join("data", "data-chem.xlsx")
```

Simple 0 1 Python 3 (ipykernel) | Idle

Mode: Command Ln 1, Col 7 2023-11-python-workshop-tdf-full-my.ipynb



# Entwickl

- Entwickl

- Jupyter

- Spyder

The screenshot displays the Spyder Python IDE interface. The main window is titled "Spyder (Python 3.9)". The top menu bar includes "File", "Edit", "Search", "Source", "Run", "Debug", "Consoles", "Projects", "Tools", "View", and "Help". The toolbar contains various icons for file operations and execution.

The code editor shows a Python script named "tdf-intro-python.py" with the following code:

```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import os
5 import pprint as pp
6
7
8
9 # In[ ]:
10
11 #filename = "Studium.txt"
12 filename = "C:/Users/[redacted]/spyder-workspace/data-chem.csv"
13 commentSign = "#"
14 separatorSign = ";"
15 decimalSign = ","
16 data = pd.read_csv(filename, sep=separatorSign, decimal=decimalSign, comment=commentSign)
17
18
19
20 # In[4]:
21 print(data)
22
23
24
25 #%%
26 data.iloc[:,1:4].boxplot()
27
28
```

The console window shows the execution of the script:

```
In [12]: runcell(0, 'C:/Users/[redacted]/spyder-workspace/tdf-intro-python.py')
In [13]: runcell([' ], 'C:/Users/[redacted]/spyder-workspace/tdf-intro-python.py')
In [14]: runcell(['[4]', 'C:/Users/[redacted]/spyder-workspace/tdf-intro-python.py')
      wavelength  PTA  ...  PTAAC after 5 Rec Runs  PTAAC after 20 Rec Runs
0      399.26491  61.88036  ...  98.79381  97.37183
1      401.12273  65.91014  ...  98.45279  95.97457
2      403.12254  66.66432  ...  98.18493  94.53716
3      405.05136  66.87938  ...  98.22182  93.39952
4      406.98018  66.34057  ...  98.40700  92.81974
...
1864  3992.64912  83.78279  ...  99.99696  100.06051
1865  3994.57794  83.83957  ...  100.00728  100.05586
1866  3996.50675  83.84588  ...  99.98987  100.04717
1867  3998.43557  83.83170  ...  99.98242  100.03501
1868  4000.36438  83.78698  ...  99.98046  100.02981

[1869 rows x 5 columns]
In [15]: runcell(3, 'C:/Users/[redacted]/spyder-workspace/tdf-intro-python.py')
In [16]:
```

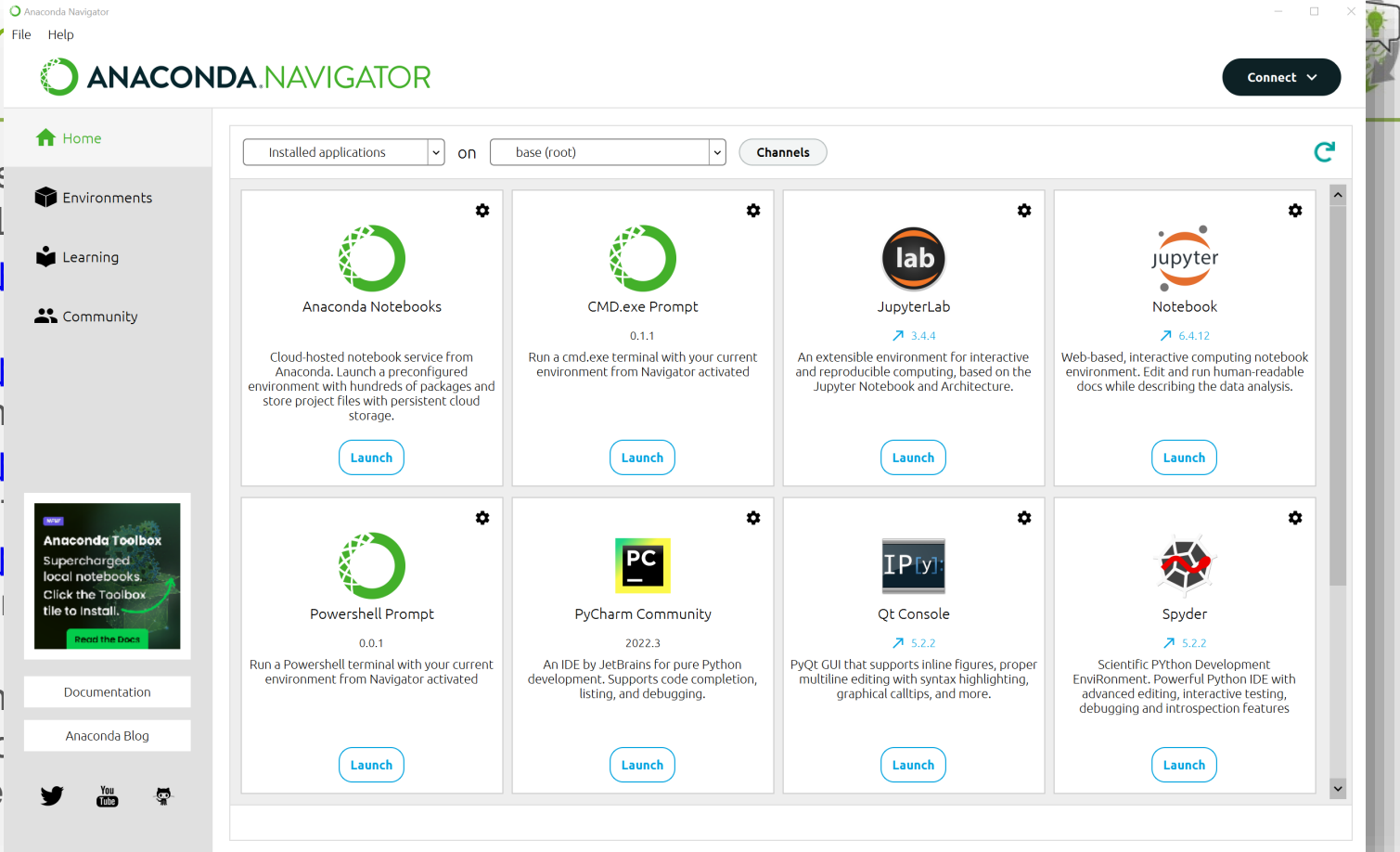
Two box plots are displayed side-by-side. The top plot shows the distribution of "wavelength" for three categories: "PTA", "PTAAC", and "PTAAC after 5 Rec Runs". The bottom plot shows the distribution of "wavelength" for three categories: "PTA", "PTAAC", and "PTAAC after 5 Rec Runs". The x-axis labels are "PTA", "PTAAC", and "PTAAC after 5 Rec Runs". The y-axis ranges from 40 to 110. The plots show that the "PTAAC" and "PTAAC after 5 Rec Runs" categories have higher median values and narrower distributions compared to the "PTA" category.



- Entwicklungsumgebung – IDE (Integrated Development Environment)
  - JupyterLab
    - <https://jupyter.org/>  Jupyter
  - Spyder
    - <https://www.spyder-ide.org/>  Spyder
  - PyCharm
    - <https://www.jetbrains.com/pycharm/> 
  - Visual Studio Code
    - <https://code.visualstudio.com/> 
- Mein Tipp zum Einstieg: Anaconda
  - Python-Distribution
  - Liefert mehrere IDEs
  - Anaconda Navigator
  - Installiert Python

# Entwicklung

- Entwicklung
  - Jupyterlab
    - <http://jupyterlab.com>
  - Spyder
    - <http://spyder-ide.org>
  - PyCharm
    - <http://pycharm.com>
  - Visual S
    - <http://visualstudio.com>
- Mein Tipp zu
  - Python-
  - Liefert n
  - Anacond
  - Installie





- Vorbereiteter Code und Live Coding
- JupyterLab
  - Abschnittsweise Ausführung
  - Markdown-Dokumentation und -Beschreibung
  - Direkte Ausgabe der Diagramme
- Code verfügbar auf Homepage
  - <https://fdm.tu-dortmund.de/informationen/tag-der-forschungsdaten-2023-einfuehrung-in-python/>
  - JupyterLab-Notebook
  - Lizenz: MIT
- Verwendete Daten basieren auf:
  - L. Hombach, A. K. Beine, “Carbon Supported Polyoxometalates as Recyclable Solid Acid Catalysts in Aqueous Reactions.”, 2023, doi: [10.22000/1106](https://doi.org/10.22000/1106)



Logo von <https://radar4chem.radar-service.eu/radar/de/home>

# Ausblick, Take-Home



- Python *einfache* Programmiersprache  
→ Automatisierung
- Kostenlos
- Mächtige Module (heute: Pandas) für Data Science
- Online gute Hilfe (Diskussionen, Code-Schnipsel)
  - z.B. <https://stackoverflow.com>
  - Websuche → „python pandas *thema*“
- Zum Einstieg: Anaconda  
<https://www.anaconda.com/download>
  - Entwicklungsumgebung JupyterLab (in Anaconda enthalten)

