Hydroinformatik - SoSe 2026

UW-BHW-414-06: Programmiersprache Python

Prof. Dr.-Ing. habil. Olaf Kolditz

[1]Helmholtz Centre for Environmental Research – UFZ, Leipzig

[2]Technische Universität Dresden – TUD, Dresden

[3]Center for Advanced Water Research – CAWR

[4]TUBAF-UFZ Center for Environmental Geosciences – C-EGS, Freiberg / Leipzig

Dresden, 15.05.2026

Technische Universität Dresden

Zeitplan: Hydroinformatik I+II

Sommersemester 2026: Stand: 06.04.2026

Nr.	KW	Datum	ID	Thema
01+02	16	17.04.2026	UW-BHW-414-01/02	Einführung in die Vorlesung, Umweltinformatik
03	16	17.04.2026	UW-BHW-414-03	Werkzeuge, Hello World (in C++)
05	17	24.04.2026	UW-BHW-414-04	Selbststudium: Software-Installationen
07	19	08.05.2026	UW-BHW-414-05	Objekt-Orientierte Programmierung: C++, Klassen
09	20	15.05.2026	UW-BHW-414-06	Programmiersprache Python
11	21	22.05.2026	UW-BHW-414-F	Modellierung, Digitalisierung, Wasser 4.0
00	22	29.05.2026		Vorlesungsfreie Woche
13	23	05.06.2026	UW-BHW-414-G	KI, Maschinelles Lernen, Neuronale Netzwerke
15	24	12.06.2026	UW-BHW-414-H	Kontinuumsmechanik, Hydromechanik
17	25	19.06.2026	UW-BHW-414-I	Differentialgleichungen, Näherungsverfahren
19	26	26.06.2026	UW-BHW-414-J	Finite-Differenzen, explizite Verfahren
21	27	03.07.2026	UW-BHW-414-K	Finite-Differenzen, implizite Verfahren
23	28	10.07.2026	UW-BHW-414-L	Gerinnehydraulik, Grundwasserhydraulik
25	29	17.07.2026	UW-BHW-414-M	Grundwasserhydraulik
27	30	24.07.2026	UW-BHW-414-N	Zusammenfassung, Klausurvorbereitung

Technische Universität Dresden

Fahrplan für heute ...

1. UW-BHW-414-06: Programmiersprache Python
   - Semesterplan
   - About
   - Installation
   - Basics
   - Grafik
2. Python
3. Jupyter
4. Übungen

Technische Universität Dresden

Python

- About
- Installation

Technische Universität Dresden

Python: About

- "Python is a programming language that lets you work more quickly and integrate your systems more effectively."
- Webseite: `https://www.python.org`
- Vorteil: funktioniert auf allen Rechnern (>> Demo)

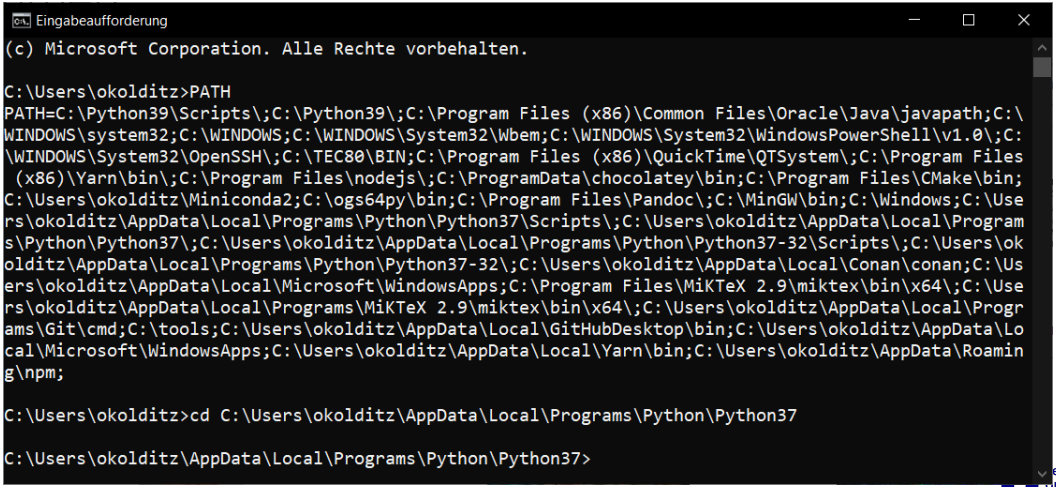
Technische Universität Dresden

Python: Installation

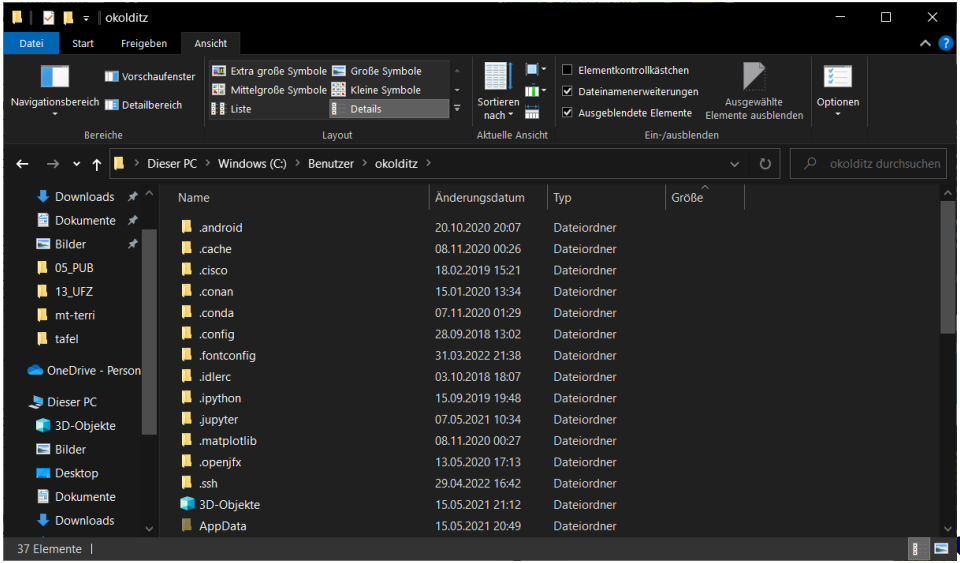
Technische Universität Dresden

Python: Installation

PATH



Python: Installation



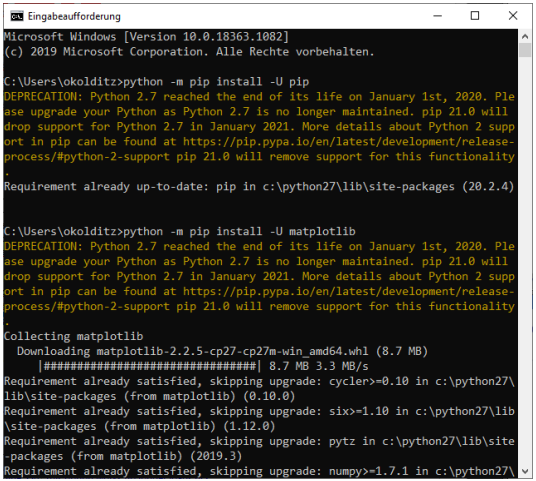
Technische Universität Dresden

Python: Module installieren

matplotlib

```
1 python -m pip install -U pip
2 python -m pip install -U matplotlib
```

Listing: Installieren von der Konsole



Python

- Basics

Technische Universität Dresden

Synthax

Import

Rechnen

Plotten

vgl. C++ Programme

Beispiel

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3  #-----------------
4  year = np.arange(11)
5  numbers = [1,1,7,4,8,7,6,7,3,2,3]
6  #-----------------
7  fig, ax = plt.subplots()
8  ax.set_title('Hydroinformatik I 2019 - Notenspiegel')
9  ax.set_ylabel('Anzahl von Noten')
10 plt.bar(year, numbers)
11 plt.xticks(year, ('1.0','1.3','1.7','...','5.0'))
12 plt.grid(True)
13 plt.show()
```

Listing: Python example for bar charts: Hydroinformatik-I-Noten.py

Technische Universität Dresden

Datentypen: EX05-data-types.py

Python-Basics

```python
#boolean
d = True
print(d)
type(d)
print(type(d))
print(d .__sizeof__())
#integer
a = 3
print(a)
type(a)
print(type(a))
print(a .__sizeof__())
#float
b = 3.1415
print(b)
type(b)
print(type(b))
print(b .__sizeof__())
#string
c = 'Hydroinformatik'
print(c)
type(c)
print(type(c))
id(c)
print(id(c))
print(c .__sizeof__())
```

Listing: Exercise: Data types: Python

```cpp
#include <iostream>
using namespace std;
int main()
{
  cout << "E04-data-types: Size of data types\n";
//  system("pwd");
  cout << "Type\tNumber of bytes\n";
  cout << "-----------------------\n";
  cout << "bool\t\t" << sizeof(bool) << endl;
  cout << "char\t\t" << sizeof(char) << "\n";
  cout << "short\t\t" << sizeof(short) << endl;
  cout << "int\t\t" << sizeof(int) << endl;
  cout << "long\t\t" << sizeof(long) << endl;
  cout << "float\t\t" << sizeof(float) << endl;
  cout << "double\t\t" << sizeof(double) << endl;
  cout << "long double\t" << sizeof(long double) <<
      endl;
  return 0;
}
```

Listing: Exercise: Data types: C++

Technische Universität Dresden

Felder anlegen

```
import numpy as np

#dynamische Felder
x = [] #leeres Feld
y = [] #leeres Feld

#statische Felder
year = np.arange(11) #x
numbers = [1,1,7,4,8,7,6,7,3,2,3] #y

#mehrdimensionale Felder
dim = 20
xarray = np.arange(-dim,dim)
yarray = np.arange(-dim,dim)
```

Listing: Python: Arrays

Felder füllen

```
for i in range(0,n):
    x.append(float(i)/float(n))
```

Listing: Python: Arrays

```python
#feste Schleife
t = [0.1,0.5,1.0,2.0]
for i in t:
    x.append(i*i)

#dynamische Schleife
for i in range(0,n):
    x.append(float(i)/float(n))
```

...

Listing: Python: Loops

Klassen: EX06-classes.py

Python-Basics

```python
class Student:
    def __init__(self, name, matrikel):
        self.name = name
        self.matrikel = matrikel

    def print_name(self):
        print("Name " + self.name)

    def print_matrikel(self):
        print("Matrikel " + str(self.
    matrikel))

    def print(self):
        self.print_name()
        self.print_matrikel()
```

Listing: Python: Klassen: Definition

```python
# Create and use instances

Suse = Student("Suse",
    192837465)
Suse.print()

Max = Student("Max",
    987654321)
Max.print()
```

Listing: Python: Klassen: Instanzen

Technische Universität Dresden

Plotting: Balkendiagramm

Python-Basics

```
1  import matplotlib.pyplot as plt
2
3  fig, ax = plt.subplots()
4  ax.set_title('Hydroinformatik I 2019 - Notenspiegel')
5  ax.set_ylabel('Anzahl von Noten')
6
7  plt.bar(marks, numbers)
8  plt.xticks(marks, ('1.0','1.3','1.7','2.0','2.3',...,'5.0'))
9  plt.grid(True)
10 plt.show()
```

...

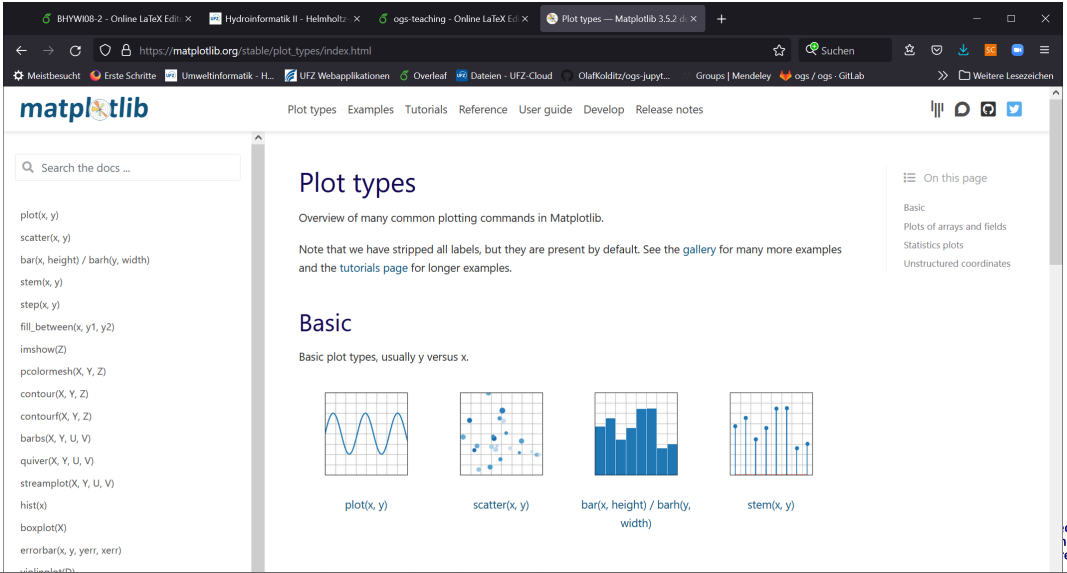
Listing: Python: Plotten

Technische Universität Dresden

Python: Grafik

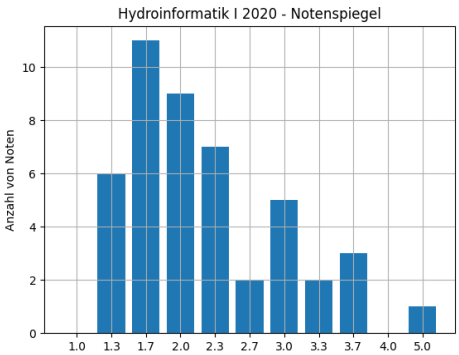
Technische Universität Dresden

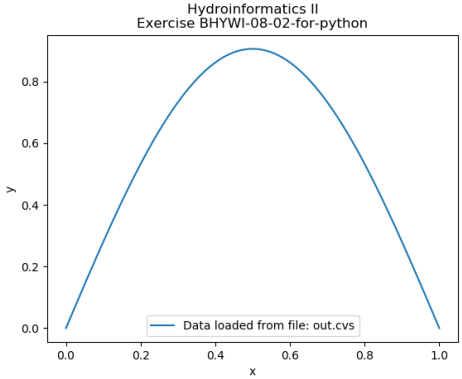
matplotlib https://matplotlib.org/



Python: Ploting (matplotlib)

Installation zusätzlicher Module: Seite 9

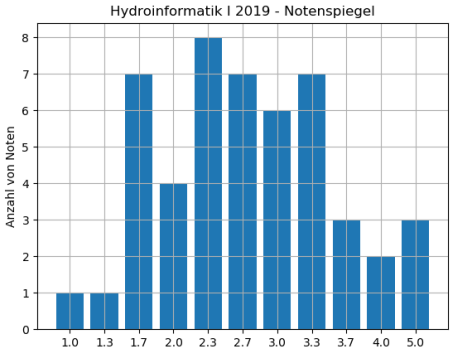




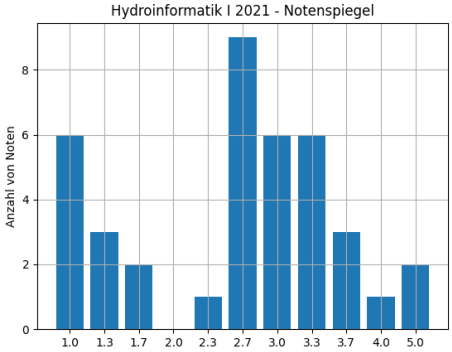
Technische Universität Dresden

Python: Ploting (matplotlib)

Normalverteilung (Gauss)



Bi-modale Verteilung



Technische Universität Dresden

Balkendiagramm (bar charts): EX07-bar-chart.py


```
1  from matplotlib.ticker import FuncFormatter
2  import matplotlib.pyplot as plt
3  import numpy as np
4  #-------
5  year = np.arange(11)
6  numbers = [1,1,7,4,8,7,6,7,3,2,3]
7  #-------
8  fig, ax = plt.subplots()
9  ax.set_title('Hydroinformatik I 2019 - Notenspiegel')
10 ax.set_ylabel('Anzahl von Noten')
11 plt.bar(year, numbers)
12 plt.xticks(year, ('1.0','1.3','1.7','2.0','2.3','2.7','3.0','3.3', '3.7',
       '4.0', '5.0'))
13 plt.grid(True)
14 plt.show()
```

Listing: Python example for bar charts: EX06-bar-chart.py

Technische Universität Dresden

Zusammenfassung

- Python: About und Installation (einschl. Module)
- Python: Basics
  - Datentypen
  - Felder (Arrays), Schleifen
  - Klassen: Definition und Instanzen
- Python: Plotten mit `matplotlib`
  - Balkendiagramm

$\Rightarrow$ Funktionen: Rechnen und Plotten

Technische Universität Dresden

Diffusionsgleichung: Analytische Lösung Tutorial: Abschnitt 2.2

- Repo: EX08-function
- C++ Variante
- C++ / Python Variante
- Python Variante

Technische Universität Dresden

MinGW + Python: Function plotter (analytical solution) #1

Übung: EX08-function

```
echo Compilation
g++ Ex08a-function.cpp
echo Execution
a.exe
echo Plotting
python EX08b-plot_data_from_file.py
echo End
```

Listing: Script file for entire workflow

Technische Universität Dresden

MinGW + Python: Function plotter (analytical solution) #2

Übung: EX08-function

```cpp
#include <cmath>
#include <fstream>
#define PI 3.14159265358979323846
int main(int argc, char *argv[])
{
  //1-Definitionen
  int numPoints = 1000;
  double x,y,alpha=1.,t=0.01;
  std::ofstream out_file;
  out_file.open("out.csv");
  //2-Berechnung
  for (int i = 0; i < numPoints+1; ++i)
  {
    x = double(i)/double(numPoints);
    y=sin(PI*x) * exp(-alpha*PI*PI*t);
    out_file << x << "," << y << std::endl;
  }
  //3-Ausgabe
}
```

Listing: C++ program for analytical solution

Technische Universität Dresden

Python: Function plotter (analytical solution) #3

Übung: EX08b-plot-data-from-file.py

```
1  import matplotlib.pyplot as plt
2  import csv
3
4  x = []
5  y = []
6  with open('out.csv','r') as csvfile:
7      plots = csv.reader(csvfile, delimiter=',')
8      for row in plots:
9          x.append(float(row[0]))
10         y.append(float(row[1]))
11
12 plt.plot(x,y, label='Data loaded from file: out.cvs')
13 plt.xlabel('x')
14 plt.ylabel('y')
15 plt.title('Hydroinformatics II\nExercise BHYWI-08-02-for-python')
16 plt.legend()
17 plt.savefig("diffusion-equation.png.png")
18 plt.show()
```

Listing: File reading and data plotting

Technische Universität Dresden

```python
import math
import matplotlib.pylab as plt
PI = 3.14159265358979323846
numPoints = 10
alpha = 1.0
t = [0.1,0.5,1.0,2.0]
x = []
y = []
for n in t:
    for i in range(0,numPoints+1):
        x.append(float(i)/float(numPoints))
        y.append(math.sin(PI*x[i]) * math.exp(-alpha*n*n))
    plt.plot(x,y,color='red',marker="o")
    x = []
    y = []
plt.xlabel('x')
plt.ylabel('u')
plt.savefig("diffusion-equation.png")
plt.show()
```

Listing: Analytical solution

KI Challenge

ChatGPT vs. DeepSeek

ChatGPT (OpenAI)

ChatGPT (Generative Pre-trained Transformer) is a large language model chatbot developed by OpenAI that can hold conversations and generate human-like text. It uses natural language processing and machine learning to understand and respond to user prompts in a variety of formats. ChatGPT can be used for tasks like answering questions, writing different types of content, translating languages, and even interacting with computers.

`https://chatgpt.com`

DeepSeek (open source AI)

DeepSeek is a Chinese AI startup (Liang Wenfeng) that has gained significant attention for developing competitive AI models at a fraction of the cost of established players like OpenAI. They have released open-source models like DeepSeek-R1, demonstrating their capabilities while also pushing the boundaries of AI development cost-effectiveness.
$>>$ cost and energy efficiency

`https://chat.deepseek.com/`

Technische Universität Dresden

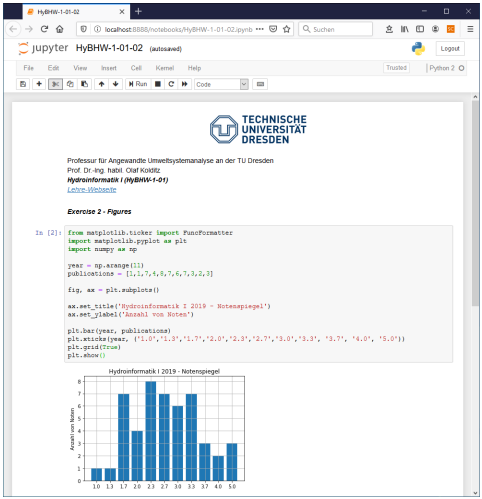
Jupyter

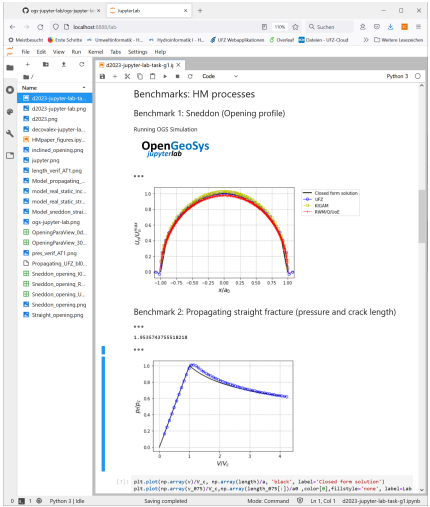
- Jupyter Notebook
- Jupyter Lab
- Browser-basiert

Jupyter

- "The Jupyter Notebook · The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, ..."
- Webseite: `https://jupyter.org/`
- Vorteil: funktioniert auf allen Rechnern
- ... ein Teil unserer (neuen) Übungen machen wir mit Jupyter Notebooks (>> Demo)

Jupyter: Example



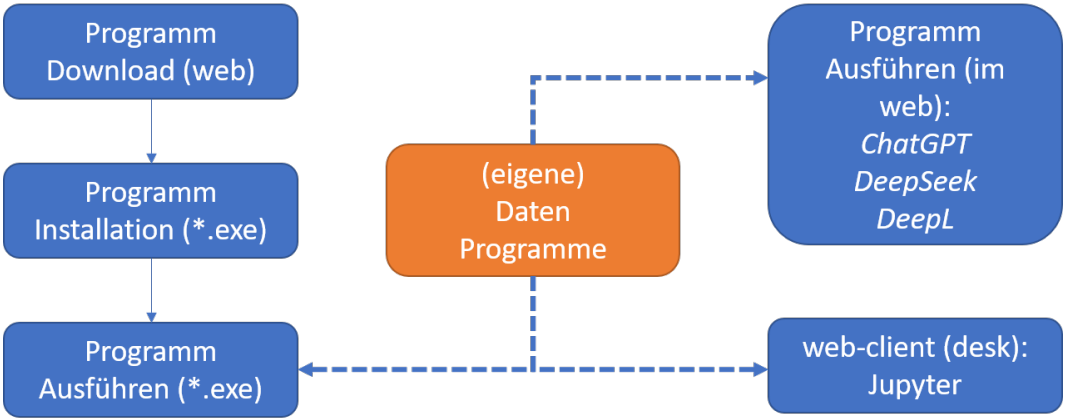


Jupyter: Installation

`https://jupyter.org/install`

Technische Universität Dresden

Desk und Web-Anwendungen



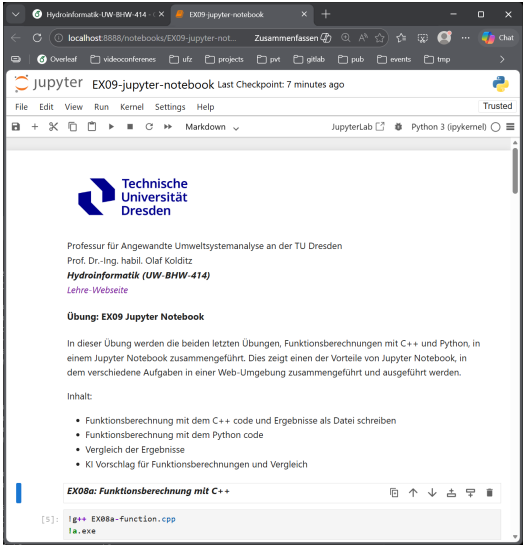
Technische Universität Dresden

Jupyter Notebook

- Zusammenführen von C++ und Python Funktionen

Zusammenführen von Funktionen (C++ und Python)

Jupyter Notebooks



Jupyter Notebooks:

- internet-fähig
- funktionieren zellenbasiert, Daten bleiben erhalten
- Beschreibung und Berechnungen, grafische Darstellungen
- ...

Technische Universität Dresden

Übungen
Informatik

Technische Universität Dresden

Übungen: Hydroinformatik

GitHub-Repo: `https://github.com/OlafKolditz/Hydroinformatik-2026`

- EX01: C++ ”Hello World”
- EX02: C++ Datentypen
- EX03: C++ Klassen
- EX04: C++ Datenbank
- EX05: Python Datentypen
- EX06: Python Klassen
- EX07: Python Balkendiagramme
- EX08: Funktionen: C++, Python
- EX09: Jupyter Notebooks
- ...

Technische Universität Dresden

Übungen
Mechanik

Technische Universität Dresden

Übungen: Hydroinformatik

GitHub-Repo: `https://github.com/OlafKolditz/Hydroinformatik-2026`

- EX20: Jupyter Notebook
- EX21: Kontinnumsmechanik: Skalarprodukt
- EX22: Hydromechanik: Divergenzfreie Strömung
- EX23: Analytische Lösung: Elliptische Gleichung
- EX24: Analytische Lösung: Parabolische Gleichung (Diffusion)
- EX25: Analytische Lösung: Transportgleichung (ADE)
- EX26: Finite-Differenzen-Methode (FDM) explizit
- EX27: Finite-Differenzen-Methode (FDM) implizit
- EX28: Gerinnehydraulik

Technische Universität Dresden