

Workshop

Workshop-Themen

- Python-Entwicklungsumgebungen:
 - Spyder
 - Jupyter
- Branchensoftware mit Python-Schnittstelle:
 - Blender
 - optional: FreeCAD
- CAMPLA/Lernstick

Spyder

Python-Entwicklungsumgebung

Zur Erstellung von Python-Programmen genügen eigentlich einfache Texteditoren, wie wir sie schon kennengelernt haben, z.B. `vi`, `nano` oder `gedit`. Ein deutlich höheres Komfort-Level und erweiterte Möglichkeiten bieten jedoch so genannte «Integrierte Entwicklungsumgebungen», die dann schon bei der Eingabe des Python-Quelltextes Fehler erkennen oder automatische Vervollständigungen anbieten.

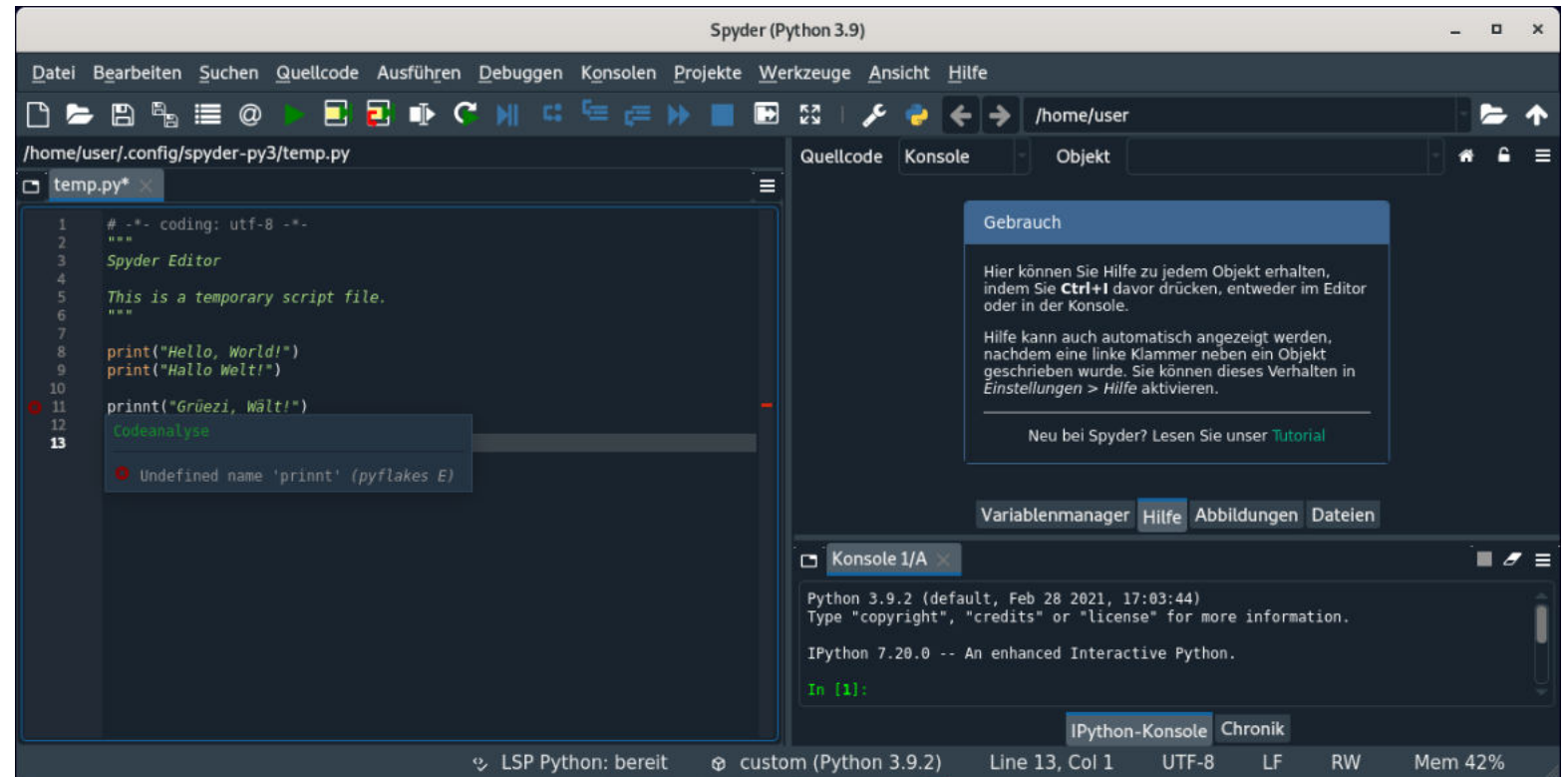
Eine relativ einfache Python-Entwicklungsumgebung ist die «Spyder IDE», siehe <https://www.spyder-ide.org>.

In der Lernumgebung können Sie einfach das Softwarepaket `spyder3` nach-installieren.

Starten Sie dann das Programm via Desktopumgebung (Dock → Anwendungen anzeigen), ignorieren eventuelle Hinweise zu Updates und stellen die Sprache via «Tools → Preferences → General → Advanced Settings» auf Deutsch um.

Unter Windows und macOS laden Sie das Installationsprogramm von folgender Seite herunter:

<https://github.com/spyder-ide/spyder/releases>



Jupyter-Einführung

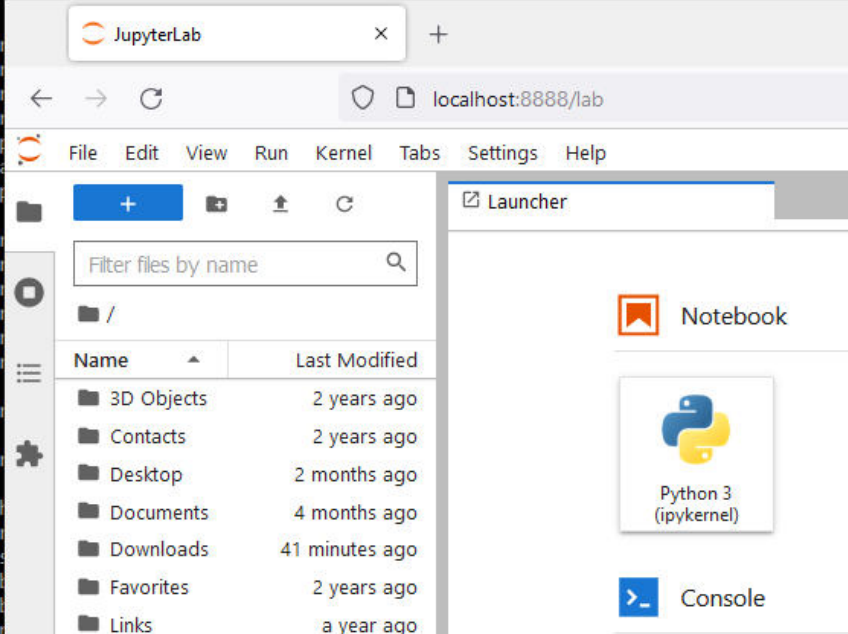
JupyterLab via Python-Standardinstallation (Windows)

- 1) Startmenü ► Windows-System ► Eingabeaufforderung starten und den Befehl `pip install jupyterlab` eingeben.
- 2) Zum Starten von Jupyterlab kann der Befehl `jupyter-lab` eingegeben werden.

```
Eingabeaufforderung - pip install jupyterlab
C:\Users\Anwender> pip install jupyterlab
Collecting jupyterlab
  Downloading jupyterlab-3.3.0-py3-none-any.whl (8.7 MB)
    |-----| 8.7 MB 2.2 MB/s
Collecting packaging
  Downloading packaging-21.3-py3-none-any.whl (40 kB)
    |-----| 40 kB 2.5 MB/s
Collecting jupyter-core
  Downloading jupyter_core-4.9.2-py3-none-any.whl (86 kB)
    |-----| 86 kB 2.8 MB/s
Collecting ipython
  Downloading ipython-8.1.1-py3-none-any.whl (750 kB)
    |-----| 750 kB 3.3 MB/s
Collecting jupyter-server==1.4
```

```
Eingabeaufforderung - jupyter-lab
Microsoft Windows [Version 10.0.19044.1415]
(c) Microsoft Corporation. Alle Rechte vorbehalten.
C:\Users\Anwender> jupyter-lab
[I 2022-03-05 22:41:38.957 ServerApp]
[I 2022-03-05 22:41:38.950 ServerApp]
[I 2022-03-05 22:41:39.544 ServerApp]
[I 2022-03-05 22:41:39.606 ServerApp]
[I 2022-03-05 22:41:39.606 LabApp]
on310\lib\site-packages\jupyterlab
[I 2022-03-05 22:41:39.606 LabApp]
python310\share\jupyter\lab
[I 2022-03-05 22:41:39.606 ServerApp]
[I 2022-03-05 22:41:39.625 ServerApp]
[I 2022-03-05 22:41:39.625 ServerApp]
[I 2022-03-05 22:41:39.625 ServerApp]
[I 2022-03-05 22:41:39.625 ServerApp]
6b
[I 2022-03-05 22:41:39.625 ServerApp]
ation).
[C 2022-03-05 22:41:39.669 ServerApp]

To access the server, open the
file:///C:/Users/Anwender
Or copy and paste one of these
http://localhost:8888/lab
or http://127.0.0.1:8888/lab
[W 2022-03-05 22:41:45.149 LabApp]
```



The screenshot shows the JupyterLab web interface in a browser window. The address bar displays `localhost:8888/lab`. The interface includes a menu bar (File, Edit, View, Run, Kernel, Tabs, Settings, Help), a file browser on the left with a search bar and a table of files, and a launcher on the right with options for Notebook and Console. The file browser table is as follows:

Name	Last Modified
3D Objects	2 years ago
Contacts	2 years ago
Desktop	2 months ago
Documents	4 months ago
Downloads	41 minutes ago
Favorites	2 years ago
Links	a year ago

JupyterLab via Python-Standardinstallation (Linux)

In der Lernumgebung kann JupyterLab via Kommandozeile wie folgt installiert werden:

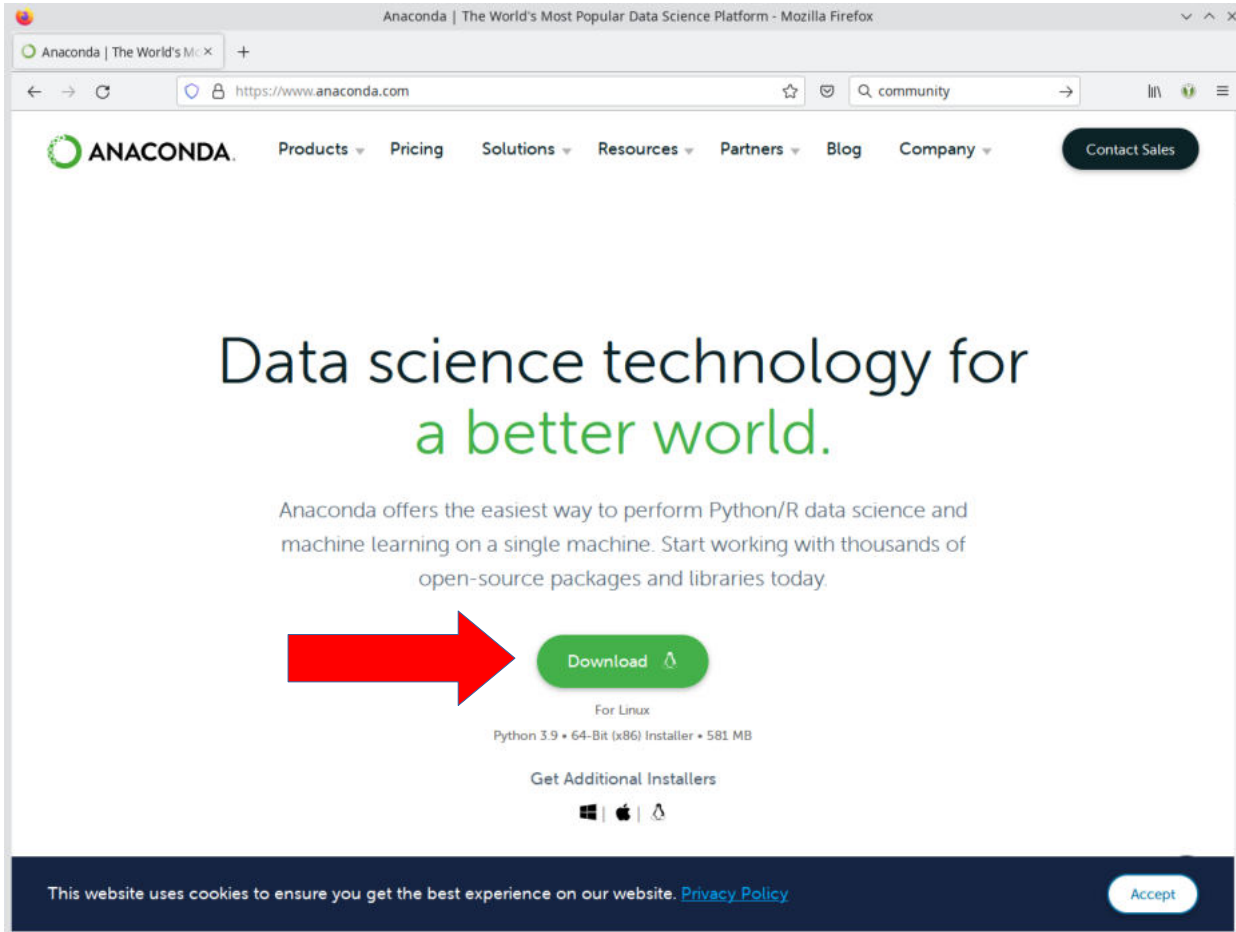
```
sudo apt install python3-pip
```

```
sudo pip3 install jupyterlab
```

Der Start erfolgt dann durch die Eingabe des Befehls:

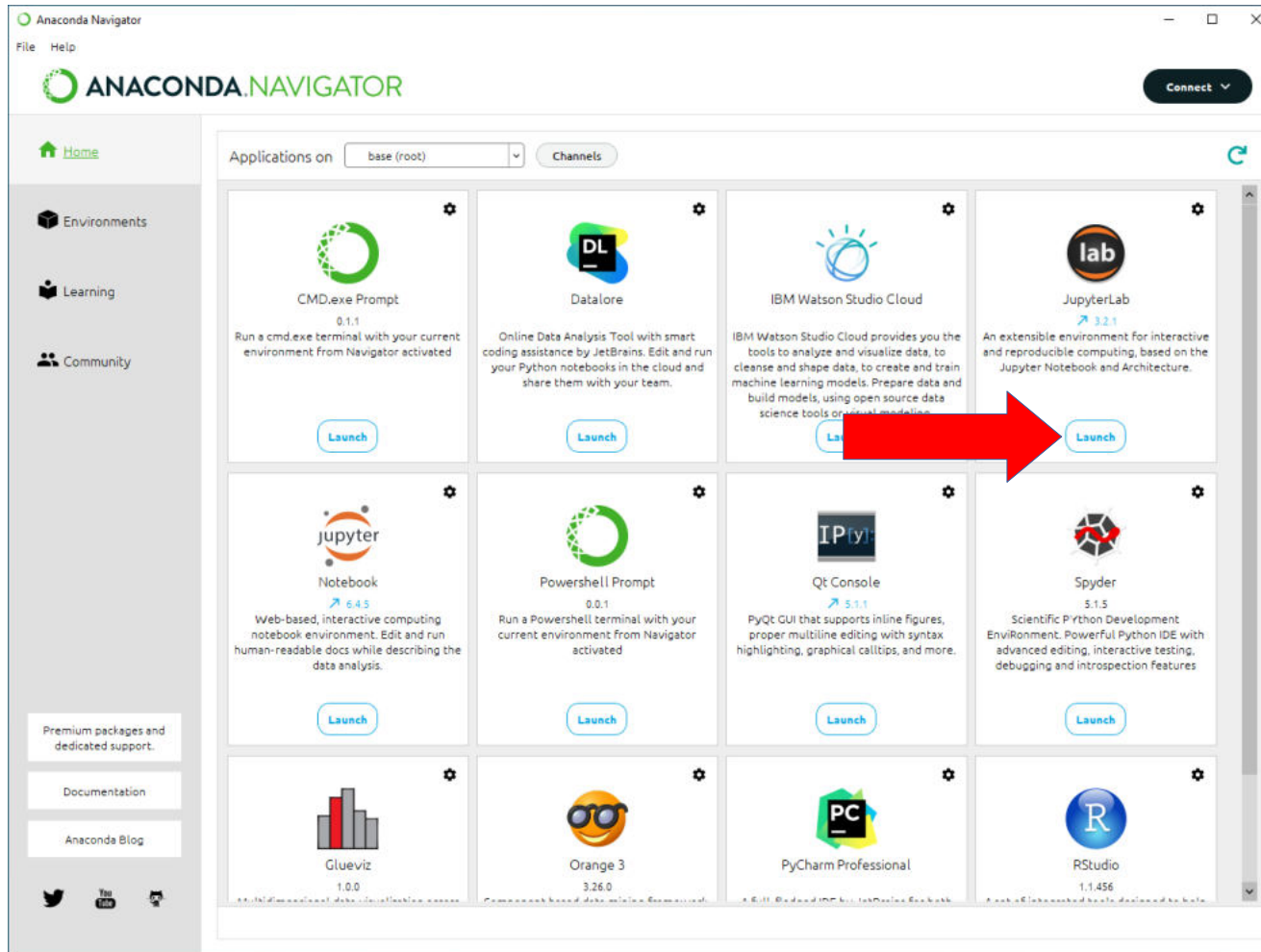
```
jupyter-lab
```


JupyterLab via Anaconda



- 1) das Installationsprogramm herunterladen
- 2) das Installationsprogramm starten
- 3) die Installation und unter Beibehaltung aller Voreinstellungen durchführen

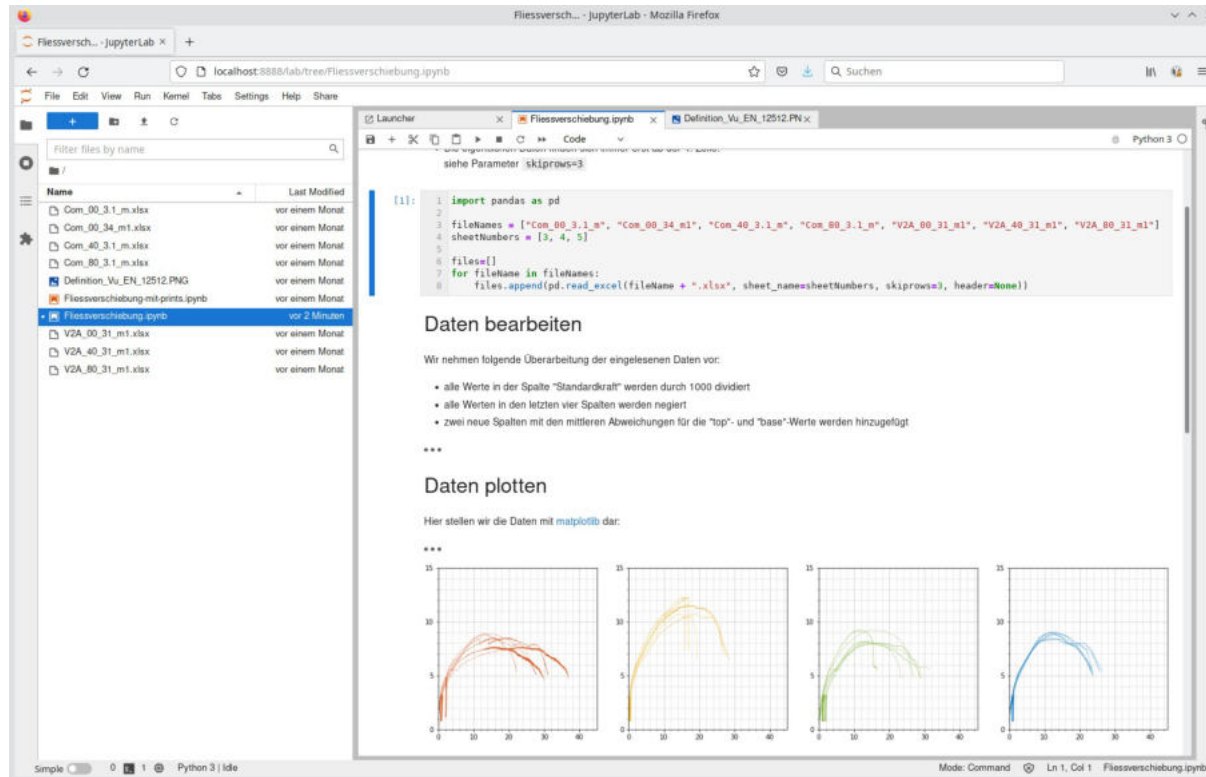
JupyterLab via Anaconda



1) über das Startmenü
«Anaconda Navigator» starten

2) JupyterLab starten

JupyterLab



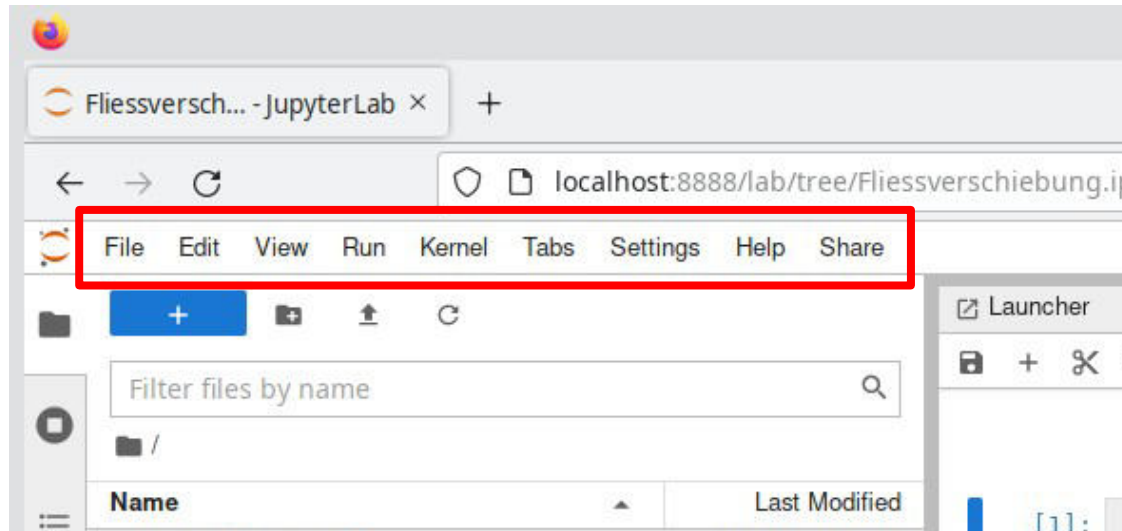
The screenshot displays the JupyterLab web interface in a Mozilla Firefox browser. The address bar shows the local host path: localhost:8888/lab/tree/Flieverschiebung.ipynb. The interface is divided into several sections:

- File Browser (Left):** A sidebar showing a file tree with folders and files. The current file, 'Flieverschiebung.ipynb', is selected and highlighted in blue. Other files include 'Com_00_3_1_m.xlsx', 'Com_00_34_m1.xlsx', 'Com_40_3_1_m.xlsx', 'Com_80_3_1_m.xlsx', 'Definition_Vu_EN_12512.PNG', and 'Flieverschiebung-m1-printa.ipynb'.
- Code Editor (Center):** A code cell containing Python code for data processing. The code imports pandas, defines file names and sheet numbers, and uses a loop to read and append data from multiple Excel files. The code is as follows:

```
[1]: 1 import pandas as pd
2     filenames = ["Com_00_3_1_m", "Com_00_34_m1", "Com_40_3_1_m", "Com_80_3_1_m", "V2A_00_31_m1", "V2A_40_31_m1", "V2A_80_31_m1"]
3     sheetNumbers = [5, 4, 5]
4
5
6     files=[]
7     for fileName in filenames:
8         files.append(pd.read_excel(fileName + ".xlsx", sheet_name=sheetNumbers, skiprows=3, header=None))
```
- Text and Plot Area (Bottom):** Below the code, there is a section titled 'Daten bearbeiten' (Data processing) with a list of instructions: 'Wir nehmen folgende Überarbeitung der eingelesenen Daten vor:' followed by three bullet points: 'alle Werte in der Spalte "Standardkraft" werden durch 1000 dividiert', 'alle Werten in den letzten vier Spalten werden negiert', and 'zwei neue Spalten mit den mittleren Abweichungen für die "top"- und "base"-Werte werden hinzugefügt'. Below this is a section titled 'Daten plotten' (Data plotting) with the text 'Hier stellen wir die Daten mit matplotlib dar:' followed by four line plots. Each plot shows a curve of data points on a grid, with the x-axis ranging from 0 to 40 and the y-axis from 0 to 25. The plots are color-coded: orange, yellow, green, and blue.

- moderne, flexible und modulare Benutzeroberfläche
- Bearbeitung von Jupyter-Notebooks und vielen weiteren Dateiformaten
- Zusammenstellung und Konfiguration von Data-Science-Workflows

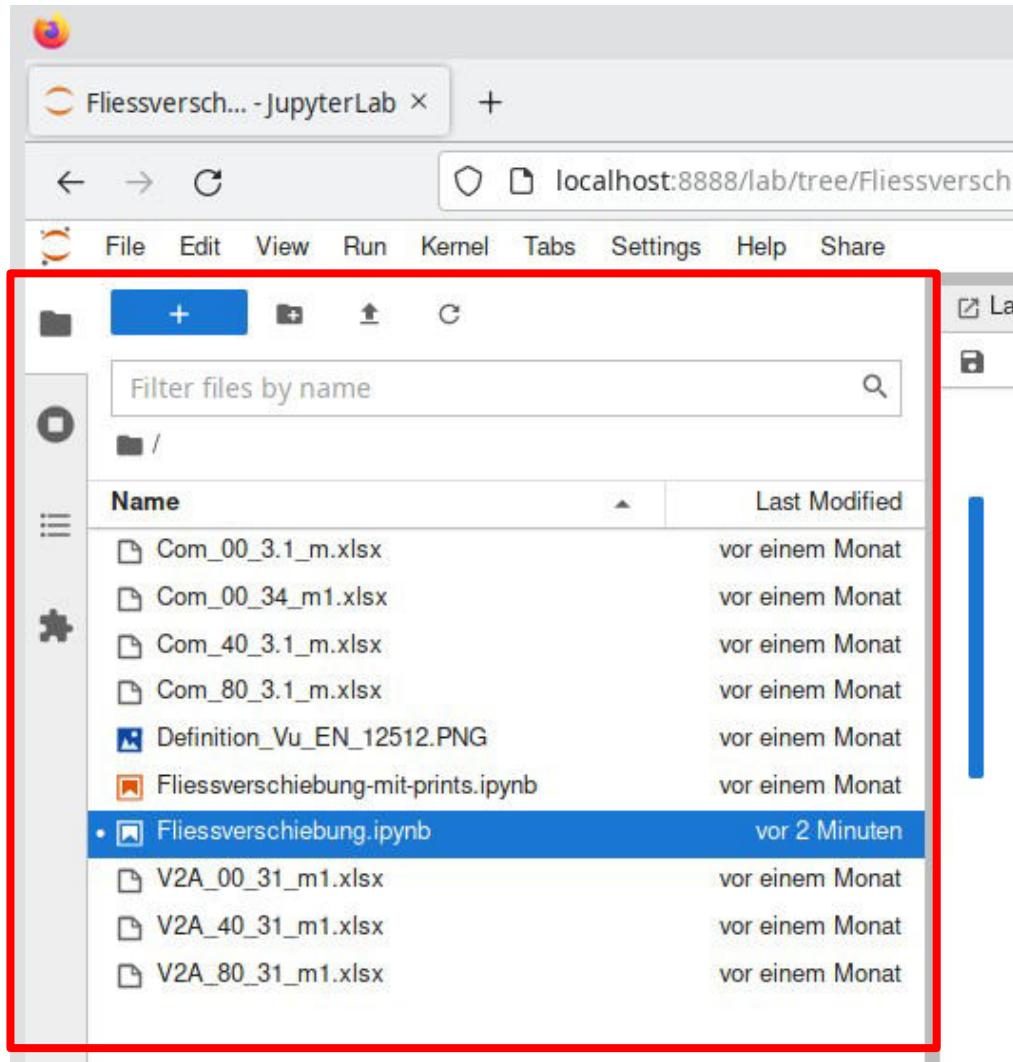
JupyterLab: Menü



- Das **Menü** beinhaltet die wichtigsten Einträge zu:
- Dateioperationen
 - Bearbeitung von Dokumenten
 - Anpassung des Aussehens von JupyterLab
 - Ausführung von verschiedenen Zellen
 - Verwaltung von verschiedenen Kernen
 - Wechseln zwischen verschiedenen Dokumenten und Aktivitäten
 - Einstellungen
 - Hilfe

JupyterLab-Erweiterungen können das Menü um weitere Einträge ergänzen (z. B. im Screenshot der Eintrag «Share» ganz rechts).

JupyterLab: Seitenleiste

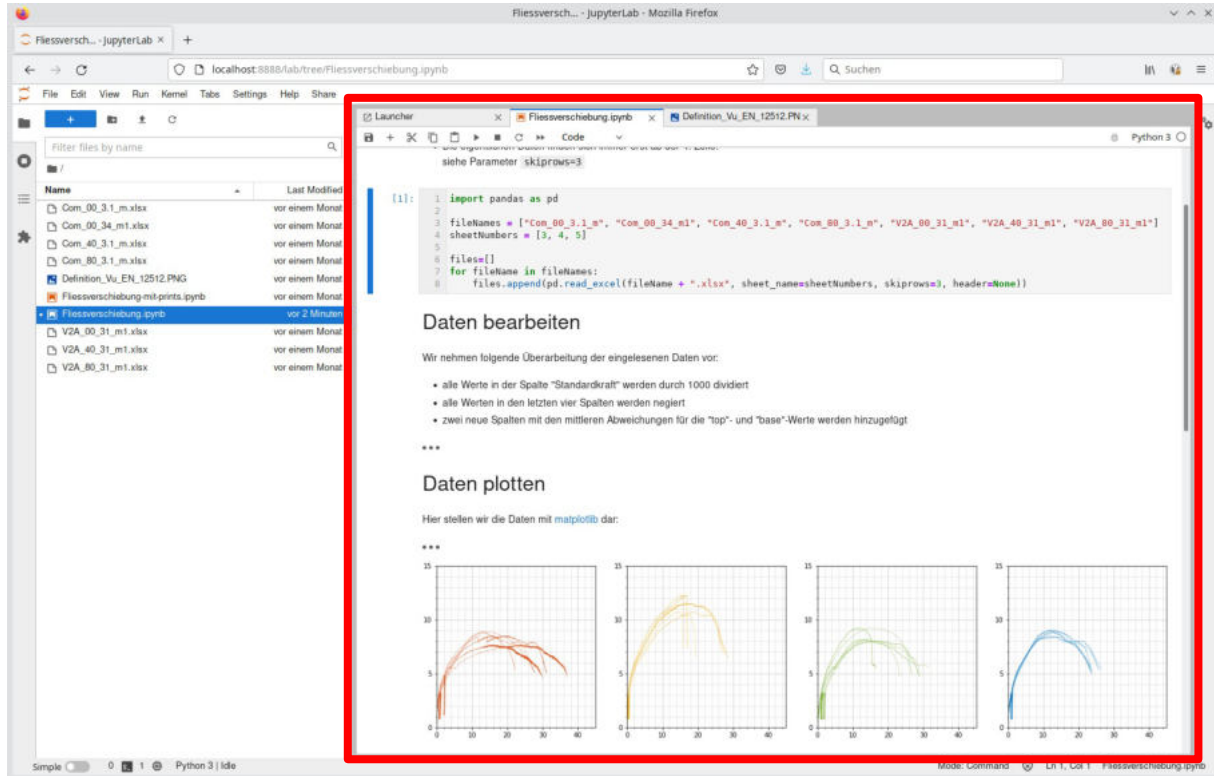


In der **Seitenleiste** befindet sich:

- ein Dateibrowser
- eine Übersicht über alle Dokumente, Kernel und Terminals
- ein Inhaltsverzeichnis zur schnellen Navigation durch ein Jupyter-Notebook
- die Verwaltung der JupyterLab-Erweiterungen

Die Seitenleiste kann durch Anklicken der Icons auf der linken Seite auf- und zugeklappt werden.

JupyterLab: Arbeitsbereich



Im Arbeitsbereich werden

- alle geöffneten Dokumente (Notebooks, Textdateien, Bilder, ...)
- Terminals
- Konsolen

... in sogenannten «Tabs» dargestellt, die sehr flexibel per Drag & Drop oder Kontextmenüs (rechte Maustaste) umsortiert, geteilt und angeordnet werden können.

JupyterLab: Arbeiten innerhalb eines Notebooks

<https://jupyterlab.readthedocs.io/en/stable/user/notebook.html>

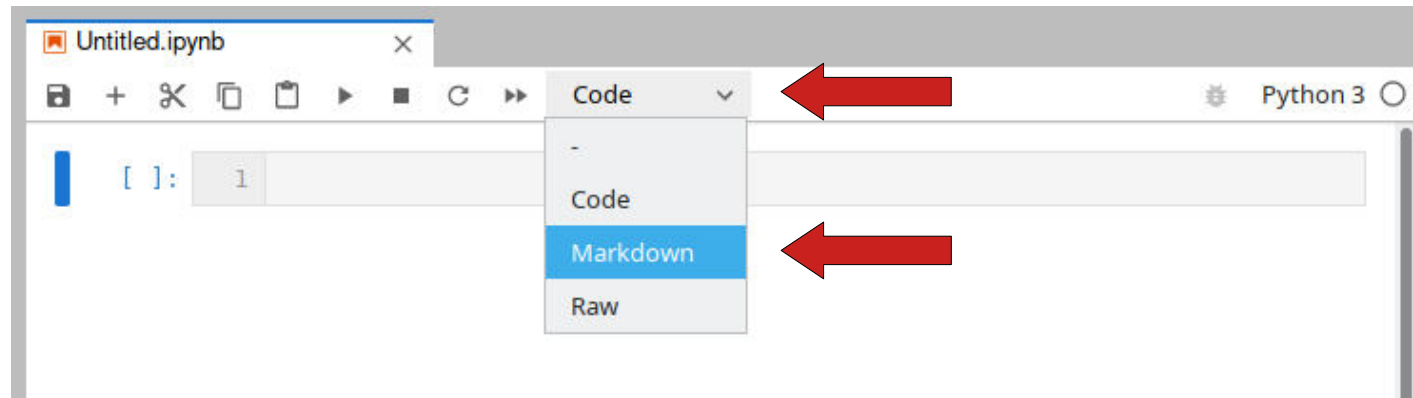
- Neues Notebook anlegen
- Zellen
 - erzeugen
 - navigieren mit Pfeiltasten
 - kopieren (c), ausschneiden (x), einfügen (v)
 - auf- und zuklappen
 - per Drag & Drop umsortieren
 - per Drag & Drop in andere Notebooks kopieren
- synchronisierte Ansichten des gleichen Dokuments erzeugen
- synchronisierte Ansichten für Ausgaben von Zellen erzeugen

Markdown-Zellen

Wenn neue Zellen angelegt werden, sind sie per Voreinstellung «Code-Zellen». Das sind Zellen, in denen Programmcode abgelegt wird, der durch den zugehörigen Kernel (z.B. Python) ausgeführt werden kann.

Für reichhaltig formatierte Kommentare und Erklärungen können in einem Jupyter-Notebook sogenannte «**Markdown-Zellen**» verwendet werden.

Der Typ einer jeden Zelle kann über das Dropdown-Menü am oberen Rand eines jeden Notebooks umgestellt werden:

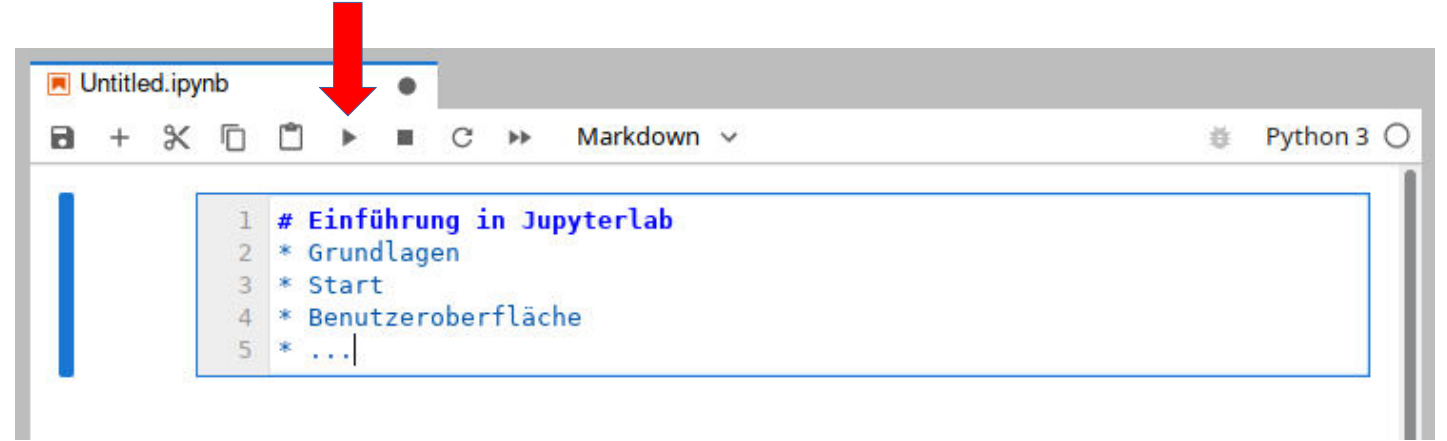


(Zellen vom Typ «Raw» werden von Jupyter-Notebooks gar nicht ausgewertet. Sie können für besondere Wünsche beim Exportieren von Notebooks in andere Formate verwendet werden, was wir jedoch hier nicht weiter betrachten werden.)

Markdown-Zellen

Markdown ist eine sehr einfache Auszeichnungssprache, um Texte zu gestalten. Eine gute Übersicht dazu ist im entsprechenden Wikipedia-Artikel zu finden, siehe:

<https://de.wikipedia.org/wiki/Markdown>



Aufgabe:

Ändern Sie den Typ einer Zelle auf `Markdown` und fügen Sie einen einfachen, jedoch mit Markdown formatierten, Text ein (siehe Wikipedia-Artikel oder Beispiele auf nächster Folie).

Eine Zelle lässt sich unter anderem mit dem Play-Icon (siehe Pfeil ► im Screenshot oben), über das Play-Menü oder die Tastenkombination `Shift+Enter` ausführen.

Wenn Sie die Markdown-Zelle ausführen, wird als Ergebnis der formatierte Text angezeigt.

Kleiner Tipp:

Mit `Ctrl+Enter` wird die Zelle ausgeführt, ohne dass auf die nächste Zelle gesprungen oder eine neue Zelle angelegt wird.

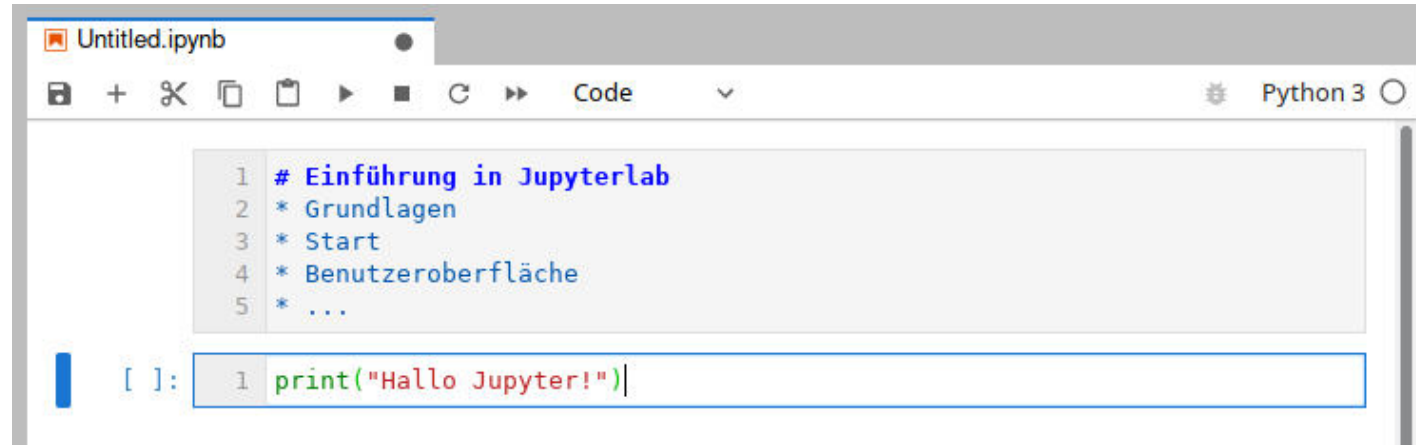
Markdown-Beispiele

```
1 # Überschrift 1
2 ## Überschrift 2
3 ### Überschrift 3, usw.
4
5 **Fett**
6
7 *Kursiv*
8
9 Zitat:
10 > Wer Wind säht, wird Sturm ernten.
11
12 horizontale Linie:
13 - - -
14
15 Das hier ist `Inline-Quelltext`, der innerhalb einer Zeile eingefügt wird.
16
17 Hier ist ein ganzer Code-Block:
18 ```python
19 message="Hello World!"
20 print(message)
21 ```
22 - das hier ist eine
23 - unnummerierte Liste
24
25 1. und hier
26 2. ist noch eine
27 3. nummerierte Liste
28
29 ein Hyperlink zur \[BFH\]\(https://www.bfh.ch\)
30
31 und der Link zu einem Bild:
32 !\[BFH-Logo\]\(https://upload.wikimedia.org/wikipedia/commons/a/a2/BFH\_Logo\_deutsch.png\)
```

Links sehen Sie ein paar typische Beispiele für mit Markdown formatierte Texte.

Das Ergebnis wird erst nach der Ausführung der Zelle (z.B. mit Shift+Enter) angezeigt.

Einschub: Edit- und Command-Mode



The screenshot shows a JupyterLab notebook window titled "Untitled.ipynb". The window has a toolbar with icons for file operations and a "Code" dropdown menu. The notebook content is displayed in a light gray editor area. The first cell contains a list of items:

```
1 # Einführung in Jupyterlab
2 * Grundlagen
3 * Start
4 * Benutzeroberfläche
5 * ...
```

The second cell is currently in Edit Mode, indicated by a blue border around the input area. It contains the following code:

```
[ ]: 1 print("Hallo Jupyter!")
```

Vielleicht ist es Ihnen schon aufgefallen: Die Zelle, die Sie gerade bearbeiten, bekommt einen kleinen blauen Rahmen. Dann befindet sich die Zelle im so genannten **Edit-Mode**, wobei alle Tastatureingaben dem Inhalt der Zelle hinzugefügt werden.

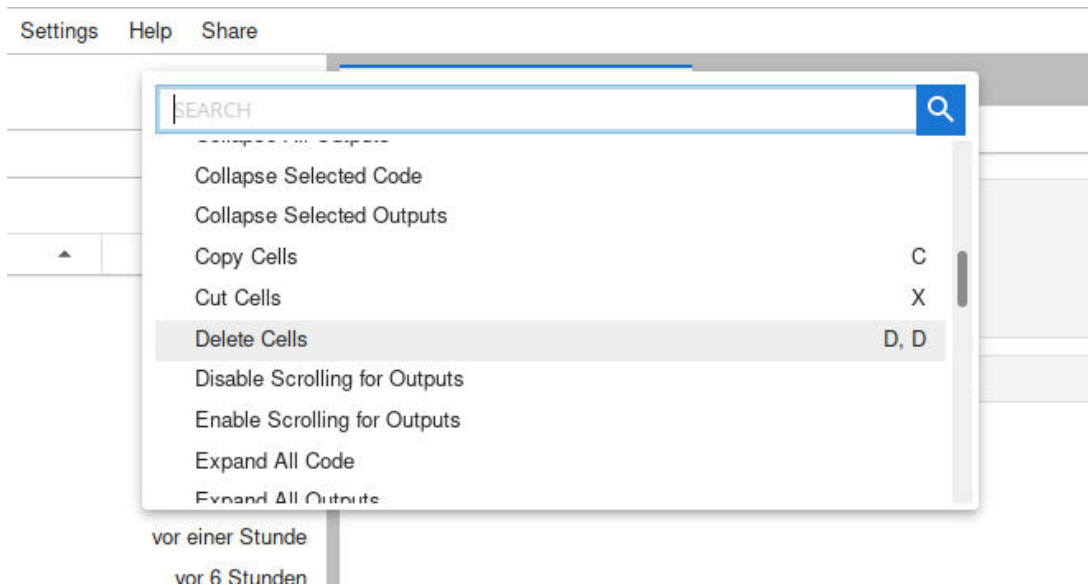
Der Edit-Mode kann durch Drücken der ESC-Taste beendet werden. Dadurch wird in den **Command-Mode** gewechselt, wobei dann alle Tastatureingaben als *JupyterLab-Befehle* interpretiert werden.

Einschub: Edit- und Command-Mode



Welche Befehle und entsprechende Tastaturkürzel zur Verfügung stehen, können Sie sich durch den Aufruf von

View ▶ Activate Command Palette anzeigen lassen.



Aufgaben:

Wechseln Sie in den Command-Mode und fügen Sie unter der aktuellen Zelle eine neue Zelle über das entsprechende Tastaturkürzel hinzu (siehe Command Palette!).

Wechseln Sie mit dem entsprechenden Tastaturkürzel in den Edit-Mode und fügen Sie der neuen Zelle einen kurzen, beliebigen Inhalt hinzu. Löschen Sie die neue Zelle über das entsprechende Tastaturkürzel.

Mathematische Formeln

Mathematische Formeln können in Markdown-Zellen mitten im Text zwischen zwei Dollar-Zeichen als LaTeX-Formeln eingegeben werden, siehe z.B.: <https://en.wikibooks.org/wiki/LaTeX/Mathematics>

Soll eine Formel auf einer Zeile allein dargestellt werden, müssen doppelte Dollar-Zeichen verwendet werden:

- 1 Die Formel `$M_{max}=q \cdot l^2/8$` bedeutet, dass das maximale Biegemoment eines Einfeldträgers mit Gleichlast in Feldmitte `M_{max}` dadurch ermittelt werden kann, indem das Produkt der Streckenlast `q` mit dem Quadrat der Stützweite des Trägers `l` durch 8 dividiert wird.
- 2
- 3 Die Formel kann selbstverständlich auch schöner und allein auf einer Zeile dargestellt werden: `$M_{max}=\frac{q \cdot l^2}{8}$`

Aufgabe:

Geben Sie eine andere Ihnen bekannte Formel in einer Markdown-Zelle ein und lassen Sie sie durch «Ausführen» der Zelle darstellen.

Die Formel $M_{max} = q \cdot l^2/8$ bedeutet, dass das maximale Biegemoment eines Einfeldträgers mit Gleichlast in Feldmitte M_{max} dadurch ermittelt werden kann, indem das Produkt der Streckenlast q mit dem Quadrat der Stützweite des Trägers l durch 8 dividiert wird.

Die Formel kann selbstverständlich auch schöner und allein auf einer Zeile dargestellt werden:

$$M_{max} = \frac{q \cdot l^2}{8}$$

Codezellen:

Führen Sie ein paar einfache Python-Befehle einzeln oder gemeinsam in verschiedenen Jupyter-Codezellen aus.

Anregungen:

```
print("Hello, World!")
```

```
print(42)
```

```
a = 13
```

```
a += 2
```

```
print(a)
```

Zellennummerierung

Vor den Codezellen erscheinen sich ständig ändernde Nummern. Diese zeigen die Ausführungsreihenfolge der jeweilige Codezellen an:

wurde als 30. Codezelle ausgeführt 

```
[30]: 1 a = 13  
      2 a += 2  
      3 print(a)
```

15

wurde als 29. Codezelle ausgeführt
(also **vor** der oberen Zelle!) 

```
[29]: 1 fruits = ["Äpfel", "Bananen", "Kirschen"]  
      2 for x in fruits:  
      3     print("Hmm, lecker " + x + "!")  
      4     print("Gibt's noch mehr?")  
      5 print("Ach schade, nichts mehr da.")
```

```
Hmm, lecker Äpfel!  
Gibt's noch mehr?  
Hmm, lecker Bananen!  
Gibt's noch mehr?  
Hmm, lecker Kirschen!  
Gibt's noch mehr?  
Ach schade, nichts mehr da.
```

wird **soeben** ausgeführt 

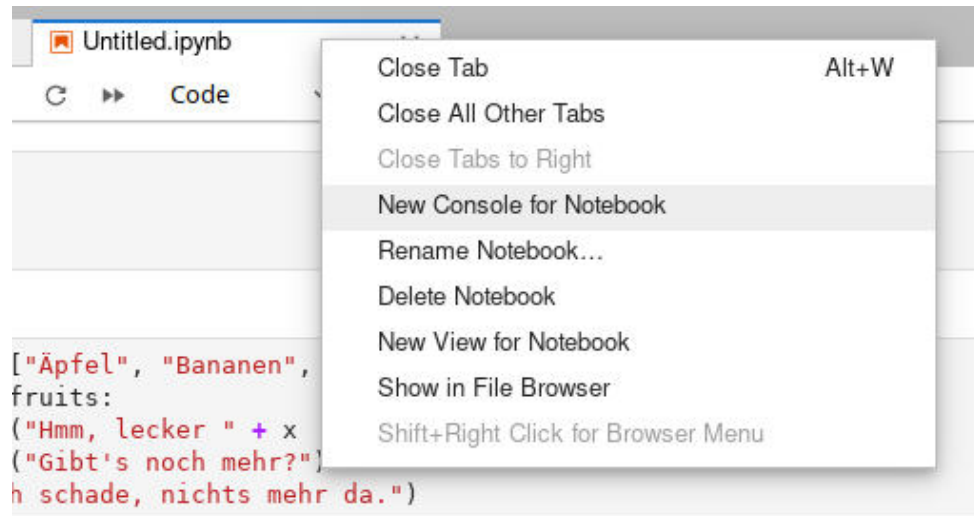
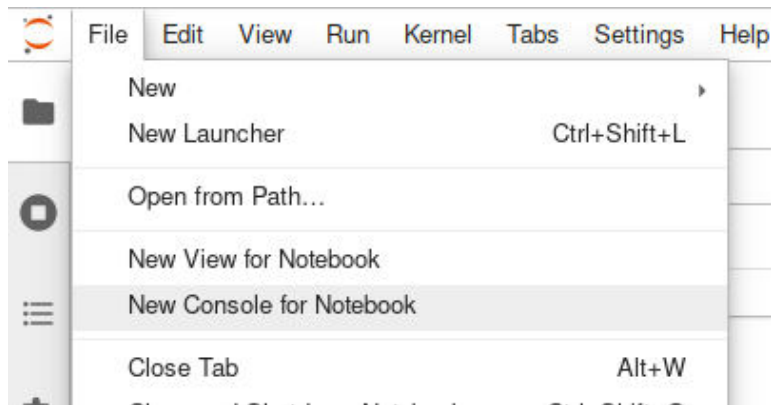
```
[*]: 1 while True:  
      2     pass
```

wurde **noch nie** ausgeführt 

```
[ ]: 1 print(42)
```

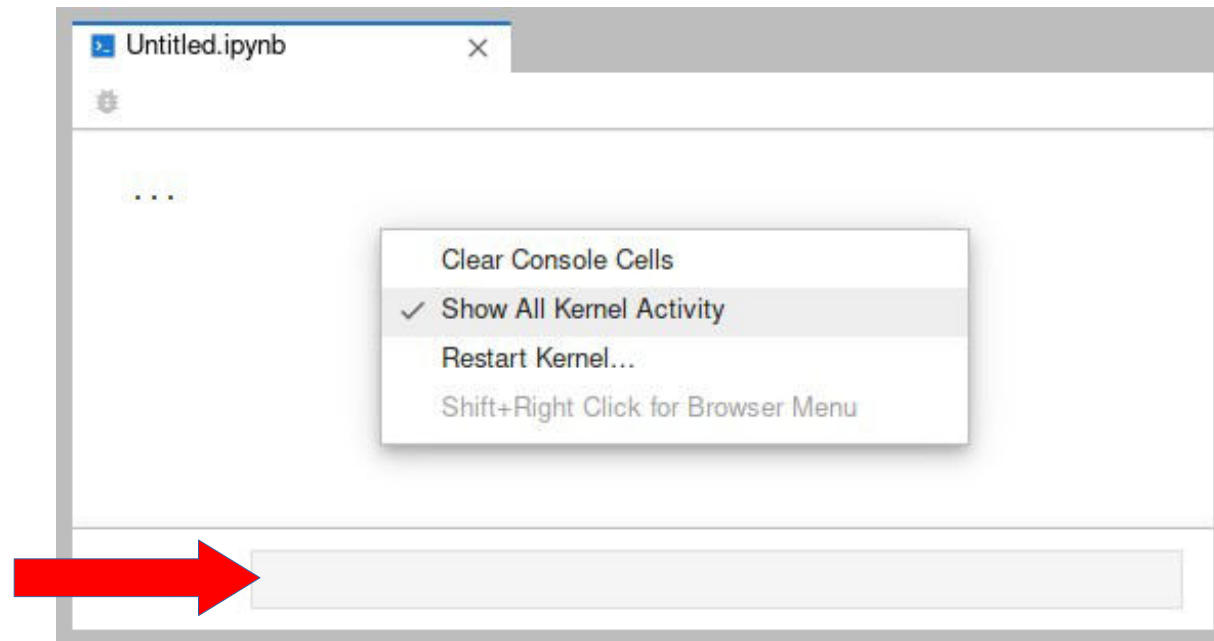

Einschub: Console

Die Console bietet die Möglichkeit, ausserhalb des eigentlichen Notebooks schnell und einfach Befehle im gleichen Kernel laufen zu lassen. Die Console kann unter anderem über das Menü «File ► New Console for Notebook» oder über das Kontextmenü (rechte Maustaste) des Notebook-Tabs gestartet werden:



Einschub: Console

Zusätzlich bietet die Console noch die Möglichkeit, alle im Kernel ausgeführten Befehle und deren Ausgeben in ihrer tatsächlichen Reihenfolge zu sehen. Die Aktivierung erfolgt via Kontextmenü ► «Show All Kernel Activity».



Codezelle für Experimente
ausserhalb des Notebooks
(Ausführung mit `Shift+Enter`)

Aufgabe: Console

Aktivieren Sie die Console und die Ausgabe sämtlicher Kernelaktivitäten (siehe vorherige Folie).

Führen Sie sowohl die Codezelle der Console als auch Codezellen Ihres Notebooks aus und beobachten Sie sowohl die Zellenummerierung im Notebook als auch die Ausgaben in der Console.

Tipp:

Die Codezelle am unteren Rad der Console hat eine History-Funktion, d.h. wenn die Codezelle leer ist, kann man mit den Pfeiltasten (hoch und runter) durch die bisher ausgeführten Befehle navigieren und diese über diesen Weg auch einfach anpassen und erneut ausführen.

Bitte ausprobieren!

```
Untitled.ipynb x
+ ✂ 📄 ▶ ■ ↺ ▶▶ Code Python 3 ○

[36]: 1 a = 13
      2 a += 2
      3 print(a)
      15

Untitled.ipynb x
...
[35]: print("Hallo, hallo, hier ein Test in der Console...")
      Hallo, hallo, hier ein Test in der Console...
[36]: a = 13
      a += 2
      print(a)
      15

[ ]:
```

Tipps & Tricks

Durch Drücken der **TAB-Taste** kann **Code automatisch vervollständigt** werden.

Rechts ein Beispiel, bei dem direkt nach der Eingabe des Textes «csv.» die TAB-Taste gedrückt wurde. Dann werden alle Methoden aus dem csv-Modul angezeigt.

Falls der Code an der Stelle schon eindeutig war, z.B. wenn Sie «csv.rea» eingeben und dann die TAB-Taste drücken, wird automatisch auf «csv.reader» vervollständigt, da es die einzige Methode ist, die mit diesem Präfix anfängt.

Bitte ausprobieren!

```
1 import csv
2
3 with open('Holzpreisindex.csv', mode='r') as
4     my_reader = csv.reader(my_file)
5
6 csv.|
7
8 s complex statement
9 #
10 c Dialect class
11 #
12 c DictReader class
13 #
14 c DictWriter class
15 #
16 c Error class
17 #
18 c excel class
19 #
20 c excel_tab class
21 #
22 f field_size_limit function
23 #
24 f get_dialect function
25 #
26 f list_dialects function
27 #
28 1_11 108.0 103.9
29 2_11 106.0 107.3
30 3_11 107.0 106.3
31 4_11 105.0 103.9
32 5_11 104.0 107.3
33 6_11 104.0 106.3
34 1_12 105.0 103.9
35 2_12 104.0 107.3
36 3_12 103.9 106.3
```

Tipps & Tricks: Dokumentation

Die Python-Module und ihre Funktionen sind im Allgemeinen sehr gut auf ihren jeweiligen Webseiten dokumentiert. Hier nochmal das Beispiel für das Modul csv:

<https://docs.python.org/3/library/csv.html>

Um aber nicht ständig zwischen Jupyter und der Dokumentation im Webbrowser wechseln zu müssen, kann man sich die **Dokumentation** eines Moduls oder einer Funktion, auf der der Textcursor gerade steht, schnell und direkt mit der Tastenkombination **Shift-TAB** anzeigen lassen (rechts das Beispiel für die Funktion reader im Modul csv).

Aufgabe:

Lassen Sie sich auf diese Art und Weise die Dokumentation des Moduls csv selbst anzeigen.

```
lzpreisindex.csv', mode='r') as my_file:  
    = csv.reader(my_file)
```

```
r
```

```
ne with he  
eader)
```

```
data
```

```
n my_reade  
(row[0], r
```

Docstring:

```
csv_reader = reader(iterable [, dialect='excel']  
                    [optional keyword args])
```

```
    for row in csv_reader:  
        process(row)
```

The "iterable" argument can be any object that returns a line of input for each iteration, such as a file object or a list. The optional "dialect" parameter is discussed below. The function also accepts optional keyword arguments which override settings provided by the dialect.

The returned object is an iterator. Each iteration returns a row of the CSV file (which can span multiple input lines).

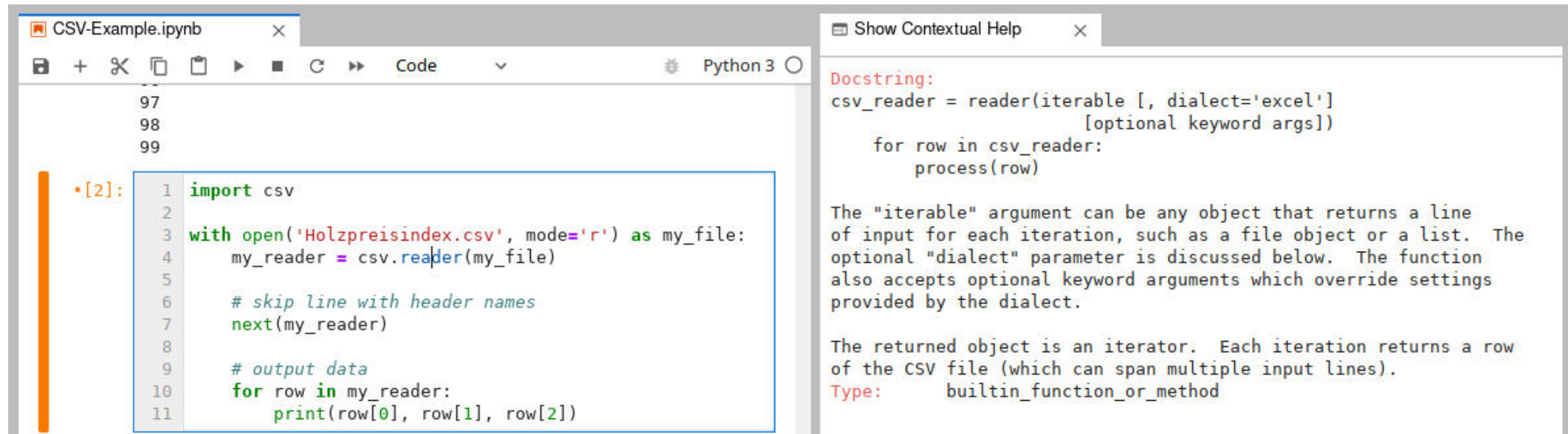
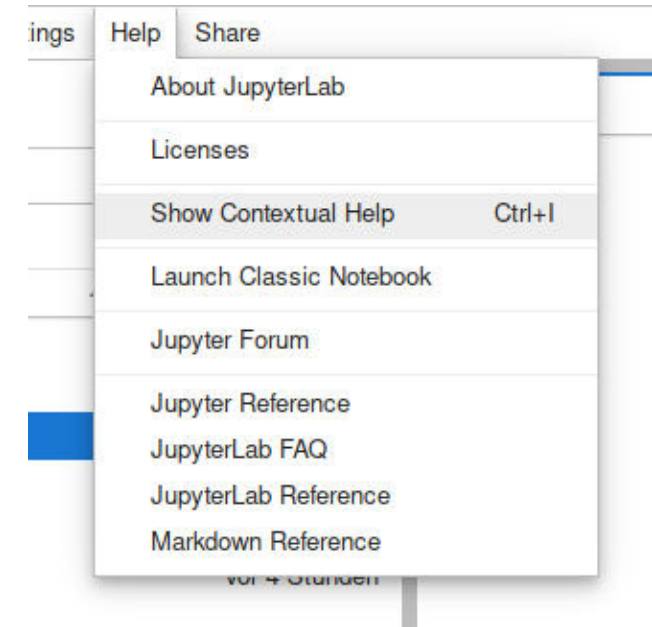
Type: builtin_function_or_method

Tipps & Tricks: Kontexthilfe

Über das Menü «Help ► Show Contextual Help» oder mit der Tastenkombination **Ctrl+I** wird die Kontexthilfe eingeblendet. In diesem Hilfefenster wird dynamisch immer die Dokumentation des Moduls oder der Funktion angezeigt, in der sich der Textcursor gerade befindet.

(Das Modul oder die Funktion muss jedoch schon mindestens einmal geladen worden sein, die Zelle also sicherheitshalber nochmal ausführen.)

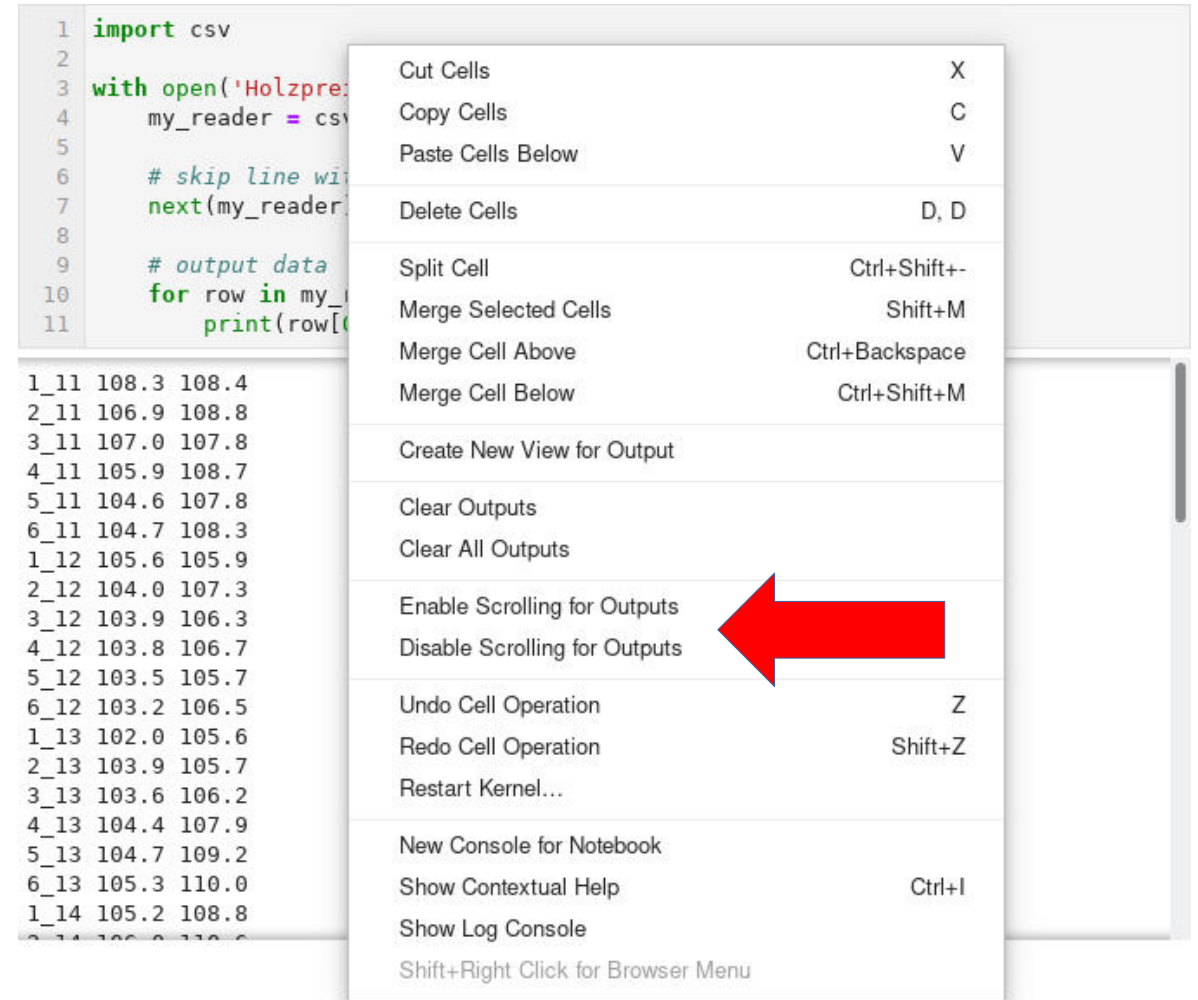
Bitte ausprobieren!



Tipps & Tricks: Scrolling von langen Ausgaben

Es kann vorkommen, dass die Ausgabe einer Codezelle sehr lang wird. Damit die Übersicht und die Navigation im Notebook nicht darunter leidet, kann für jede Zelle festgelegt werden, ob die Ausgaben vollständig oder nur in einem kleinen scrollbaren Abschnitt angezeigt werden.

Diese Funktion erreicht man über das Kontextmenü (rechte Maustaste) einer jeden Zelle.



```
1 import csv
2
3 with open('Holzpreis.csv') as f:
4     my_reader = csv.reader(f)
5
6     # skip line with header
7     next(my_reader)
8
9     # output data
10    for row in my_reader:
11        print(row[0], row[1], row[2])
```

1_11 108.3 108.4
2_11 106.9 108.8
3_11 107.0 107.8
4_11 105.9 108.7
5_11 104.6 107.8
6_11 104.7 108.3
1_12 105.6 105.9
2_12 104.0 107.3
3_12 103.9 106.3
4_12 103.8 106.7
5_12 103.5 105.7
6_12 103.2 106.5
1_13 102.0 105.6
2_13 103.9 105.7
3_13 103.6 106.2
4_13 104.4 107.9
5_13 104.7 109.2
6_13 105.3 110.0
1_14 105.2 108.8
2_14 106.0 110.0

- Cut Cells X
- Copy Cells C
- Paste Cells Below V
- Delete Cells D, D
- Split Cell Ctrl+Shift+-
- Merge Selected Cells Shift+M
- Merge Cell Above Ctrl+Backspace
- Merge Cell Below Ctrl+Shift+M
- Create New View for Output
- Clear Outputs
- Clear All Outputs
- Enable Scrolling for Outputs ←
- Disable Scrolling for Outputs
- Undo Cell Operation Z
- Redo Cell Operation Shift+Z
- Restart Kernel...
- New Console for Notebook
- Show Contextual Help Ctrl+I
- Show Log Console
- Shift+Right Click for Browser Menu

Tipps & Tricks: Rückgängig / Wiederherstellen

Während man eine Zelle im Editiermodus bearbeitet, funktionieren die Operationen *Rückgängig* und *Wiederherstellen* wie in jeder üblichen Textverarbeitung über die Tastaturkombinationen **Ctrl+Z** und **Ctrl+Y**.

Im Commandmodus geht es etwas schneller mit nur **Z** und **Shift+Z**.

The screenshot shows a Jupyter Notebook cell with the following Python code:

```
1 import csv
2
3 with open('Holzpreise.csv') as f:
4     my_reader = csv.reader(f)
5
6     # skip line with header
7     next(my_reader)
8
9     # output data
10    for row in my_reader:
11        print(row)
```

The output of the cell is a list of rows of data:

```
1_11 108.3 108.4
2_11 106.9 108.8
3_11 107.0 107.8
4_11 105.9 108.7
5_11 104.6 107.8
6_11 104.7 108.3
1_12 105.6 105.9
2_12 104.0 107.3
3_12 103.9 106.3
4_12 103.8 106.7
5_12 103.5 105.7
6_12 103.2 106.5
1_13 102.0 105.6
2_13 103.9 105.7
3_13 103.6 106.2
4_13 104.4 107.9
5_13 104.7 109.2
6_13 105.3 110.0
1_14 105.2 108.8
2_14 106.0 110.0
```

A context menu is open over the output, listing various actions. A red arrow points to the 'Undo Cell Operation' option, which has the keyboard shortcut 'Z'.

Action	Shortcut
Cut Cells	X
Copy Cells	C
Paste Cells Below	V
Delete Cells	D, D
Split Cell	Ctrl+Shift+-
Merge Selected Cells	Shift+M
Merge Cell Above	Ctrl+Backspace
Merge Cell Below	Ctrl+Shift+M
Create New View for Output	
Clear Outputs	
Clear All Outputs	
Enable Scrolling for Outputs	
Disable Scrolling for Outputs	
Undo Cell Operation	Z
Redo Cell Operation	Shift+Z
Restart Kernel...	
New Console for Notebook	
Show Contextual Help	Ctrl+I
Show Log Console	
Shift+Right Click for Browser Menu	

Blender

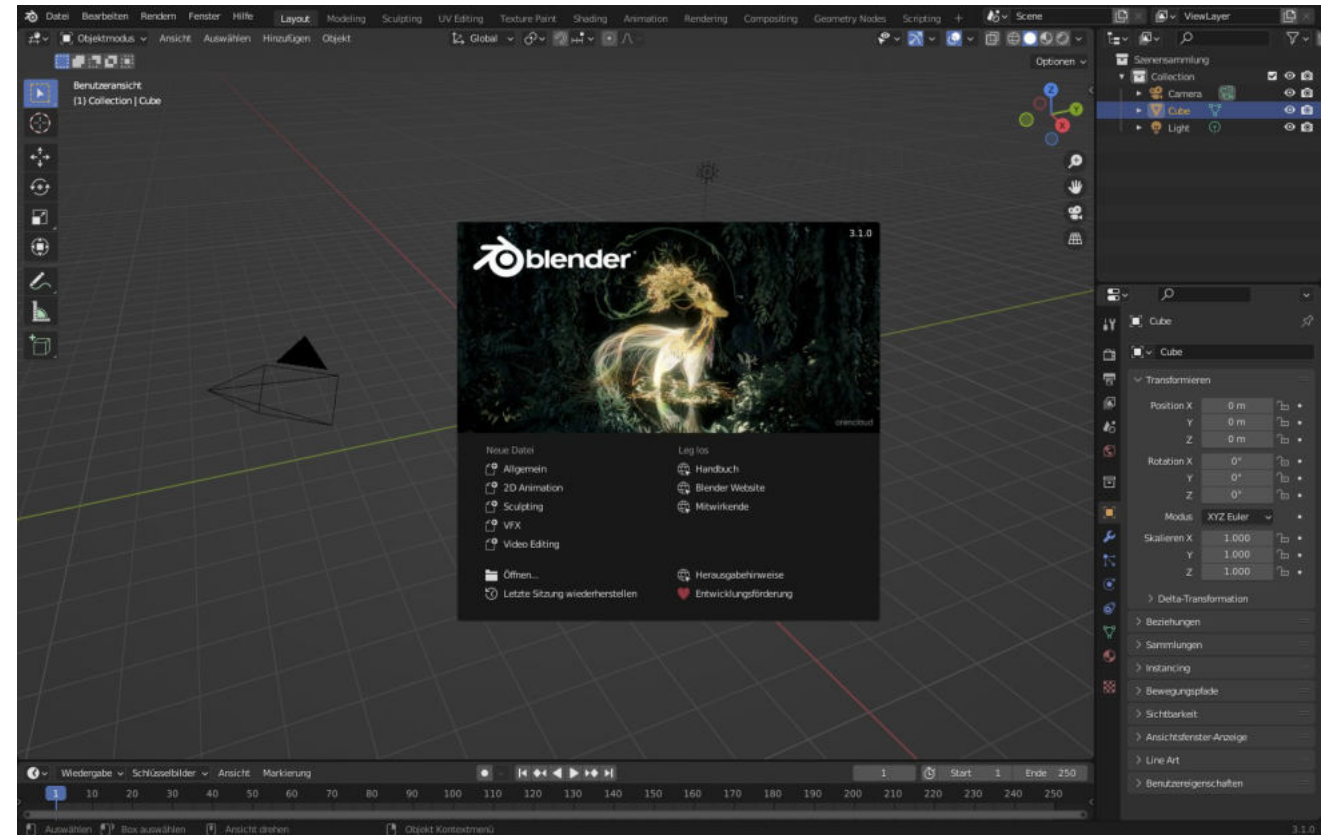
Einführung Blender



Zur Erzeugung hochqualitativer grafischer Darstellungen von CAD-Zeichnungen werden darauf spezialisierte Programme, sogenannte 3D-Grafiksuiten, verwendet.

Blender ist eine freie und leistungsstarke Software zur Erstellung von dreidimensionalen Grafiken, Animationen und Spielen, siehe:

<https://www.blender.org>



FreeCAD (optional)

Einfaches CAD-Programm? → FreeCAD



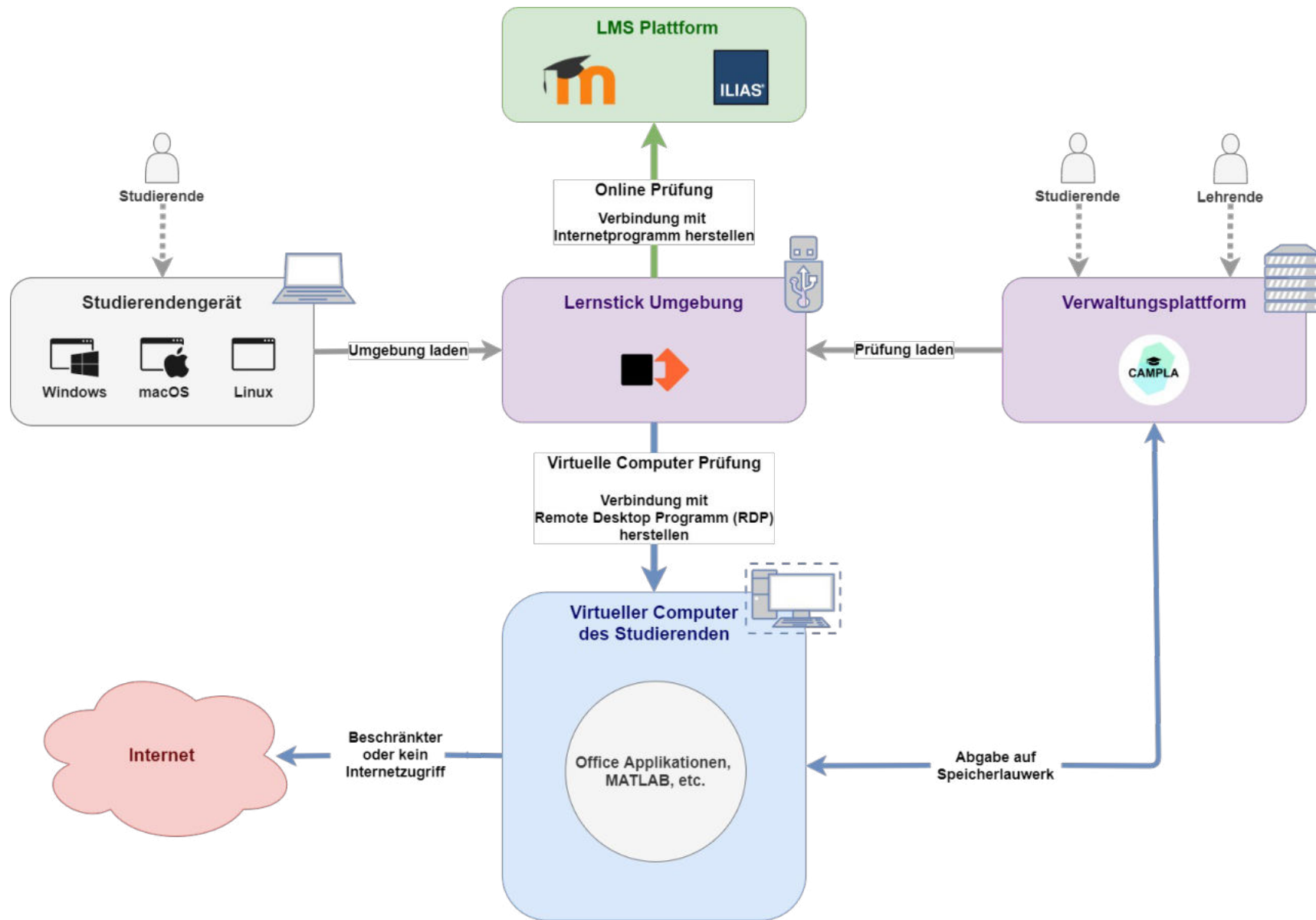
<https://www.freecad.org>



CAMPLA / Lernstick

CAMPLA / Lernstick

- Verwaltungsoberfläche CAMPLA
 - Zugriff auf die Prüfung direkt geht leider nur im BFH-Netzwerk...
- Lernstick ohne CAMPLA



CAMPLA Systemlandschaft



Hauptseite - FreeCAD Do x Neuer Tab

Mit Google suchen oder Adresse eingeben

FreeCAD Wiki FreeCAD-Reporting Blender Documentation BlenderBIM

FreeCAD GitHub Blender B

Blender 3.6.2

Benutzeransicht (1) Collection | Cube

studio blender.org

Neue Datei Allgemein 2D Animation Sculpting VFX Video Editing

Leglos Handbuch Blender Website Mitwirkende

Herausgabenweise Entwicklungsförderung

Positionen: X 0 m, Y 0 m, Z 0 m

Rotationen: X 0°, Y 0°, Z 0°

Skalierung: X 1.00, Y 1.00, Z 1.00

Wiedergabe Keying Ansicht Markierung

FreeCAD 0.21.0

Beispiele

- ArchDetail.FCStd 225Kb
- EngineBlock.FCStd Werner Mayer 78Kb
- FemCalculixCantile ver2D.FCStd 116Kb
- FemCalculixCantile ver3D.FCStd 148Kb
- FemCalculixCantile ver3D_newSolver.F CStd 158Kb
- Schenkel.stp 576Kb
- draft_test_objects.F CStd FreeCAD Developers 117Kb

Daten angepasst werden

Menü Bearbeiten -> Einstellungen -> Start -> Zusätzlichen Ordner anzeigen

Spyder (Python 3.11)

Gebrauch

Hier können Sie Hilfe zu jedem Objekt erhalten, indem Sie **Ctrl+I** davor drücken, entweder im Editor oder in der Konsole.

Hilfe kann auch automatisch angezeigt werden, nachdem eine linke Klammer neben ein Objekt geschrieben wurde. Sie können dieses Verhalten in *Einstellungen* > *Hilfe* aktivieren.

Konsole 1/A x

```
Python 3.11.2 (main, Mar 13 2023, 12:18:29) [GCC 12.2.0]
Type "copyright", "credits" or "license()" for more information.

IPython 8.5.0 -- An enhanced Interactive Python.

In [1]:
```

Python-Konsole Chronik

