



Predictive Maintenance

Einführung

Agenda

1. Organisatorisches rund um den Kurs

- i. Termine
- ii. Klausur, Übungen, etc.
- iii. Einbettung der Veranstaltung

2. Unsere Tools

- i. Anaconda
- ii. JupyterLab
- iii. Spyder

3. Predictive Maintenance



Organisatorisches rund um den Kurs

Regensburg School of Digital Sciences (RSDS)



OSTBAYERISCHE
TECHNISCHE HOCHSCHULE
REGENSBURG

REGENSBURG SCHOOL
OF DIGITAL SCIENCES

- Dieses Modul wird (auch) im **Rahmen der Regensburg School of Digital Sciences** angeboten
- Wenn dieses Modul in Ihrem **Modulkatalog** eingebettet ist, dann ist eine **Anrechnung von Hause aus** möglich
- **Andernfalls** müssen Sie mit Ihrer **Prüfungskommission** über die Anrechenbarkeit Rücksprache halten

Teilnahmebestätigung

- Nach erfolgreicher Teilnahme können Sie sich – zusätzlich zur eingetragenen Note – eine Teilnahmebestätigung ausstellen lassen
- Wenden Sie sich hierzu bitte nach bestandener Prüfung an **Frau Manon Georg**



First Step

0

So what?

Bitte an allen CIP-Pool PCs
nachschauen, ob Anaconda
installiert ist!

Anaconda Installers

Windows

Python 3.7

64-Bit Graphical Installer (466 MB)

32-Bit Graphical Installer (423 MB)

Python 2.7

64-Bit Graphical Installer (413 MB)

32-Bit Graphical Installer (356 MB)

MacOS

Python 3.7

64-Bit Graphical Installer (442 MB)

64-Bit Command Line Installer (430 MB)

Python 2.7

64-Bit Graphical Installer (637 MB)

64-Bit Command Line Installer (409 MB)

Linux

Python 3.7

64-Bit (x86) Installer (522 MB)

64-Bit (Power8 and Power9) Installer (276 MB)

Python 2.7

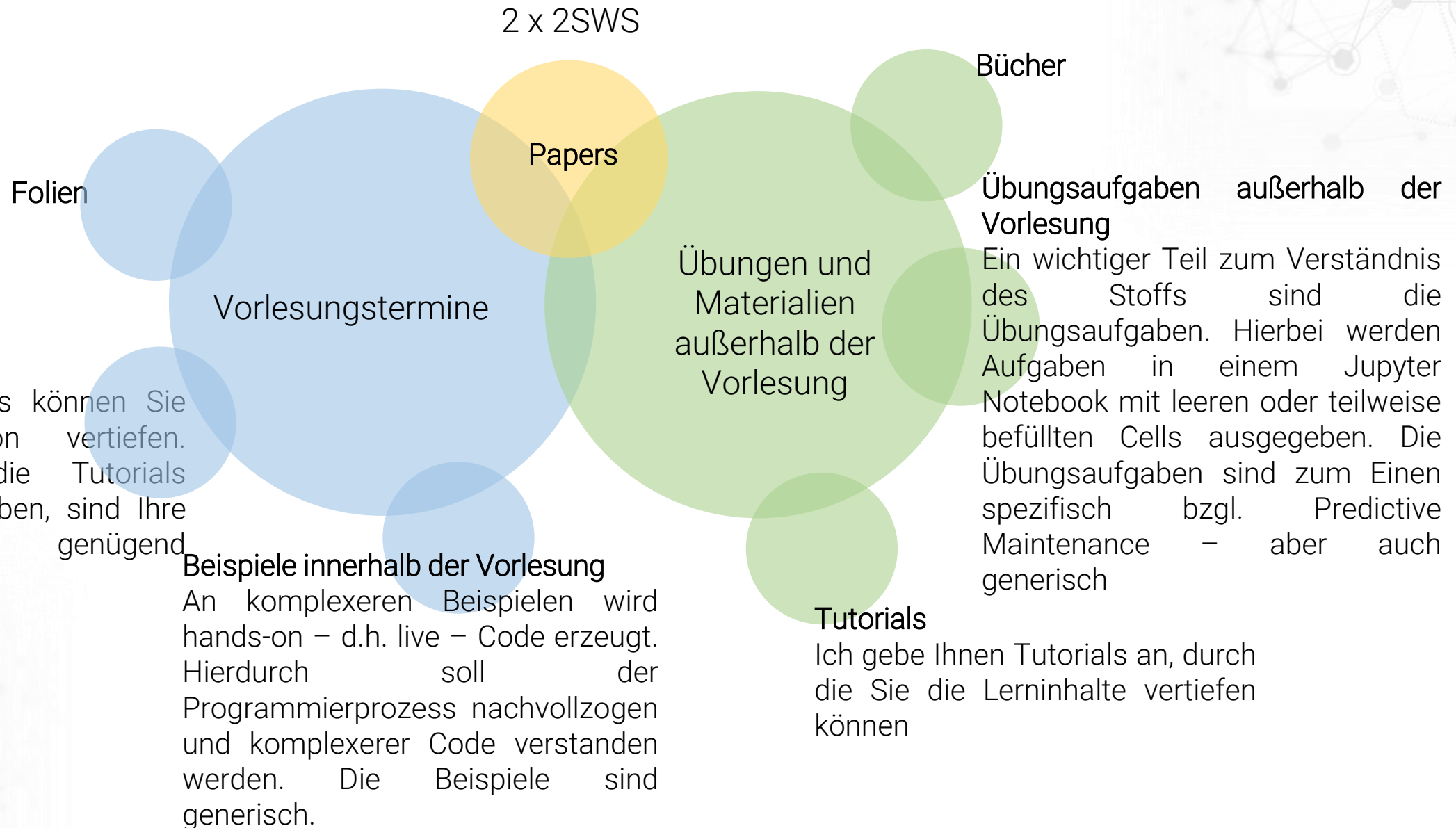
64-Bit (x86) Installer (477 MB)

64-Bit (Power8 and Power9) Installer (295 MB)

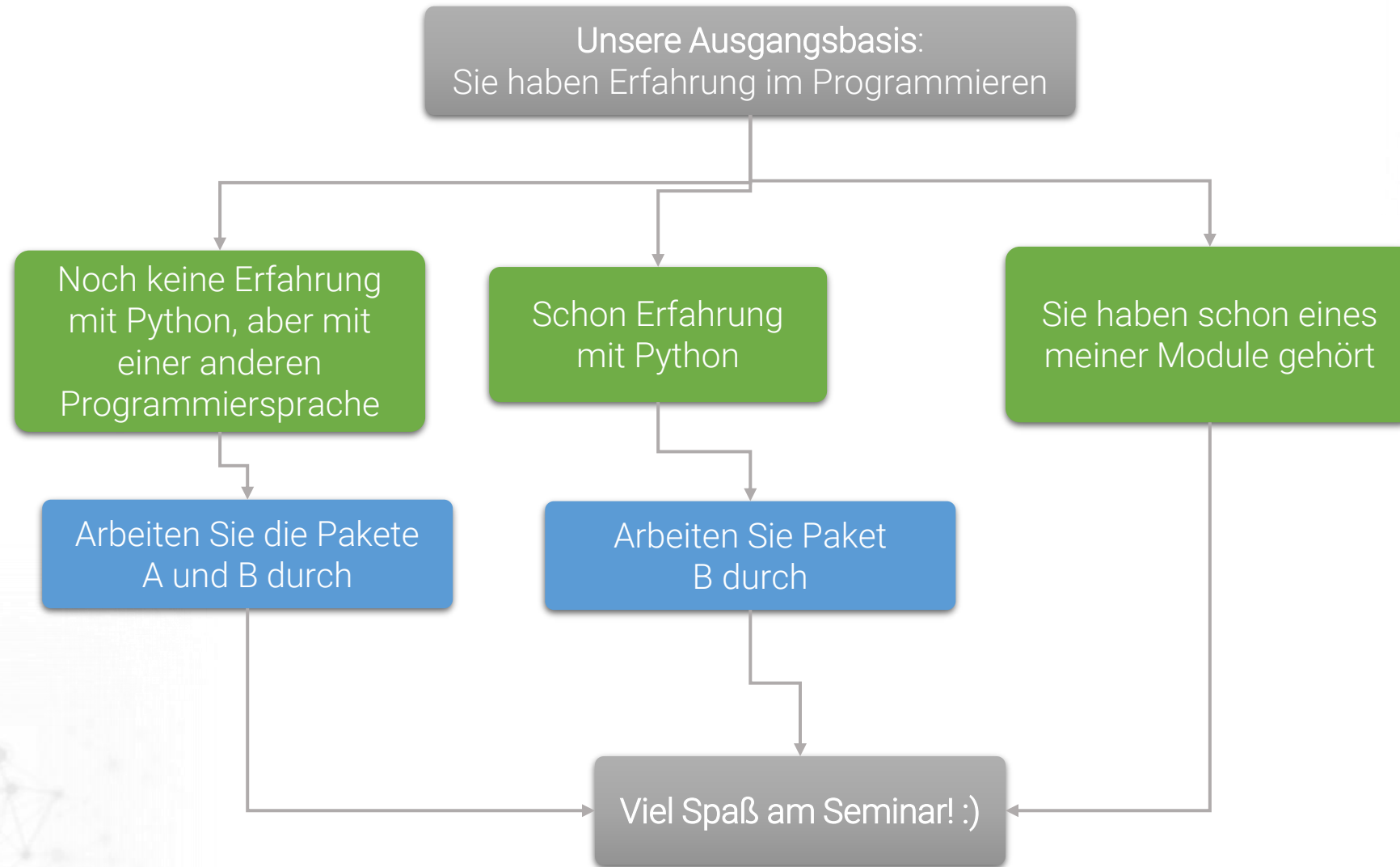
Download Anaconda

<https://www.anaconda.com/products/individual>

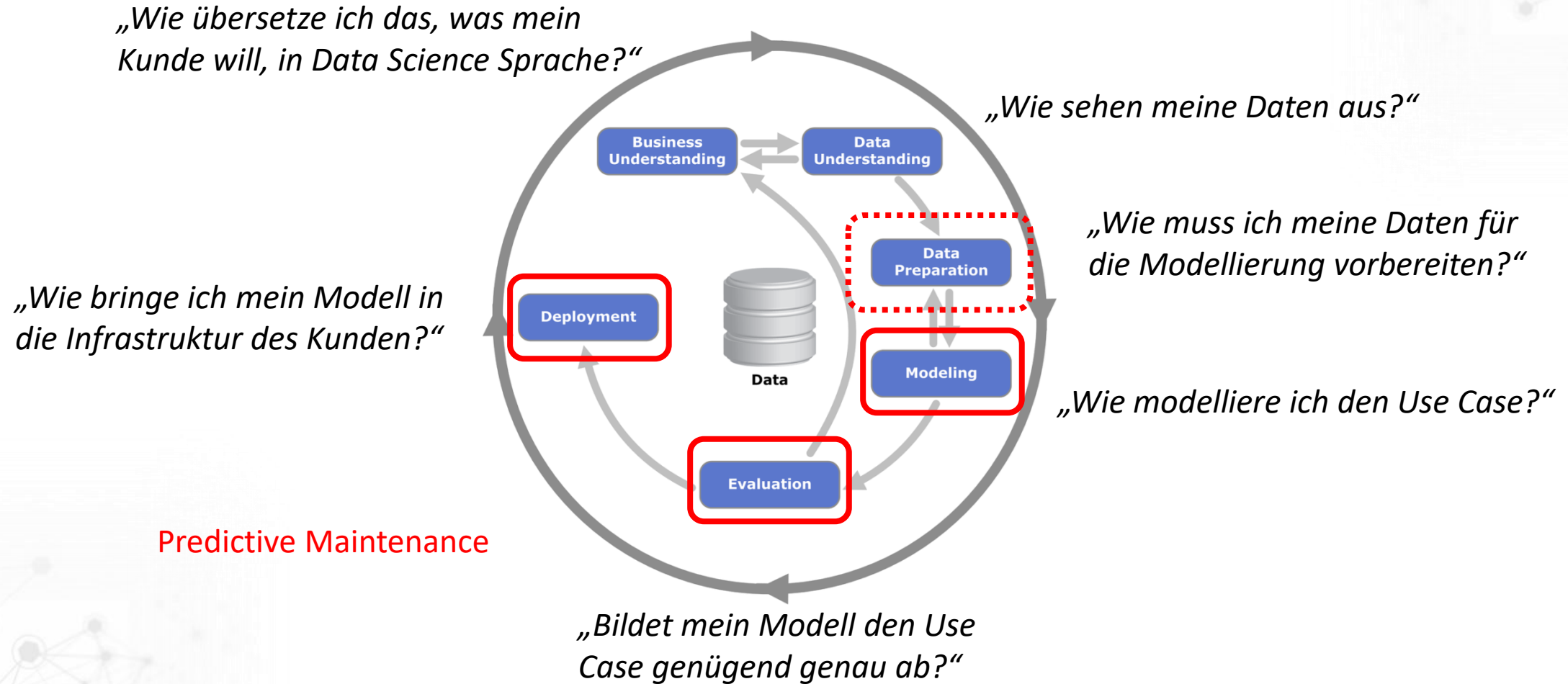
Struktureller Aufbau dieser Vorlesung



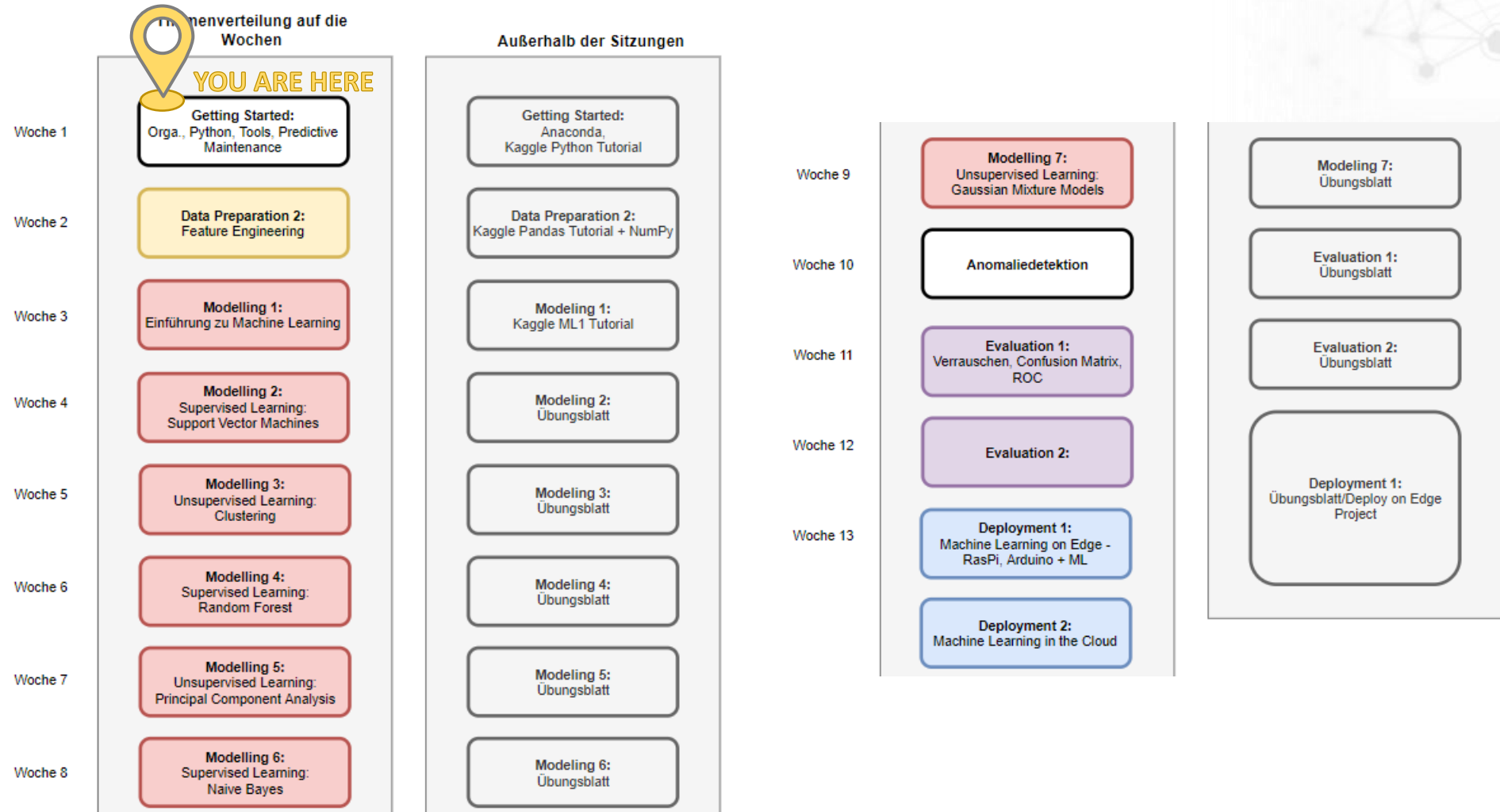
Schätzen Sie sich selbst ein



CRISP-DM als Grundlage



Konkrete Themen der einzelnen Termine



Präsenzzeit/Eigenstudium: Aufteilung Ihrer Zeit

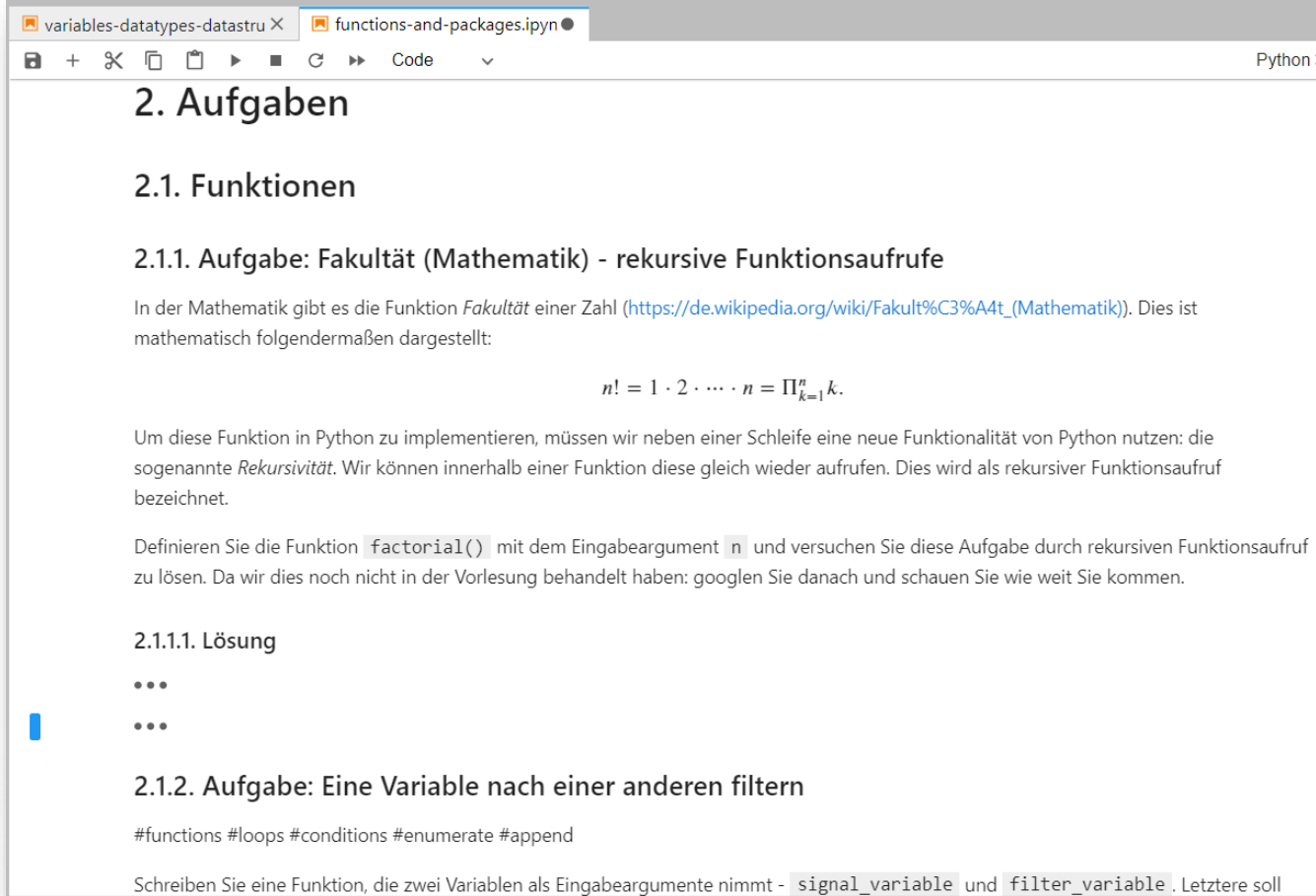
Gesamtarbeitsaufwand
150 Stunden

- Entspricht 5 ECTS
 - Präsenzzeit: 14 Wochen x 4 SWS = 42h
 - Eigenstudium: 108h
- $108\text{h} / 14\text{ Wochen} \sim 7,7\text{h/Woche}$

Aufteilung dieser Zeit:

- Bearbeitung der Übungsaufgaben
- Bearbeitung der Tutorials
- Lesen entsprechender Literatur

Übungen



variables-datatypes-datastru X functions-and-packages.ipynb Python 3

2. Aufgaben

2.1. Funktionen

2.1.1. Aufgabe: Fakultät (Mathematik) - rekursive Funktionsaufrufe

In der Mathematik gibt es die Funktion *Fakultät* einer Zahl ([https://de.wikipedia.org/wiki/Fakult%C3%A4t_\(Mathematik\)](https://de.wikipedia.org/wiki/Fakult%C3%A4t_(Mathematik))). Dies ist mathematisch folgendermaßen dargestellt:

$$n! = 1 \cdot 2 \cdot \dots \cdot n = \prod_{k=1}^n k.$$

Um diese Funktion in Python zu implementieren, müssen wir neben einer Schleife eine neue Funktionalität von Python nutzen: die sogenannte *Rekursivität*. Wir können innerhalb einer Funktion diese gleich wieder aufrufen. Dies wird als rekursiver Funktionsaufruf bezeichnet.

Definieren Sie die Funktion `factorial()` mit dem Eingabeargument `n` und versuchen Sie diese Aufgabe durch rekursiven Funktionsaufruf zu lösen. Da wir dies noch nicht in der Vorlesung behandelt haben: googlen Sie danach und schauen Sie wie weit Sie kommen.

2.1.1.1. Lösung

...

2.1.2. Aufgabe: Eine Variable nach einer anderen filtern

#functions #loops #conditions #enumerate #append

Schreiben Sie eine Funktion, die zwei Variablen als Eingabeargumente nimmt - `signal_variable` und `filter_variable`. Letztere soll

- Jede Woche wird ein Übungsblatt mit Aufgaben gestellt
- Das **Format**: JupyterNotebook
- Die Übungsaufgaben werden dann in den Übungsterminen **durchgearbeitet**
- Während der Übungen können Sie sich austauschen– ca. 5-10min pro Aufgabe – dann besprechen wir die Aufgabe gemeinsam

0

So what?

Vor der Übung die Übungsaufgaben (eigenständig!) durcharbeiten!

Prüfung



Prüfung

- Am Ende des Semesters gibt es eine Prüfung im CIP-Pool
- Dauer: 90 Minuten + 30 Minuten Treffen vorab im CIP-Pool
- Hilfsmittel: alle (bis auf Anwendungen wie z.B. ChatGPT)
- Datum und weitere Infos gibt es dann **während des Semesters**

Tutorials

- Klare Empfehlung: arbeiten Sie neben der Vorlesung und den Übungen auch Tutorials durch
- Die Tutorials werden den einzelnen Themenblöcken zugeordnet (s. vorhergehende Folien)

<https://www.kaggle.com/learn/overview>



Python

Learn the most important language for data science.



Intro to Machine Learning

Learn the core ideas in machine learning, and build your first models.



Intermediate Machine Learning

Learn to handle missing values, non-numeric values, data leakage and more. Your models will be more accurate and useful.



Data Visualization

Make great data visualizations. A great way to see the power of coding!



Pandas

Solve short hands-on challenges to perfect your data manipulation skills.



Feature Engineering

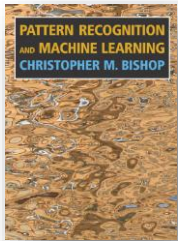
Discover the most effective way to improve your models.



Data Cleaning

Master efficient workflows for cleaning real-world, messy data.

Literatur



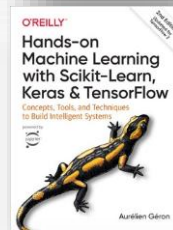
Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer. 2006.

- Vertiefendes Buch zu Machine Learning generell
- <https://www.microsoft.com/en-us/research/uploads/prod/2006/01/Bishop-Pattern-Recognition-and-Machine-Learning-2006.pdf>

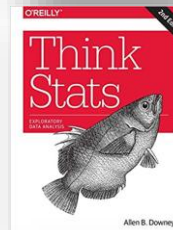


Jake VanderPlas. *Python Data Science Handbook: Essential Tools for working with Data*. O'Reilly UK Ltd. 2016.

- <https://github.com/jakevdp/PythonDataScienceHandbook>



Aurélien Géron. *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition*. O'Reilly Media, Inc. 2019.



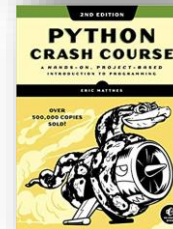
Allen B. Downey. *Think Stats: Exploratory Data Analysis*. O'Reilly UK Ltd. 2014.

- <https://github.com/AllenDowney/ThinkStats2>



Allen B. Downey. *Think Python: How to Think Like a Computer Scientist*. O'Reilly UK Ltd. 2015.

- <https://github.com/AllenDowney/ThinkPython2>



Eric Matthes. *Python Crash Course, 2nd Edition: A Hands-On, Project-Based Introduction to Programming*. No Starch Press. 2019.

Git und GitLab



- Alle Unterlagen finden Sie im zugehörigen **Git-Repository**
- Nutzen Sie gerne `git clone` und `git pull`, um sich die Unterlagen **runterzuladen** bzw. zu **aktualisieren**
- Wenn Sie kein Git verwenden möchten, dann sind in unserem **ELO-Kurs** auch die Unterlagen **verlinkt**

https://gitlab.oth-regensburg.de/gom39655/predictive_maintenance

ELO ist unser zentraler Punkt für

- Informationsaustausch außerhalb der Termine
- Materialsammlungen (Folien, Literatur, URLs, JupyterNotebooks, Umfragen, Forum, etc.)


Termine

- Übung: Montags, 15:15–16:45
- Vorlesung: Donnerstags, 13:30–15:00


Zoom-Link zur Vorlesung und Übung


<https://oth-regensburg.zoom.us/j/99743011625?pwd=dCthYmREeVNaVS9VZ25XaCtrcm94Zz09>


Forum zum Kurs

 Fragen & Antworten: Diskussionsforum zum Kurs

Anaconda: Open Source Data Science Distribution

 Download-Link Anaconda

 User Guide „JupyterLab Interface“

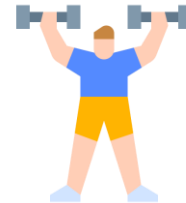
 User Guide „Jupyter Notebooks“

Unsere Kommunikationsformen



Regelmäßige Teilnahme an der Vorlesung +
Fragen unbedingt erwünscht!

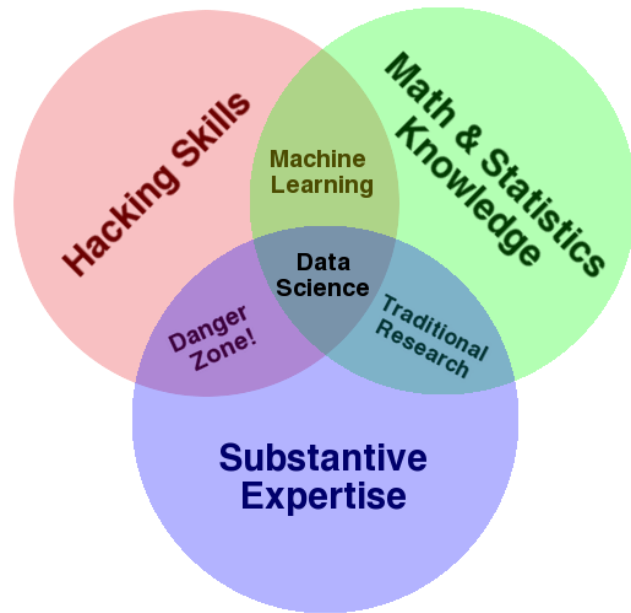
Regelmäßige Teilnahme an den Übungen +
aktive Teilnahme!



Fragen & Antworten

Außerhalb der Veranstaltungen: kommen Sie
auf mich zu, wenn es Fragen gibt. Am besten
über unser „Fragen & Antworten“-Forum
→ Gerne auch Diskussionen untereinander!

What is Data Science/Machine Learning?



Um vom BuzzWord-Bingo wegzukommen:
fundiertes Wissen nötig in

- Programmierung
- Mathematik/Statistik
- Domäne



Data Science/Machine Learning ohne Mathe



<https://i.imgur.com/0rW2b1s.gif>



So what?

- Data Science ohne Mathe ist wie blind Autofahren
- Der Kurs ist anwendungsorientiert – aber wir werden auch etwas Mathematik brauchen, um vor allem die Machine Learning Verfahren zu verstehen



Fragen?

Zum organisatorischen Teil





Unsere Tools

Anaconda und Virtual Environments
JupyterLab



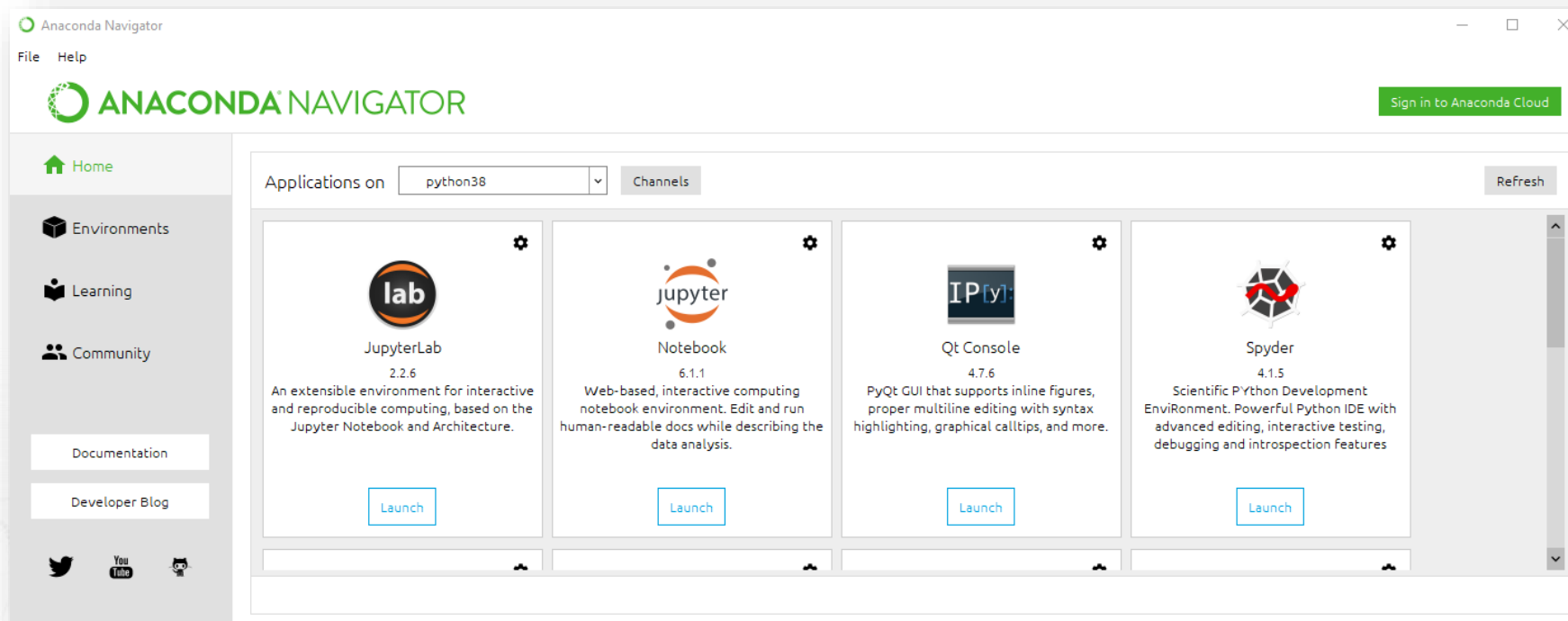
Anaconda



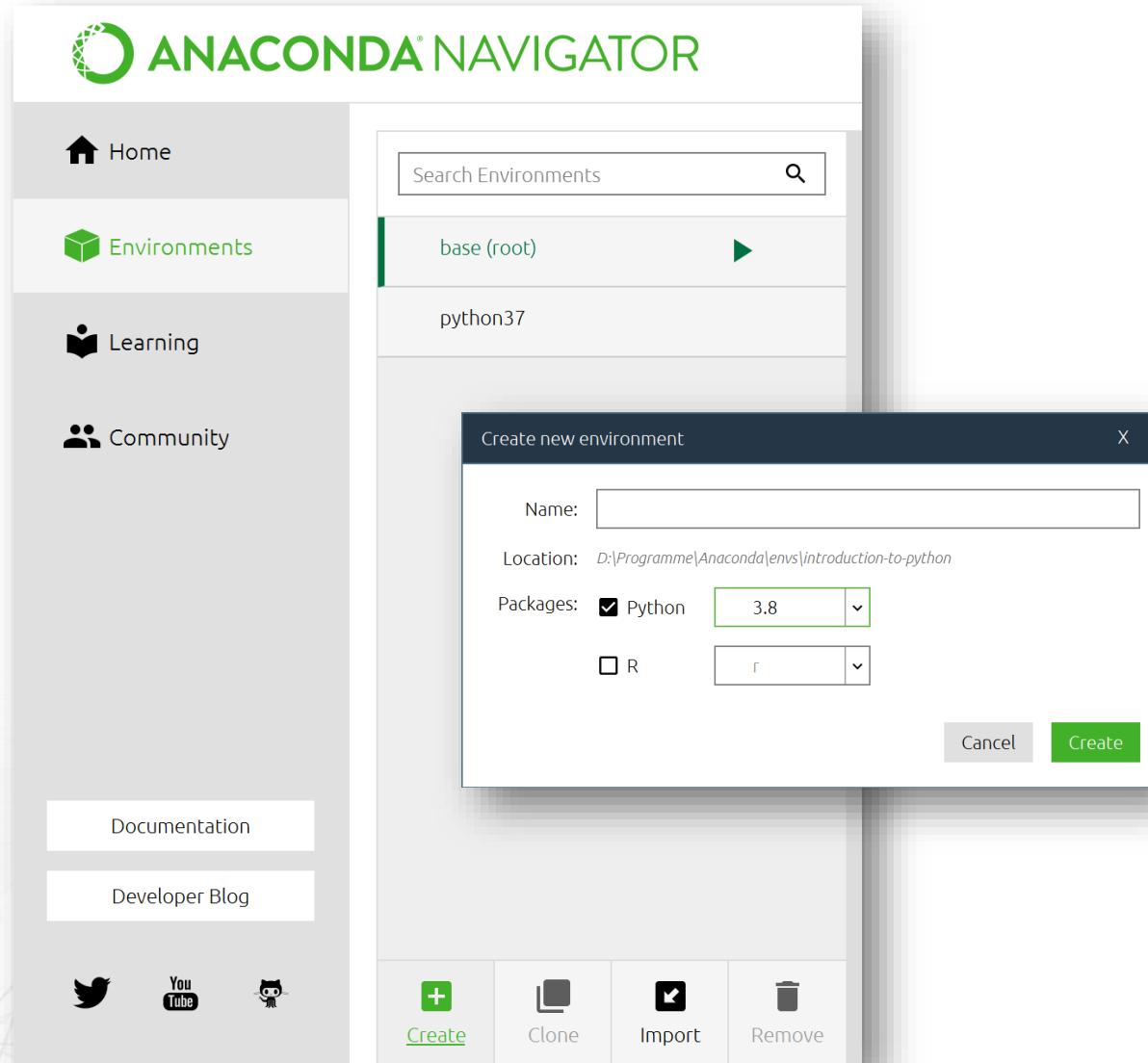
- Anaconda ist eine Open-Source-Distribution für die Programmiersprachen **Python** und R
- Vereinfacht Installation/Nutzung von Tools und Packages

Anaconda: „Single Point of Truth“

- Die meisten unserer Python-Aktivitäten/-Tools gründen auf Anaconda
- Vorteil: kein Chaos



Anaconda: Virtual Environment



- Virtual Environments stellen abgeschlossene Programmierumgebung dar
- Es macht Sinn VEs projekt- bzw. aufgabenbasiert zu erzeugen
- Anaconda bietet einfache GUI hierfür

Wir erzeugen uns ein VE mit dem Namen data-science-mit-python mit dem Anaconda Navigator wie links dargestellt

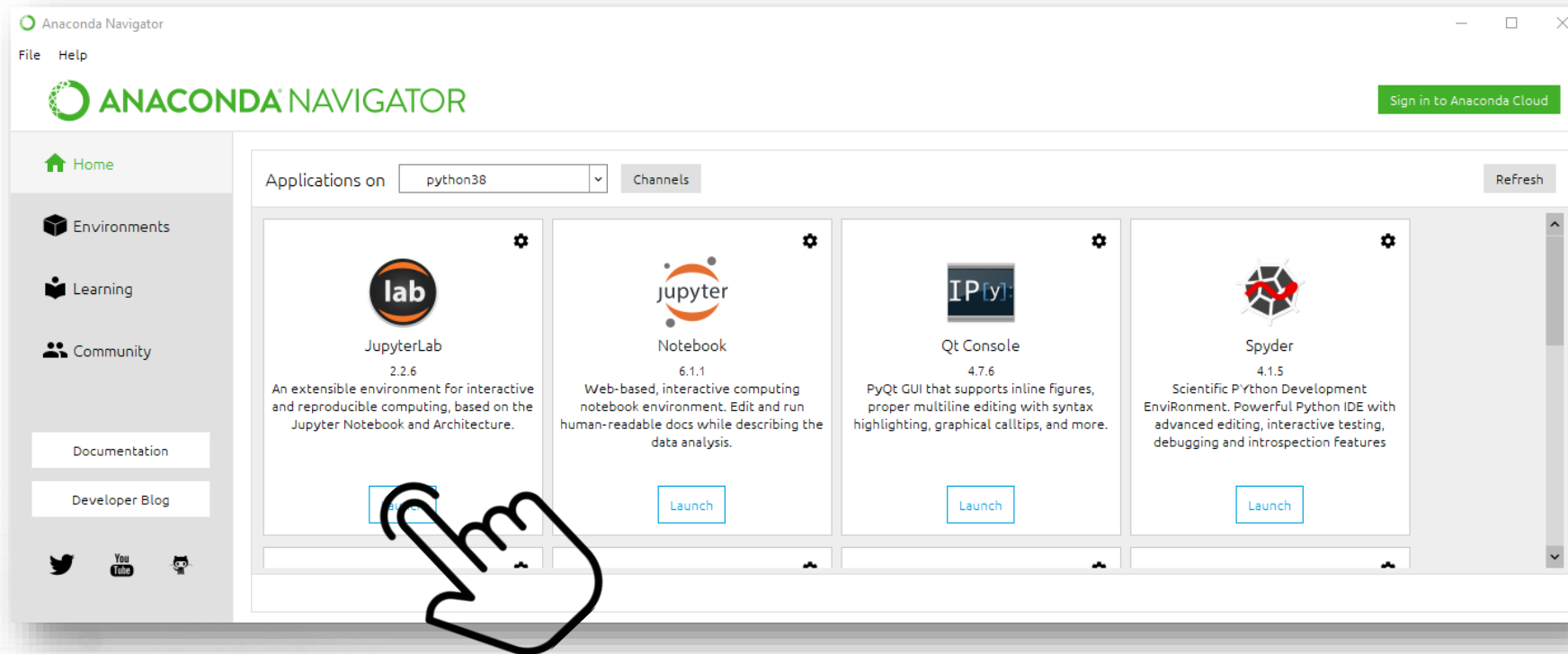
Das VE ist nach Erzeugung aktiv!
(grüner Pfeil neben VE)



So what?

- Ich habe Ihnen ein conda environment im Repo/in ELO zur Verfügung gestellt
- Bitte nutzen Sie dieses → Import

Anaconda: „Single Point of Truth“



Wir installieren

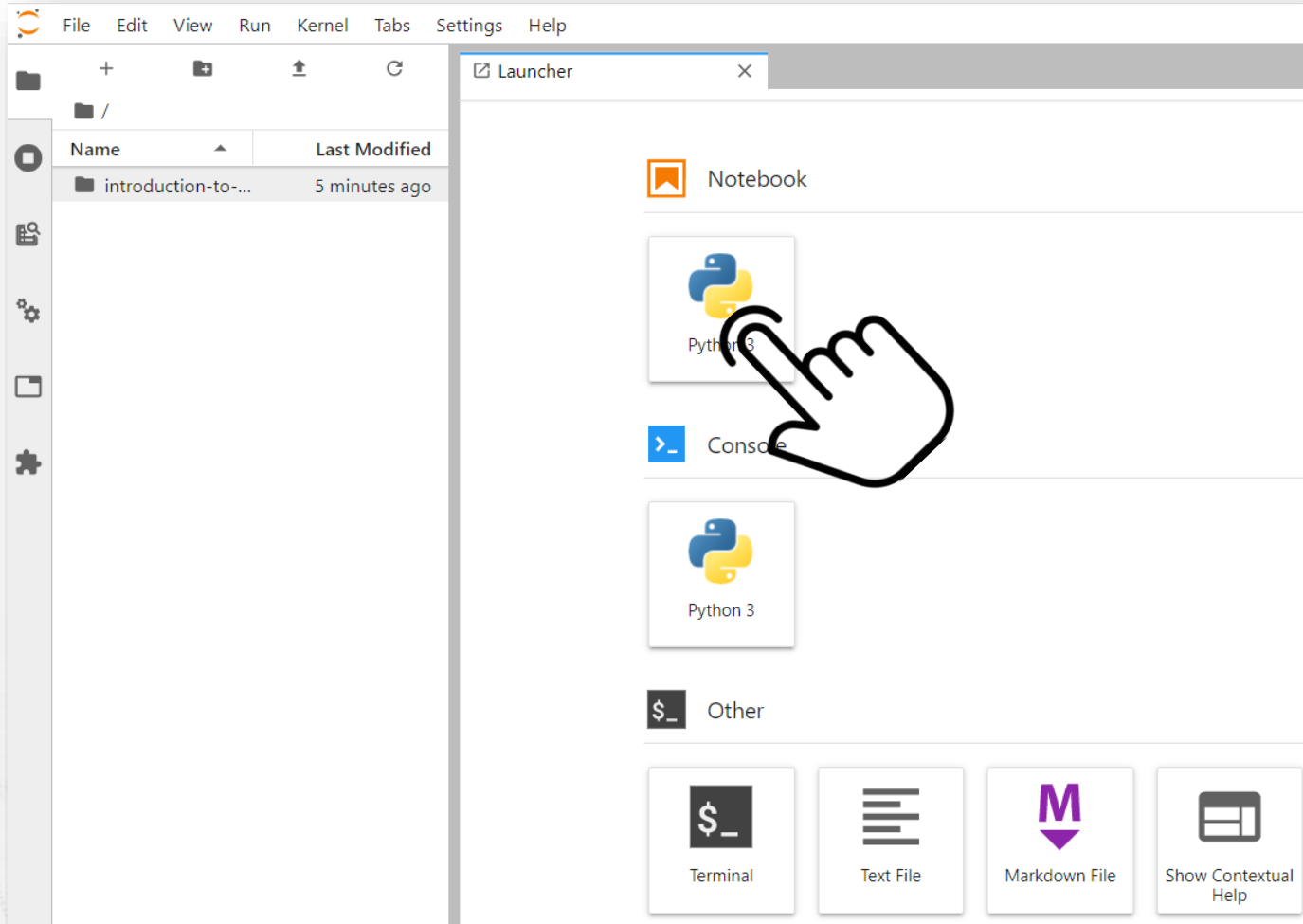
- Spyder
- JupyterLab

JupyterLab/Notebook



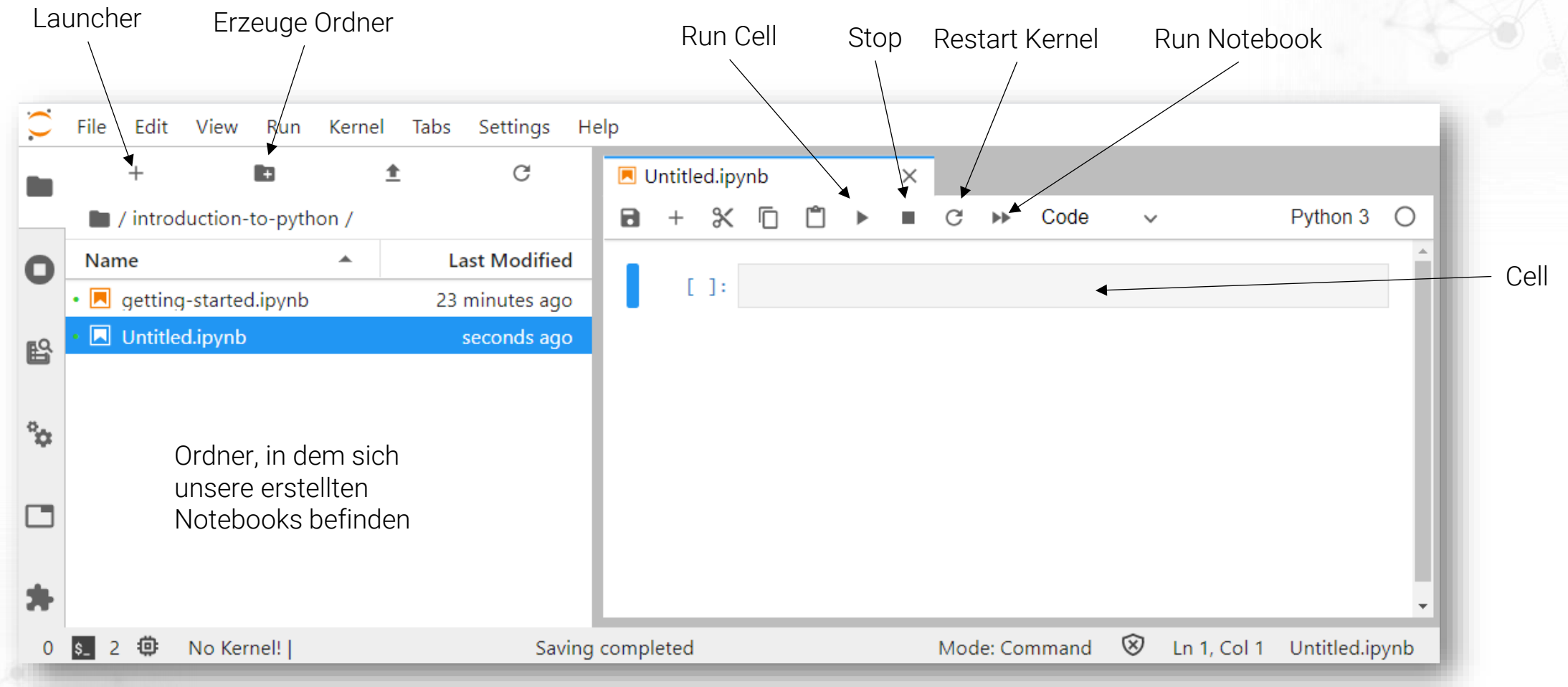
- Jupyter Notebook ist eine web-basierte interaktive Umgebung, mit der Jupyter-Notebook-Dokumente erstellt werden können
- Ein Jupyter-Notebook-Dokument ist ein JSON-Dokument mit einem versionsabhängigen Schema, das aus einer Liste von Eingabe- und Ausgabezellen besteht, die jeweils Code, Text und Plots enthalten können
- JupyterLab ist eine neuere Umgebung zur Ausführung von Jupyter Notebooks

JupyterLab: Launcher



- Nach dem Start sehen wir den Launcher
- Hier kann ein neues Notebook erzeugt werden

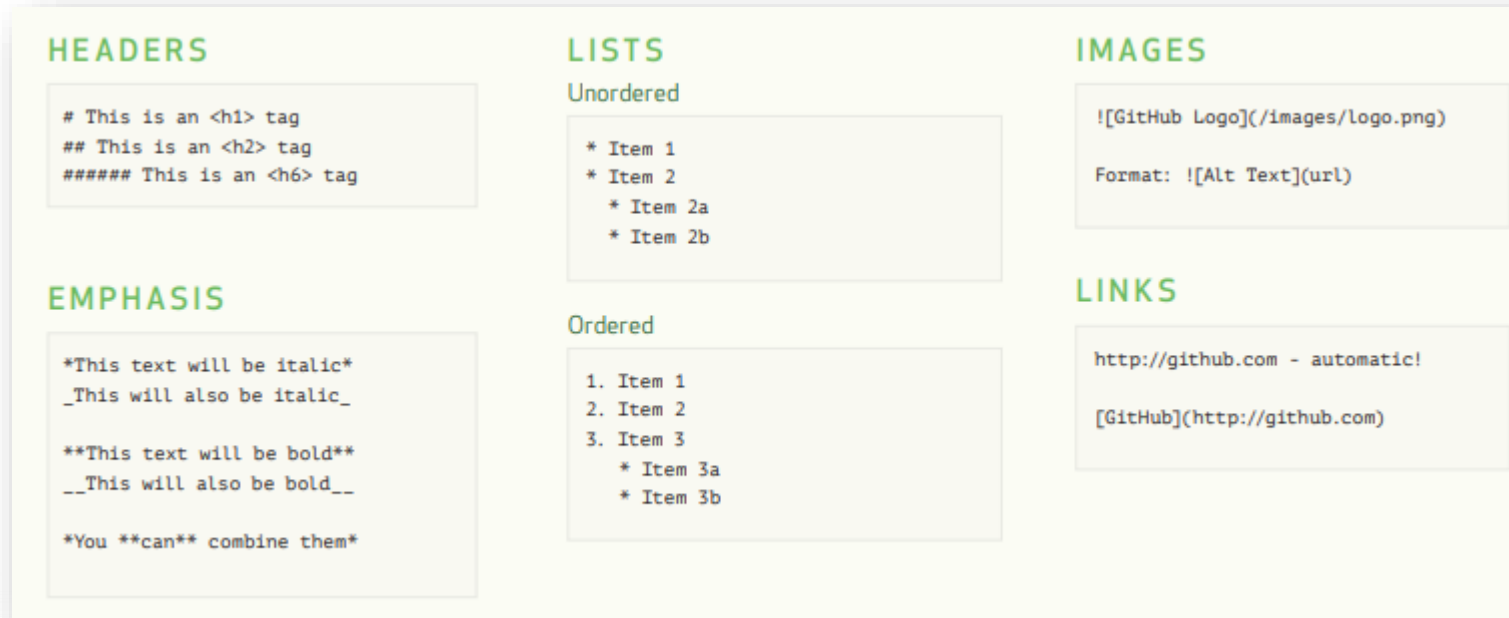
JupyterLab: Interface und Notebooks



Jupyter Notebook Cheat Sheet

https://cheatography.com/weidadeyue/cheat-sheets/jupyter-notebook/pdf_bw/

JupyterLab: Notebooks: Cell Types



Drei Cell Types:

- Code
 - Hier wird der Python Code geschrieben und ist dann ausführbar
- Markdown
 - In diesem Cell Type wird dokumentiert
 - Markdown ist eine einfache sog. Auszeichnungssprache für die Gliederung und Formatierung von Text
 - Markdown versteht auch LaTeX
- Raw (wird kaum benötigt)

Markdown Cheat Sheet

<https://www.heise.de/mac-and-i/downloads/65/1/1/6/7/1/0/3/Markdown-CheatSheet-Deutsch.pdf>

JupyterLab: Setup Working Directory

- Anaconda Prompt öffnen
- Zeile suchen
- Pfad eingeben
- Auskommentieren (# am Zeilenanfang entfernen)
- Speichern
- JupyterLab ausführen

```
C:\> Administrator: C:\WINDOWS\system32\cmd.exe  
  
(introduction-to-python) C:\Users\Ich>jupyter notebook --generate-config  
Writing default config to: C:\Users\Ich\.jupyter\jupyter_notebook_config.py  
  
(introduction-to-python) C:\Users\Ich>
```

```
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276
```

```
## Gets or sets a lower bound on the open file handles process  
# This may need to be increased if you run into an OSError: [Er  
# open files. This is not applicable when running on Windows.  
#c.NotebookApp.min_open_files_limit = 0  
  
## Dict of Python modules to load as notebook server extensions.  
# be used to enable and disable the loading of the extensions. T  
# will be loaded in alphabetical order.  
#c.NotebookApp.nbserver_extensions = {}  
  
## The directory to use for notebooks and kernels.  
#c.NotebookApp.notebook_dir = ''  
  
## Whether to open in a browser after starting. The specific bro  
# platform dependent and determined by the python standard libr  
# module, unless it is overridden using the --browser (Notebook  
# configuration option.  
#c.NotebookApp.open_browser = True  
  
## Hashed password to use for web authentication.  
#  
# To generate, type in a python/IPython shell:
```

Suchen und ersetzen

Suchen Ersetzen In Dateien suchen Hervorheben

Suchen nach: ☐

☐ In Auswahl

☒ Am Ende von vorne beginnen

Suchmodus

☒ Normal

☐ Erweitert (\n, \r, \t, \0, \x...)

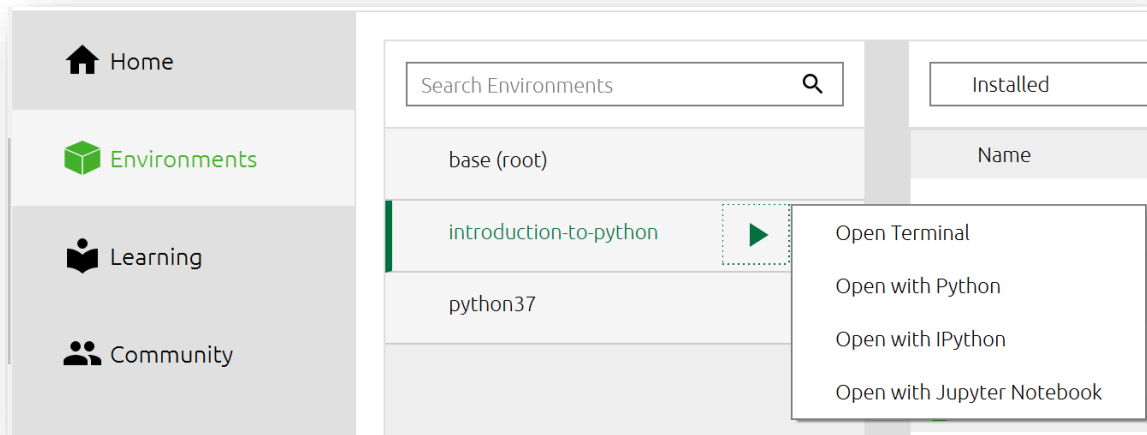
☐ Reguläre Ausdrücke ☐ . findet \r und \n

☒ Transparenz

☒ Wenn inaktiv

☐ Immer

Anaconda: Prompt und Installation von Packages



Wir öffnen ein Terminal für unsere VM und tippen ein:

```
pip install numpy
```

- Unserer VM liegt ein Terminal zugrunde
- Hierüber können z.B. Installationen von Python-Erweiterungen – sog. Packages – durchgeführt werden
- Mittels pip können Packages installiert werden

JupyterLab: Funktionen und Anwendung



- User Guide „JupyterLab Interface“:
<https://jupyterlab.readthedocs.io/en/stable/user/interface.html>
- User Guide „Jupyter Notebooks“:
<https://jupyterlab.readthedocs.io/en/stable/user/notebook.html>
- User Guide „Terminals“:
<https://jupyterlab.readthedocs.io/en/stable/user/terminal.html>
- User Guide „File and Output Formats“:
https://jupyterlab.readthedocs.io/en/stable/user/file_formats.html
- User Guide „Extensions“:
<https://jupyterlab.readthedocs.io/en/stable/user/extensions.html>

Spyder



“Spyder is a powerful scientific environment written in Python, for Python, and designed by and for scientists, engineers and data analysts. It features a unique combination of the advanced editing, analysis, debugging, and profiling functionality of a comprehensive development tool with the data exploration, interactive execution, deep inspection, and beautiful visualization capabilities of a scientific package.”

<https://docs.spyder-ide.org/current/first-steps-with-spyder.html>

PyCharm: eine IDE für Python

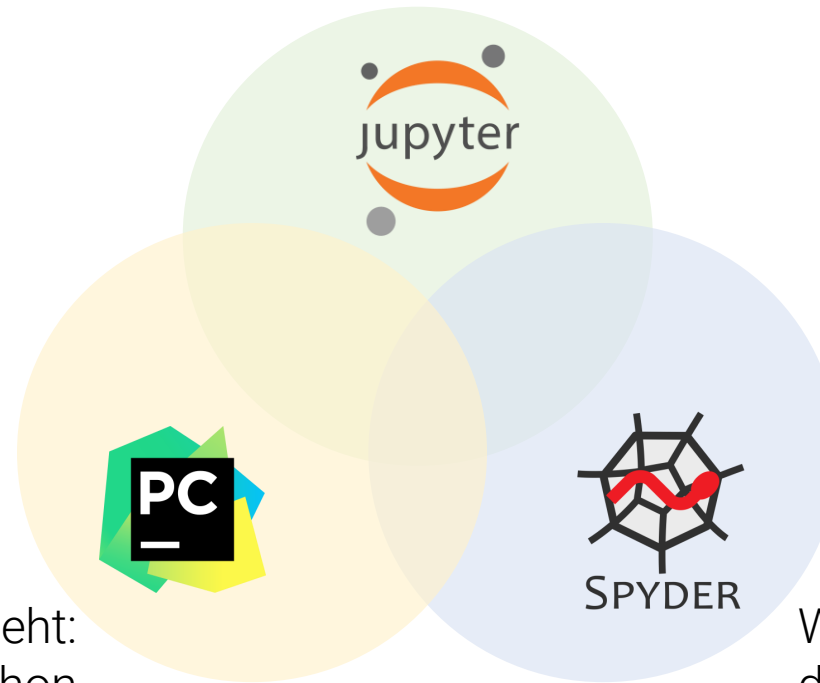


- PyCharm ist eine Integrierte Entwicklungsumgebung (IDE) für Python
- Im Gegensatz zu JupyterLab nutzen wir PyCharm, wenn wir längere Programme schreiben
→ also im Bereich der Softwareentwicklung

Was für was?

Wenn es um Freiheit und Kreativität geht:
Scientific Computing und **Datenexploration** mit Python

Wir nutzen hauptsächlich JupyterLab. Wenn wir größere Programme schreiben, dann steigen wir auf PyCharm um. Spyder nutzen wir nur in Ausnahmefällen und kann als „zur Vollständigkeit erwähnt“ betrachtet werden



Wenn es an's Eingemachte geht:
Softwareentwicklung mit Python

Wenn keine Zeit für Tests ist:
direktes **Debugging** mit Python

Fallback Lösung: Google Colab

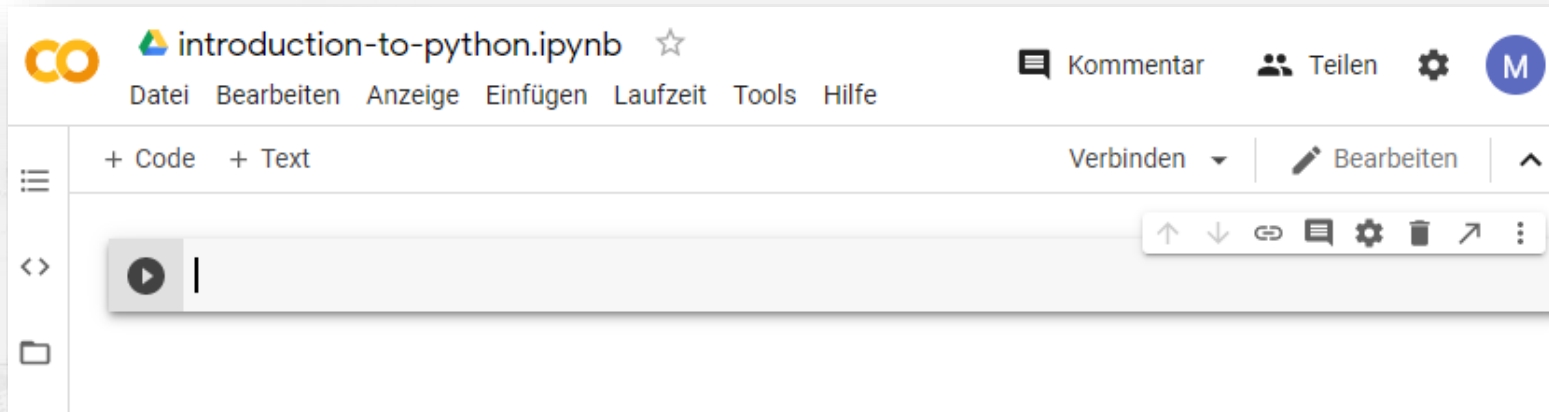


colab.research.google.com ▾ [Diese Seite übersetzen](#)

Google Colab

Colab notebooks allow you to combine executable code and rich text in a single document, along with images, HTML, LaTeX and more. When you create your ...

- Frei verfügbare Rechenressourcen
- Jupyter Notebooks direkt im Browser erstellen und ausführen
- Wenn es Probleme mit obigen Installationen gibt, dann kann Google Colab genutzt werden



Dateitypen: .py- und .ipynb-Dateien

 `hello_world.py` `getting-started.ipynb`

.py-Dateien

- Diese Endung bezeichnet Dateien, in denen Python-Source Code als Skript organisiert ist
- Es handelt sich hierbei um die originäre Dateiendung für Python-Programme
- Diese Dateien können von IDEs gelesen werden, die Python verstehen

.ipynb-Dateien

- Diese Endung bezeichnet Jupyter Notebook Dateien
- Sie werden im sogenannten JSON-Format abgespeichert
 - Aufgebaut wie ein Dictionary in Python
- Kann von Standard-IDEs meist nicht gelesen werden
 - Oft durch Plugins lesbar



Fragen?

Zu den Tools





Predictive Maintenance

Another BuzzWord?

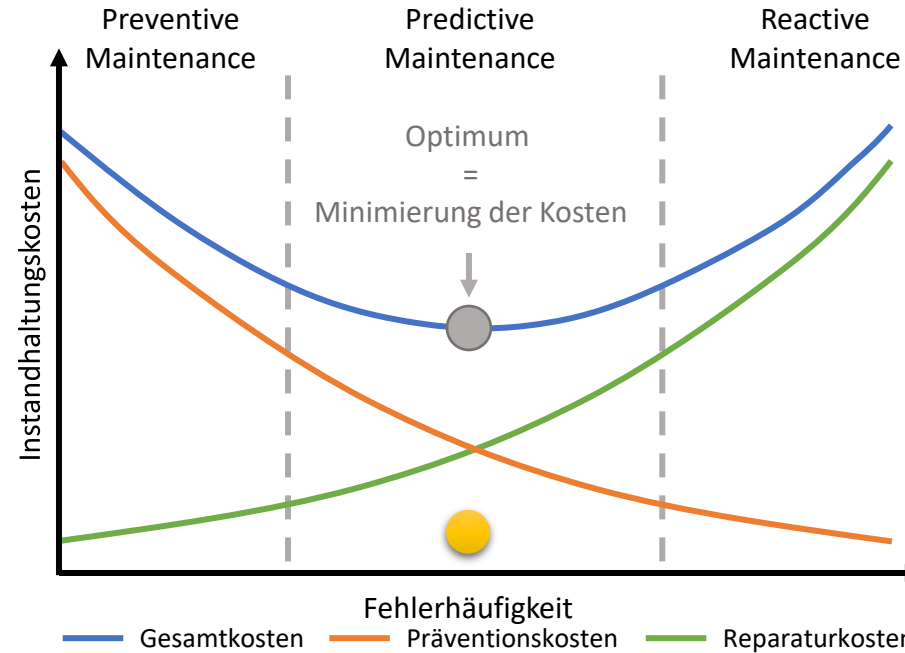
Begriffsklärung: Predictive Maintenance

Betriebsstunden: >5000



Preventive Maintenance

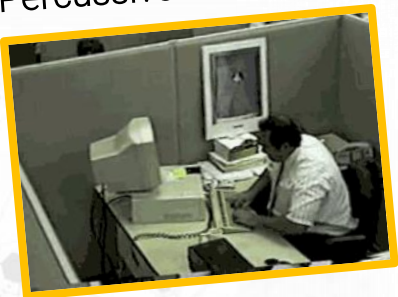
- Ist eine Routine zur regelmäßigen Überprüfung
- mit dem Ziel Probleme zu beheben bevor sie sich entwickeln.



Reactive Maintenance

- wird nach der Fehlererkennung durchgeführt
- und zielt darauf ab die ursprüngliche Funktionalität wieder herzustellen

Percussive Maintenance



Just in time



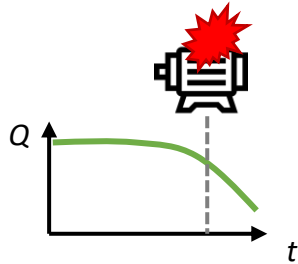
Predictive Maintenance

- lernt von historischen und gegebenenfalls in Echtzeit verfügbaren instandhaltungsrelevanten Daten.
- Durch die Prognose von zukünftigen Ereignissen kann die Frage „Was wird wann passieren?“ beantwortet werden.



Was denken Sie?
Wie minimieren wir diese Kosten?

Was ist Predictive Maintenance unter der Haube?

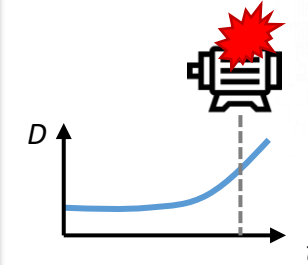


Remaining Useful Life (RUL) Prediction

- KI-Modell lernt den Abnutzungsvorrat einer Maschine vorherzusagen
- KI-Modell im Betrieb: gibt an wie lange der Abnutzungsvorrat noch ausreichen wird

Time to Failure (TTF) Prediction

- KI-Modell lernt die Zeit bis zum nächsten Fehler vorherzusagen
- KI-Modell im Betrieb: gibt an wann und wie wahrscheinlich der nächste Fehler sein wird

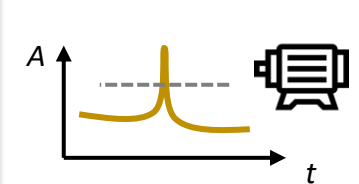


Fault Classification

- KI-Modell lernt Schadensklassen zu unterscheiden
- KI-Modell in Produktion meldet, ob Maschine im Normalzustand ist oder in einer bestimmten Schadensklasse

Anomaly Detection

- Anlernen von KI-Modellen an „Gutzeiträumen“
- KI-Modell in Produktion meldet Anomalien



Abgrenzung/Begriffsklärung: Predictive Maintenance zu... bzw. Verwechslungsgefahr

Condition-based Monitoring

- Behandelt die Zustandsüberwachung von – z.B. Maschinen – anhand der Überprüfung von Rohsignalen und/oder Kennwerten (oft KPIs genannt)
- Sobald eine prädiktive Komponente auftaucht, gruppieren wir das Verfahren unter den Begriff Predictive Maintenance

Visualisierung/visuelle Exploration

- Werden Daten visualisiert und anhand derer Schlüsse durch Menschen gezogen, dann spricht man von visueller Exploration
- Fehlt also eine komputative Komponente, die für uns das Schlussfolgern übernimmt, so betrachten wir dies nicht als PRM

0

So what?

- Das „Predictive“ in PRM bedeutet für uns, dass nicht wir, sondern der Computer Schlüsse zieht
- Wenn wir es tun – z.B. anhand der Interpretation von Grafiken – dann sind wir in der visuellen Datenexploration unterwegs
- D.h. PRM benötigt *per definitionem* komputative bzw. maschinelle Verfahren, die eigenständig Vorhersagen treffen können
→ Machine Learning ist Kern von Predictive Maintenance



Fragen?

Zu allem

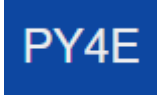


Quellen

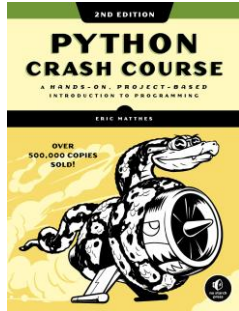


<https://i.gifer.com/g0bL.gif>

Weitere Quellen



www.py4e.com



Python Crash Course, 2nd Edition

A Hands-On, Project-Based Introduction to Programming
by Eric Matthes



[Document icons created by Yoteyo - Flaticon](https://www.flaticon.com/free-icons/document "document icons")