



Predictive Maintenance

Einführung

Kurz zu mir



Goldhacker, Markus
Prof. Dr. rer. nat.

Tel.: (0941) 943-5150

Fax: (0941) 943-1428

Homepage

[markus.goldhacker@oth-](mailto:markus.goldhacker@oth-regensburg.de)
[regensburg.de](http://www.oth-regensburg.de)

Fakultät Maschinenbau
Galgenbergstr. 30
Raum: B 001

Data Science in der
Automatisierungstechnik / Predictive
Maintenance

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)

Big Data, Blockchain und User Experience hören sich für Sie wie Fremdwörter an?
Das wollen wir ändern!

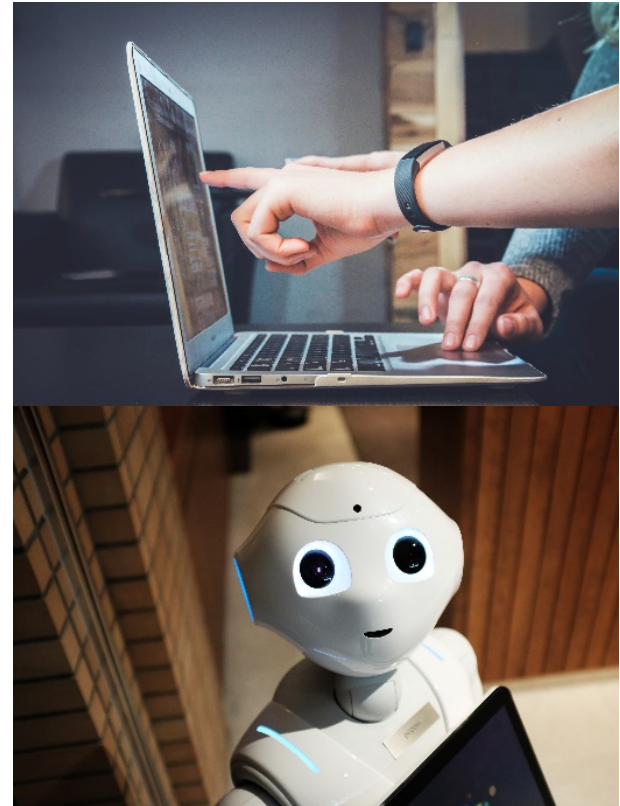
- Digitalisierung durchdringt alle Lebensbereiche
- Der Bedarf an Technologiekompetenzen in Deutschland steigt
- Neue Arbeitsformen und veränderte Tätigkeitsanforderungen verlangen ein verändertes Set an Fähigkeiten

Erwerben Sie mit der Regensburg School of Digital Sciences digitale Schlüsselkompetenzen!
Aktuell bieten wir für Studierende der OTH Regensburg Kurse in folgenden Bereichen an:

- Digitalisierung, Technologiefolgen und angewandte Ethik
- Digitale Transformation
- Data Science

Sprechen Sie uns bei Fragen gerne an. Weitere Informationen finden Sie hier:

www.rsds.info



Fotos: John Schnobrich (oben), Alex Knight (unten), beide über unsplash.com

First Step

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

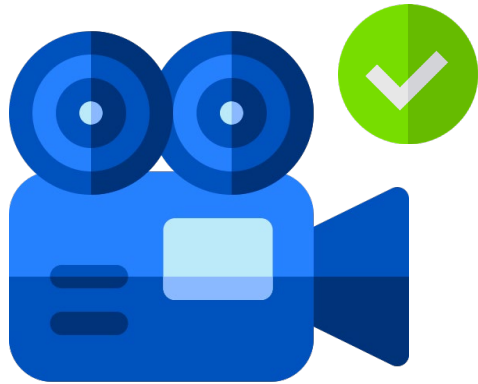
64-Bit (x86) Installer (477 MB)

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

Download Anaconda 3.8

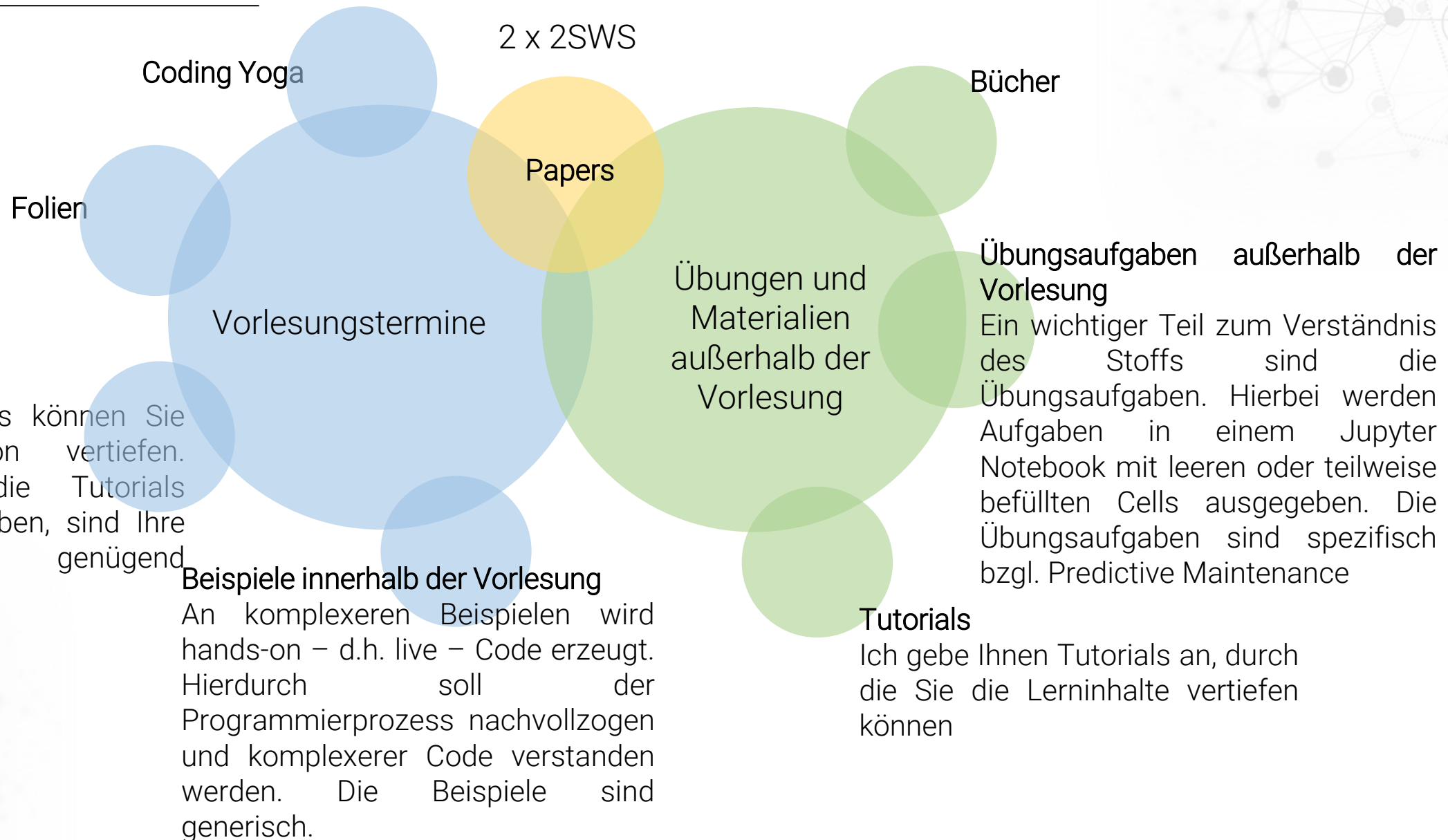
<https://www.anaconda.com/products/individual>

Aufzeichnung der Veranstaltung



- Die Veranstaltung wird aufgezeichnet, um Studierenden, die die Kurse nicht oder nur teilweise besuchen können, die Teilnahme zu ermöglichen
- Wenn Sie den Zoom-Meetings beitreten, dann erklären Sie sich damit einverstanden

Struktureller Aufbau dieser Vorlesung



Inhaltlicher Aufbau dieser Vorlesung

Supervised und
Unsupervised
Modelle alternieren

Unterschiedliche
„Flughöhen“

Es wird darauf abgezielt,
dass sowohl ohne
Mathe, als auch mit, ML
& KI verständlich wird

Algorithmenspezifisches
und ML-generisches

Wir werden generische
ML-Konzepte innerhalb
spezifischer Algorithmen
betrachten, aber sie
auch als generische
Konzepte verstehen

Theorie:
Klassische Machine
Learning Verfahren
und Beispiele

Anwendung:
Predictive
Maintenance
Übungsaufgaben

**Theorie und
Anwendung:**
Predictive
Maintenance Papers
– lesen, verstehen,
umsetzen

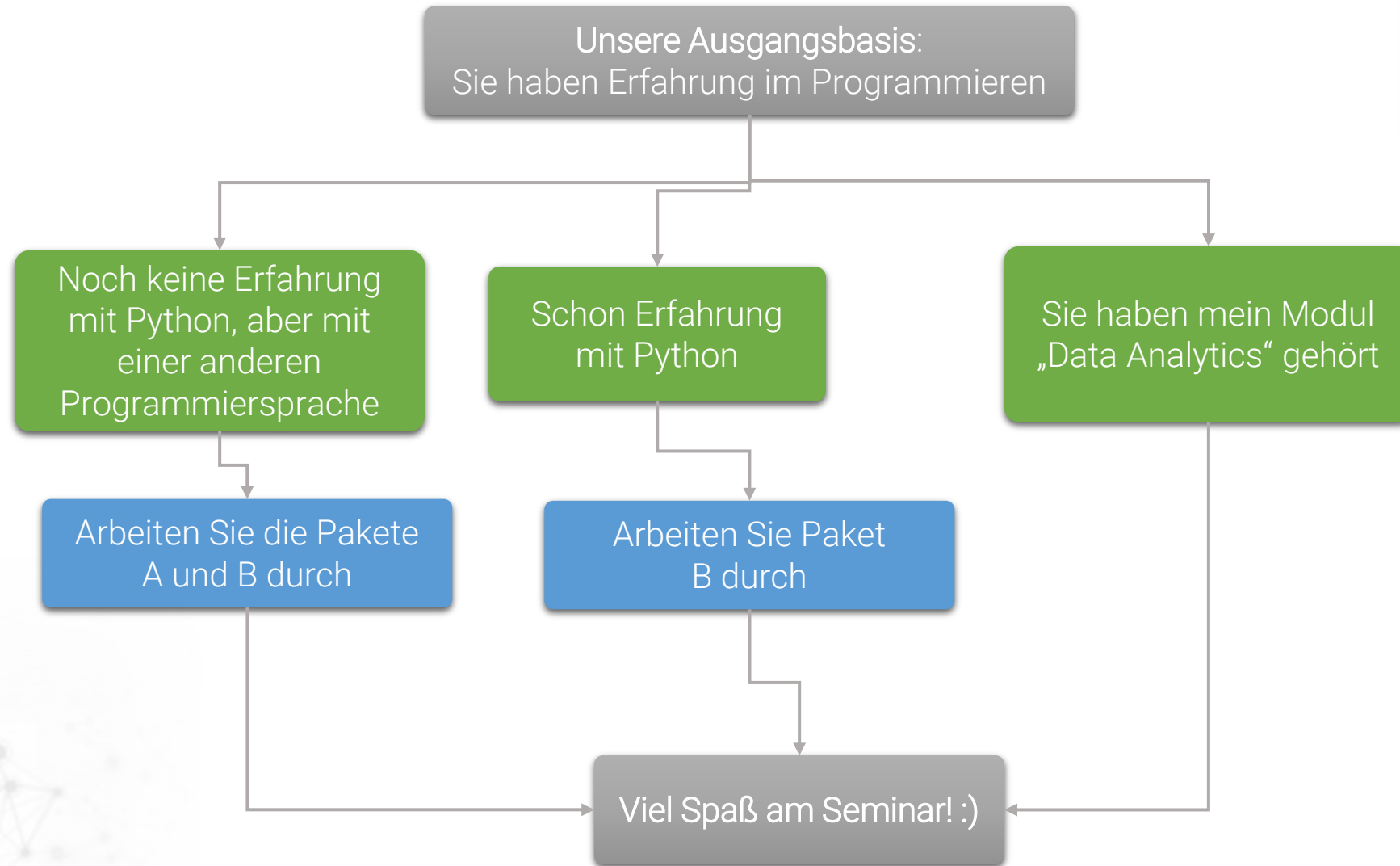
RUL, TTF, Fault
Classification,
Anomaliedetektion
erarbeiten wir uns vor
allem durch die
Übungsaufgaben

0

So what?

- Wir beschäftigen uns mit einem „Cutting-Edge“ Thema
- Daher ist es für uns wichtig in aktuelle Literatur zu schauen
- Und ML Methoden im Kontext der Predictive Maintenance zu verstehen und umzusetzen

Schätzen Sie sich selbst ein



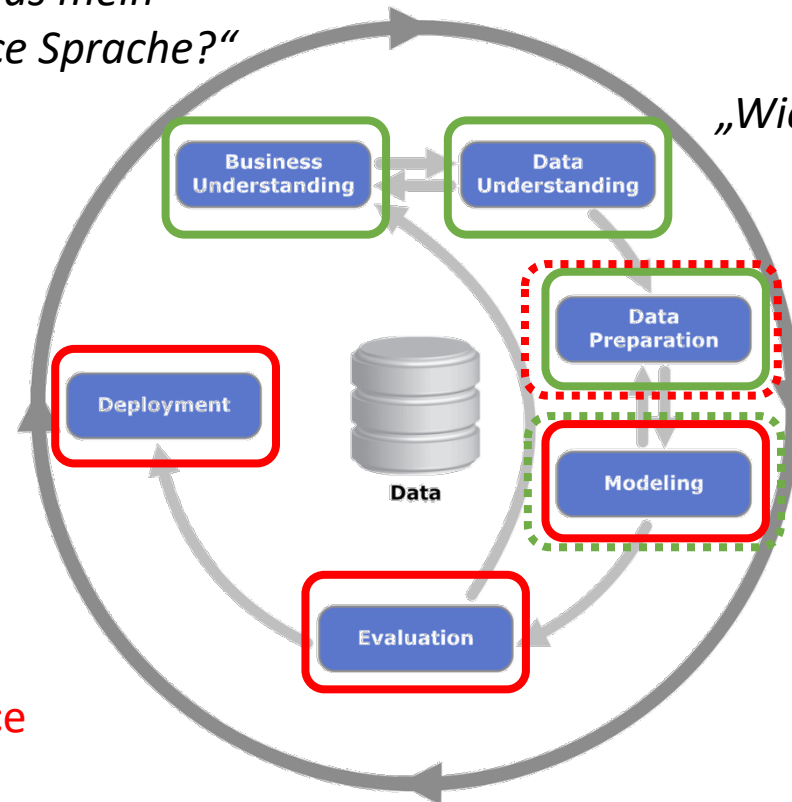
CRISP-DM als Grundlage

Data Analytics

„Wie übersetze ich das, was mein Kunde will, in Data Science Sprache?“

„Wie bringe ich mein Modell in die Infrastruktur des Kunden?“

Predictive Maintenance



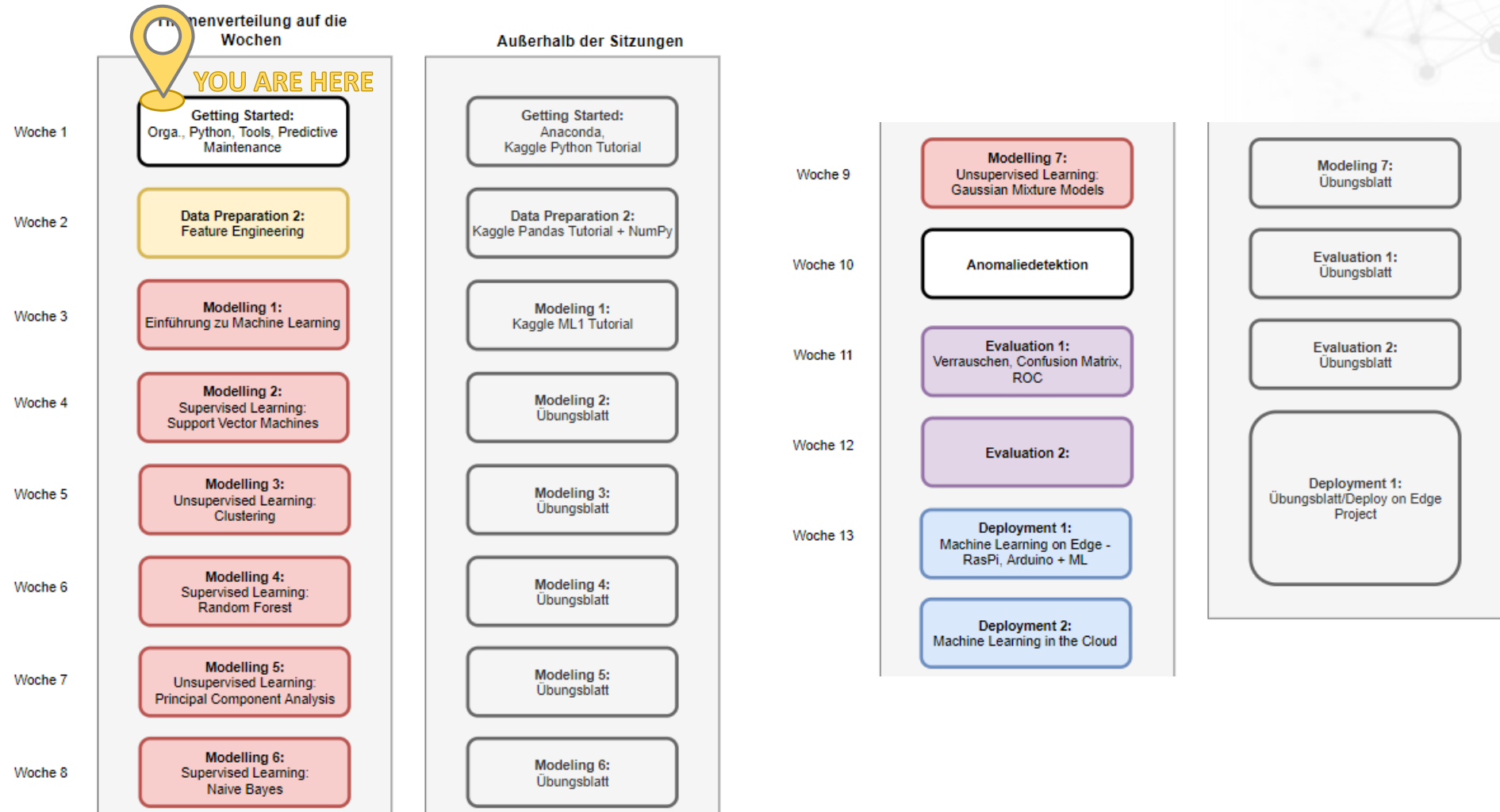
„Wie sehen meine Daten aus?“

„Wie muss ich meine Daten für die Modellierung vorbereiten?“

„Wie modelliere ich den Use Case?“

„Bildet mein Modell den Use Case genügend genau ab?“

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

0

So what?

Vor der Übung die Übungsaufgaben
(eigenständig!) durcharbeiten!

- Jede Woche wird ein Übungsblatt mit Aufgaben gestellt
- Das **Format**: JupyterNotebook
- Die Übungsaufgaben werden dann in den Übungsterminen **durchgearbeitet**
- Zoom-Links für Vorlesung und Übung ist derselbe
- Während der Übung verteilen wir uns auf Breakout Session – ca. 5-10min pro Aufgabe – dann besprechen wir die Aufgabe gemeinsam
- Gerne Screen-Share während Übung!

Prüfungsleistungen



Elektronische Prüfung

- Am Ende des Semesters gibt es eine elektronische Prüfung im CIP-Pool
 - Dauer: 90 Minuten
 - Hilfsmittel: keine Einschränkung
 - 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

Sie werden feststellen: im Data Science Bereich ist vieles Open Source und die Community ist sehr hilfsbereit

→ Viele frei verfügbare Bücher!



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

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



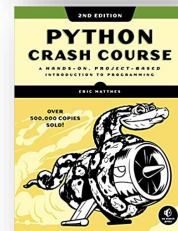
Allen B. Downey. *Think Stats: Exploratory Data Analysis*. O'Reilly UK Ltd. 2014.

→ <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.

GRIPS 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

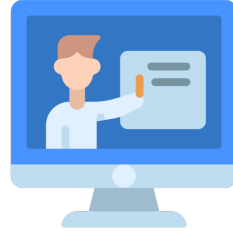


User Guide „JupyterLab Interface“



User Guide „JupyterNotebooks“

Unsere Kommunikationsformen



Regelmäßige Teilnahme an der Vorlesung +
Fragen unbedingt erwünscht!

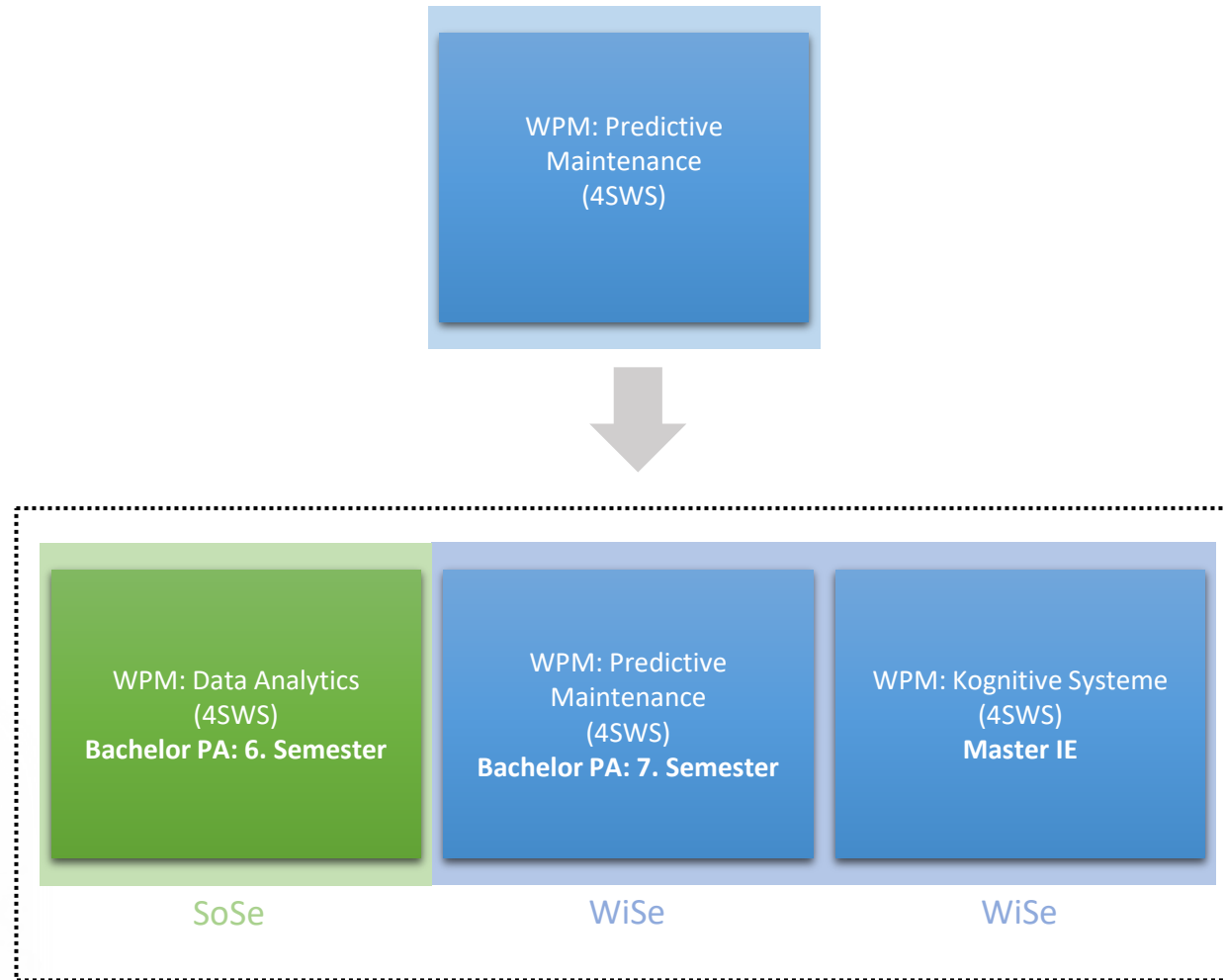
Regelmäßige Teilnahme an den Übungen +
aktive Teilnahme!



Fragen & Antworten

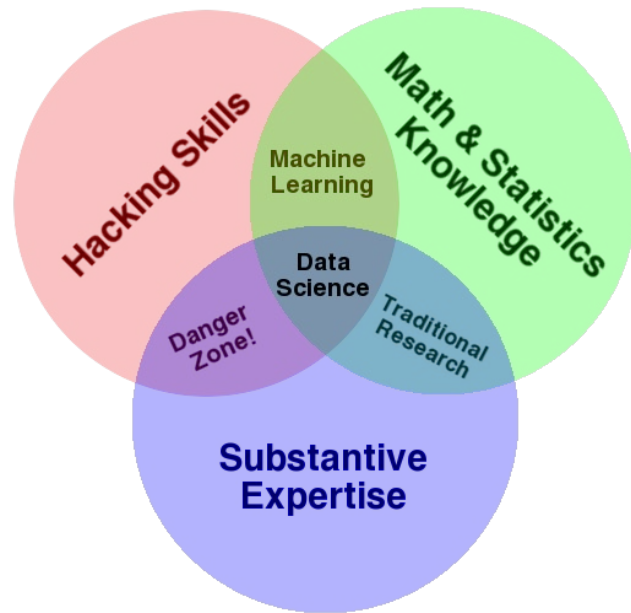
Außerhalb der Veranstaltungen: kommen Sie
auf mich zu, wenn es Fragen gibt. Am besten
über unser „Fragen & Antworten“-Forum
→ Gerne auch Diskussionen untereinander!

Einbettung in zukünftige Lehrveranstaltungen



Disclaimer
Under Construction: hierbei handelt es sich um vorläufige Planungen, die sich noch ändern können!

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



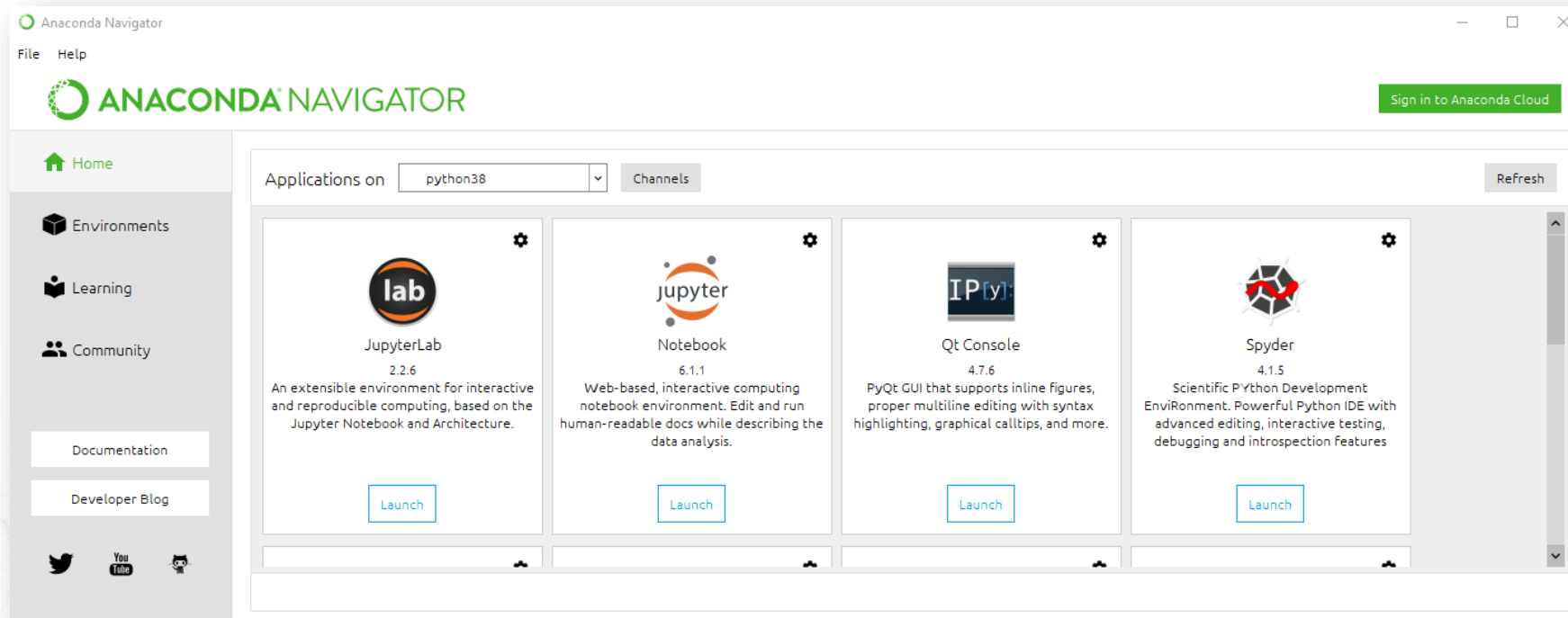
Anaconda



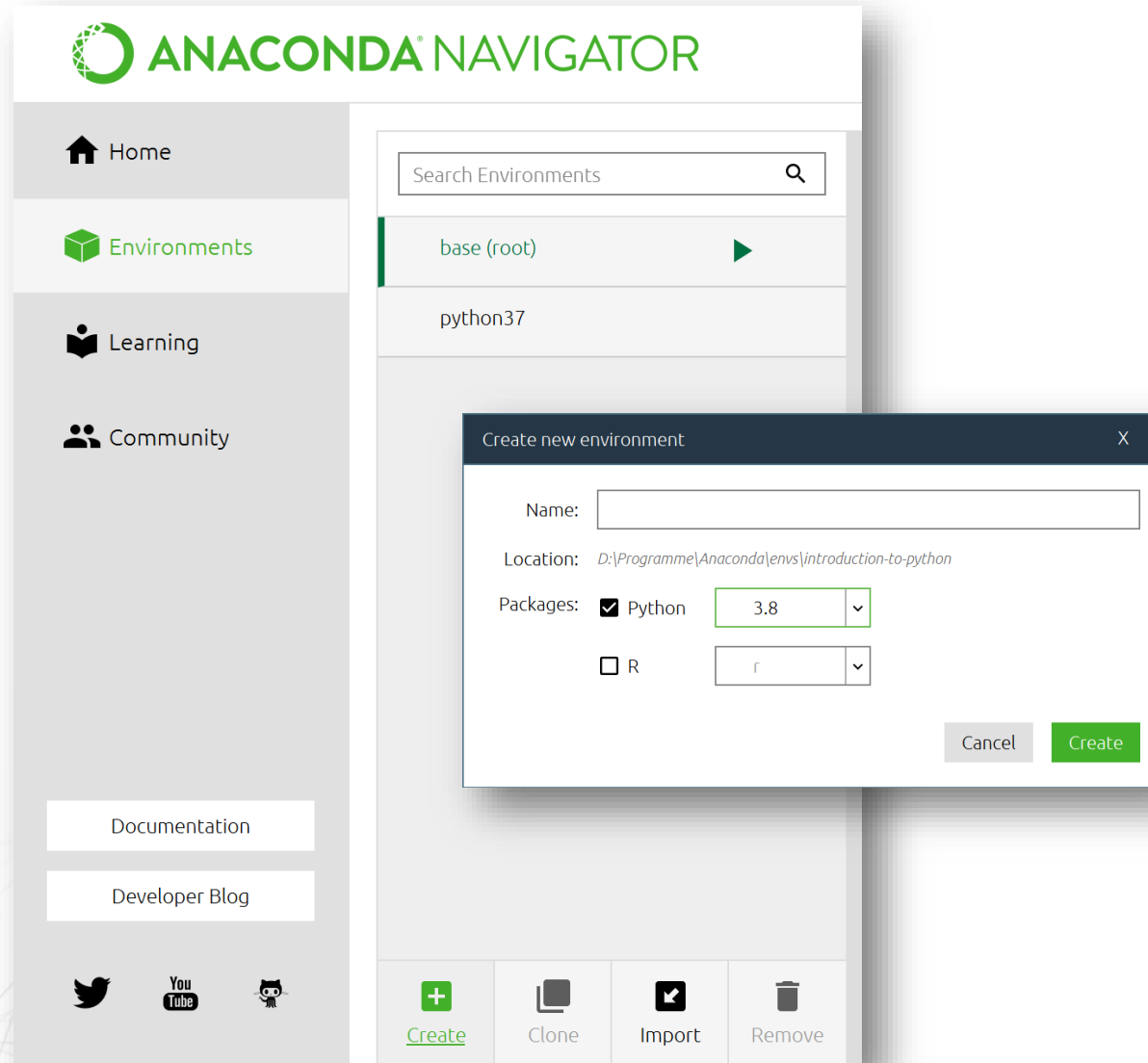
- Anaconda ist eine Open-Source-Distribution für die Programmiersprachen **Python** und R
- Vereinfacht Installation/Nutzung von Tools und Packages

Anaconda: „Single Point of Truth“

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



Anaconda: Virtual Environment

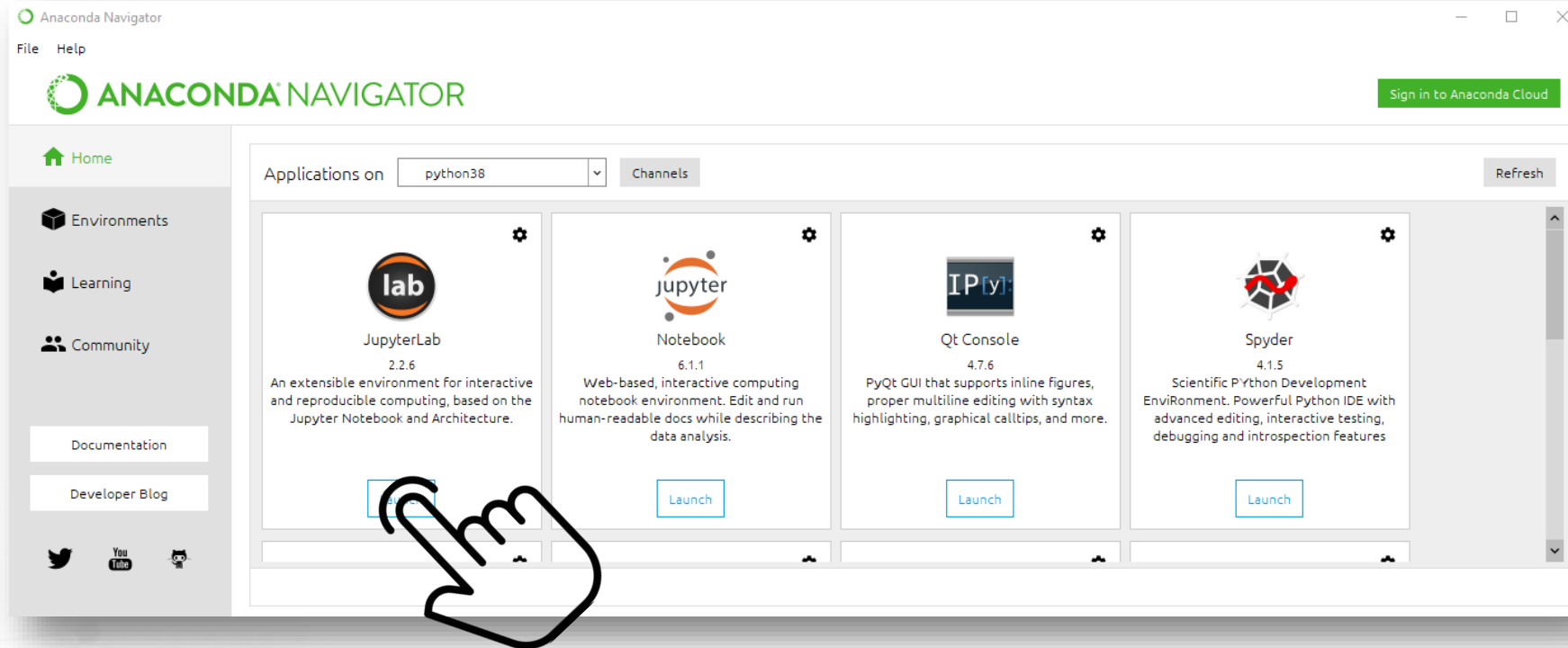


- Virtual Environments stellen abgeschlossene Programmierumgebung dar
- Es macht Sinn VEs projekt- bzw. aufgabenbasiert zu erzeugen
- Anaconda bietet einfache GUI hierfür

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)

Anaconda: „Single Point of Truth“



Wir installieren

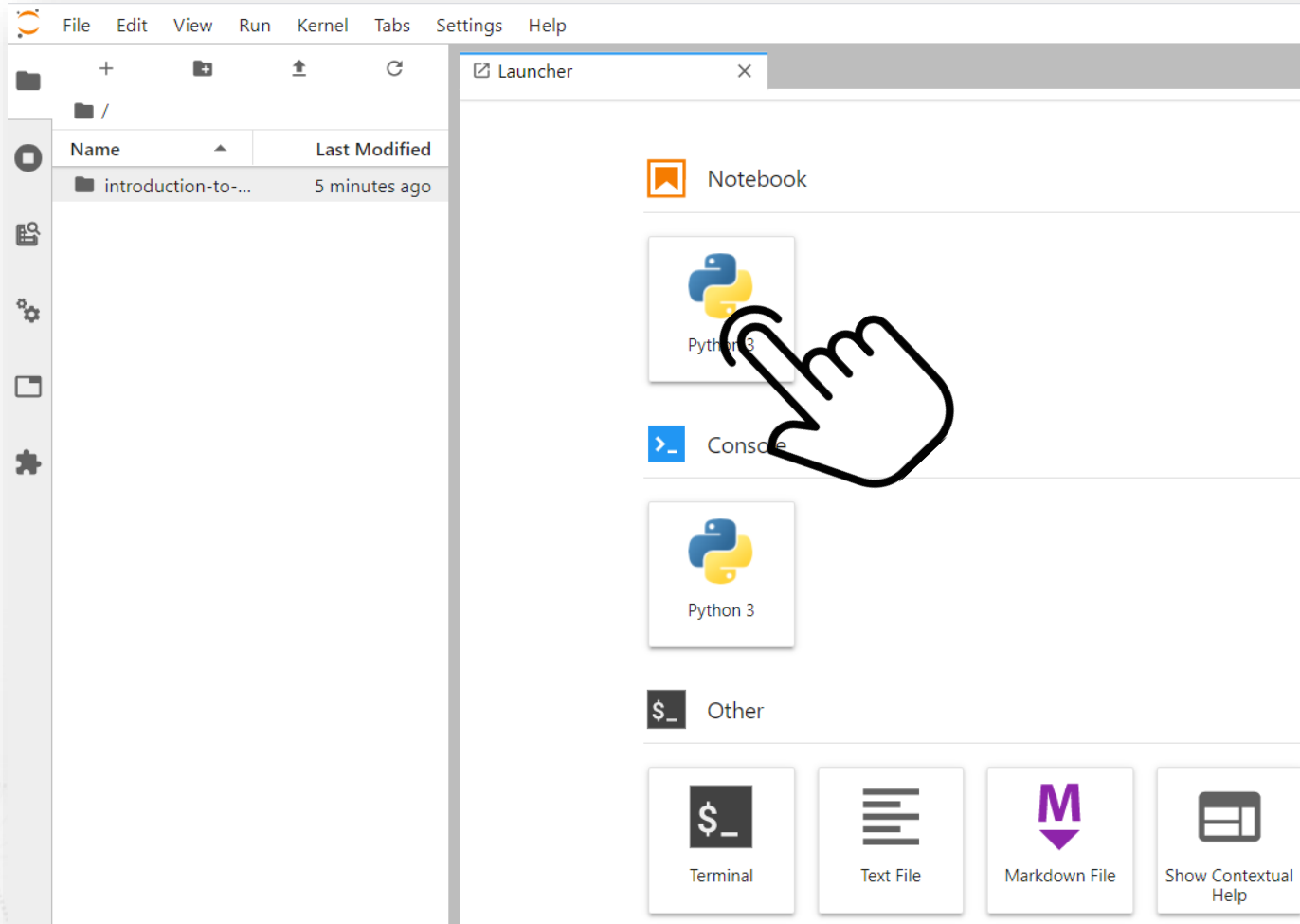
- Spyder
- JupyterLab

JupyterLab/Notebook



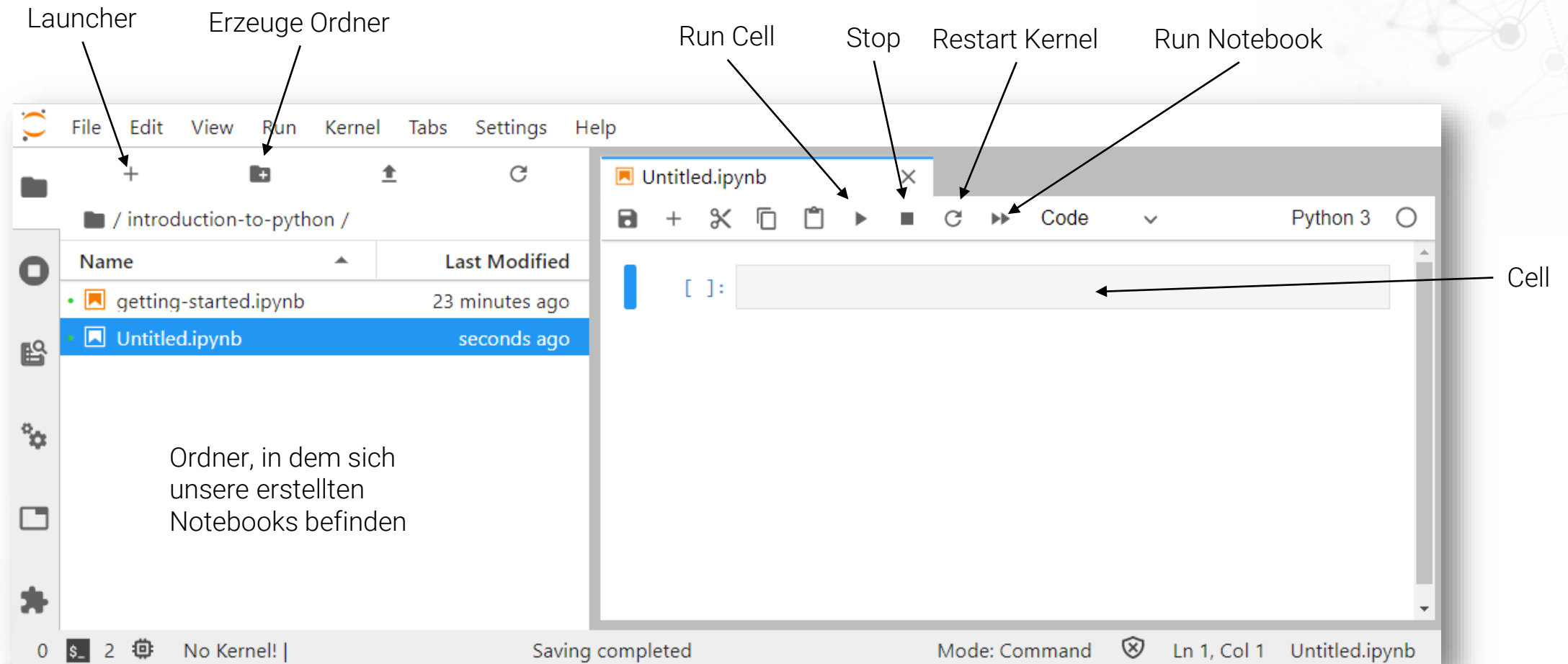
- Jupyter Notebook ist eine web-basierte interaktive Umgebung, mit der Jupyter-Notebook-Dokumente erstellt werden können
- Ein Jupyter-Notebook-Dokument ist ein JSON-Dokument mit einem versionsabhängigen Schema, das aus einer Liste von Eingabe- und Ausgabezellen besteht, die jeweils Code, Text und Plots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

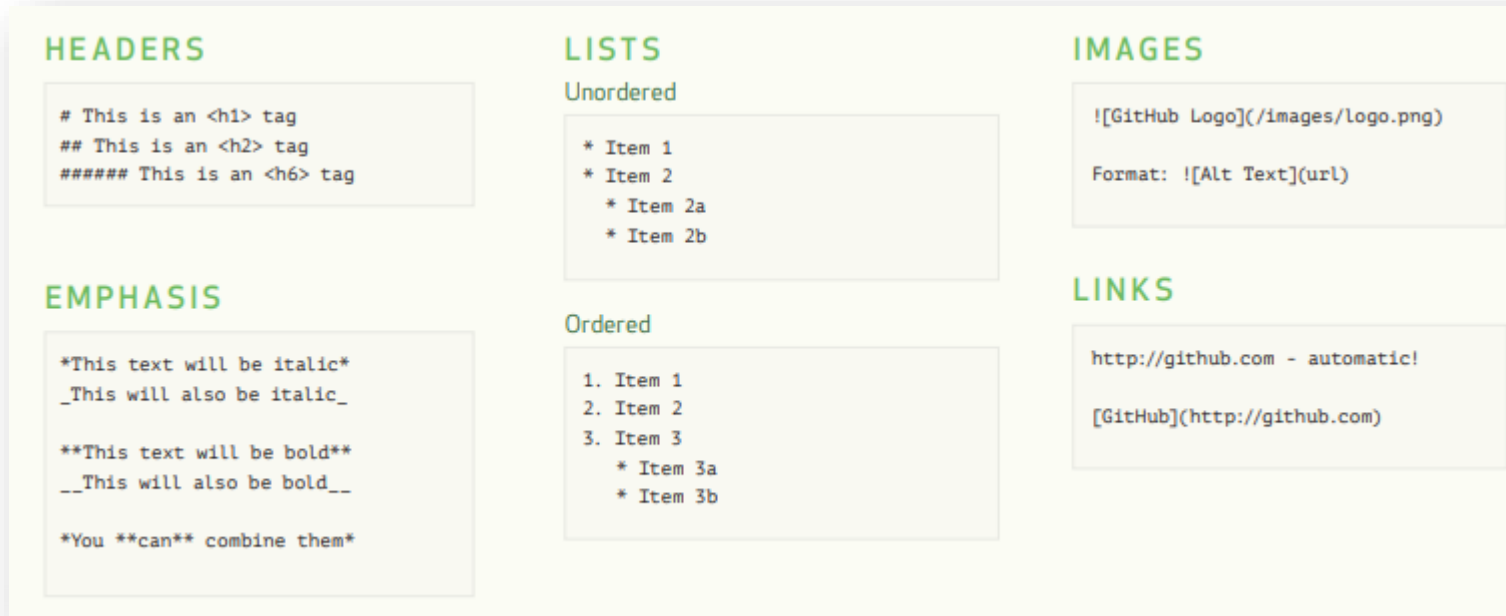
JupyterLab: Interface und Notebooks



Jupyter Notebook Cheat Sheet

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

JupyterLab: Notebooks: Cell Types



HEADERS

```
# This is an <h1> tag
## This is an <h2> tag
##### This is an <h6> tag
```

EMPHASIS

```
*This text will be italic*
_This will also be italic_

**This text will be bold**
__This will also be bold__

*You **can** combine them*
```

LISTS

Unordered

```
* Item 1
* Item 2
  * Item 2a
  * Item 2b
```

Ordered

```
1. Item 1
2. Item 2
3. Item 3
  * Item 3a
  * Item 3b
```

IMAGES

```
![GitHub Logo](/images/logo.png)

Format: ![Alt Text](url)
```

LINKS

```
http://github.com - automatic!

[GitHub](http://github.com)
```

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 ## Gets or sets a lower bound on the open file handles process
256 # This may need to be increased if you run into an OSError: [Er
257 # open files. This is not applicable when running on Windows.
258 #c.NotebookApp.min_open_files_limit = 0
259
260 ## Dict of Python modules to load as notebook server extensions.
261 # be used to enable and disable the loading of the extensions. T
262 # will be loaded in alphabetical order.
263 #c.NotebookApp.nbserver_extensions = {}
264
265 ## The directory to use for notebooks and kernels.
266 #c.NotebookApp.notebook_dir = ''
267
268 ## Whether to open in a browser after starting. The specific bro
269 # platform dependent and determined by the python standard libr
270 # module, unless it is overridden using the --browser (Notebook
271 # configuration option.
272 #c.NotebookApp.open_browser = True
273
274 ## Hashed password to use for web authentication.
275 #
276 # 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

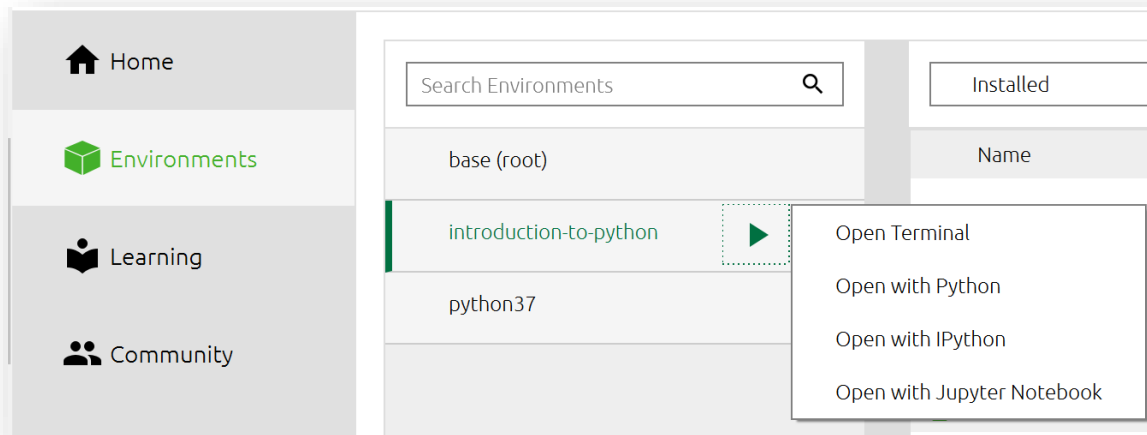
☒ Transparenz

☒ Wenn inaktiv

☐ Immer

```
264
265 ## The directory to use for notebooks and kernels.
266 c.NotebookApp.notebook_dir = 'D:/code/git'
267
```

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

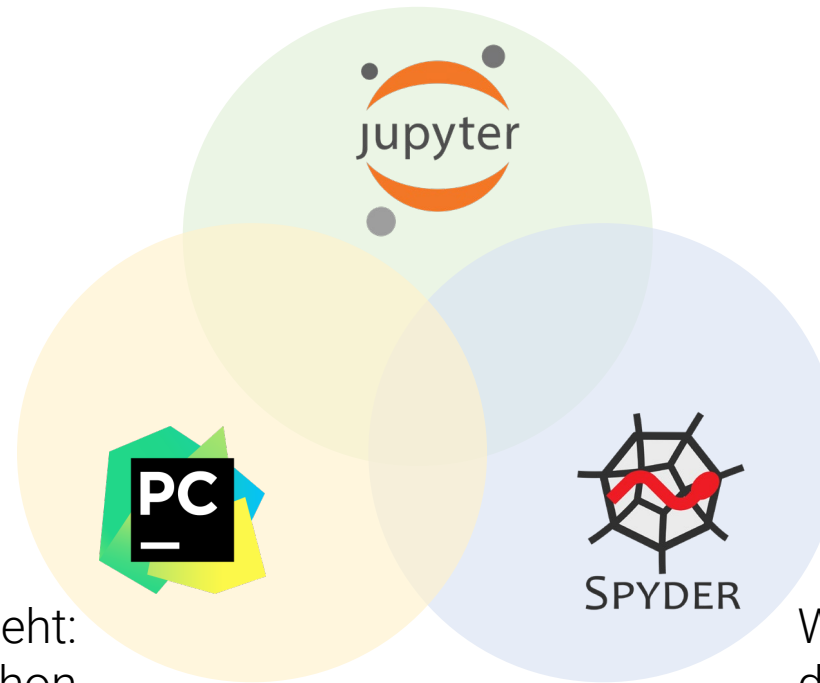


- PyCharm ist eine Integrierte Entwicklungsumgebung (IDE) für Python
- Im Gegensatz zu JupyterLab nutzen wir PyCharm, wenn wir längere Programme schreiben
→ also im Bereich der Softwareentwicklung

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

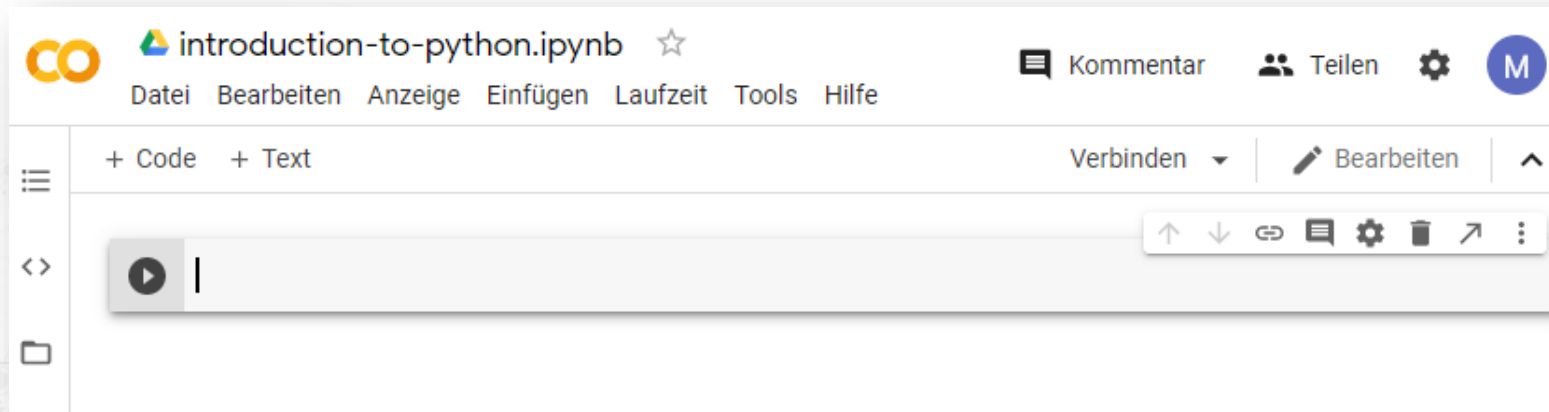


colab.research.google.com ▾ [Diese Seite übersetzen](#)

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`

.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

 `getting-started.ipynb`

.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?

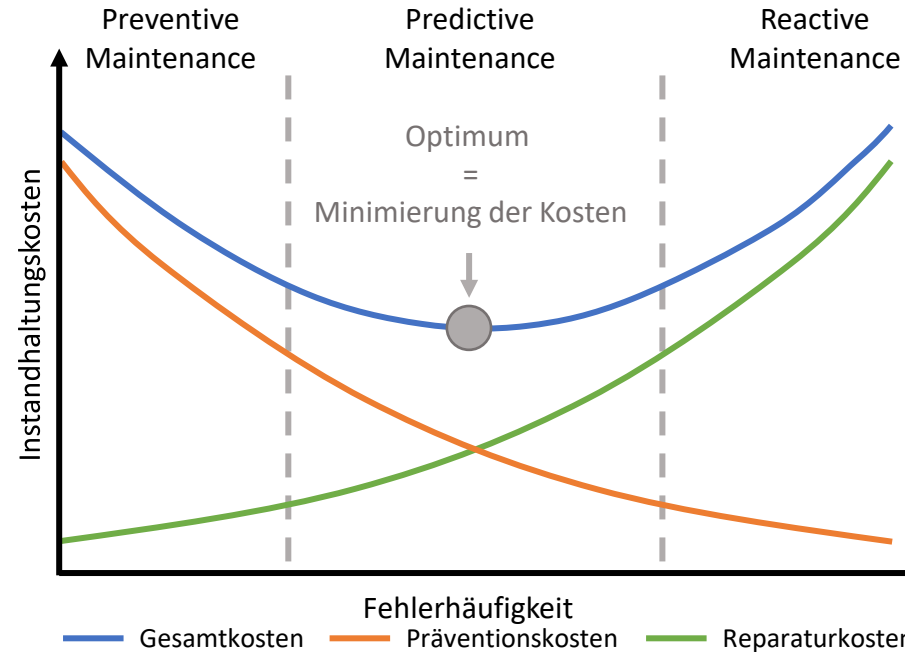
Begriffsklärung: Predictive Maintenance

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

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

- Regressiv

Time to Failure (TTF) Prediction

- Regressiv

Fault Classification

- Klassifikation

Anomaly Detection

- Unsupervised

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 für uns 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* computationale bzw. maschinelle Verfahren, die eigenständig Vorhersagen treffen können
→ Machine Learning ist Kern von Predictive Maintenance

Umfragen



<https://www.menti.com>

- Da es sich um ein neues Modul handelt, das sich im Aufbau befindet würde ich Sie gerne öfter die verschiedensten Fragen zu den Terminen stellen
 - Für jede Umfrage finden Sie einen Code im GRIPS
- Sie helfen mir durch Ihre Antworten den Kurs anzupassen und für zukünftige Studenten die richtige Richtung vorzugeben!

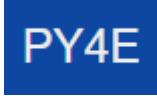


Fragen?

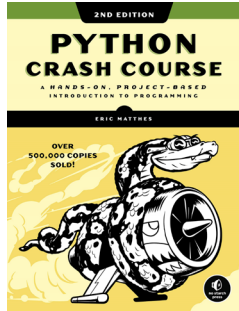
Zu allem



Weitere Quellen



www.py4e.com



Python Crash Course, 2nd Edition

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