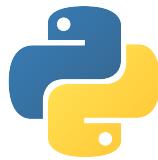


Handout zum Python-Workshop des WiChem Kiel e.V.

Niclas Grocholski & Konstantin Köster

12. Januar 2023



1 Erläuterungen zum Handout

Dieses Handout hilft dir als Ausgangspunkt für deine ersten Schritte mit Python, einen Einblick in die Geschichte der Programmiersprache zu erhalten (Abschnitt 2), notwendige (und nicht notwendige, aber nützliche) Programme und Tools kennenzulernen und zu installieren (Abschnitt 3) sowie über den Workshop hinausgehende Inhalte in Form von Lernmedien verschiedenster Arten an die Hand zu bekommen (Abschnitt 4).

Formatierungen im Handout sind wie folgt gewählt:

- ▶ externe Links sind als **grüner Text** oder in entsprechenden Bildern hinterlegt,
 - ▶ Schaltflächen in Programmen und Tasten auf der Tastatur deines Computers sind als **Knöpfe** dargestellt,
 - ▶ Einzugebender Text ist als **roter Text in Festbreitenschrift** dargestellt. Komponenten mit **(WINKELKLAMMERN IN GROSSBUCHSTABEN)** müssen bei der Eingabe dem Inhalt entsprechend ersetzt werden,
 - ▶ besonders wichtige Anmerkungen zu den Installationen stehen in
- !** grünen Boxen.

2 Eine kleine Einführung in Python

2.1 Was ist Python?

Python ist eine interpretierte Skriptsprache, die in den späten 1980er Jahren von Guido van Rossum am Nationalen Forschungsinstitut für Mathematik und Informatik in den Niederlanden entwickelt wurde. Die erste Version von Python wurde 1991 veröffentlicht und Version 1.0 wurde 1994 herausgegeben. Python 2.0 wurde im Jahr 2000 veröffentlicht und die 2.x-Versionen waren bis Dezember 2008 die vorherrschenden Versionen. Zu diesem Zeitpunkt entschied sich das Entwicklerteam für die Version 3.0, die einige kleine, aber wichtige Änderungen enthielt, die nicht abwärtskompatibel mit den 2.x-Versionen waren. Python 2 und 3 sind sich sehr ähnlich und einige Funktionen von Python 3 wurden in Python 2 zurückportiert. Aber im Allgemeinen sind sie weiterhin nicht ganz kompatibel. Für Python 2 wurde ein offizielles End-of-Life-Datum für den 1. Januar 2020 festgelegt, seitdem es nicht mehr weiterentwickelt wird.

Der Name Python leitet sich übrigens nicht von der Schlange ab, sondern von der britischen Komikertruppe Monty Python's Flying Circus, deren Fan Guido war und vermutlich immer noch ist. In der Python-Dokumentation finden sich immer wieder Verweise auf Sketche und Filme von Monty Python.

2.2 Warum genau Python?

Wenn du Programme schreiben willst, gibt es buchstäblich Dutzende gängiger Programmiersprachen, aus denen du wählen kannst. Warum solltest du daher Python wählen? Hier einige Merkmale, die Python zu einer attraktiven Wahl machen:

1. Python ist **beliebt** und hat eine **aktive Community**

Python hat in den letzten Jahren stark an Popularität gewonnen. Führende Unternehmen und Softwarefirmen auf der ganzen Welt setzen auf Python, darunter Facebook, Google, Netflix, Instagram und andere und auch in der akademischen Welt setzt sich Python zunehmend durch. Quantitativ ausgedrückt: mit einem Wachstum von 8.7 % über die letzten fünf Jahre und einem Gesamtanteil von über 28 % ist Python im Dezember 2022 auf Platz 1 des *PopularitY of Programming Language Index*.

Kein Programmierer ist eine Insel; er ist auf Dokumentationen und Unterstützung angewiesen, damit er bei unerwarteten Problemen oder neuen Fragestellungen eine Anlaufstelle hat, die Antworten bietet. Python wird von einem extrem großen Ökosystem von Bibliotheken und Paketen unterstützt, was es oft zur ersten Wahl für neue Entwickler macht. Die Python-Community umfasst Entwickler aller Qualifikationsstufen und bietet einfachen Zugang zu Dokumentationen, Anleitungen, Tutorials und vielem mehr. Gleichzeitig ist die Python-Community äußerst aktiv. Wenn Entwickler unter Zeitdruck stehen und dringend Hilfe benötigen, können sie mit der Community zusammenarbeiten, um schnelle und effektive Lösungen zu finden.

2. Python ist **kostenlos**

Der Python-Interpreter wird unter einer von der „Open Source Initiative“ (OSI) genehmigten Open-Source-Lizenz entwickelt und kann daher frei installiert, verwendet und weitergegeben werden — auch für kommerzielle Zwecke. Es gibt eine Version des Interpreters für praktisch jede Plattform, die es gibt, einschließlich aller Varianten von Unix, Windows, macOS, Smartphones und Tablets und wahrscheinlich alles andere, von dem du jemals gehört hast. Es gibt sogar eine Version für das halbe Dutzend Leute, die noch OS/2 benutzen.

3. Python ist **einfach zu lernen**

Eine der größten Hürden für diejenigen, die in die Programmierung einsteigen wollen, ist, dass Programmiersprachen wirklich ihre eigenen Sprachen sind; sie haben ihre eigenen Regeln, ihren eigenen Syntax, ihre eigenen grammatischen Strukturen usw. und sie erfordern oft das Erlernen eines völlig neuen Vokabulars. Python aber ist anders. Wie kaum eine andere Programmiersprache liest und schreibt sich Python sehr ähnlich wie die englische Standardsprache. Es verwendet einen vereinfachten Syntax mit dem Schwerpunkt auf natürlicher Sprache, was Anfängern das Erlernen der Sprache erleichtert.

4. Python ist **flexibel**

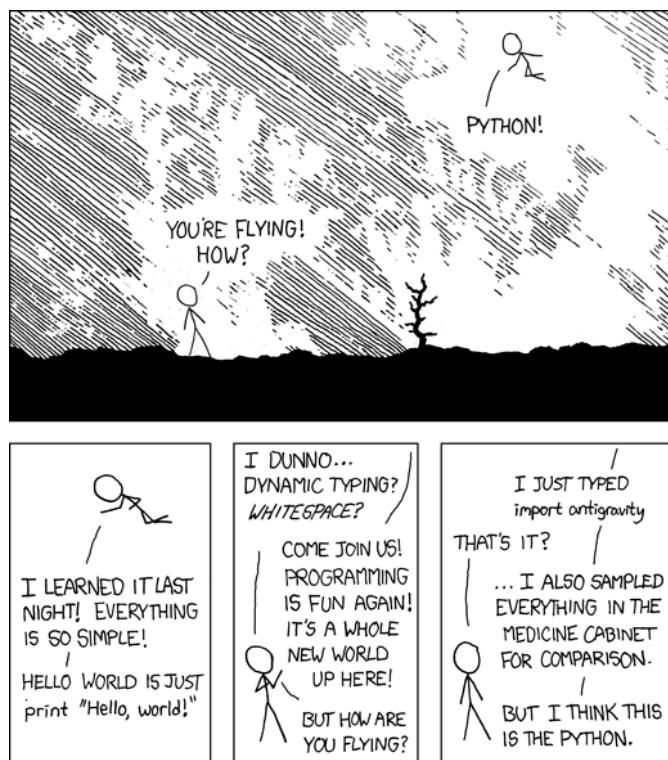
Python wird oft als Allzweckprogrammiersprache bezeichnet. Das bedeutet, dass Python im Gegensatz zu domänen-spezifischen Sprachen, die nur für bestimmte Anwendungstypen konzipiert sind, für die Entwicklung fast aller Arten von Anwendungen in allen Branchen und Bereichen verwendet werden kann.

Python wird mit großem Erfolg in den Bereichen Webentwicklung, Data Science, Data Engineering und sogar maschinellen Lernens und künstlicher Intelligenz eingesetzt. Unterstützt von einer Reihe von Frameworks und Bibliotheken gibt es praktisch keine Programmieraufgabe, die Python nicht bewältigen kann.

5. Python ist **effizient, schnell und zuverlässig**

Gelegentlich fragt sich ein Entwickler, der sich auf eine andere Programmiersprache spezialisiert hat, „Warum ist Python langsam?“ — und ja, im Vergleich zu einigen anderen Sprachen wie Java, C#, Go, JavaScript oder C++ ist Python oft etwas langsamer in der Ausführung — in der heutigen Welt ist die Entwicklungszeit eines Programms jedoch deutlich wichtiger als die Laufzeit des Computers. Und was die Zeit der Entwicklung bis zur Marktreife angeht, ist Python einfach unschlagbar.

Darüber hinaus ist Python effizient, zuverlässig und ermöglicht es Entwicklern, mit einem Minimum an Aufwand leistungsstarke Anwendungen zu erstellen, sodass Projekte, die in Python geschrieben wurden, mit Anwendungen mithalten, die in anspruchsvolleren Sprachen geschrieben wurden.



3 Installations- und Nutzungsanleitungen

3.1 Der Python Interpreter

3.1.1 Was ist ein Python Interpreter?

Ein Interpreter ist ein Programm, das andere Programme ausführt. Wenn du Python-Programme ausführst, findet ein mehrstufiger Prozess im Hintergrund statt. Der Interpreter wandelt den von dir geschriebenen Quellcode in eine ZwischenSprache, den sogenannten *bytecode*, um, die wiederum in die native Sprache („Maschinensprache“) übersetzt wird, die vom Computer ausgeführt werden kann.

3.1.2 Installationsanleitung



In diesem Abschnitt wird die Installation des Python-Basisinterpreters beschrieben. Zur Installation des Interpreters über Anaconda vergleiche Abschnitt 3.2.2.

Python ist eine Open-Source Programmiersprache, der durch Updates regelmäßig neue Funktionen hinzugefügt werden. Die aktuellste Python-Version ist 3.11.

Auf vielen Systemen ist Python bereits vorinstalliert. Du kannst versuchen, in einer Konsole¹ den Python-Befehl `py --version` auszuführen, um zu überprüfen, ob bereits ein Interpreter installiert ist. Ist Python installiert, erscheint eine Antwort, die die aktuell installierte Versionsnummer enthält. Wenn die Versionsnummer Python 2.X.Y lautet (wobei X und Y beliebige Zahlen sind), verwendest du Python 2, das nicht mehr unterstützt wird und für die Entwicklung neuen Softwarecodes nicht geeignet ist. Du musst dann Python ebenso neu installieren, wie wenn keine Installation auf deinem Computer vorhanden wäre.

Im Folgenden wird die Installation für **Windows** erklärt. Online finden sich Anleitungen zur Installation von Python auf **macOS**, **Linux**, **iOS** und **Android**. Unter Windows gibt es zwei Installationsmethoden:

1. **Der Microsoft Store:** Die einfachste Installationsmethode unter Windows ist die Installation über die Microsoft Store-App.
2. **Das vollständige Installationsprogramm** (empfohlen): Bei diesem Ansatz wird Python direkt von der Website [python.org](https://www.python.org) heruntergeladen.

Die beiden offiziellen Python-Installationsprogramme für Windows sind nicht identisch. Das Microsoft Store-Paket hat einige wichtige Einschränkungen. Das Microsoft Store-Paket ist „hauptsächlich für den interaktiven Gebrauch bestimmt“. Das heißt, das Microsoft Store-Paket ist für Personen gedacht, die Python zum ersten Mal verwenden. Das Microsoft Store-Paket hat Einschränkungen, die es für eine professionelle Entwicklungsumgebung ungeeignet machen. Insbesondere hat es keinen vollständigen Schreibzugriff auf gemeinsam genutzte Speicherorte wie TEMP oder die Windows-Registry.

Die Microsoft Store-Version: Die Microsoft Store-Version kann manuell über den Microsoft Store² mit dem Suchwort `Python` gesucht werden. Es sollte das Suchergebnis mit der neusten Python-Version gewählt werden, was nach aktuellem Stand Python 3.10 ist. Alternativ dazu kannst du eine Konsole öffnen und den Befehl `python` eingeben, der dich in den Microsoft Store zur neusten Python-Version bringt. Klicke auf `Installieren`, warte, bis die Anwendung heruntergeladen ist und folge dem Installationsprozess. Gib in einer Konsole anschließend `python --version` ein, um zu bestätigen, dass Python auf deinem Computer installiert wurde.

¹Tastenkombination + → „cmd“ oder „PowerShell“ eingeben → drücken

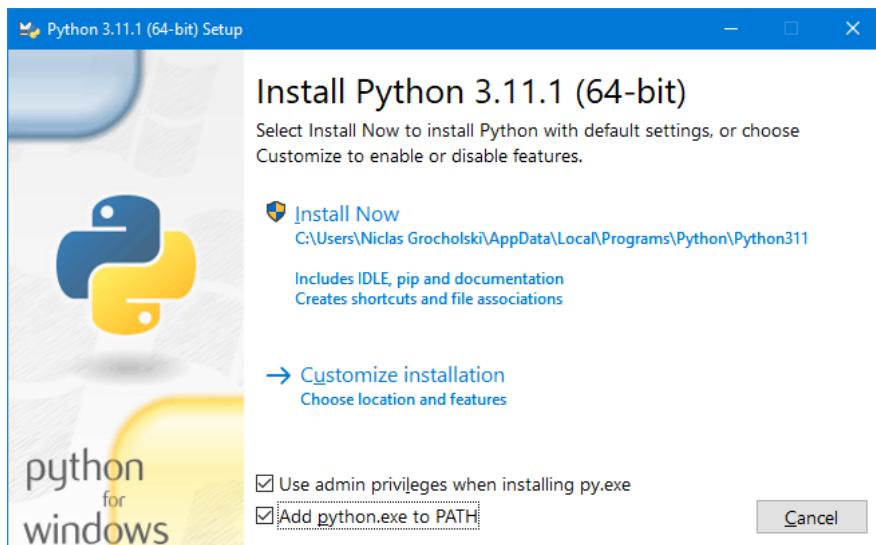
²Startmenü mit -Taste öffnen → „Microsoft Store“ suchen → oberstes Ergebnis (App) anklicken

Das vollständige Installationsprogramm: Um das vollständige Installationsprogramm zu laden, führe die folgenden Schritte aus:

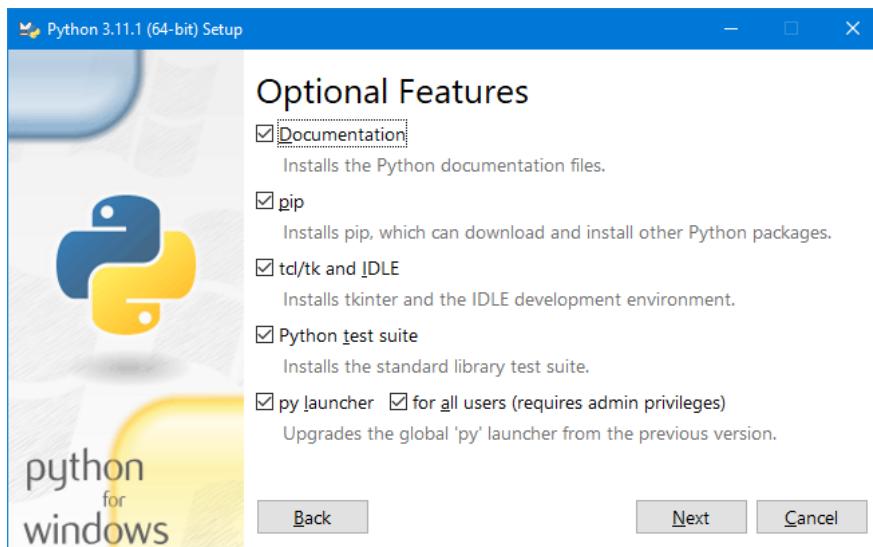
1. Öffne die [Windows Download-Seite für Python](#)
2. Klicke unter der Überschrift „Python Releases for Windows“ auf den Link [Latest Python 3 Release - Python 3.11.X](#) ($X \geq 1$)
3. Scrolle nach unten und wähle entweder den 32- oder 64-bit „Windows Installationsassistenten“ (einer der Installationsassistenten sollte die Beschreibung „Recommended“ erhalten)
4. Führe den Installationsassistenten aus dem Download-Ordner aus

Für den Installationsassistenten gibt es die folgenden Dinge anzumerken:

- ▶ Der Standardinstallationspfad befindet sich im AppData-Verzeichnis³ des aktuellen Windows-Benutzers
 - ▶ über die Schaltfläche [Customize installation](#) kannst du den Installationspfad anpassen und zusätzliche Funktionen, die installiert werden, einschließlich pip und IDLE, anpassen
- !** Es wird empfohlen, alle optionalen Features mitzuinstallieren und den Installationspfad nicht zu ändern.
- ▶ das Kontrollkästchen [Install launcher for all users \(recommended\)](#) ist standardmäßig aktiviert. Das bedeutet, dass jeder Benutzer auf dem Rechner Zugriff auf den py.exe Launcher hat. Du kannst dieses Kästchen deaktivieren, um Python auf den aktuellen Windows-Benutzer zu beschränken
 - ▶ Das Kontrollkästchen [Add Python 3.11 to PATH](#) ist standardmäßig nicht markiert
- !** Es gibt mehrere Gründe, warum du Python nicht im PATH haben willst, informiere dich also vorher über die Implikationen, die es mit sich bringt, diese Option zu aktivieren. Es ist auch nach der Installation jederzeit möglich, Python zum PATH hinzuzufügen.



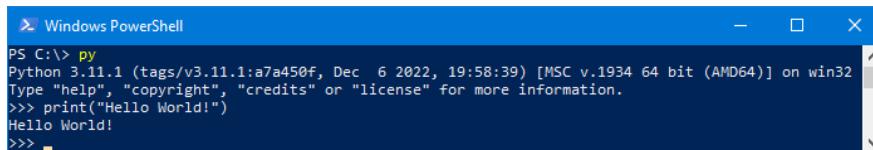
³Startmenü mit -Taste öffnen → %appdata% suchen → oberstes Ergebnis (Ordner) öffnen



3.1.3 Nutzungsanleitung

! Python über eine Kommandozeile aufzurufen, funktioniert nur, wenn Python zum PATH hinzugefügt wurde.

Read-Eval-Print Loop (REPL): Eine einfache Methode Python-Code auszuführen, ist die interaktive Sitzung. Um unter **Windows** eine interaktive Python-Sitzung zu starten, öffne eine Kommandozeile, gib `python` oder alternativ `py`⁴ ein und drücke die `Eingabe`-Taste. Die Standard-Eingabeaufforderung für den interaktiven Modus ist `>>>`, sobald du also diese Zeichen am Zeilenanfang siehst, weißt du, dass du Python-Code in die Eingabezeile tippen kannst. Der eingetippte Code wird mit der `Eingabe`-Taste ausgeführt⁵. Sind mehrere Python-Installationen auf deinem Computer vorhanden, wird standardmäßig auf die neuste Python-Installation zurückgegriffen. Um andere Python-Versionen anzusteuern, nutze entweder den Befehl `python<VERSIONSNUMMER>` oder den Befehl `py -<VERSIONSNUMMER>`, z.B. mit der Versionsnummer `3.10`. Wenn du interaktiv in der Konsole arbeitest, wird jeder Ausdruck und jede Anweisung, die du eingibst, sofort ausgewertet und ausgeführt. Sobald du die Kommandozeile schließt (entweder über die Befehle `quit()` oder `exit()` oder alternativ über die Tastenkombination `Strg` + `Z` mit `Eingabe`), geht der Code verloren.



Skripte: Ein Skript ist eine Textdatei mit Python-Instruktionen, die direkt ausgeführt wird. Python-Skripte haben die Dateiendung `.py`. Um Befehle aus einem Skript auszuführen, erstelle zunächst eine ausführbare Datei. Dies ist unter **Windows** wie folgt möglich:

1. Wenn nicht bereits aktiviert, aktiviere im Datei-Explorer unter *Ansicht* in der Gruppe *Ein-/Ausblenden* das Kontrollkästchen *Dateinamenerweiterungen*,
2. Erstelle im Datei-Explorer eine neue Textdatei mit beliebigen Namen ohne Leerzeichen⁶ über *Rechtsklick* → `[Neu]` → `[Textdokument]` und ändere die Dateiendung von `.txt` zu `.py`.

⁴mit Jupyter installiert (vergleiche Abschnitt 3.3), steht auch der Befehl `ipython` zur Verfügung

⁵mehrere Zeilen einzutippen, verlangt den Zeilenfortsetzungsoperator `\`

⁶Leerzeichen werden in der Regel durch Unterstriche `_` ersetzt

Starte in einer Konsole im gleichen Ordner, wo das Skript liegt⁷ nachdem zu dein Skript mit Code gefüllt hast den Befehl `py <DATEINAME>.py` und das Python-Skript wird ausgeführt.

Installation von Pakten mit pip: Funktionen in Python, wie die Unterstützung zur Auswertung von Massenspektrometriedaten oder die Erstellung diskreter Eventsimulationen, werden in sogenannten *Paketen* (auch *Module* oder *Bibliotheken* genannt) geliefert, die standardmäßig über den Paketmanager pip aus dem *Python Package Index* (PyPI) installiert werden. Ein Modul setzt dabei eine bestimmte Python-Version sowie die Installation anderer Module, sogenannten *Abhängigkeiten*, voraus. Das Paketverwaltungssystem wird in einer beliebigen Kommandozeile bedient. Wichtige Befehle sind:

- ▶ `py -m pip install <PAKETNAME>` um zusätzliche Pakete zu installieren. Zur [Installation über Git](#), benutze den Befehl `py -m pip install git+<REPOSITORY-URL>`,
- ▶ `py -m pip uninstall <PAKETNAME>` um ein Paket zu deinstallieren,
- ▶ `py -m pip list` um alle installierten Pakete aufzulisten.

Weitere Informationen zu pip findest du in der [offiziellen Python-Dokumentation](#).

! Sollten Probleme beim Ausführen von pip auftreten, muss das Paket eventuell neu installiert werden.

! Die hier vorgestellte Methode installiert alle Pakete in die selbe Umgebung. Um virtuelle Umgebungen zu nutzen, vergleiche den Ansatz mit conda aus Abschnitt 3.2.3.

3.1.4 Online-Interpreter

Auf Webseiten wie python.org, pythonfiddle.com, oder [programiz.com](https://programiz.com/python-tutorial) existieren Online-Python-Interpreter, die nicht installiert werden müssen und als Alternative zur lokalen Kommandozeile (vergleiche Abschnitt 3.1.3) für kleinere Code-Tests geeignet sind.

3.2 Anaconda

3.2.1 Was ist Anaconda?

Wie in Abschnitt 3.1.3 erklärt, werden Funktionen in Python in Paketen geliefert, die zuweilen von Extern installiert werden müssen. Einzelne Pakete setzen eine bestimmte Python-Version sowie andere Pakete zum Arbeiten voraus. Bei einer großen Anzahl installierter Pakete kann es unter Umständen zwischen verschiedenen Paketen zu Inkompatibilitäten kommen. Conda ist ein Open-Source-Paket- und Umgebungsverwaltungssystem für Windows, macOS und Linux, das als häufig genutzte Alternative zu pip erlaubt, Pakete und deren Abhängigkeiten in voneinander unabhängigen *virtuellen Umgebungen* zu installieren, auszuführen und zu aktualisieren. Conda ermöglicht somit, parallel Umgebungen mit zueinander inkompatiblen Python-Installationen und Paketabhängigkeiten zu schaffen. Alternativen zu conda sind [poetry](#), [virtualenv](#) und [pipenv](#).

Conda wird mit [Anaconda](#) geliefert, einer Python-Distribution mit Fokus auf der Verarbeitung großer Datenmengen, Vorhersageanalysen und wissenschaftlichem Rechnen, die neben einer großen Anzahl vorinstallierter Pakete unter anderem die Entwicklungsumgebung Spyder (vergleiche Abschnitt 3.4.1) und die Webanwendung JupyterLab (vergleiche Abschnitt 3.3.1) enthält.

In abgespeckter Version, nur mit conda, einer Python-Installation sowie einer kleinen Anzahl nützlicher Pakete, ist Anaconda in Form der [Miniconda](#)-Distribution verfügbar.

⁷ein Wechsel des aktiven Ordners in der Konsole ist mit dem Befehl `cd "<PFAD>"` möglich

3.2.2 Installationsanleitung

! Python ist in Anaconda mit enthalten. Eine separate Installation gemäß der Anleitung in Abschnitt 3.1.2 ist deshalb nicht erforderlich.

Im Folgenden wird die Anaconda-Installation auf **Windows** beschrieben. Zur Installation auf macOS und Linux, vergleiche die [Anaconda Installationsanleitung](#).

Anaconda benötigt eine 64-bit Windows-Installation mit Windows 8 oder neuer sowie mindestens 5 GB freien Festplattenspeicher. Folge zur Installation diesen Schritten:

1. Downloade den Anaconda-Installationsassistenten von der Startseite von [anaconda.com](#)

2. Öffne den Installationsassistenten

! Installiere Anaconda nicht als Administrator, es sei denn, du benötigst zwingend Administratorrechte zur Installation.

! Um Berechtigungsfehler zu vermeiden, starte das Installationsprogramm nicht aus dem Ordner „Favoriten“.

3. Klicke auf **Next**

4. Akzeptiere die Lizenzbedingungen über den Knopf **I Agree**

5. Es wird empfohlen, die Installation nur für das eigene Benutzerkonto über den Knopf **Just me (recommended)** durchzuführen. Wähle die Installation für alle Benutzer nur, wenn alle Benutzerkonten des Computers auf die Distribution zugreifen sollen (dies benötigt Windows-Administratorrechte). Klicke anschließend auf **Next**

6. Wähle einen Zielordner für die Installation von Anaconda und klicke auf **Next**. Installiere Anaconda in einen Verzeichnispfad, der keine Leerzeichen oder Unicode-Zeichen enthält. Weitere Informationen zu Zielordnern findest du in den [FAQ](#)

7. Wähle, ob du Anaconda zu deinen PATH-Umgebungsvariablen hinzufügen und Anaconda als Standard-Python registrieren möchtest

! Es ist nicht empfohlen, Anaconda zu den PATH-Umgebungsvariablen hinzuzufügen, da dies andere Software beeinträchtigen kann. Wenn du nicht vorhast, mehrere Versionen von Anaconda oder mehrere Versionen von Python zu installieren und auszuführen, bleibe hier bei den Standardeinstellungen.

8. Klicke auf **Install** und warte, bis die Installation abgeschlossen ist

9. Klicke auf **Next**, um die Installation von JetBrains DataSpell zu überspringen

10. Schließe den Installationsassistenten über **Finish**

! Sollten während der Installation Probleme auftreten, deaktiviere deine Antivirensoftware vorübergehend und aktiviere sie nach Abschluss der Installation wieder. Wenn du das Programm für alle Benutzer installiert hast, deinstalliere Anaconda und installiere es nur für dich als Benutzer neu. Weitere Troubleshooting-Informationen sind auf der Anaconda Website verfügbar.

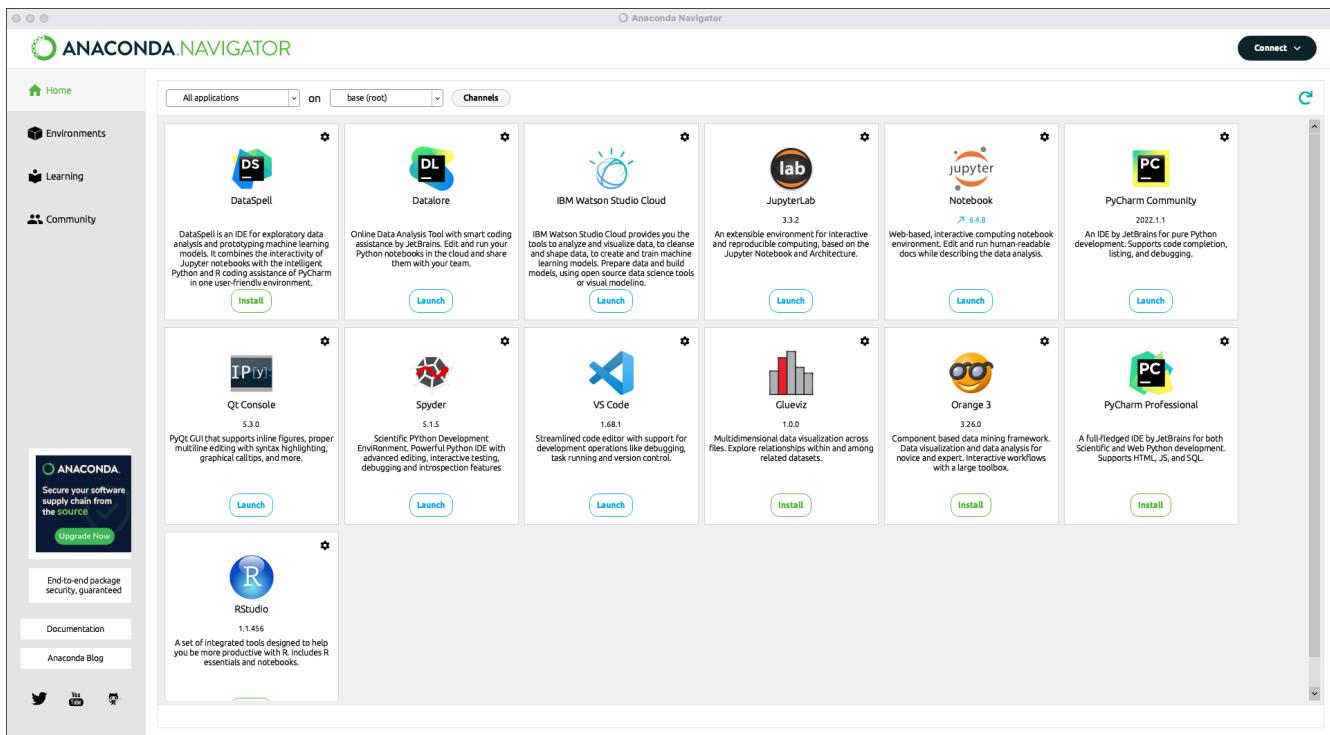
3.2.3 Nutzungsanleitung

Die Anaconda Distribution enthält conda und den *Anaconda Navigator*, sowie Python und hunderte von wissenschaftlichen Paketen. Mit der Installation von Anaconda sind auch all diese Pakete installiert worden. Anaconda Navigator ist eine grafische Desktop-Benutzeroberfläche, mit der Anwendungen gestartet und conda-Pakete, -Umgebungen und -Kanäle einfach verwaltet werden können, ohne Kommandozeilenbefehle zu verwenden (wie mit pip aus Abschnitt 3.1.3 zwangsläufig notwendig ist). Conda kann aber selbstredend auch über die Kommandozeilenschnittstelle *Anaconda Prompt* bedient werden.

Im Folgenden werden der Anaconda Navigator und Anaconda Promt mit ihren wichtigsten Funktionen vorgestellt. Sollten dir dort verwendete Begriffe nicht bekannt sein, findest du eine Übersicht dieser im [Anaconda Glossar](#).

Anaconda Navigator: Anaconda Navigator kann aus dem Startmenü geöffnet werden⁸. Es öffnet sich zunächst das Home-Fenster. Das linke Menü ist Zugang zu den folgenden Fenstern:

- **Home**: Die Startseite wird beim Start von Navigator standardmäßig geöffnet. Auf der Startseite werden alle verfügbaren Anwendungen (so auch die in den Abschnitten 3.3 und 3.4 behandelten Anwendungen) angezeigt, die mit Navigator verwaltet werden können. Hier können installierte Anwendungen ausgeführt und neue hinzugefügt werden. Anwendungen können mit spezifischen Umgebungen verbunden sein.



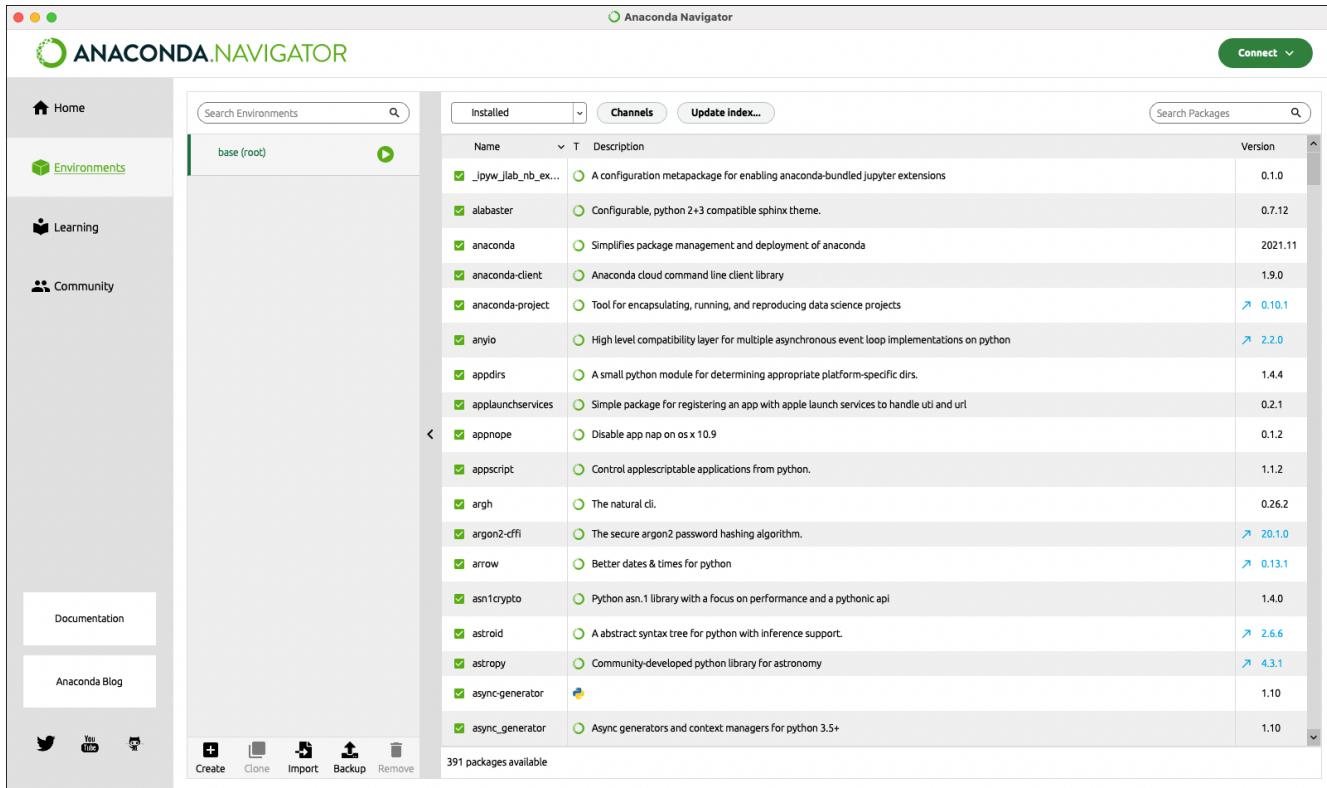
- **Environments**: Auf dieser Seite können installierte Umgebungen und Pakete verwaltet sowie neue hinzugefügt werden

- **Umgebungen**: Am unteren Ende der Liste der Umgebungen befinden sich die Schaltflächen zum Erstellen , Klonen , Importieren , Sichern und Entfernen von Umgebungen. Die Standardumgebung heißt base. Um in eine bestimmte Umgebung zu wechseln, klicke auf diese in der Umgebungsliste. Über das

⁸Startmenü mit -Taste öffnen → „Anaconda Navigator“ suchen → oberstes Ergebnis (App) anklicken

Start-Feld der aktiven Umgebung  können Terminals und Jupyter Notebooks mit der Umgebung geöffnet werden. Weitere Informationen zu Umgebungen gibt es auf der [Anaconda Website](#)

- **Pakete:** Pakete werden für jede Umgebung separat verwaltet. Änderungen, die du an Paketen vornimmst, werden nur für die aktive Umgebung angewendet. Neue Pakete können über den -Dropdown-Filter und die Suchfunktion über der Paketliste gesucht und dann installiert werden. Installierte Pakete werden über einen Klick auf ihre Checkbox  verwaltet. Weitere Informationen zu Paketen gibt es auf der [Anaconda Website](#)



- ▶  **Learning:** Auf der Seite Lernen kannst du mehr über Navigator, die Anaconda-Plattform und Open Data Science erfahren. Klicke auf die Schaltflächen Dokumentation oder Training, um die Lernkacheln zu filtern. Klicke auf eine beliebige Kachel, um sie in einem Browserfenster zu öffnen
- ▶  **Community:** Hier kannst du mehr über kostenlose Support-Foren und soziale Netzwerke im Zusammenhang mit Navigator erfahren. Klicke auf die Schaltflächen Forum oder Social, um die Kacheln zu filtern. Klicke auf eine beliebige Kachel, um sie in einem Browserfenster zu öffnen.

Anaconda Prompt: Die Kommandozeilenversion des Navigators kann aus dem Startmenü geöffnet werden⁹. Eine Übersicht über Anaconda-spezifische Befehle gibt es über  sowie auf der [Anaconda Website](#). Wichtiger an dieser Stelle sind die conda-Befehle, die über  aufgelistet oder über die [Conda-Website](#) eingesehen werden können. Wichtige Befehle sind:

- ▶  `<KOMMANDONAME> --help`, um sich Informationen zu einem bestimmten Befehl anzeigen zu lassen,

⁹Startmenü mit -Taste öffnen → „Anaconda Prompt (Anaconda3)“ oder „Anaconda Powershell Prompt (Anaconda3)“ suchen → oberstes Ergebnis (App) anklicken

- ▶ `conda create --name <UMGEBUNGNAME> python=3.<X>` zum Erstellen einer neuen Umgebung mit der Python-Version der angegebenen Nummer,
- ▶ `conda activate <UMGEBUNGNAME>` zum Aktivieren einer neuen Umgebung (die aktuelle Umgebung wird im Klammern vor der Befehlseingabe angezeigt),
- ▶ `conda env list`, um alle erstellten Umgebungen aufzulisten,
- ▶ `conda env remove --name <UMGEBUNGNAME>`, um eine spezifische Umgebung zu löschen,
- ▶ `conda install <PAKETNAME1> <PAKETNAME2> ...` zum Installieren eines oder mehrerer Pakete in die aktuell aktivierte Umgebung. Hat Anaconda das Paket nicht in seinem Verzeichnis, kann es alternativ über pip (vergleiche Abschnitt 3.1.3) installiert werden. Pakete können über `conda search <PAKETNAME>` im Verzeichnis gesucht werden,
- ▶ `conda list` um alle installierten Pakete der aktuell aktivierten Umgebung anzuzeigen.

Zusätzlich kann der Navigator über den Befehl `anaconda-navigator` geöffnet werden.

3.3 JupyterLab

3.3.1 Was ist JupyterLab?

Notebook-Dokumente (oder „notebooks“) sind von der JupyterLab-App erstellte Dokumente, die sowohl Computercode als auch Rich-Text-Elemente (Absätze, Gleichungen, Abbildungen, Links usw.) enthalten. Die JupyterLab-Software ist in vielen Forschungsbereichen zur Defacto-Standardprogrammierumgebung geworden. JupyterLab-Notebooks arbeiten mit sogenannten Zellen, wobei sich in einem Notebook Codezellen und Textzellen abwechseln können. Code wird direkt in den Zellen ausgeführt, Texte werden in `Markdown` und `LATEX` verfasst. Dadurch können Hypothesen und Annahmen über die Daten formuliert und anschließend mithilfe von Code direkt überprüft werden. So entstehen reproduzierbare Forschungstagebücher und im fortgeschrittenen Stadium publikationsreife Artikel. Neben Notebooks, die den Kern der JupyterLab-App ausmachen, bietet die Umgebung auch einen Texteditor, eine Code-Konsole und ein Dateiverwaltungssystem.

JupyterLab wird von den Mitarbeitenden des [Project Jupyter](#) verwaltet. JupyterLab ist ein Ableger des IPython-Projekts. Der Name Jupyter leitet sich von den Programmiersprachen ab, die das Projekt unterstützt: [Julia](#), Python und [R](#). Die JupyterLab-App ist eine Server-Client-Anwendung, die die Bearbeitung und Ausführung von Notebook-Dokumenten über einen Webbrowser ermöglicht. Die JupyterLab-App kann auf einem lokalen Desktop ausgeführt werden, der keinen Internetzugang benötigt, oder sie kann auf einem entfernten Server installiert werden und über das Internet zugänglich sein (vergleiche Abschnitt 3.3.4).

3.3.2 Installationsanleitung

Das JupyterLab ist nicht im Lieferumfang von Python enthalten. Jupyter muss also extra installiert werden. Dies ist über pip mit `pip install jupyterlab` bzw. über conda mit `conda install -c conda-forge jupyterlab` möglich.

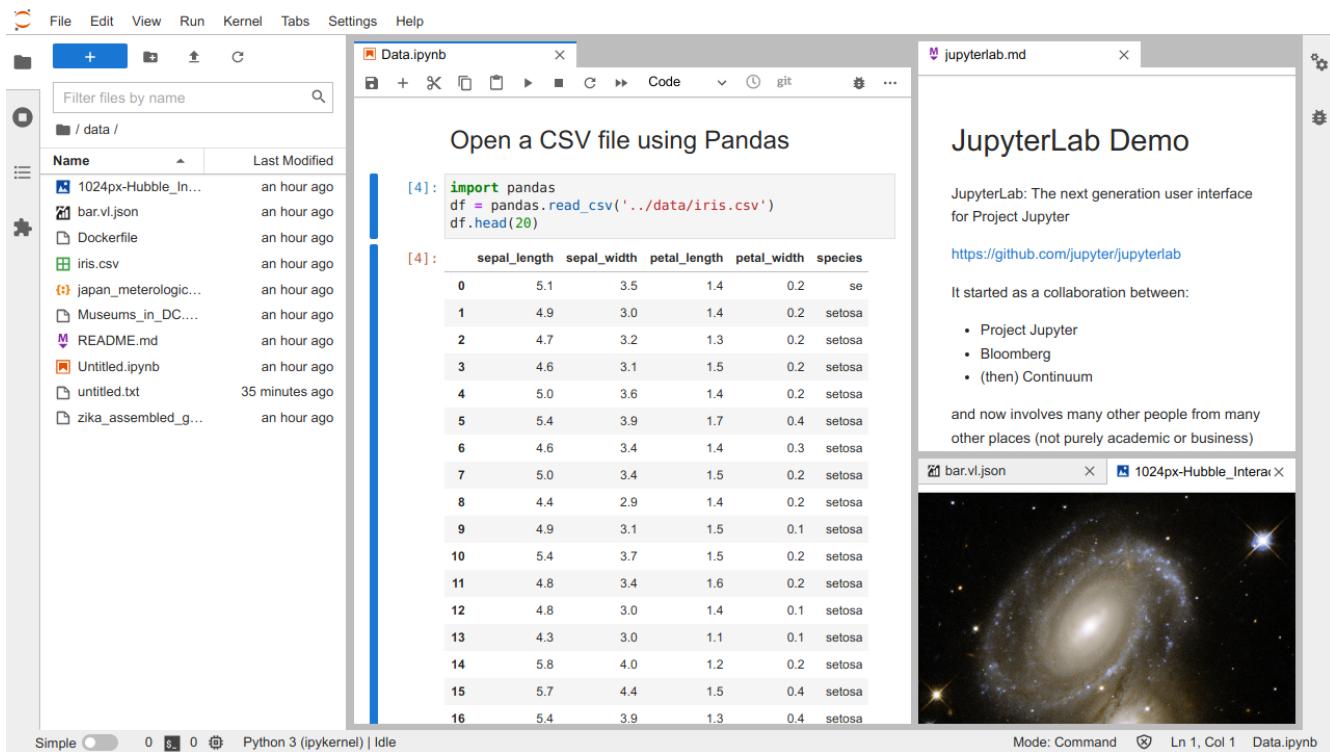
3.3.3 Nutzungsanleitung

JupyterLab starten: Sobald JupyterLab installiert ist, kann die Anwendung in einer Konsole (auch in Anaconda Prompt) mit dem Befehl `jupyter lab` gestartet werden. JupyterLab wird dann automatisch im Standardbrowser geöffnet. Wenn sich deine Notebook-Dateien nicht im aktuellen Verzeichnis, aus dem der Befehl gestartet wurde, befinden, kannst du beim Starten von JupyterLab den Pfad zum eigenen Arbeitsverzeichnis als Argument mit `jupyter lab --notebook-dir="<PFAD>"` übergeben oder in der Konsole manuell in das entsprechende Verzeichnis wechseln. Vermeide es am Besten, JupyterLab

aus deinem Root-Verzeichnis (C: unter Windows) aus zu starten, um das Risiko einer Änderung von Systemdateien zu verringern.

Du kannst auf eine existierende Instanz von JupyterLab zugreifen, indem du die URL des Notebook-Servers `http(s)://(SERVER):(PORT)/(LAB-LOCATION)/lab` in den Browser eingibst. Alternativ zeigt der Befehl `jupyter notebook list` alle aktiven Notebooks. Weitere Informationen dazu findest du in der [JupyterLab Dokumentation](#).

Die Benutzeroberfläche: Die JupyterLab-Benutzeroberfläche besteht aus einem Hauptarbeitsbereich mit Registerkarten für Dokumente und Aktivitäten, einer ausklappbaren linken Seitenleiste und einer Menüleiste am oberen Ende des Browserfensters.



Die Menüleiste am oberen Rand von JupyterLab enthält Menüs der obersten Ebene, in denen die in JupyterLab verfügbaren Aktionen mit den entsprechenden Tastenkombinationen angezeigt werden. Die Standardmenüs sind:

- ▶ `File`: Aktionen im Zusammenhang mit Dateien und Verzeichnissen,
- ▶ `Edit`: Aktionen, die sich auf die Bearbeitung von Dokumenten und andere Aktivitäten beziehen,
- ▶ `View`: Aktionen, die das Aussehen von JupyterLab verändern,
- ▶ `Run`: Aktionen zum Ausführen von Code in verschiedenen Aktivitäten wie Notizbüchern und Codekonsolen,
- ▶ `Kernel`: Aktionen zur Verwaltung von Kerneln, die separate Prozesse zur Ausführung von Code sind,
- ▶ `Tabs`: eine Liste der geöffneten Dokumente und Aktivitäten im Andockfenster,
- ▶ `Settings`: allgemeine Einstellungen und ein Editor für erweiterte Einstellungen,
- ▶ `Help`: eine Liste mit Links zur JupyterLab- und Kernel-Hilfe.

Die linke Seitenleiste enthält eine Reihe von häufig genutzten Registerkarten, darunter (von oben nach unten):

- ▶ einen Dateibrowser,
- ▶ eine Liste offener Tabs, laufender Kernels und Terminals,
- ▶ das [Inhaltsverzeichnis](#),
- ▶ den [Erweiterungsmanager](#).

Die rechte Seitenleiste enthält:

- ▶ den Eigenschaftsinspektor (aktiv in Notebooks),
- ▶ den [Debugger](#).

Die Seitenleisten können durch Auswahl von „Linke Seitenleiste anzeigen“ oder „Rechte Seitenleiste anzeigen“ im Menü [View](#) oder durch Klicken auf die aktive Registerkarte der Seitenleiste ein- oder ausgeklappt werden. Mehr Informationen über das Interface von JupyterLab sind [in der JupyterLab Dokumentation](#) zu finden.

Arbeiten mit Notebooks: Notebooks haben die Dateiendung .ipynb. Sie bestehen, wie eingangs erwähnt, aus Zellen. Zellen können einzeln ausgeführt werden, wobei Ergebnisse von Codezelloperationen, wie z.B. die Deklaration von Variablen, bis zum Kernel-Neustart erhalten bleiben. So kann mit Ergebnissen aufwendigerer Berechnungen in anderen Zellen einfach weitergearbeitet werden, ohne dass alle Berechnungen (wie bei einem klassischen Python-Skript) erneut zusammen, von vorne ausgeführt werden müssen.

Neue Zellen können über den -Knopf in der oberen Leiste hinzugefügt und über den Ausschneiden-Knopf wieder entfernt werden. Die grün umrandete Zeile ist die aktuell aktive Zelle, wobei mit einem Mausklick oder den Pfeiltasten (im Kommando-Modus ohne blinkenden Cursor in der Zelle) zwischen Zellen hin- und hergewechselt werden kann. Die ausgewählte Zelle kann mit den - und -Knöpfen im Notebook verschoben werden. Mit einem Dropdown-Menü kann die Zelle als Code- oder Textzelle deklariert werden.

Das Projekt wird über den -Knopf gespeichert und die aktive Zelle über den -Knopf ausgeführt und mit dem -Knopf unterbrochen. Um den Kernel neu zu starten (und damit alle gespeicherten Variablen zu löschen), ist der -Knopf da.

Ausführliche Videotutorials zu Notebooks sind [in der JupyterLab Dokumentation](#) oder als [RealPython Tutorial](#) zu finden. Hilfreich zum schnelleren Arbeiten sind außerdem [Tastenkombinationen](#), wie z.B. zum Einfügen neuer Zellen, Neustart des Kernels, etc.

3.3.4 Jupyter Notebooks in der Cloud

Während JupyterLab ab Version 3.1 [kollaboratives Arbeiten](#) auch mit einem lokal ausgeführten Server bietet, stellt eine handvoll Unternehmen bereits seit längerer Zeit „online data science notebook“-Lösungen an, die über Cloud-Hosting betrieben werden. Zu diesen Anbietern zählen z.B. [Deepnote](#), [Kaggle](#), [Google Colab](#), oder [CoCalc](#). Vorteil dieser Lösungen ist, dass keine eigene Einrichtung einer JupyterLab-Umgebung notwendig ist, eine aufpolierte Programmierumgebung, unter Umständen mit zusätzlichen Funktionen, geboten wird und dass die Dienste plattformübergreifend Echtzeitkollaboration in einem Notebook erlauben, was Teamarbeit deutlich einfacher gestaltet. Rechenleistung ist in den kostenlosen Versionen dieser Anbieter gegenüber einer lokalen Maschine dafür begrenzt und der Funktionsumfang einer lokalen Python-IDE (vergleiche Abschnitt 3.4) wird von keinem der Anbieter erreicht.

Im Workshop verwenden wir den Anbieter Deepnote. Online findest du einen [Crashkurs](#) zu Deepnote, der dir einen ersten Einblick in die Plattform bietet und das Bearbeiten der Übungsaufgaben im Workshop erleichtert.

The screenshot shows a Jupyter Notebook interface. On the left, there's a sidebar with a navigation menu including 'Galahad Sciences' (legend@galahad.com), 'All Projects / Analytics / Customer churn model', 'Search', 'Integrations', 'Settings & Members', 'Workspace' (with 'Data exploration', 'Analytics' (selected), 'Marketing team', 'Inbound lead scoring', 'Q1 A/B test results', 'Customer churn model' (selected), 'Internal tooling', 'Data pipelines'), 'Private' (with 'Fundraising resources', 'Revenue + customers summary', 'Ops modelling'), 'Explore', 'Documentation & Help', and 'Deepnote'. The main area has a title 'Customer churn model' and a description about customer retention being a primary KPI. Below that is a code cell:

```

1 # Imports
2 import pandas as pd
3
4 customers = pd.read_csv("./customers.csv")
5
6 Jacqueline
7
8 def load_and_process_img(path_to_img):
9     img = load_img(path_to_img)
10    img = tf.keras.applications.vgg19.preprocess_input(img)
11    return image
12
13 Dataframe import successful 🎉

```

Below the code is a visualization titled 'Visualization of customers' showing a line chart of 'Retention' over 'Week' (0 to 21) for different 'Cohort' groups. The chart shows a general downward trend in retention over time.

3.4 PyCharm

3.4.1 Was ist Pycharm?

PyCharm ist eine Integrierte Entwicklungsumgebung (IDE) für Python, also ein Programm, welches Werkzeuge bereitstellt, das häufig wiederkehrende Aufgaben beim Programmieren abnimmt und so hilft, sich auf die eigentliche Aufgabe, das Entwickeln/Programmieren von Software, fokussieren zu können. Während es generell möglich ist, Quellcode in einem einfachen Texteditor zu schreiben, kann eine IDE den Schreibprozess von Code deutlich angenehmer gestalten und so beschleunigen. Zu den Features von PyCharm, die ein Texteditor nicht bietet, gehören:

- ▶ **die automatische Vervollständigung:** beschleunigt die korrekte Codierung, indem sie Eingaben vervollständigt
- ▶ **Schnellkorrekturen:** erkennt häufige Fehler und bietet sprachspezifische Korrekturen
- ▶ **Code Intentions:** schlägt Optimierungen und Verbesserungen für gängige Python-Muster vor
- ▶ **Code Refactoring:** nimmt die Arbeit ab, die beim Refactoring (Prozess der Umstrukturierung von Code, ohne dessen ursprüngliche Funktionalität zu verändern) häufig anfällt
- ▶ **Templates:** automatisiert wiederkehrende Aufgaben
- ▶ **Code Navigation:** analysiert die Struktur und Semantik des eigenen Codes und des von ihm verwendeten Codes, um Möglichkeiten zu bieten, sich zwischen zusammenhängenden Codesegmenten zu bewegen

Häufig genutzte Alternativen zu PyCharm sind **Spyder**, eine Open-Source Software als Teil der Anaconda-Distribution und **Visual Studio Code**, eine IDE für mehrere Programmiersprachen von Microsoft.

Von PyCharm existiert eine kostenlose Community-Version sowie eine kostenpflichtige Professional-Version. Mit der

Community-Version können reine Python-Projekte erstellt werden. Die Professional-Version bietet weitere Unterstützungen, z. B. für HTML, JavaScript und SQL. Für Studierende bietet JetBrains, das Unternehmen hinter PyCharm, kostenlose, jährlich zu erneuernde Lizenzen für PyCharm Professional.

3.4.2 Installationsanleitung

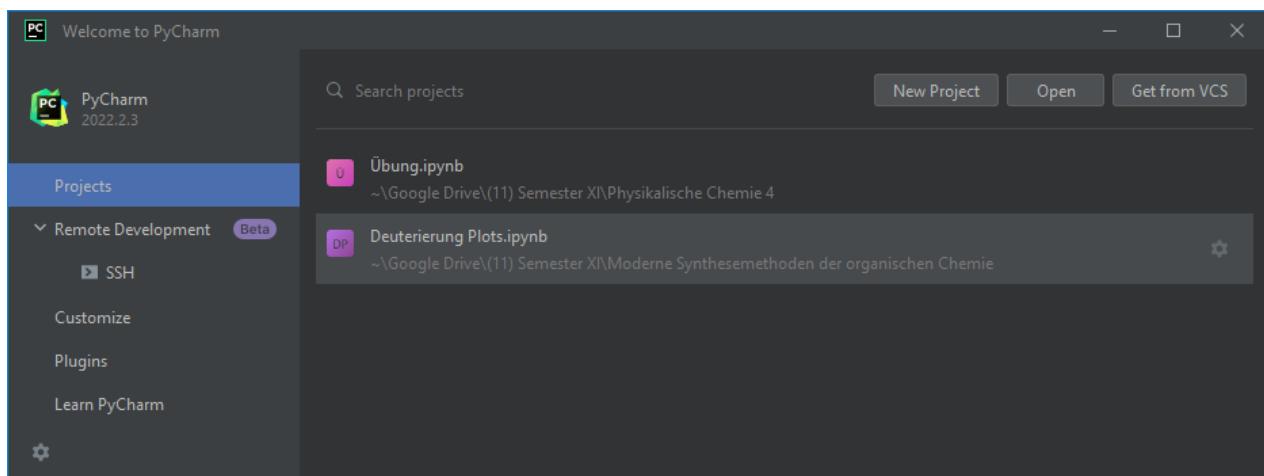
PyCharm benötigt eine 64-bit Installation von Windows 8 oder neuer mit mindestens 4 GB RAM sowie mindestens 3.5 GB freien Festplattenspeicher. Um PyCharm Professional zu nutzen, stelle zunächst außerdem sicher, dass du einen Python Interpreter (\geq Version 3.6) installiert hast.

1. Erstelle einen [Account mit deiner Universitäts-E-Mail-Adresse](#).
2. Lade anschließend die entsprechende [PyCharm Professional Version für dein Betriebssystem](#) (Windows / macOS / Linux) herunter. Es wird empfohlen, die „standalone installation“ und nicht die „JetBrains Toolbox App“ zur Installation zu verwenden. Die aktuelle PyCharm Version ist 2022.3.
3. Unter **Windows**: führe den Installer aus und folge den Schritten des Assistenten. Die folgenden Optionen stehen bei der Installation zur Verfügung:
 - ▶ **64-bit launcher**: fügt ein Startsymbol auf dem Desktop hinzu
 - ▶ **Open Folder as PyCharm Project**: fügt dem Ordner-Kontextmenü eine Option hinzu, die das Öffnen des ausgewählten Verzeichnisses als PyCharm-Projekt ermöglicht
 - ▶ **.py**: stellt eine Verbindung zu Python-Dateien her, um sie standardmäßig in PyCharm zu öffnen
 - ▶ **Add launchers dir to the PATH**: ermöglicht das Ausführen dieser PyCharm-Instanz von der Konsole aus, ohne den Pfad zu ihr anzugeben

Weitere Informationen zur Installation (auch für macOS und Linux) sind auf der [Website von PyCharm](#) zu finden.

3.4.3 Konfigurationsanleitung

Der Willkommensbildschirm: Sobald PyCharm gestartet wurde, ist der Willkommensbildschirm zu sehen, der Ausgangspunkt für die Arbeit mit der IDE ist und die Möglichkeit zur Konfiguration des Programms bereitstellt. Dieser Bildschirm erscheint auch, wenn alle geöffneten Projekte geschlossen werden.



Die Registerkarten auf der linken Seite werden verwendet, um zum jeweiligen Willkommensdialog zu wechseln:

- ▶ **Projects**: Zeigt zuletzt geöffnete Projekte an. Bietet die Möglichkeit, ein neues lokales Projekt anzulegen, ein lokales Projekt zu öffnen, oder ein Projekt aus einem Versionskontrollsystem (wie Git) zu laden. Eine Anleitung zur Erstellung eines simplen Projektes ist auf der [Website von PyCharm](#) zu finden – da wir im Workshop mit Jupyter arbeiten, soll an dieser Stelle lediglich auf die Projekterstellung verwiesen werden
- ▶ **Customize**: Anpassung von Farben, Schriftgröße und Tastenbelegung sowie Zugang zu allen Einstellungen über **All Settings...**. Alternativ sind alle Einstellungen in einem geöffneten Projekt über den Menüpunkt **File** → **Settings** aufrufbar. Eine Übersicht der Standardtastenbelegungen ist auf der [Website von PyCharm](#) zu finden
- ▶ **Plugins**: Verwaltung von Erweiterungen. Beliebte Erweiterungen sind [Rainbow Brackets](#), [String Manipulation](#), [CSV Editor](#), [GitToolBox](#) und viele mehr! Weitere Informationen zu Erweiterungen sind auf der [Website von PyCharm](#) zu finden
- ▶ **Learn PyCharm**: Zugang zu IDE Trainingskursen, die ebenso über die [Website von PyCharm](#) aufgerufen werden können

PyCharm Professional aktivieren: Um PyCharm Professional zu aktivieren, klicke unter dem Zahnradssymbol in der linken unteren Ecke des Willkommensbildschirms auf den Dialog **Manage Licenses**. Hier besteht nun die Option, sich über das Feld **Activate New License** zur Authentifizierung im Browser seiner Wahl im JetBrains-Account der Universitäts-E-Mail-Adresse anzumelden. Weitere Informationen zur Aktivierung der akademischen Lizenz sind auf der [Website von PyCharm](#) zu finden.

Einrichtung eines Python-Interpreters: PyCharm bietet die Möglichkeit, zwischen mehreren installierten Python-Interpreters (z.B. verschiedener Python-Versionen) zu wechseln. Um Python-Code in PyCharm auszuführen (und nicht nur zu schreiben und anzuschauen), muss mindestens einen Python-Interpreter konfiguriert sein. Es ist dabei sowohl möglich, einen Systeminterpreter (vergleiche Abschnitt 3.1.3) als auch eine virtuelle Umgebung von conda (vergleiche Abschnitt 3.2.3), poetry, virtualenv, oder pipenv zu verwenden.

Um einen Interpreter aus einem geöffneten Projekt heraus zu konfigurieren,

1. drücke **Strg** + **Alt** + **S**, um die Einstellungen zu öffnen und gehe über das linke Navigationsmenü in das Untermenü **Project: <PROJEKTNNAME>** → **Python Interpreter**¹⁰. Hier können [Pakete bestehender Interpreter hinzugefügt und gelöscht](#) und neue Interpreter hinzugefügt werden,
2. klicke auf das Dropdownmenü **Add Interpreter** rechts neben der Liste verfügbarer Interpreter (die „<No interpreter>“ ausgewählt haben sollte),
3. wähle **Add Local Interpreter**.

Es öffnet sich das „Add Python Interpreter“-Dialogfeld. Hier besteht die Möglichkeit, einen Systeminterpreter über die linke Menüoption **System Interpreter** oder eine virtuelle Umgebung aus conda über die linke Menüoption **Conda Environment** zu konfigurieren.

Der Systeminterpreter benötigt einen Pfad zur Python-Programmdatei (`python3.<VERSIONSNUMMER>.exe`). Normalerweise sollten alle Python-Installationen in einem Dropdownmenü zur Auswahl stehen. Ist dies nicht der Fall, kannst du über den **...**-Knopf rechts des Dropdown-Menüs auch händisch zur Python-Programmdatei navigieren. Bestätige die Auswahl im Dialogfeld über den **OK**-Knopf und der neue Interpreter ist hinzugefügt.

Eine bereits über Anaconda erstellte virtuelle conda-Umgebung (vergleiche Abschnitt 3.2.3) wählst du im „Interpreter“-Dropdownmenü aus. Alternativ zu Anaconda Navigator und Anaconda Prompt besteht hier die Möglichkeit, auch aus

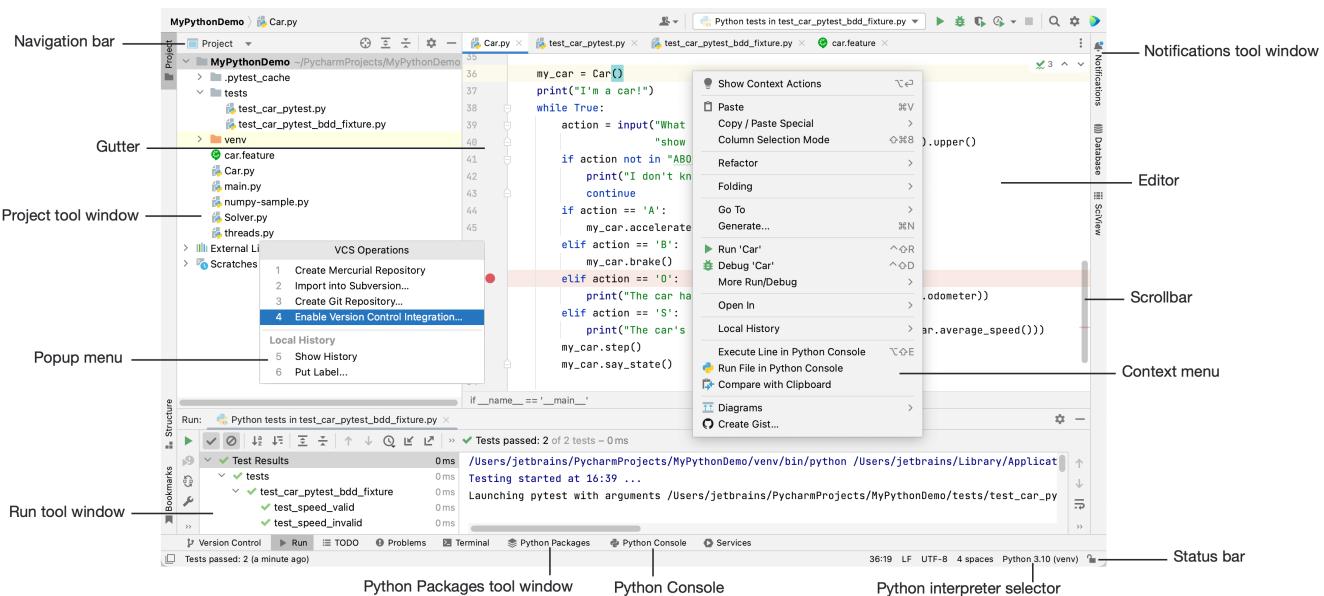
¹⁰das Menü ist, wie weiter oben erklärt, ebenso aus dem Willkommensbildschirm aufrufbar. Hier ist der Menüpunkt **Python Interpreter** aber verständlicherweise einer der Hauptmenüpunkte und nicht unter **Project: <PROJEKTNNAME>** einsortiert.

PyCharm eine neue virtuelle Umgebung zu erzeugen. Bestätige die Auswahl im Dialogfeld über den **OK**-Knopf und der neue Interpreter ist hinzugefügt.

Bestehende Interpreten können in einem geöffneten Projekt einfach über den „Python interpreter selector“ der Statusleiste (vergleiche Abschnitt 3.4.4) gewechselt werden. Hier wird auch angezeigt, dass dem Projekt eventuell noch kein Interpreter zugeordnet ist. Weitere Informationen zur Konfiguration von Interpreters in PyCharm findest du auf der [PyCharm-Website](#).

3.4.4 Nutzungsanleitung

Die grafische Benutzeroberfläche eines Projektes: Abhängig von der Menge der Plugins, der PyCharm-Edition und den Einstellungen kann die eigene IDE anders aussehen und sich anders verhalten. Grundlegend sieht das Interface eines geöffneten Projektes aber wie folgt aus:



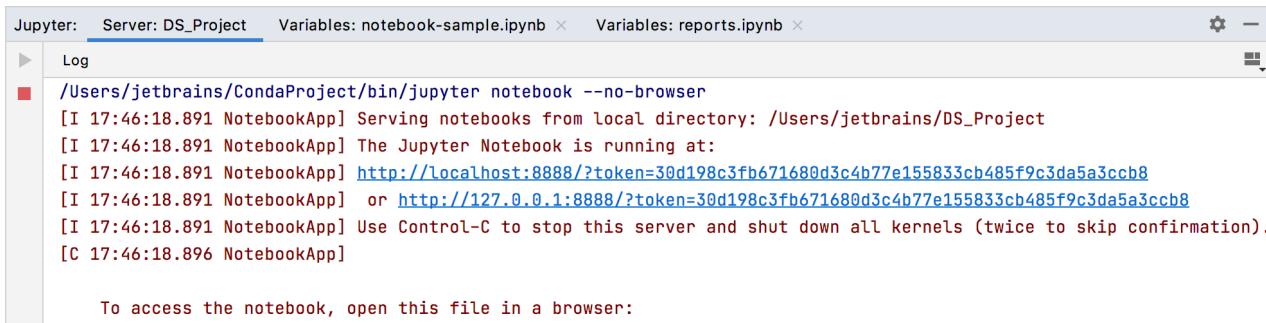
- ▶ **Navigation Bar:** Die Navigationsleiste am oberen Rand ist eine Alternative zur Projektansicht, in der du durch die Struktur deines Projekts navigierst und Dateien zur Bearbeitung öffnen kannst. Verwende die Schaltflächen rechts der Navigationsleiste, um Anwendungen auszuführen **▶**, zu debuggen **🐞**, zu durchsuchen **🔍** und Einstellungen aufzurufen **⚙️**.
- ▶ **Gutter:** Verschiedene Symbole, die hier erscheinen, z. B. das gelbe Glühbirnensymbol **💡**, helfen Schnellkorrekturen und andere Aktionen zu erkennen. Wenn du auf einen solchen Aktionsindikator klickst oder **Alt** + **Enter** drückst, wird eine Aktionsliste mit allen Schnellreparaturen und anderen Aktionen geöffnet, die an der aktuellen Cursorposition verfügbar sind.
- ▶ **Project tool window:** Im Projektwerkzeugfenster kannst du dein Projekt aus verschiedenen Blickwinkeln betrachten und verschiedene Aufgaben ausführen, wie z. B. neue Elemente (Verzeichnisse, Dateien, Klassen usw.) erstellen, Dateien im Editor öffnen, zu erforderlichen Codefragmenten navigieren und vieles mehr.
- ▶ **Popup menu:** Popup-Menüs bieten schnellen Zugriff auf Aktionen, die mit dem aktuellen Kontext zusammenhängen.
- ▶ **Run tool window:** Dieses Fenster zeigt die von deiner Anwendung erzeugten Ausgaben an. Das Aussehen der einzelnen Registerkarten hängt von der Art der ausgeführten Anwendung ab und kann zusätzliche Toolboxen und Bereiche enthalten.

- ▶ **Python Packages tool window:** In diesem Fenster können mit pip Pakete aus PyPI in den aktuell ausgewählten Python-Interpreter installiert werden.
- ▶ **Python Console:** Die Python-Konsole ermöglicht die zeilenweise Ausführung von Python-Befehlen und -Skripten wie in Abschnitt 3.1.3 erklärt.
- ▶ **Python interpreter selector:** Zeigt den Python-Interpreter an, der derzeit verwendet wird. Klicken hier drauf, um einen neuen Python-Interpreter hinzuzufügen, einen vorhandenen Interpreter auszuwählen oder die Liste der Interpretereinstellungen zu öffnen (vergleiche Abschnitt 3.4.3).
- ▶ **Status bar:** Im linken Teil der Statusleiste am unteren Rand des Hauptfensters werden die neuesten Ereignismeldungen und Beschreibungen von Aktionen angezeigt, wenn du mit dem Mauszeiger darüber fährst. Klicke auf eine Meldung in der Statusleiste, um sie im „Benachrichtigungen“-Werkzeugfenster zu öffnen.
Die Statusleiste zeigt auch den Fortschritt von Hintergrundaufgaben an. Du kannst dann auf  klicken, um den **Manager für Hintergrundaufgaben** anzuzeigen.
- Der rechte Teil der Statusleiste enthält Widgets, die den Gesamtstatus des Projekts und der IDE anzeigen und Zugriff auf verschiedene Einstellungen bieten. Je nach Plugins und Konfigurationseinstellungen kann sich der Satz an Widgets ändern.
- ▶ **Context menu:** Du kannst mit der rechten Maustaste auf verschiedene Elemente der Benutzeroberfläche klicken, um die im aktuellen Kontext verfügbaren Aktionen anzuzeigen. Klicke z.B. mit der rechten Maustaste im Editor, um Aktionen für das aktuelle Codefragment anzuzeigen.
- ▶ **Editor:** Verwende den Editor, um deinen Quellcode zu lesen, zu schreiben und zu untersuchen.
- ▶ **Notifications tool window:** Immer wenn es ein wichtiges Ereignis oder einen Vorschlag in der IDE gibt, wird eine Benachrichtigung angezeigt.

Jupyter-Unterstützung: Mit der in PyCharm verfügbaren Jupyter Notebook-Integration kannst du Jupyter Notebooks einfach bearbeiten, ausführen und debuggen und die Ausführungsergebnisse einschließlich Streamdaten, Bildern und anderen Medien untersuchen.

Das Layout der Jupyter Notebook-Unterstützung ähnelt sich generell dem Jupyter Lab-Layout (vergleiche Abschnitt 3.3.3). Detaillierte Informationen zum [Layout](#) und [Bearbeiten](#) von Notebooks findest du auf der PyCharm Website.

Über ein dezidiertes [Tool-Fenster](#) am unteren Ende der Anwendung kann das Notebook übrigens außerhalb von PyCharm in einem Webbrowser geöffnet werden, sollte dir das Layout in der Anwendung nicht gefallen. Klicke dazu auf einen der Links im entsprechenden Fenster. Es ist empfehlenswert, die entsprechende .ipynb-Datei danach in PyCharm zu schließen.



The screenshot shows the PyCharm interface with the 'Log' tab selected in the 'Jupyter' tool window. The title bar indicates 'Jupyter: Server: DS_Project'. There are two tabs open: 'Variables: notebook-sample.ipynb' and 'Variables: reports.ipynb'. The log output is as follows:

```

Jupyter: Server: DS_Project Variables: notebook-sample.ipynb × Variables: reports.ipynb ×
Log
[✓] /Users/jetbrains/CondaProject/bin/jupyter notebook --no-browser
[I 17:46:18.891 NotebookApp] Serving notebooks from local directory: /Users/jetbrains/DS_Project
[I 17:46:18.891 NotebookApp] The Jupyter Notebook is running at:
[I 17:46:18.891 NotebookApp] http://localhost:8888/?token=30d198c3fb671680d3c4b77e155833cb485f9c3da5a3ccb8
[I 17:46:18.891 NotebookApp] or http://127.0.0.1:8888/?token=30d198c3fb671680d3c4b77e155833cb485f9c3da5a3ccb8
[I 17:46:18.891 NotebookApp] Use Control-C to stop this server and shut down all kernels (twice to skip confirmation).
[C 17:46:18.896 NotebookApp]

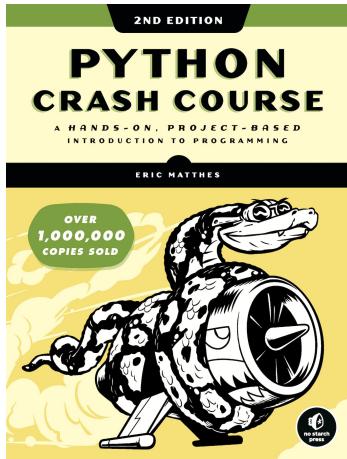
To access the notebook, open this file in a browser:

```

4 Weitere Ressourcen zu Python

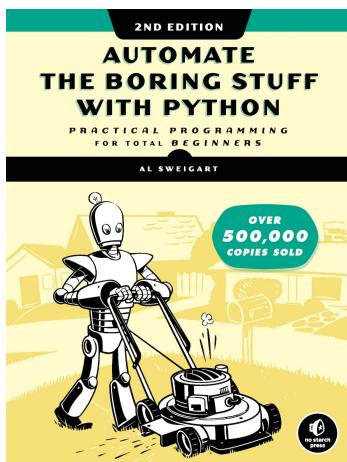
Generell ist es möglich, mit jeder der folgenden Ressourcen Erfolg haben — wichtig ist, das theoretische Wissen schnell in der Praxis anzuwenden, sei es bei einem aktuellen Projekt oder indem du selber etwas herumprobierst. Nur so lernst du dazu und wirst besser im Programmieren.

4.1 Bücher



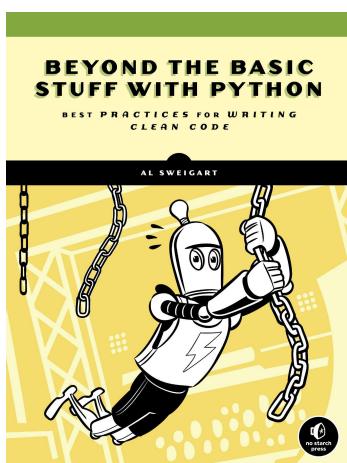
E. Matthes, *Python Crash course, A hands-on, project-based introduction to programming*, 2. Aufl., No Starch Press, San Francisco, **2019**.

„In the book's first half, you'll learn basic programming concepts, such as variables, lists, classes, and loops, and practice writing clean code with exercises for each topic. You'll also learn how to make your programs interactive and test your code safely before adding it to a project. In the second half, you'll put your new knowledge into practice with three substantial projects: a Space Invaders-inspired arcade game, a set of data visualisations with Python's handy libraries, and a simple web app you can deploy online.“



A. Sweigart, R. Górczyński, *Automate the Boring Stuff with Python, Practical Programming for Total Beginners*, 2. Aufl., Helion, Gliwice, **2021**.

„In this fully revised second edition of the best-selling classic Automate the Boring Stuff with Python, you'll learn how to use Python to write programs that do in minutes what would take you hours to do by hand—no prior programming experience required. You'll learn the basics of Python and explore Python's rich library of modules for performing specific tasks, like scraping data off websites, reading PDF and Word documents, and automating clicking and typing tasks. The second edition of this international fan favourite includes a brand-new chapter on input validation, tutorials on automating Gmail and Google Sheets, and tips on automatically updating CSV files.“



A. Sweigart, *Beyond the Basic Stuff with Python*, No Starch Press, San Francisco, **2020**.

„More than a mere collection of advanced syntax and masterful tips for writing clean code, you'll learn how to advance your Python programming skills by using the command line and other professional tools like code formatters, type checkers, linters, and version control. Sweigart takes you through best practices for setting up your development environment, naming variables, and improving readability, then tackles documentation, organization and performance measurement, as well as object-oriented design and the Big-O algorithm analysis commonly used in coding interviews.“

4.2 Websites



LearnPython



RealPython

Antworten zu Coding-Problemen gibt es oft auch auf [stackoverflow.com](#), [geeksforgeeks.org](#), [programiz.com](#), oder [towardsdatascience.com](#). Die Qualität der Informationen dieser Webseiten variiert von Fall zu Fall.

4.3 YouTube-Tutorials

YouTube bietet eine große Anzahl umfangreicher Tutorials zu allen erdenklichen Python-verwandten Themengebieten. Während der große Vorteil des Angebotes darin liegt, dass es kostenlos ist, sollte man sich bewusst sein, dass die Qualität des dargebotenen Inhalts von Tutorial zu Tutorial stark schwanken kann.

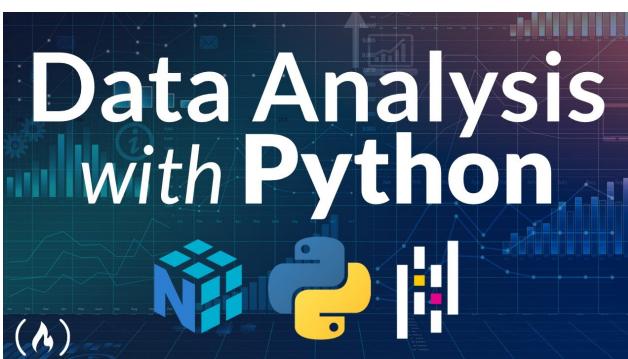
4.3.1 Themenübergreifende Tutorials



Allgemeiner Einführungskurs
(6:15 h, Python 3.7)



Allgemeiner Einführungskurs
(12:00 h, Python 3.9)



Einführungskurs zu Data Analysis
(10:00 h)



Einführungskurs zu Material Informatics
(19:20 h)

4.3.2 Tutorials zu wichtigen Bibliotheken



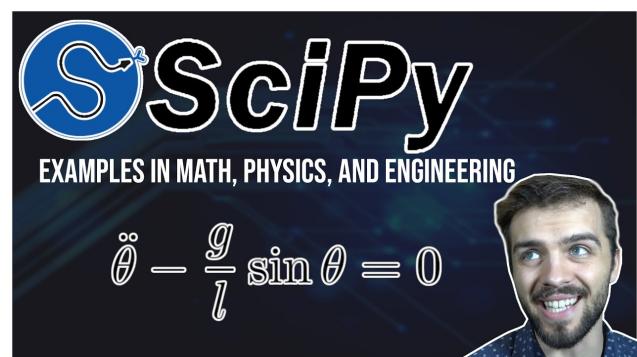
Einführungskurs in NumPy
(1:00 h, NumPy 1.13)



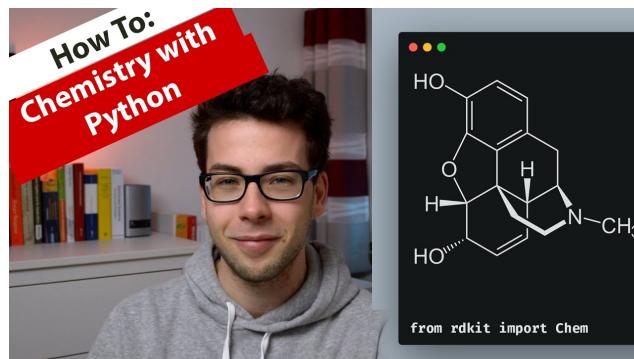
Einführungskurs in Pandas
(1:00 h, Pandas 0.22.0)



Einführungskurs in Matplotlib (ohne Seaborn)
(1:30 h, Matplotlib 1.3.0)



Einführungskurs in SciPy
(1:30 h, SciPy 1.4)



Einführungskurs in RDKit
(17:00 min, RDKit 2019.09.1)

4.4 Offizielle Ressourcen verschiedener Bibliotheken

- ▶ **Python Allgemein:** [Offizielles Tutorial](#) zu grundlegenden Funktionalitäten von Python und Funktionen der Python-Standardbibliothek sowie [PEP 8](#) mit Konventionen für Code und [PEP 257](#) mit Konventionen für Doc Strings
- ▶ **NumPy:** [Übersicht über NumPy](#) inklusive „[Quickstart Guide](#)“ und „[the absolute Basics for Beginners Guide](#)“
- ▶ **Pandas:** [Paketübersicht](#), „[Getting started tutorials](#)“ zu ausgewählten Basisfunktionalitäten, „[10 minutes to pandas](#)“ mit weiteren, darauf aufbauenden Anleitungen zu ausgewählten Themenkomplexen und [Pandas Cheat Sheet](#)
- ▶ **SciPy:** „[User Guide](#)“ aller Sub-Module

- ▶ **Matplotlib:** Übersicht von [Diagrammtypen](#), eine große Auswahl an [Anfänger- und Fortgeschrittenentutorials](#), [Beispiele von Diagrammen](#) zum Nachbauen und [Cheat Sheets/Handouts](#)

Alle Funktionalitäten der jeweiligen Pakete, sprich welche Funktionen und Klassen diese zu Verfügung stellen und welche Argumente jene Funktionen und Klassen akzeptieren, sind in ihren sogenannten *API Referenzen* aufgelistet.

4.5 Kurse auf „massive open online course“ Plattformen

Viele solcher Kurse kosten Geld, wobei sich auf einigen Plattformen das System etabliert hat, den Kurs selbst kostenlos, ein Zertifikat über die Absolvierung des Kurses aber kostenpflichtig anzubieten. Bekannte Plattformen sind beispielsweise [Coursera](#), [Codecademy](#), [Udemy](#), oder [Skillshare](#).

4.6 Initiativen

- ▶ **TechLabs:** Eine gemeinnützige Lernplattform für junge Menschen aller Fachrichtungen. TechLabs bietet eine technische Ausbildung in Form des *Digital Shaper Program* an. Dieses Programm zielt darauf ab, junge Menschen mit technischem Fachwissen sowie methodischen und sozialen Fähigkeiten auszustatten, die heute und in unserer (zunehmend automatisierten) zukünftigen Arbeitswelt von großer Bedeutung sind. Das viermonatige Programm besteht neben Online-Kursen aus einer Projektphase, in der das erworbene Wissen direkt in der Praxis zusammen mit einem erfahrenen Mentor angewendet wird. Die Teilnahme ist kostenlos.
- ▶ **pythoninchemistry:** Die Seiten von [pythoninchemistry.org](#) wurden an der University of Bath zusammengestellt und sind Ressourcen für Schüler:innen und Lehrer:innen gleichermaßen. Sie zielen darauf ab, eine Einführung in die Programmiersprache Python mit einem Verständnis der Chemie zu verbinden. Auf dieser Seite wird eine Reihe von Ressourcen vorgestellt, die den Einstieg in Python und die Nutzung von Jupyter-Notebooks behandeln.

4.7 Angebote der Christian-Albrechts-Universität zu Kiel

- ▶ **Kurse des Lehrstuhls für Service Analytics:** Der Inhalt dieses Workshops beruht auf dem Inhalt der Vorlesung „Computational Modeling for Business“. Neben dieser und anderen Vorlesungen bietet der Lehrstuhl die Möglichkeit, eigene Programmierprojekte in Form von Forschungsseminaren zu absolvieren.
- ▶ **opencampus.sh:** „Werde einzig, nicht artig“ — opencampus.sh ist ein unabhängiges, regionales Bildungscluster und somit ein Update für die deutsche Bildungslandschaft und die Innovationskultur des Nordens. Die Teilnehmenden verwirklichen hier eigene Projekte und Ideen. Außerdem gibt es bei opencampus jede Menge Jobperspektiven zu entdecken. Jedes Semester sind etwa 600 Menschen mit dabei. Opencampus bietet vor allem Kurse zum Thema künstliche Intelligenz, wobei dort in Python programmiert wird. Kurse von Opencampus SH können im Wahlpflichtbereich des Chemie- und Wirtschaftschemiestudiums angerechnet werden.
- ▶ **Zertifikatsstudium Informatik:** Das Institut für Informatik bietet für Studierende anderer Fächer die Möglichkeit an, tiefere Kenntnisse in der Informatik zu sammeln und über die erfolgreichen Leistungen ein Zertifikat zu erhalten. Der Umfang des Zertifikatsstudiums beträgt mindestens 30 Leistungspunkte und kann in Teilen nach individuellen Interessen gestaltet werden. Der Einstieg in das Informatik-Zertifikat erfolgt über grundlegende Module, welche auch im Rahmen des Profils Fachergänzung im 2-Fächerstudiengang oder im Nebenfach vieler anderer Studiengänge belegt werden können.