



Einführung in die Programmierung mit Python

September 2023

INHALT

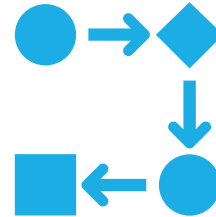
- Intro
- Colab und Python
- Grundlagen Python
- Objektorientierte Programmierung
- Pakete und Module
- Style Guide
- Integrierte Entwicklungsumgebung
- Links und Bücher



INTRO



KURSORGANISATION



ZEITPLANUNG

- 5 Tage, Mo - Fr
- Start: 09:00 Uhr
- Ende: 16:30 Uhr
- Pause nach 90 Min

VORGEHEN

- Grundlagen/Basiswissen
- Beispiele
- Training/Fallstudien

VERSCHIEDENES

- WLAN
- Pinboard

INFO BOX
Vorläufige Zeitplanung,
Änderungen möglich

ZEITPLANUNG



1. Tag

- Begrüßung Vorstellung, Zielsetzung, Programm
- grundlegende Konzepte von Python
- Erste Schritte, Anweisungen
- Zahlen und Zeichenketten
- Entwicklungsumgebung
- Erste Programme

2. Tag

- Wiederholung
- Standard-Datentypen
- Variable/Zuweisungen
- Rechenoperatoren
- Verzweigungen,
- Build-In-Funktionen
- Formatierungen

3. Tag

- Wiederholung
- Logische Operatoren
- Schleifen/Iterationen

- Beenden von Schleifen/Iterationen
- Ein-/Ausgaben, Dateien lesen/schreiben
- Eigene Funktionen
- Listen, Tupel, Sets, Dictionary

4. Tag

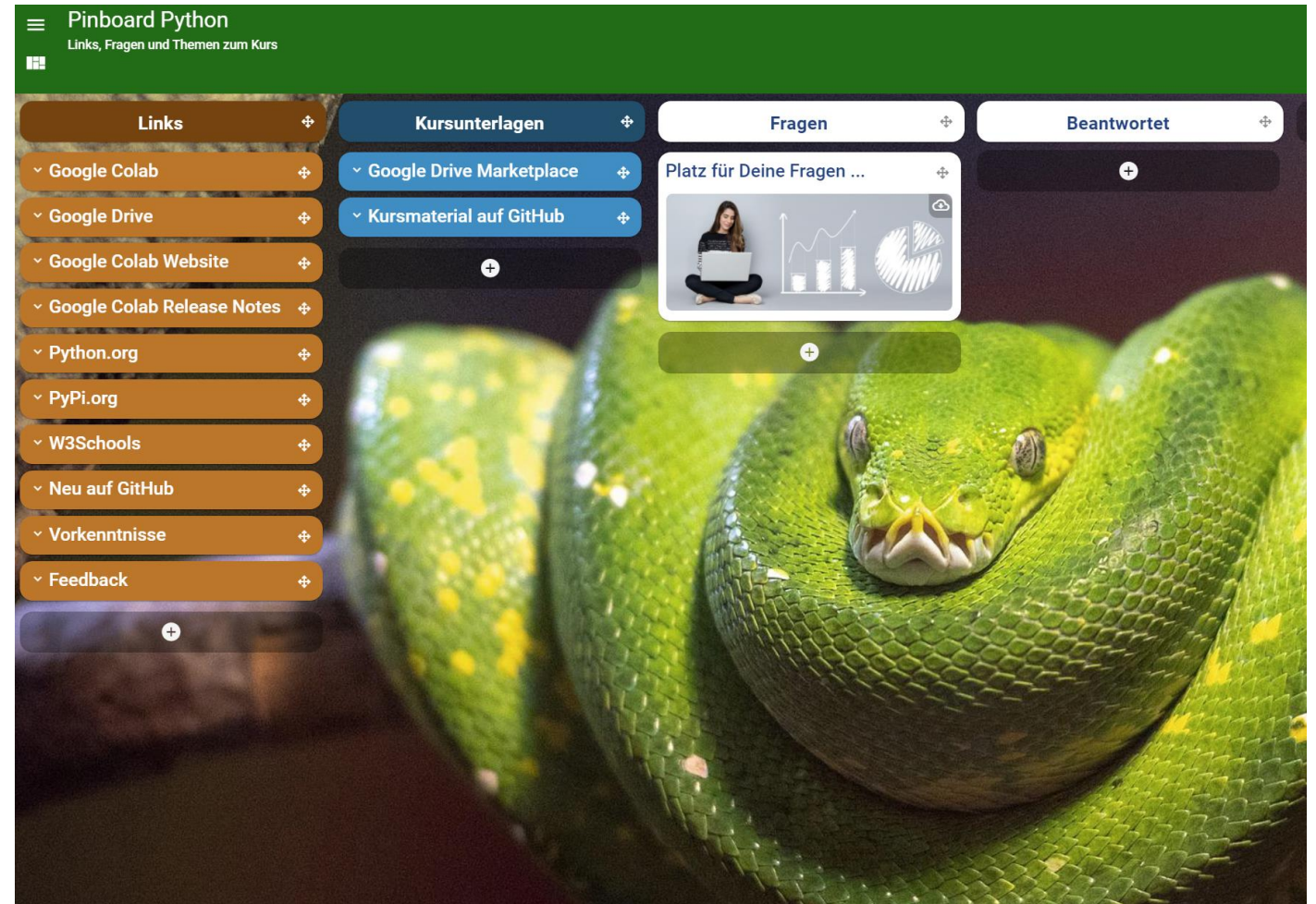
- Wiederholung
- Python-Bibliotheken/Module
- Eigene Module und Pakete
- Umgang mit Programmfehlern
- Objektorientierte Programmierung

5. Tag

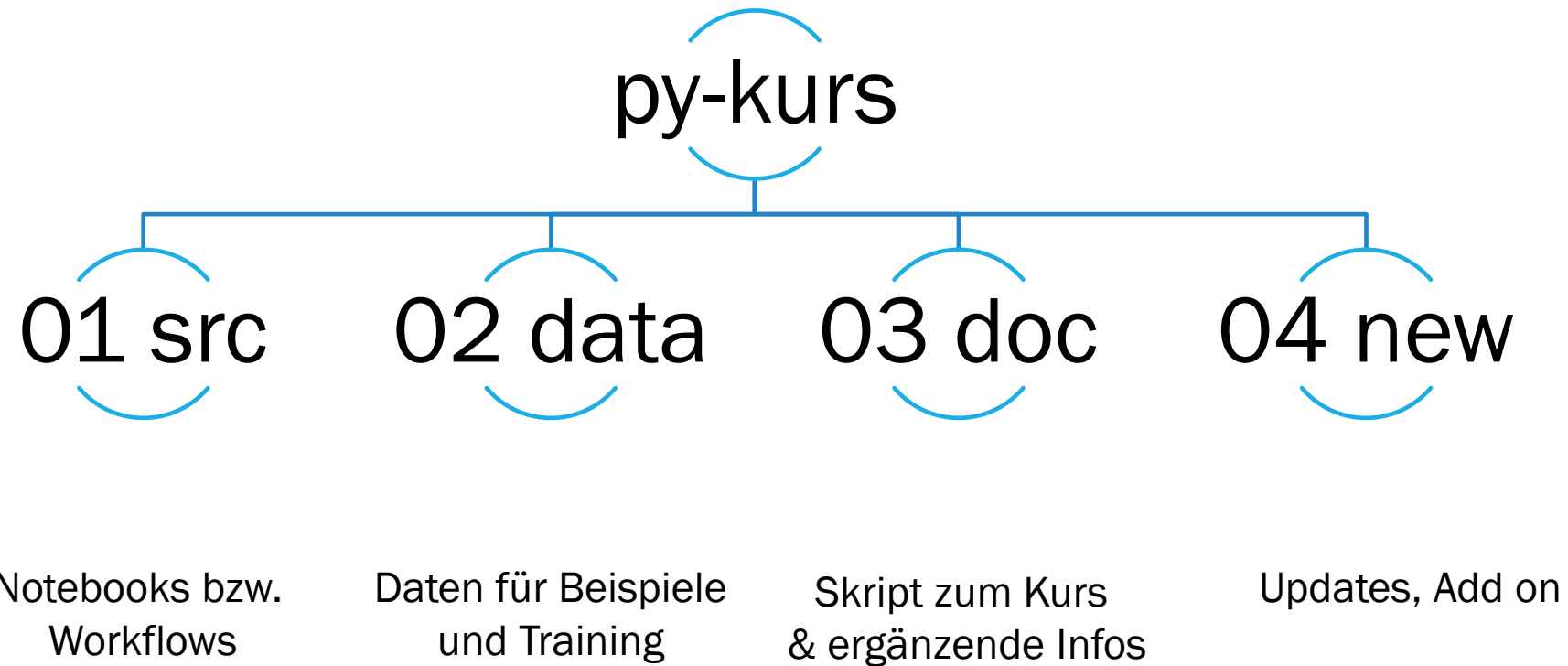
- Wiederholung
- Objektorientierte Programmierung
- Graphische Benutzeroberfläche
- Style-Guide
- Einblick: Socket, Concurrent Programming, Data structures & algorithms, Advanced Python-Code
- Kursevaluation

PINBOARD PYTHON

bit.ly/44696zA



STARTERKIT





COLAB UND PYTHON



GOOGLE COLAB(ORATORY)



- Google Colaboratory, kurz Colab, ist eine kostenlose Entwicklungsumgebung, die vollständig in der Cloud arbeiten wird.
- In Colab können Jupyter-Notebooks erstellt, bearbeiten und ausgeführt werden.
- Colab unterstützt viele beliebte Machine-Learning-Bibliotheken, die einfach in ein Notebook geladen werden können.
- Colab erlaubt es unterschiedliche Laufzeitumgebungen zu definieren in denen man neben einer CPU auch GPUs und TPUs verwenden kann.

CPU = Central Processing Unit, GPU = Graphics Processing Unit, TPU = Tensor Processing Unit

EXKURS: CPU/GPU/TPU

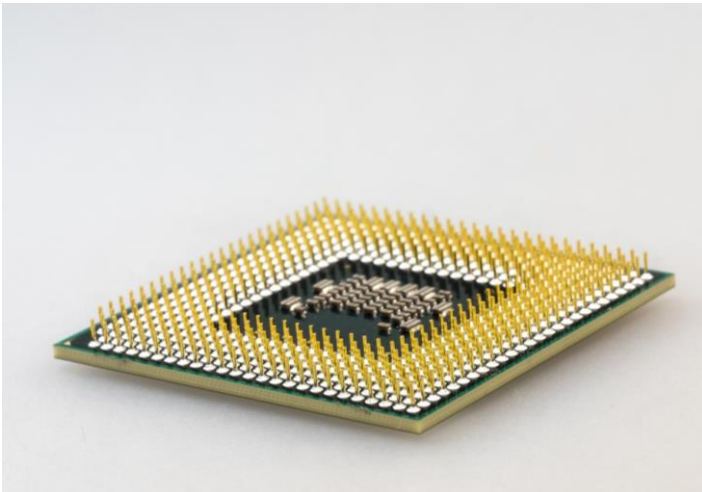


Bild von [Michael Schwarzenberger](#) auf [Pixabay](#)

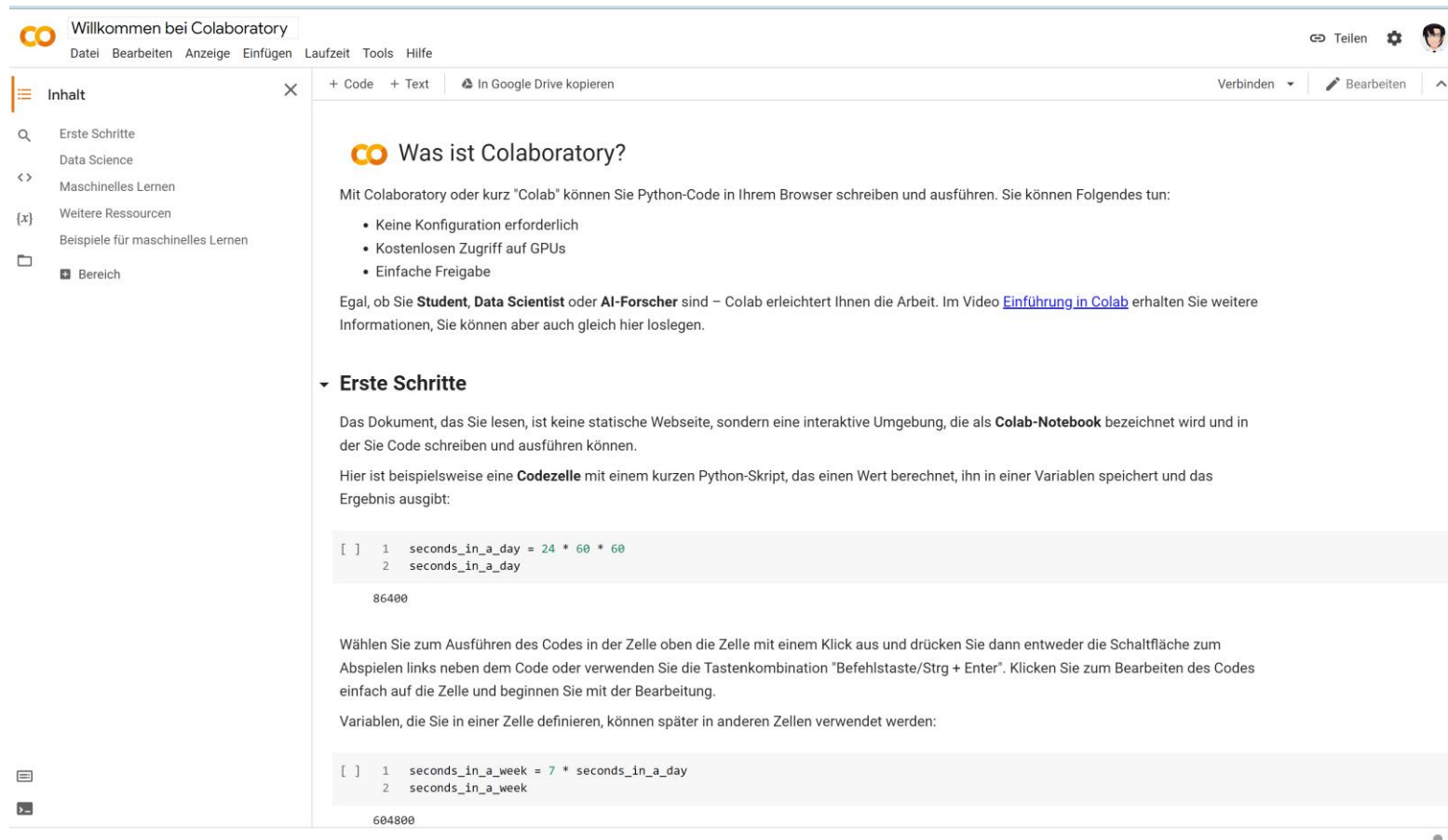
- CPU (Central Processing Unit) ist ein **Allzweckprozessor**, der die meisten Verarbeitungsaufgaben in einem Computer ausführt. Es kann eine Vielzahl von Funktionen verarbeiten, ist jedoch nicht für bestimmte Aufgaben wie Grafikverarbeitung oder Matrizenberechnungen optimiert.
- GPU (Graphics Processing Unit) wurde **speziell** für die **Grafik- und Videoverarbeitung** entwickelt. Es verfügt über viele kleine Kerne, die für die Parallelverarbeitung optimiert sind, wodurch es für Aufgaben wie das Rendern von 3D-Bildern, das Abspielen von Videos und das Ausführen komplexer Simulationen viel schneller als eine CPU ist.
- TPU (Tensor Processing Unit) ist ein benutzerdefinierter Chip, der **speziell** für Aufgaben des **maschinellen Lernens** entwickelt wurde, insbesondere mit dem TensorFlow-Framework. Es ist für Matrixberechnungen und andere Operationen optimiert, die häufig im Deep Learning verwendet werden, wodurch es viel schneller und effizienter für das Training großer neuronaler Netze ist.
- Zusammenfassend lässt sich sagen, dass die CPU ein Allzweckprozessor ist, die GPU auf die Grafikverarbeitung spezialisiert ist und die TPU auf maschinelles Lernen spezialisiert ist.



JUPYTER

- Die Jupyter App ist eine Anwendung, die das Bearbeiten und Ausführen von Programmiersprachen, u.a. Python, über einen Webbrowser ermöglicht.
- Der Name Jupyter bezieht sich auf die drei wesentlichen Programmiersprachen **Julia**, **Python** und **R** und ist auch eine Hommage an Galileos Notizbucheinträge zur Entdeckung der Jupitermonde.
- Mit der Jupyter App kann man Notizbücher erstellen. Diese Notizbücher (Dateiendung .ipynb) enthalten:
 - Programmcode, der ausgeführt werden kann,
 - Markdown-Zeilen, das sind Textzeilen mit Formatierungsangaben.
- Die Jupyter-Notebooks werden v.a. für interaktive wissenschaftliche Analysen und Berechnungen, z.B. Data Analytics und Machine Learning, verwendet.

GOOGLE COLAB - EINSTIEG



Willkommen bei Colab Grundlagen

- Menueleiste
- Obere Symbolleiste
- Linke Symbolleiste
- Befehlspalette
- Einstellungen
- Teilen
- Zellen
 - Code
 - Text/Kommentar

PYTHON STARK GEFRAGT

Der Popularity of Programming Language Index (PYPL-Index) misst, wie oft Programmiersprachen-Tutorials gegoogelt werden.

GitHub ist weltweit eines der größten Code-Repositories (netzbasierter Dienst zur Versionsverwaltung für Software) mit einer riesigen Entwickler-Community.

PYPL Ranking: Wie oft werden Tutorials gegoogelt?



1	Python	30,17%
2	Java	17,18%
3	JavaScript	8,21%
4	C#	6,76%
5	C/C++	6,71%
6	PHP	6,13%
7	R	3,81%
8	Objective-C	3,56%
9	Swift	1,82%
10	Matlab	1,8%

Quelle: PYPL

5 | FiveTeams

GitHub Ranking: Popularität auf GitHub



1	JavaScript	20,15%
2	Python	15,87%
3	Java	11,40%
4	Go	8,80%
5	TypeScript	7,50%
6	C++	6,90%
7	Ruby	6,20%
8	PHP	5,0%
9	C#	3,70%
10	C	2,90%

Quelle: GitHub

5 | FiveTeams

Quelle: Abgefragt 17.03.2023
[Programmiersprachen 2023: Das große Ranking \(fiveteams.com\)](https://www.fiveteams.com/)

PROGAMMIERSPRACHEN – EIN ERSTER EINSTIEG

Natürliche & formale Sprachen

- Natürliche Sprache:
Eine von Menschen gesprochene Sprache oder eine Gebärdensprache, die aus einer historischen Entwicklung entstanden ist.
- Formale Sprache:
Eine abstrakte Sprache, zur Anwendung in der Linguistik, der Logik oder der Informatik.

Syntax und Semantik

- Syntax: Formaler Aufbau, Regelsystem
 - Buchstaben/Zeichen
 - Wörter, Ausdruck
 - Sätze, Anweisung
- Semantik: Bedeutung sprachlicher Zeichen und Zeichenfolgen



PYTHON – EINE PROGRAMMIERSPRACHE

- Python ist eine universelle, üblicherweise interpretierte, höhere Programmiersprache.
- Die Sprache wurde Anfang der 1990er Jahre von Guido van Rossum entwickelt.
- Sie fördert einen gut lesbaren, knappen Programmierstil.
- Python unterstützt mehrere Programmierparadigmen, z. B. die objektorientierte und funktionale Programmierung.
- Als die Entwicklung/Implementierung von Python begann, las Guido van Rossum auch die veröffentlichten Drehbücher aus "Monty Python's Flying Circus", einer BBC-Comedy-Serie aus den 1970er Jahren.
- Van Rossum dachte, er brauche einen Namen, der kurz, einzigartig und leicht mysteriös sei, also beschloss er, die Sprache Python zu nennen.

INTERPRETER VS COMPILER

Merkmal	Interpretierende Sprachen	Compilierende Sprachen
Ausführung	Quellcode wird während der Laufzeit interpretiert und ausgeführt	Quellcode wird in eine ausführbare Form übersetzt, bevor es ausgeführt wird
Portabilität	In der Regel portabler als compilierende Sprachen, da sie auf verschiedenen Betriebssystemen ohne zusätzliche Anpassungen ausgeführt werden können	Erfordern in der Regel, dass das Programm für jedes Zielsystem separat übersetzt wird
Performance	In der Regel langsamer als compilierende Sprachen, da sie den Quellcode während der Laufzeit interpretieren müssen	In der Regel schneller als interpretierende Sprachen, da sie den Code in eine ausführbare Form übersetzen
Fehlererkennung	Können Fehler schneller erkennen, da sie den Quellcode Schritt für Schritt während der Laufzeit ausführen	Bieten eine stärkere Typsicherheit, da sie den Code auf Typfehler prüfen können
Debugging	Einfacher, da der Entwickler den Quellcode während der Laufzeit untersuchen und Änderungen vornehmen kann	Schwieriger, da Entwickler den ausführbaren Code untersuchen müssen
Flexibilität	In der Regel flexibler als compilierende Sprachen, da sie es dem Entwickler ermöglichen, den Quellcode zur Laufzeit zu ändern	Erfordern in der Regel, dass der Quellcode erneut kompiliert wird, um Änderungen zu berücksichtigen
Speicher-Management	In der Regel automatisch durch die Sprache übernommen	Erfordert in der Regel manuelles Speicher-Management durch den Entwickler
Beispiele	Python, JavaScript, Ruby, Lua, PHP	C, C++, Java, Swift, Rust, Cobol, Fortran

MERKMALE VON SKRIPTSPRACHEN

Python ist eine Skriptsprachen, die über einen Interpreter ausgeführt wird.

Merkmale sind:

- implizit deklarierte Variablen (erfolgt durch Wertzuweisung),
- dynamische Typisierung (Typisierung erfolgt zum Zeitpunkt der Nutzung),
- automatische Speicherverwaltung (Belegung und Freigabe von Arbeitsspeicher),
- unmittelbare Ausführung durch Interpretation des Quelltextes ohne getrennte Übersetzungsphase.

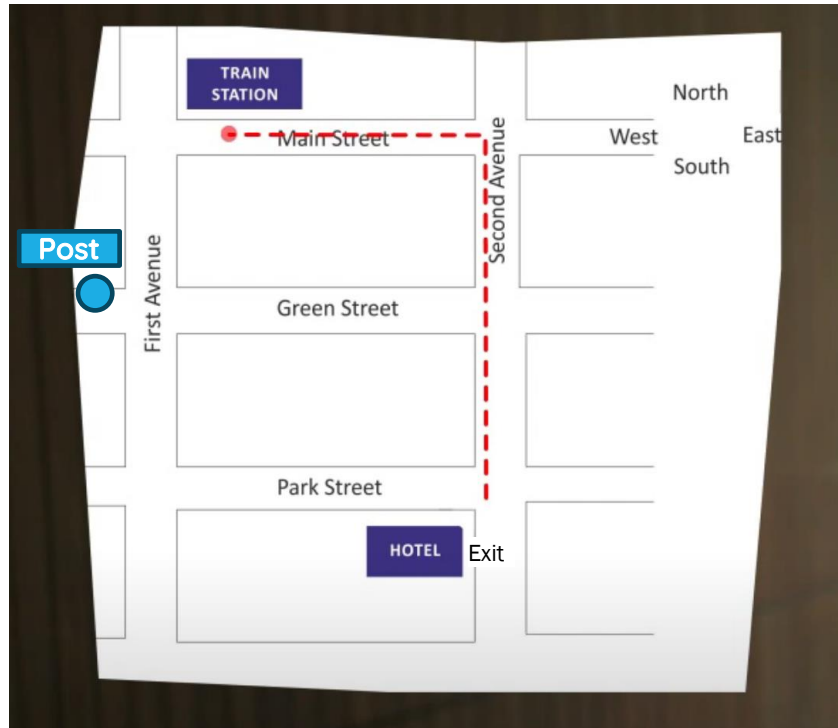
Quelle: Abgefragt 29.08.2021
[Allgemeine Python-FAQ – Python 3.9.7-Dokumentation](#)



GRUNDLAGEN PYTHON



EINFACHE ABLÄUFE / SEQUENZEN



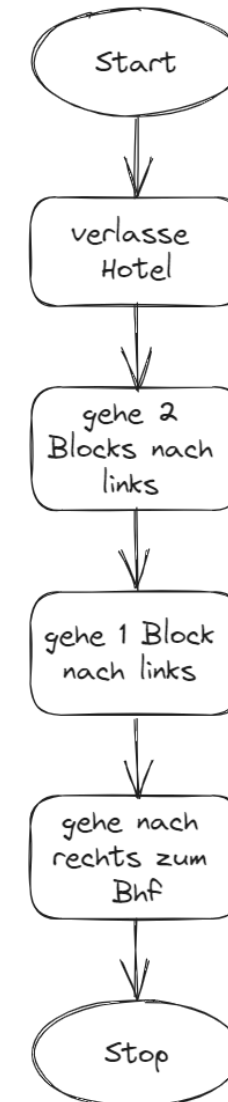
- Ein Python Programm besteht im Regelfall aus mehreren Anweisungen.
- Die einzelnen Anweisungen repräsentieren die Schritte vom Start zum Ziel.
- Die Anweisungen werden sequenziell (nacheinander) ausgeführt.
- Einfache Programme werden häufig nach dem EVA-Prinzip gestaltet (Eingabe – Verarbeitung – Ausgabe).

Quelle: Abgefragt 30.08.2021 [1.1.2 - CODE YOURSELF - Representation of Algorithms - YouTube](#)

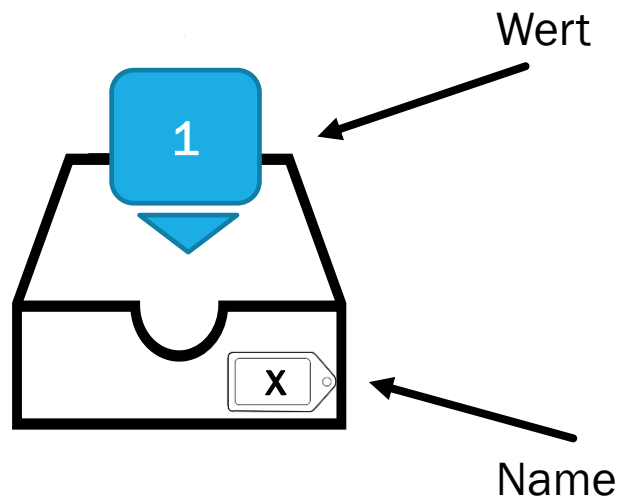
PROGRAMMABLAUFPLAN

- Ein Programmablaufplan (PAP) ist ein Ablaufdiagramm für ein Computerprogramm.
- Es ist eine grafische Darstellung und beschreibt die Folge von Aktionen bzw. Operationen zur Lösung einer Aufgabe.
- Über Symbole wird der Programmablauf bzw. der Datenfluss dargestellt.

Beispiel: Weg zum Bahnhof



ZUWEISUNGEN - VARIABLEN



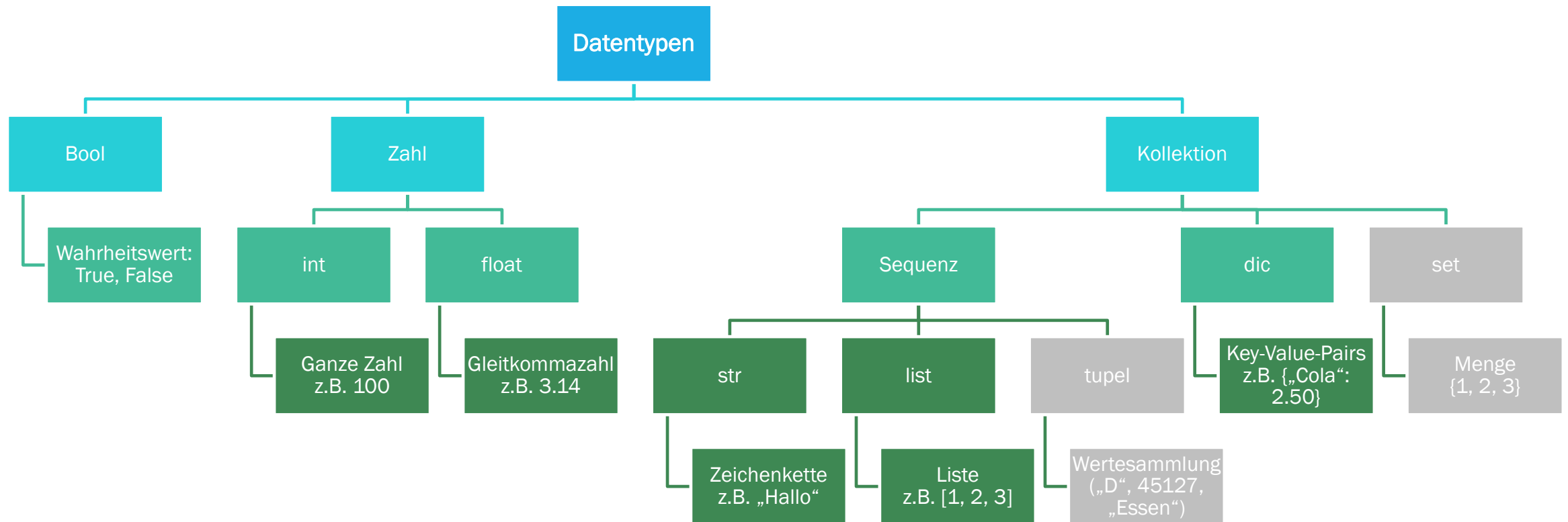
- Zuweisungen sind die häufigsten Befehle in einem Programm.
- Die einfachste Form besteht aus einem Namen gefolgt von einem Wert

Name ← Wert

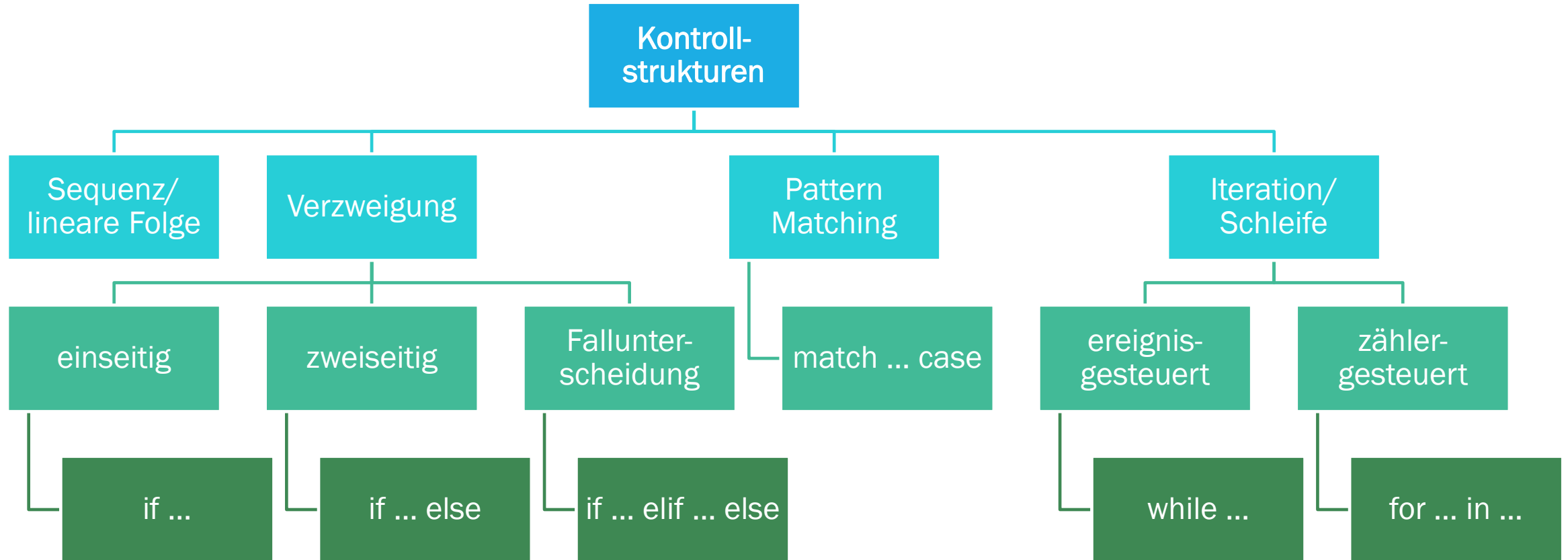
Beispiel: x ← 1

- Eine Variable ist ein „Behälter“, in dem man einen Wert aufbewahrt (speichert).
- Über dem Namen der Variablen kann man auf den Wert zugreifen.

WESENTLICHE DATENTYPEN



KONTROLLSTRUKTUREN



OPERATOREN (1/2)

Symbol	Rechenoperator
+ -	Vorzeichen
+ - * /	Grundrechenarten
//	Ganzzahlige Division
%	Rest der ganzzahligen Division
**	Exponentialfunktion
=	Zuweisung
+= oder -=	Zuweisen & Addition oder Subtraktion
*= oder /=	Zuweisen & Multiplikation o. Division

Symbol	Stringoperatoren
+	String verbinden
*	String vervielfachen

OPERATOREN (2/2)

Operator	Vergleichsoperator
==	Gleichheit testen
!=	Ungleichheit testen
< >	kleiner, größer
<= >=	kleiner-gleich, größer-gleich
in	testen, ob in Aufzählung enthalten

Operator	Logikoperatoren
or	logisches Oder
and	logisches Und
not	logisches Nicht

FUNKTIONEN

- Funktionen sind neben den Datentypen, Operatoren und Kontrollstrukturen die Bausteine einer jeden Programmiersprache.
- Python stellt vordefinierte Funktionen/Standardfunktionen zur Verfügung.
- Eine Funktion liefert ein vordefiniertes Ergebnis, sie löst eine Teilaufgabe in einem Programm.
- Sie werden für einfache und komplexe Aufgaben eingesetzt.
- Beispiele:

```
In [1]: 1 print("Hello Python")
```

```
Hello Python
```

```
In [2]: 1 len("Hello Python")
```

```
Out[2]: 12
```

genereller Aufbau:
Funktionsname(Argumente)



OBJEKTORIENTIERTE PROGRAMMIERUNG (OOP)



OBJEKTORIENTIERTE PROGRAMMIERUNG (OOP)

- Die objektorientierte Programmierung (kurz OOP) ist ein auf dem Konzept der Objektorientierung basierendes Programmierparadigma.
- Die Grundidee besteht darin, die Architektur einer Software an den Grundstrukturen desjenigen Bereichs der Wirklichkeit auszurichten, der die gegebene Anwendung betrifft.
- Ein Modell dieser Strukturen wird in der Entwurfsphase aufgestellt. Es enthält Informationen über die auftretenden Objekte und deren Verallgemeinerungen.
- Die Umsetzung dieser Denkweise erfordert die Einführung verschiedener Konzepte, insbesondere Klassen, Vererbung und Polymorphie.

GRUNDIDEE OBJEKTORIENTIERTE PROGRAMMIERUNG (OOP)

- Konzeption von formalen Modellen (Strukturen, Objekte, Beziehungen), die sich auf die reale Welt beziehen lassen.
- Übertragung des Modell-Bausteine in die Programmierung (z.B. Datentyp: Auto, Mensch, Straße).
- Ziel der objektorientierten Programmierung ist es, die Flexibilität und die Wartbarkeit von Programmen zu erhöhen.
- Bei OOP liegt der Fokus darauf, sich mit dem zu lösenden Problem auf Elemente der realen Welt zu beziehen und das Problem in Bezug auf diese Objekte und ihr Verhalten darzustellen.

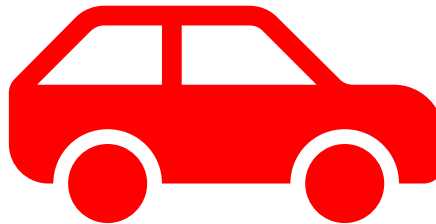
GRUNDBEGRIFFE OOP

Klasse – legt die prinzipielle Gestalt (Attribute) und Fähigkeiten (Methoden) der Objekte fest. (~Datentyp)

Klasse: Auto

Methoden:

- Fahren
- Parken
- Tanken
- ...



Attribute:

- Hersteller
- Modell
- Leistung
- Baujahr
- ...

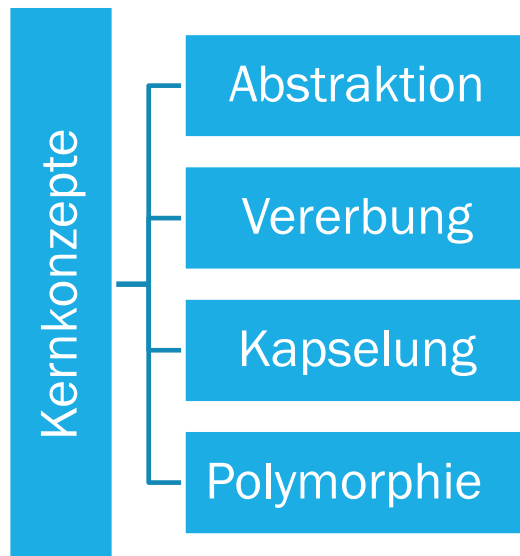
Methode – sind Fähigkeiten der Objekte. (~Funktion)

Attribut – kennzeichnet die Eigenschaften und somit die Unterschiede zwischen den Objekten. (~Bauplan des Datentyps)

Objekte: BMW X1, : Audi A3, ...

Objekt/Instanz – konkrete Ausprägung (~ Variable)

KERNKONZEPTE DER OBJEKTORIENTIERTE PROGRAMMIERUNG



- **Abstraktion:** Die Abstraktion ist der Prozess, bei dem komplexe Systeme oder Prozesse auf ihre wesentlichen Merkmale reduziert werden. In der objektorientierten Programmierung werden Klassen verwendet, um Abstraktionen zu erstellen, die die gemeinsamen Merkmale und Verhaltensweisen von Objekten darstellen.
- **Vererbung:** Vererbung ist ein Konzept, das es ermöglicht, eine neue Klasse auf der Grundlage einer vorhandenen Klasse zu erstellen. Die neue Klasse erbt die Eigenschaften und Methoden der vorhandenen Klasse und kann diese nach Belieben erweitern oder ändern. Dies ermöglicht es, effektiveren und wiederverwendbaren Code zu schreiben.
- **Kapselung:** Die Kapselung bezieht sich auf die Idee, dass Objekte bestimmte Informationen vor der Außenwelt verbergen können und nur ausgewählte Methoden und Eigenschaften für den Zugriff durch andere Objekte freigeben. Durch Kapselung wird die Interaktion mit Objekten auf eine definierte und kontrollierte Weise durchgeführt.
- **Polymorphie:** Polymorphie ermöglicht es, dass ein Objekt unterschiedliche Formen oder Verhaltensweisen annimmt, basierend auf dem Kontext, in dem es verwendet wird. Das bedeutet, dass ein Objekt in der Lage ist, verschiedene Methoden oder Eigenschaften bereitzustellen, je nachdem, wie es verwendet wird.

INIT

```
class Auto:
    def __init__(self): # Methode zur Erstellung des Bauplans der Klasse
        self.hersteller = None
        self.modell = None
        self.leistung = None
    def __str__(self): # Methode zur Ausgabe einer Klasse
        return f"{self.hersteller}, {self.modell}, {self.leistung}"
```

- `def __init__` wird aufgerufen, wenn eine neue Instanz einer Klasse erstellt wird.
- Es wird verwendet, um die Eigenschaften/Attribute der Instanz zu initialisieren.
- Es liefert damit den Bauplan, nach dem das Objekt bzw. die Instanz erstellt wird.

STR

```
class Auto:
    def __init__(self): # Methode zur Erstellung des Bauplans der Klasse
        self.hersteller = None
        self.modell = None
        self.leistung = None
    def __str__(self): # Methode zur Ausgabe einer Klasse
        return f"{self.hersteller}, {self.modell}, {self.leistung}"
```

- Die `__str__`-Methode in Python ist eine spezielle Methode, die verwendet wird, um einen lesbaren String einer Instanz einer Klasse zu erstellen.
- Wenn eine Instanz einer Klasse in Python als String gedruckt wird, ruft Python automatisch die `__str__`-Methode auf, um den String der Instanz zu erzeugen.
- Ohne `__str__`-Methode erfolgt die Ausgabe einer Referenz: `<__main__.Person object at 0x7f946bbf9dc0>`

SELF

```
class Auto:
    def __init__(self): # Methode zur Erstellung des Bauplans der Klasse
        self.hersteller = None
        self.modell = None
        self.leistung = None
    def __str__(self): # Methode zur Ausgabe einer Klasse
        return f"{self.hersteller}, {self.modell}, {self.leistung}"
```

- Die Aufgabe von self besteht darin, die Referenz des ausführenden Objekts zurückzuliefern.
- Aber, warum benötigt man das?
- Ganz einfach: Mit der Klasse, die wir oben definieren, geben wir ganz allgemein den Bauplan für unsere Objekte vor. Von dieser Klasse können wir aber beliebig viele Objekte erzeugen.
- Das Programm muss wissen, auf welchem Objekt es aktuell arbeitet.
- Genau deshalb existiert das self in Python. Dort wird nämlich automatisch beim Aufruf der init-Methode die Referenz auf das aktuelle Objekt mitübergeben.

FUNKTIONEN UND METHODEN

- Funktionen können direkt über ihren Namen aufgerufen werden – Methoden benötigen zusätzlich immer ihr Objekt.
- Die Schreibweise:
 - *Funktionen: funktionsname()*
 - *Methoden: objekt.methode()*
- Funktionen sind unabhängig. Einer Funktion kann beliebiges übergeben werden, mit dem dann weitergearbeitet wird.
- Methoden sind festgelegt – sprich jedes Objekt verfügt über bestimmte Möglichkeiten (sprich Methoden). Die Kunst ist nur zu wissen, was es an Methoden bereits „standardmäßig“ gibt.



PAKETE & MODULE



MODUL

- In Python ist ein Modul eine Datei, die Funktionen, Klassen und Variablen enthält.
- Module werden verwendet, um den Code in wiederverwendbare Teile aufzuteilen, was dazu beitragen kann, den Code effizienter zu gestalten und ihn einfacher zu warten.
- Ein Modul kann über das import-Statement in einem anderen Python-Skript oder Modul geladen werden, um auf dessen Funktionen, Klassen und Variablen zuzugreifen.
- Das import-Statement wird in der Regel am Anfang des Skripts oder Moduls verwendet.
- Python verfügt über eine Vielzahl von Standardmodulen, die in der Standardbibliothek enthalten sind und für eine Vielzahl von Aufgaben verwendet werden können.
- Es gibt auch eine große Anzahl von Drittanbietermodulen, die von der Python-Community entwickelt wurden und für spezielle Aufgaben oder Anwendungen nützlich sein können.

OFT EINGESETZT MODULE (1/2)

Web

- Requests: <https://pypi.org/project/requests/>
- Django: <https://pypi.org/project/Django/>
- Flask: <https://pypi.org/project/Flask/>
- Twisted: <https://twistedmatrix.com/trac/>
- BeautifulSoup:
<https://pypi.org/project/beautifulsoup4/>
- Selenium: <https://selenium-python.readthedocs.io/>

Data Science

- Numpy: <https://numpy.org/>
- Pandas: <https://pandas.pydata.org/>
- Matplotlib: <https://matplotlib.org/>
- Nltk: <https://www.nltk.org/>
- Opencv: <https://opencv-python-tutroals.readth...>

OFT EINGESETZT MODULE (2/2)

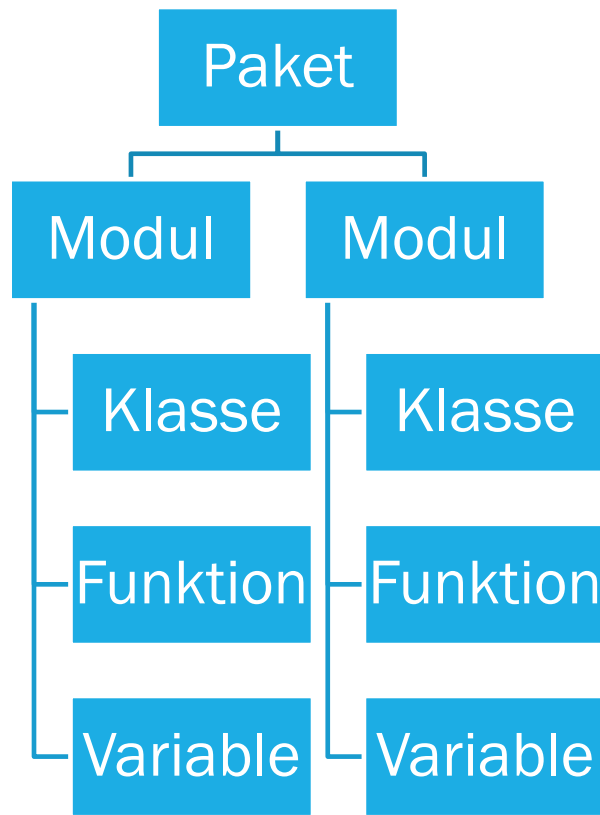
Machine Learning

- Tensorflow: <https://www.tensorflow.org/>
- Keras: <https://keras.io/>
- PyTorch: <https://pytorch.org/>
- Sci-kit Learn: <https://scikit-learn.org/stable/>

GUI

- Kivy: <https://kivy.org/#home>
- PyQt5: <https://pypi.org/project/PyQt5/>
- Tkinter: <https://wiki.python.org/moin/TkInter>
- Ipywidget: <https://ipywidgets.readthedocs.io/en/latest/>

PAKETE & MODULE



Verzeichnis

datei.py

Python-Code

Python-Code

Python-Code



STYLE GUIDE



STYLEGUIDE FÜR PYTHON-CODE (1/2)

- Einrückungen werden verwendet, um den Code lesbar zu gestalten.
- Eine Einrückungsebene in Python entspricht genau 4 Leerzeichen (keine Tabulatorzeichen).
- Jede Zeile soll aus höchstens 79 Zeichen bestehen.
- Lange Zeilen können über mehrere Zeilen unterbrochen werden, indem Ausdrücke in Klammern umgebrochen werden.
- Vermeiden Sie überflüssige Leerzeichen:
 - zu Anfang in Klammern
 - unmittelbar vor einem Komma, Semikolon oder Doppelpunkt
 - unmittelbar vor der offenen Klammer

Quelle: Abgefragt 30.08.2021 [PEP 8 – Style Guide for Python Code | Python.org](#)

STYLEGUIDE FÜR PYTHON-CODE (2/2)

- Am Anfang jeder Python-Datei und definierter Funktion steht ein Doc-String der kurz den Inhalt beschreibt.
- Doc-Strings beginnen mit `"""`, enden ebenfalls mit `"""` und in einer Zeile für sich selbst stehen.
- Jede Zeile eines Kommentars beginnt mit einem `#` und einem einzelnen Leerzeichen.
- Umgeben Sie Code-Blöcke der obersten Ebene mit zwei Leerzeilen.
- Variablennamen: Bestehen ausschließlich aus Kleinbuchstaben: Unterstriche sind erlaubt, wenn dies die Lesbarkeit des Codes verbessert.
- Funktionsnamen: wie bei Variablennamen.
- Paket- und Modulnamen: wie bei Variablennamen.

Quelle: Abgefragt 30.08.2021 [PEP 8 – Style Guide for Python Code | Python.org](#)



INTEGRIERTE ENTWICKLUNGSUMGEBUNG



INTEGRIERTE ENTWICKLUNGSUMGEBUNG

(IDE - INTEGRATED DEVELOPMENT ENVIRONMENT)

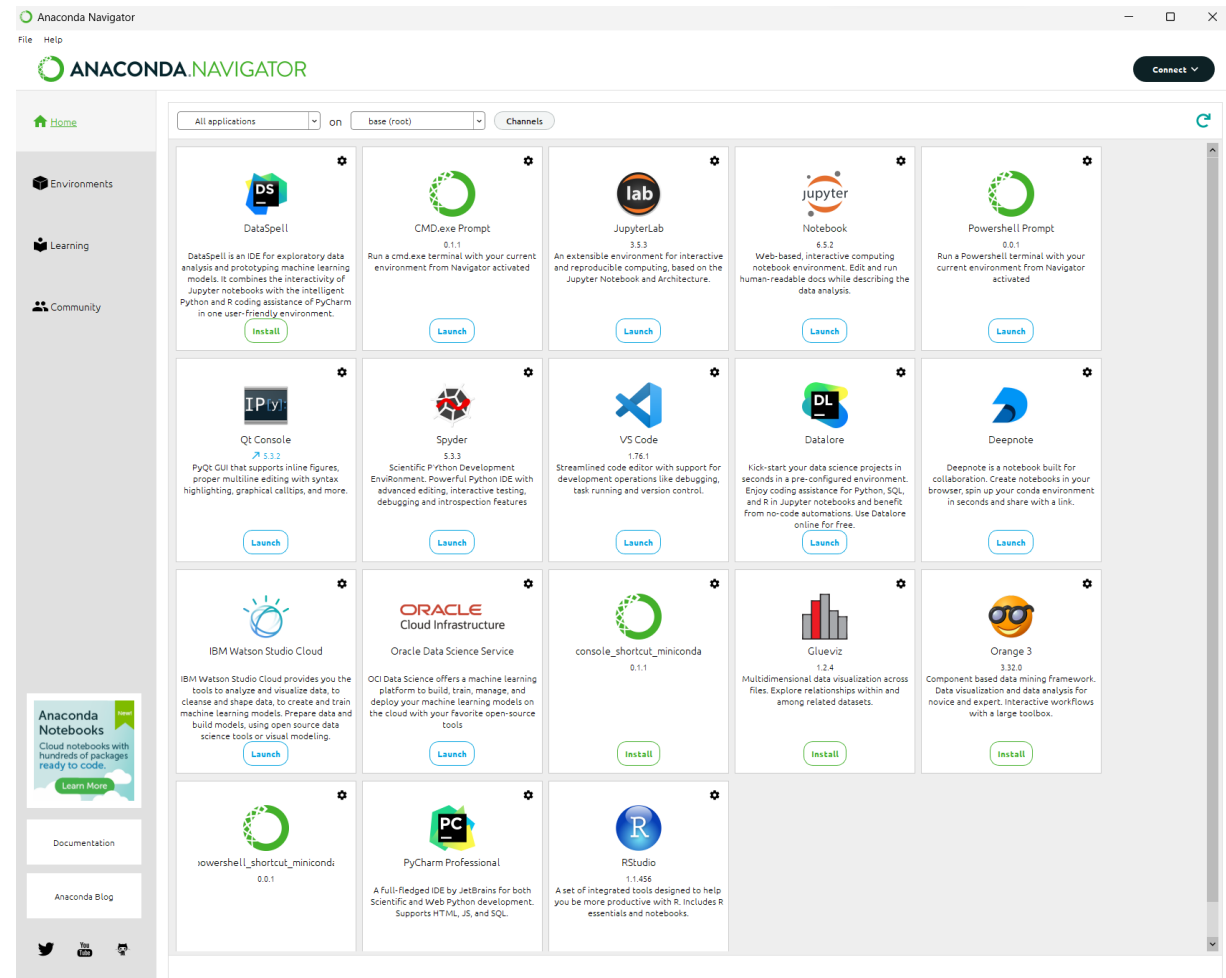
- Eine integrierte Entwicklungsumgebung ist eine Sammlung von Apps/Funktionen, mit denen die Aufgaben der Softwareentwicklung möglichst ohne Medienbrüche bearbeitet werden können.
- IDEs stellen hilfreiche Werkzeuge bereit, die Softwareentwicklern häufig wiederkehrende Aufgaben unterstützt, z.B. Arbeits(zwischen)ergebnisse verwaltet, Code erstellen/ändern, Code dokumentieren, Code testen, etc. Entwickler werden dadurch von formalen Arbeiten entlastet und können ihre eigentliche Aufgabe, das Entwickeln/ Programmieren von Software, mit Systemunterstützung effizient ausführen.
- IDEs gibt es für nahezu alle Programmiersprachen und Plattformen.
- Bekannte Python-DIE's sind:
 - Visual Studio Code
 - Jupyter
 - PyCharm
 - Spyder

Siehe z.B. auch: [10 Beste Python-IDE für Supercharge-Entwicklung und -Debugging \(geekflare.com\)](https://www.geekflare.com/best-python-ide-for-supercharge-development-and-debugging/)

ANACONDA

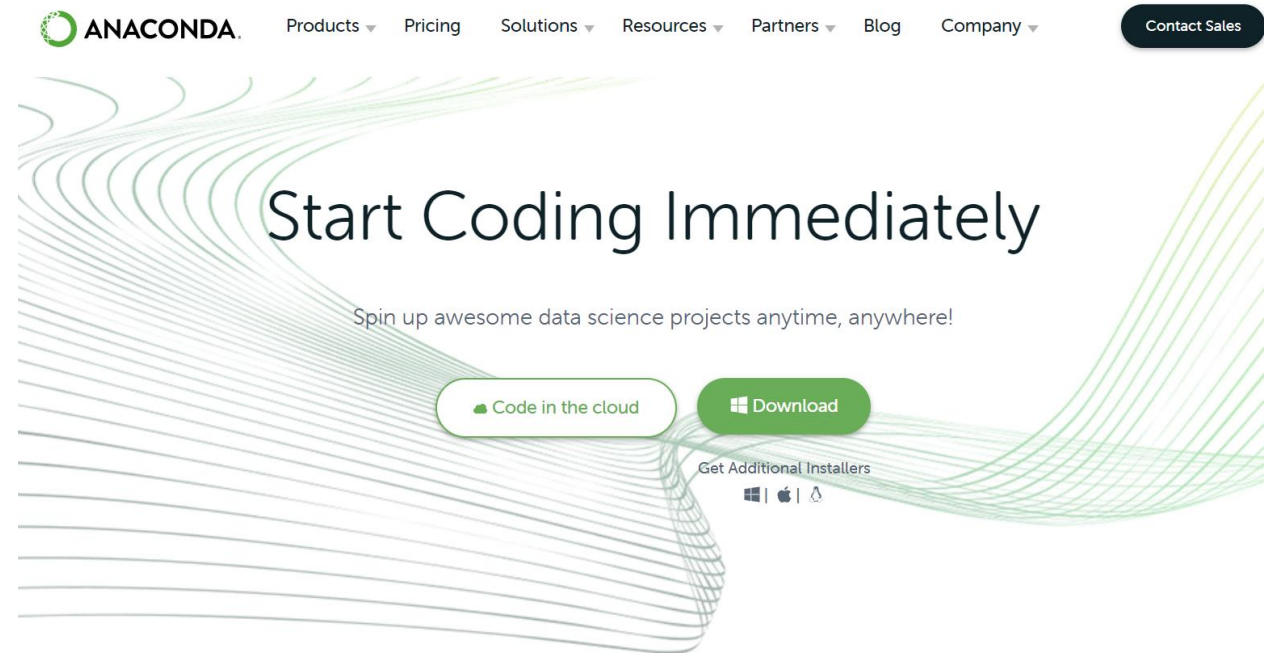
Anaconda ist eine Distribution für die Programmiersprachen Python und R, die unter anderem die Entwicklungsumgebung Visual Studio Code und Jupyter Notebook enthält.

Das Ziel der Distribution ist die Vereinfachung von Paketmanagement und Softwareverteilung.



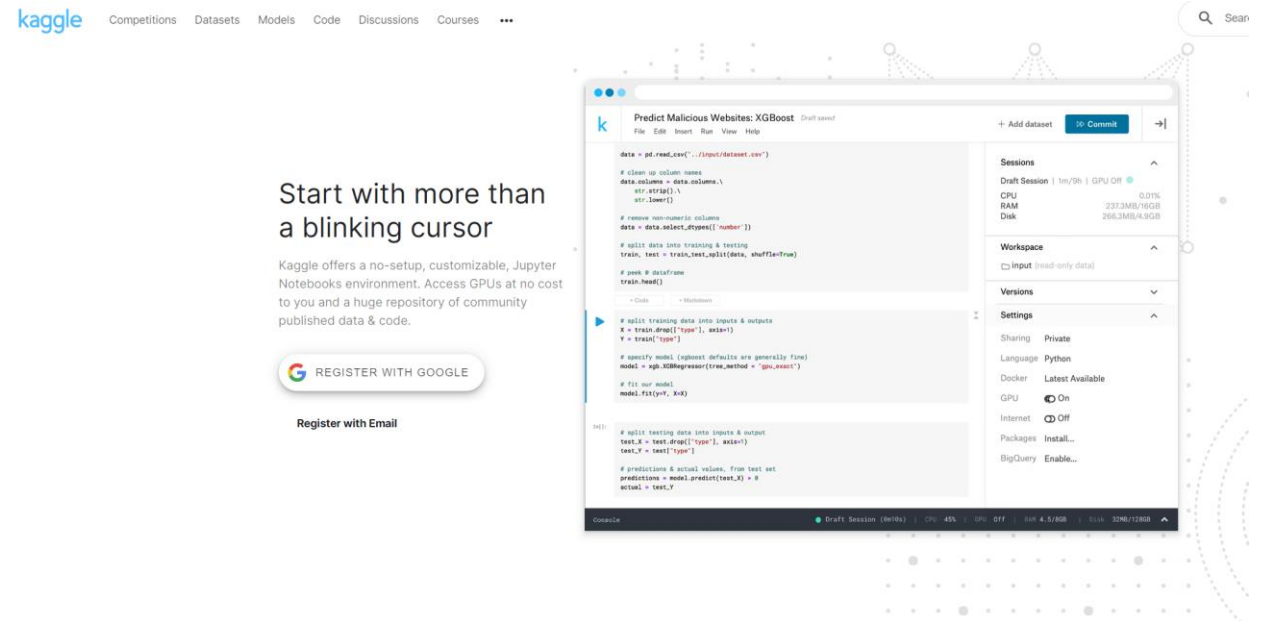
ANACONDA

Anaconda bietet neben dem Navigator auch einen in der Cloud gehosteten Notebook-Service mit einer vollständig geladenen und schlüsselfertigen interaktiven Entwicklungsumgebung. Läuft auf jedem Browser, jedem System. 100 % Installations- und Konfigurationsfrei.



KAGGLE

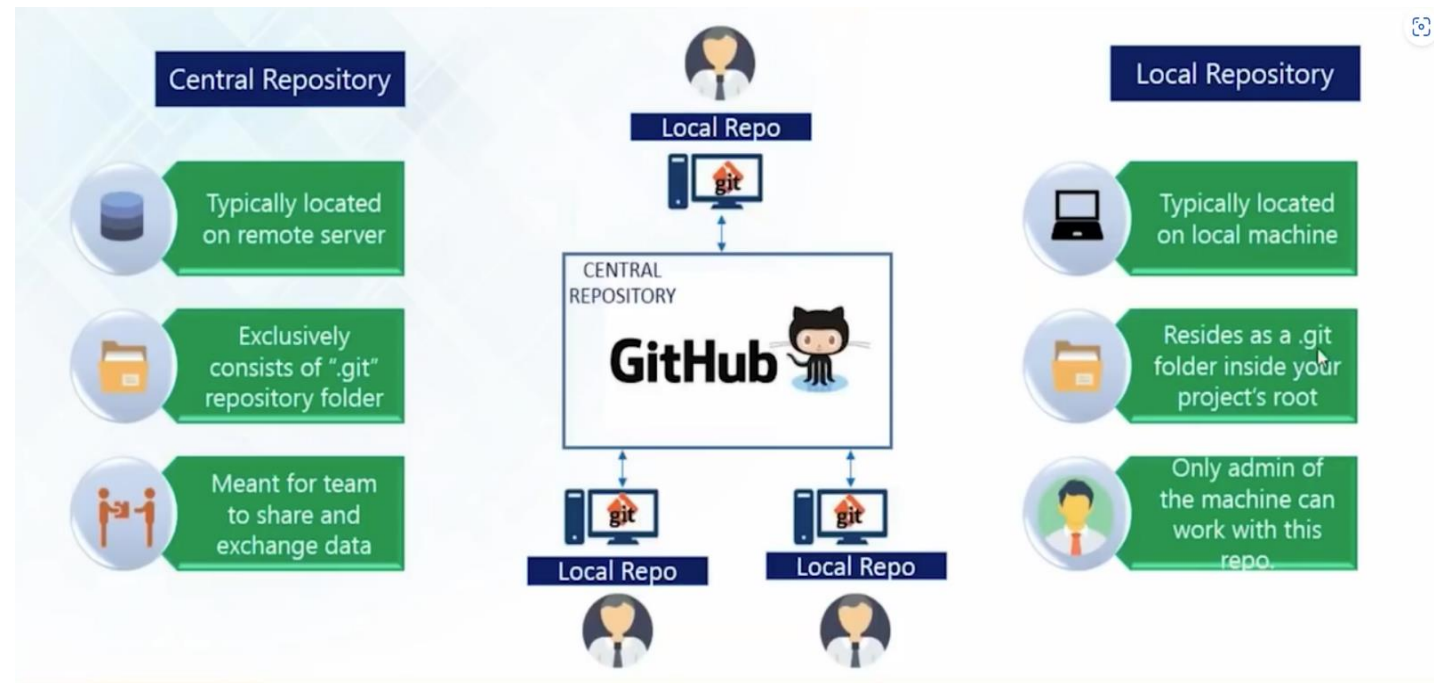
Kaggle ist eine Online-Community, die sich an Datenwissenschaftler richtet. Der Hauptzweck von Kaggle ist die Organisation von Data-Science-Wettbewerben. Kaggle bietet auch eine öffentliche Datenplattform, eine Cloud-basierte Workbench für die Datenwissenschaft und KI-Ausbildung an.



The image shows the Kaggle website interface. At the top, the Kaggle logo is followed by navigation links: Competitions, Datasets, Models, Code, Discussions, Courses, and a menu icon. Below the navigation bar, there's a section titled "Start with more than a blinking cursor". This section includes a brief description: "Kaggle offers a no-setup, customizable, Jupyter Notebooks environment. Access GPUs at no cost to you and a huge repository of community published data & code." Below this text are two buttons: "REGISTER WITH GOOGLE" (with the Google logo) and "Register with Email". To the right of the text is a screenshot of a Kaggle notebook titled "Predict Malicious Websites: XGBoost". The notebook interface shows a code editor with Python code for data loading, preprocessing, and model training using XGBoost. On the right side of the notebook, there's a sidebar with various settings and status information, including "Sessions" (Draft Session), "Workspace" (Input), "Versions", "Settings" (Sharing, Language, Docker, GPU, Internet, Packages, BigQuery), and system resources (CPU, RAM, Disk).

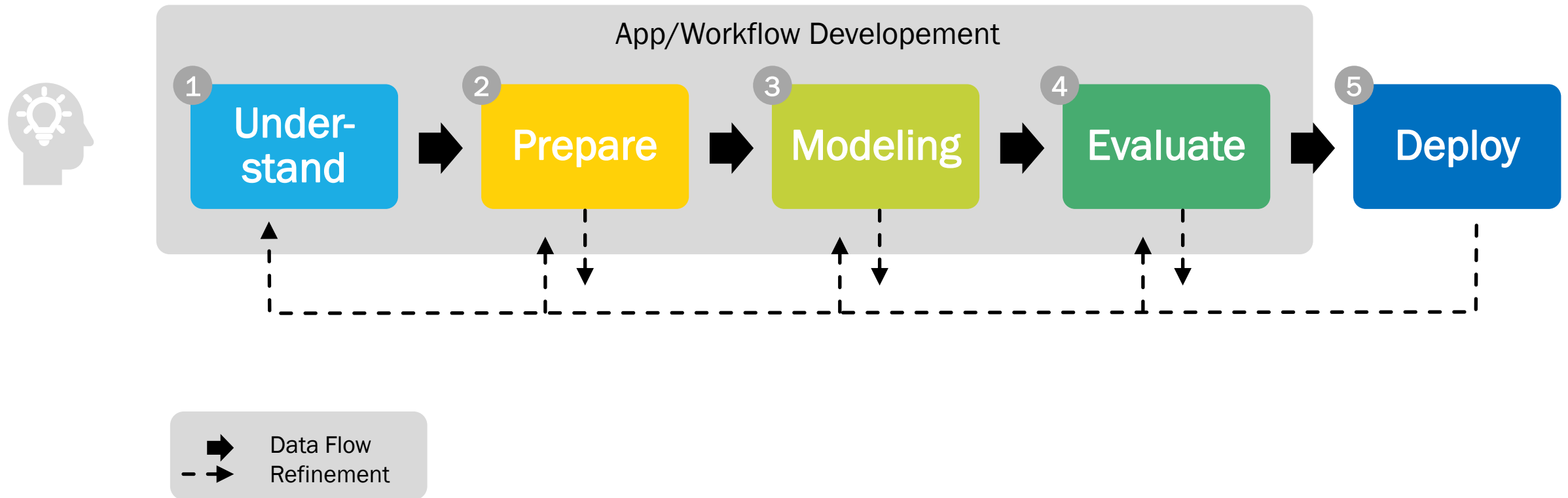
GIT & GITHUB

- Git (Global Information Tracker) ist ein Versionskontrollsystem, das entwickelt wurde, um das gemeinsame Arbeiten an Softwareprojekten zu vereinfachen.
- Es ermöglicht die Verwaltung von verschiedenen Versionen eines Codes, der von verschiedenen Personen bearbeitet wird.
- GitHub ist eine Webplattform, die auf Git basiert und Entwicklern eine einfache Möglichkeit bietet, Git-Repositories zu hosten und zu verwalten.
- GitHub ermöglicht es auch anderen Entwicklern, einen Beitrag zu einem Projekt zu leisten oder eine Kopie des Codes herunterzuladen und darauf aufzubauen.



Quelle: [Difference between Git and GitHub - Stack Overflow](#)

DEVELOPEMENT PROCESS



RE-USE

Gute
Option

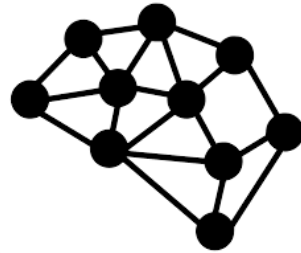


Bild von [Shirley Hirst](#) auf [Pixabay](#)

- Finde einen vorhandenen, ähnlichen Workflow.
- Nimm den einfachsten Workflow, der zu finden ist.
- Analysiere diesen Workflow und baue ein Verständnis auf.
- Tausche das Dataset aus, nimm die erforderlichen Anpassungen vor.
- Bringe den angepassten/veränderten Workflow zum Laufen.

KI ALS SUPPORTER / ASSISTANT

Kann
helfen



- Data Preparation:
KI kann genutzt werden, um bessere Entscheidungen bei der Datenvorverarbeitung zu treffen und wertvolle Erkenntnisse zu gewinnen.
- Feature Engineering:
KI kann bei der Entwicklung von Features helfen, indem es Hinweise gibt, Daten in einem bestimmten Format strukturiert und mit relevanten Informationen anreichert.
- Model Optimization:
KI kann dazu genutzt werden, ein bestehendes Machine Learning-Modell zu verbessern, indem es z.B. die Vorhersagegenauigkeit erhöht oder die Trainingszeit verkürzt.
- Codecompletion/-generation:
KI kann dazu genutzt werden Code zuschreiben bzw. Code zu überprüfen oder zu erklären.
- ...

VERSCHIEDENES

Python EXE Datei erstellen



Dauer: 3:38

[\(69\) Python Datei in EXE umwandeln - YouTube](#)

Jupyter Notebook in .py konvertieren



```
pip install ipynb-py-convert  
ipynb-py-convert abc.ipynb abc.py
```

[python - Convert JSON IPython notebook \(.ipynb\) to .py file - Stack Overflow](#)

PYTHON AUF DEM TABLET



- CARNET – Jupyter
- Pyto – Python 3
- Python3IDE



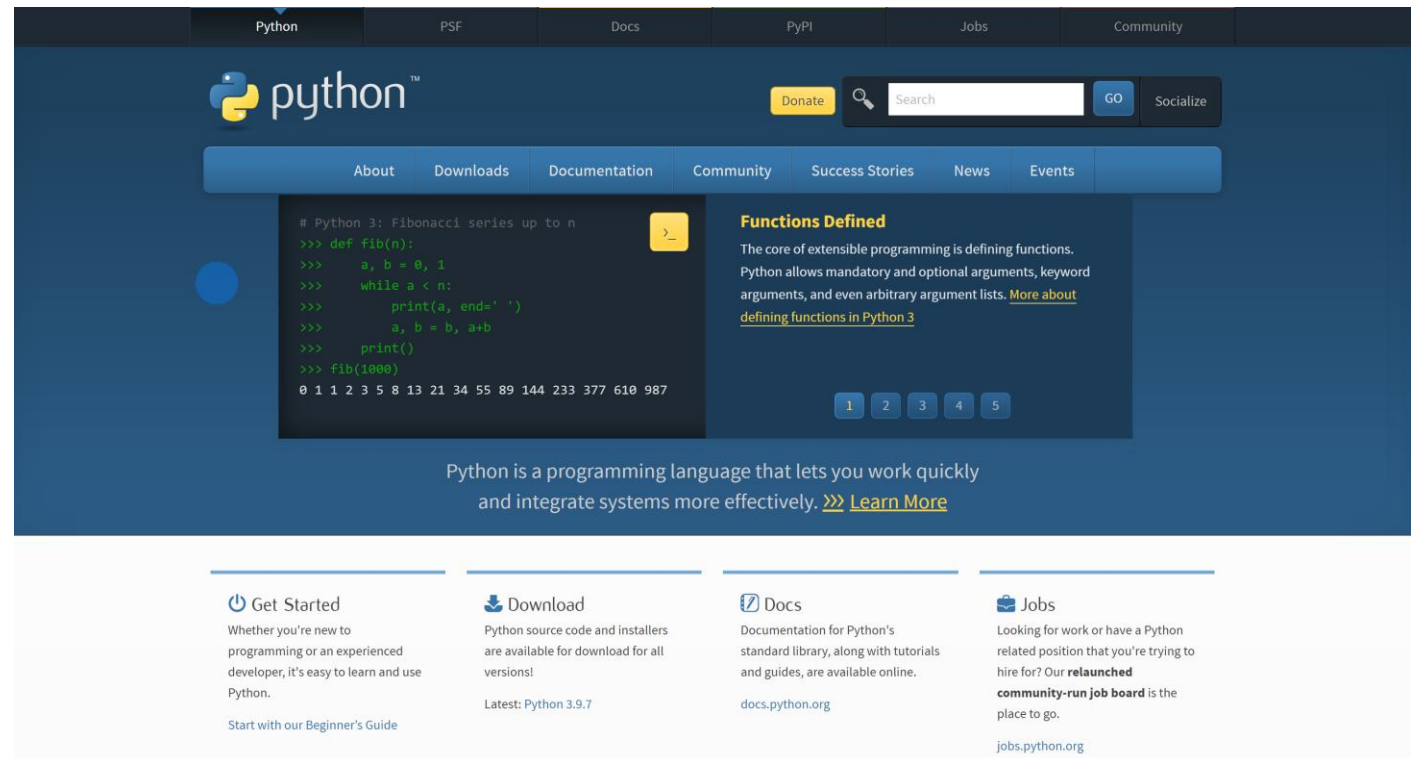
- Pydroid3



LINKS, BÜCHER



PYTHON.ORG



PYPI.ORG

Python-Paketindex

[[Mehr Details](#)]



The screenshot shows the PyPI.org homepage. At the top, there is a navigation bar with links for 'Hilfe', 'Sponsoren', 'Einloggen', and 'Registrieren'. The main heading reads 'Finde, installiere und veröffentliche Python-Pakete mit dem Python Package Index'. Below this is a search bar with the placeholder text 'Projekte suchen' and a magnifying glass icon. A link 'Oder [Projekte durchstöbern](#)' is provided. A statistics bar shows '440.462 Projekte', '4.285.092 Veröffentlichungen', '7.812.776 Dateien', and '682.282 Benutzer'. The main content area features the 'python Package Index' logo and a description: 'Der Python Package Index (PyPI) ist ein Software-Verzeichnis der Programmiersprache Python.' It also includes links for 'Learn about installing packages' and 'Erfahren sie hier wie sie ihren Python-Code für PyPI vorbereiten'. The footer contains links for 'Hilfe', 'Über PyPI', 'Mitwirken bei PyPI', and 'PyPI verwenden'.

Hilfe Sponsoren Einloggen Registrieren

Finde, installiere und veröffentliche Python-Pakete mit dem Python Package Index

Projekte suchen

Oder [Projekte durchstöbern](#)

440.462 Projekte 4.285.092 Veröffentlichungen 7.812.776 Dateien 682.282 Benutzer

python™
Package Index

Der Python Package Index (PyPI) ist ein Software-Verzeichnis der Programmiersprache Python.

PyPI hilft dabei, von der Python-Community entwickelte und geteilte Software zu finden und zu installieren. [Learn about installing packages](#).

Paket-Entwickler benutzen PyPI, um ihre Software zu veröffentlichen. [Erfahren sie hier wie sie ihren Python-Code für PyPI vorbereiten](#).

Hilfe Über PyPI Mitwirken bei PyPI PyPI verwenden



LINKS (1/2)

Eine Liste mit Links zu ...

- Python
- Lernplattform
- Fehlerhilfe
- Google Colab
- Anaconda

Thema	Link
Python	Welcome to Python.org
	The Python Standard Library – Python 3.9.6 documentation
	Our Documentation Python.org
Lernplattform	Dashboard HackerRank
	The Python Tutorial – Python 3.9.6 documentation
	Python Tutorial (w3schools.com)
	Python Tutorials – Real Python
Fehlerhilfe	Stack Overflow - Where Developers Learn, Share, & Build Careers
	Das deutsche Python-Forum - Foren-Übersicht
	Google-Suche
Google Colab	Willkommen bei Colaboratory - Colaboratory (google.com)
	(31) Complete Beginner's Tutorial to Google Colab - YouTube
Anaconda	Anaconda The World's Most Popular Data Science Platform

VIELE
TUTORIALS
AUF
YOUTUBE



LINKS (2/2)

Eine Liste mit Links zu ...

- Markdown
- Styleguide
- Jupyter CheatSheet
- VS Code
 - Intro
 - Debugger
- Git/GitHub

Thema	Link
Markdown	Basic Syntax Markdown Guide
Styleguide	PEP 8 -- Style Guide for Python Code Python.org
	styleguide Style guides for Google-originated open-source projects
	PEP 8: The Style Guide for Python Code
CheatSheet	Jupyter Notebook CheatSheet (edureka.co)
VS Code	
Intro	Visual Studio Code - Code Editing. Redefined
	Visual Studio Code Introductory Videos
	(3) VSCode Tutorial For Beginners - Getting Started With VSCode - YouTube
Debugger	(3) How to Use a Debugger - Debugger Tutorial - YouTube
Git/GitHub	(3) Git And GitHub in ~30 Minutes - YouTube

BÜCHER

