



Einführung in die Programmierung mit Python

Februar 2024

LIZENZ

Die Charts & das Kursmaterial wurden – soweit nicht anders angegeben – von Ralf Bendig erstellt.

Lizenz CC BY 4.0.

Titelseite: Bild mit DALL-E erstellt.

INHALT

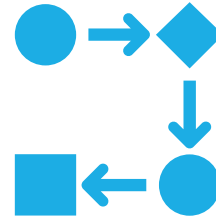
- Intro
- Colab und Python
- Grundlagen Python
- Objektorientierte Programmierung
- Pakete und Module
- Coding & KI – Just Good Friends
- Integrierte Entwicklungsumgebung
- Entwicklungsprozess
- Style Guide
- Anhang



INTRO



KURSORGANISATION



ZEITPLANUNG

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

VORGEHEN

- Grundlagen/Basiswissen
- Beispiele
- Training & Übungen
- Ergebnisse stellen Teilnehmer vor

VERSCHIEDENES

- WLAN
- Pinboard

KURSinHALT

- Standard-Datentypen: Integer, Float, String, Listen, Dictionary
- Zuweisungen/Variablen, Globale/lokale Variablen
- Rechenoperatoren, Vergleichsoperatoren, Logische Operatoren
- Kontrollstrukturen: Verzweigungen, Schleifen
- Standard-Funktionen und -Methoden, eigene Funktionen,
- Ein-/Ausgaben, Dateien lesen/schreiben, Formatierung
- Nutzung Python-Bibliotheken/Module, eigene Module
- Umgang mit Programmfehlern
- Objektorientierte Programmierung (Klassen, Attribute, Methoden, Objekte/Instanzen)
- Graphische Benutzeroberfläche
- Direkt ausführbare Python-Programme (EXE-Dateien), Konvertierung Notizbücher in .py Dateien

HINWEISE ZUM SKRIPT



Bild von [Susanne Weitzhofer](#) auf [Pixabay](#)

- Das Skript ist eine strukturierte Sammlung von Themen zur Programmierung mit Python.
- Die wesentlichen Themenfelder, wie Module, Verzweigungen, Schleifen, Klassen, etc., werden dargestellt.
- Das Skript ist als Begleitmaterial zum Kurs gedacht und kein Lehrbuch.
- Hinweise zu Lehrbüchern für Python finden sich in der Literaturliste im Anhang.

PINBOARD PYTHON

bit.ly/44696zA

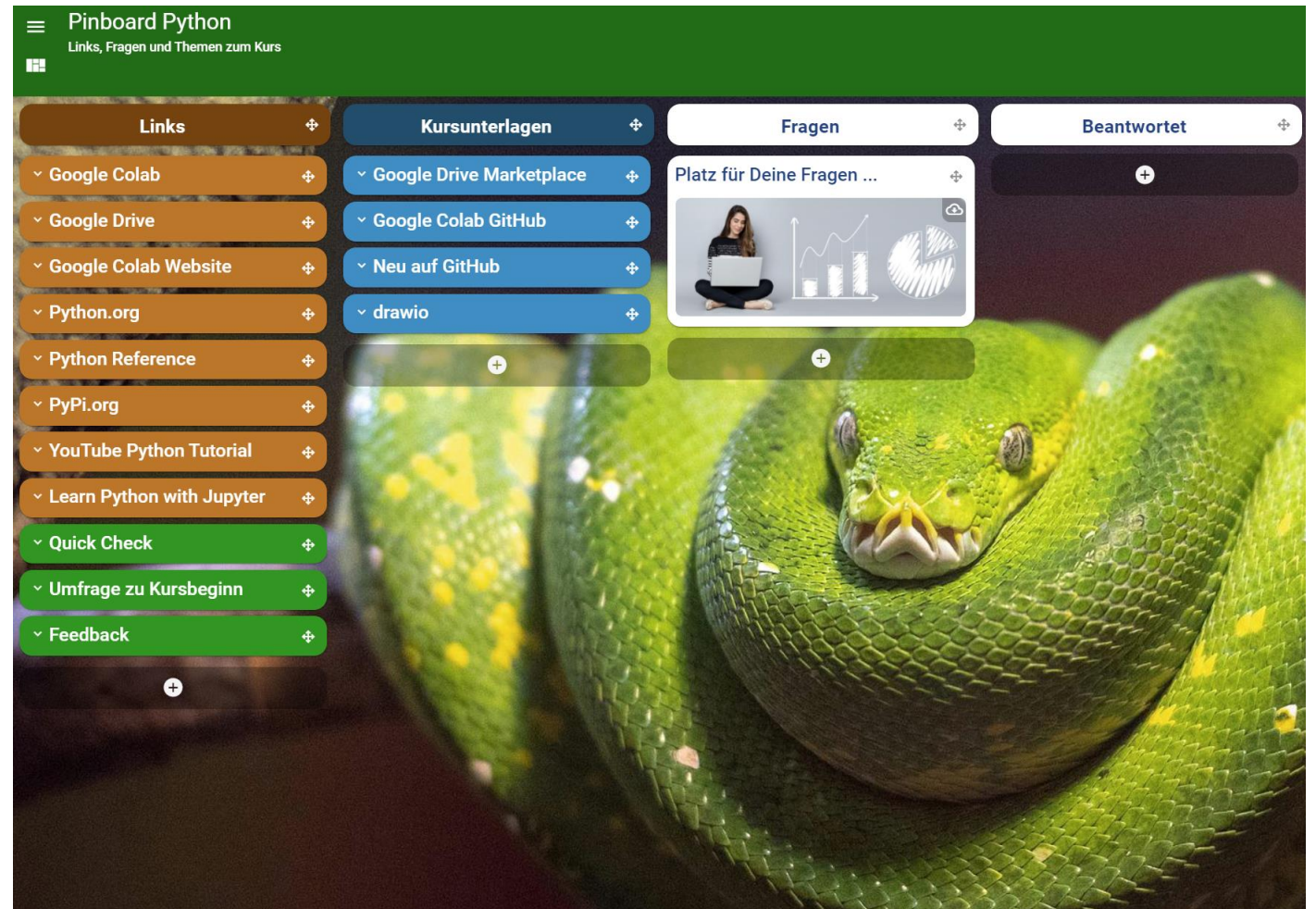


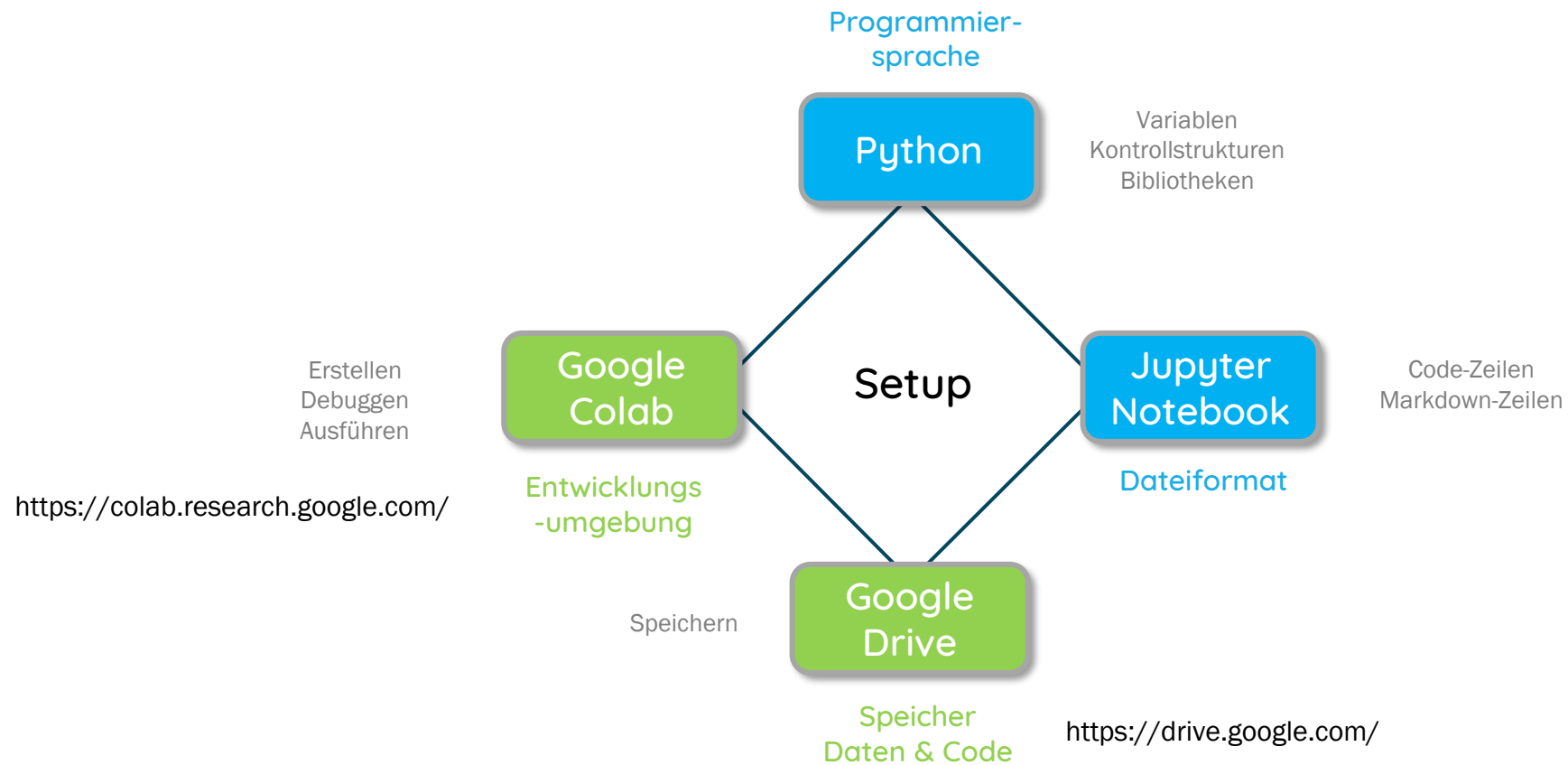
Bild von [Kerri Afford](#) auf [Pixabay](#)



COLAB UND PYTHON



IT-SETUP KURS



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-Module, 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

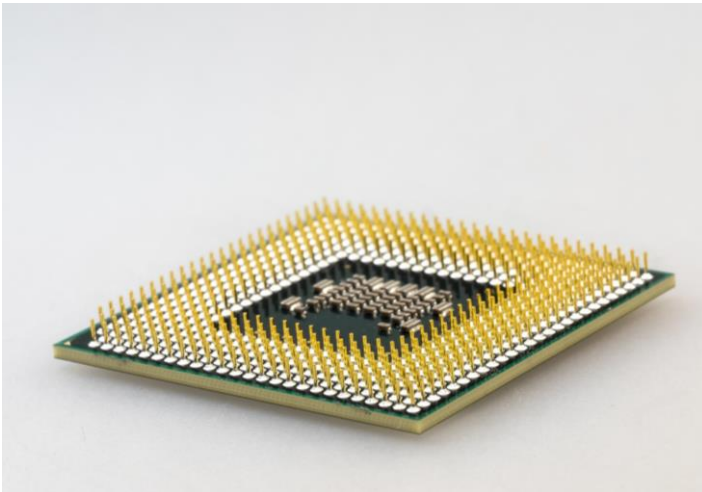


Bild von [Michael Schwarzenberger](#) auf [Pixabay](#)

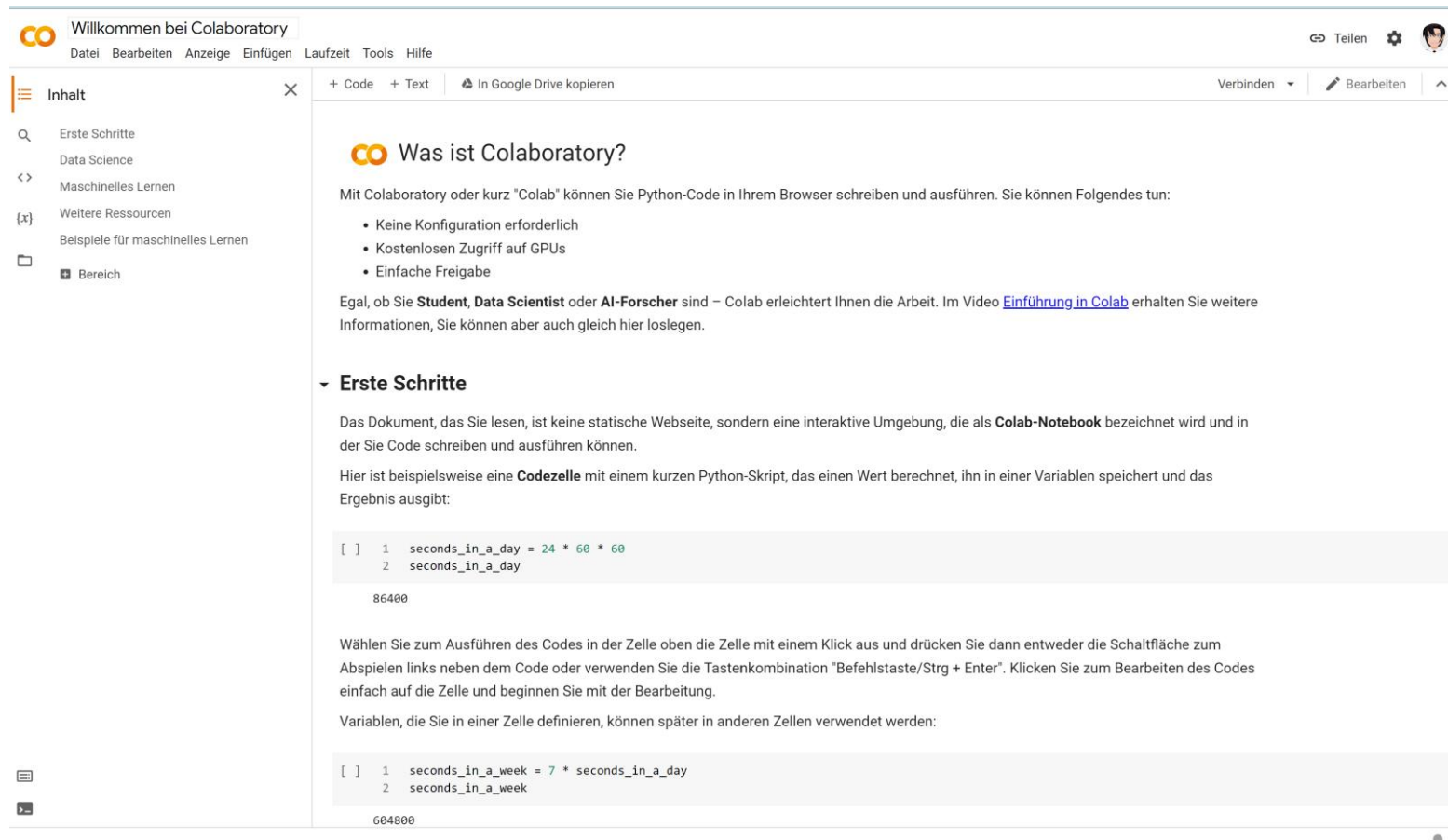
- CPU (Central Processing Unit) ist ein **Allzweckprozessor**, der die meisten Verarbeitungsaufgaben in einem Computer ausführt. Es kann eine Vielzahl von Funktionen verarbeiten, ist jedoch nicht für bestimmte Aufgaben wie Grafikverarbeitung oder Matrizenberechnungen optimiert.
- GPU (Graphics Processing 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

TASTATUREINSTELLUNGEN / SHORTCUTS GOOGLE COLAB

Tastatureinstellungen

Editor-Tastaturbelegungen	
default	
<input checked="" type="checkbox"/> Mit der Eingabetaste werden Vorschläge akzeptiert	
Tastenkombinationen	
Wenn Sie eine Tastenkombination hinzufügen oder ändern möchten, klicken Sie auf die entsprechende Tastenkombination und geben Sie dann die neue Kombination ein. Beachten Sie, dass Ctrl+M als Präfix für Tastenkombinationen mit mehreren Tasten verwendet werden kann.	
Tastenkombination	ipynb herunterladen
Tastenkombination	.py herunterladen
Ctrl+/	Aktuelle Zeile kommentieren
Tastenkombination	Alle Ausgaben löschen
Tastenkombination	Alle Laufzeiten auf Werkseinstellungen zurücksetzen
Ctrl+Shift+A	Alle Zellen auswählen
Ctrl+F9	Alle Zellen in Notebook ausführen
Ctrl+H	Alle suchen/ersetzen
Ctrl+I	Alle/ausgewählte Abschnitte maximieren
Ctrl+J	Alle/ausgewählte Abschnitte minimieren
Ctrl+M -	An Cursorposition teilen
Tastenkombination	Ansicht mit einem Tab ansehen
Tastenkombination	Auf die zuletzt ausgeführte Zelle fokussieren
Ctrl+M I	Ausführung unterbrechen
Ctrl+M O	Ausgabe einblenden/ausblenden
Tastenkombination	Ausgabevollbild anzeigen
Ctrl+M	Laufzeit neu starten
Tastenkombination	Laufzeit neu starten und alle Zellen im Notebook ausführen
Tastenkombination	Laufzeitprotokolle ansehen
Tastenkombination	Layout des minimierten Abschnitts speichern
Ctrl+M Z oder Ctrl+Shift+Z	Letzte Zellaktion rückgängig machen
Tastenkombination	Linken Bereich in einen Tab verschieben
Ctrl+^	Maximieren/Minimieren des aktuellen Abschnitts aktivieren/deaktivieren
Shift+Click	Mehrere Zellen auswählen
Tastenkombination	Mit einer Laufzeit verbinden
Tastenkombination	Mit einer benutzerdefinierten GCE-VM verbinden
Tastenkombination	Mit lokaler Laufzeit verbinden
Tastenkombination	Neues Notebook erstellen
Ctrl+P	Notebook drucken

Tastatureinstellungen

Tastenkombination	Ausgewählte Ausgaben löschen
Ctrl+M K	Ausgewählte Zellen nach oben verschieben
Ctrl+M J	Ausgewählte Zellen nach unten verschieben
Ctrl+F10	Ausgewählte Zellen und alle folgenden Zellen ausführen
Tastenkombination	Ausgewählte Zellen zusammenführen
Ctrl+Shift+Enter	Auswahl ausführen
Shift+Down	Auswahl erweitern, um nächste Zelle einzuschließen
Shift+Up	Auswahl erweitern, um vorherige Zelle einzuschließen
Tastenkombination	Automatische Ausführung der ersten Zelle oder des ersten Abschnitts bei jeder Ausführung aktivieren
Ctrl+Space oder Tab	Automatische Vervollständigung
Ctrl+Shift+P	Befehlspalette anzeigen
Tastenkombination	Code ein-/ausblenden
Ctrl+Alt+P	Code-Snippets-Bereich anzeigen
Tab	Code-docstring-Hilfe aktivieren/deaktivieren
Tastenkombination	Codezelle hinzufügen
Ctrl+M A	Codezelle oben einfügen
Ctrl+M B	Codezelle unten einfügen
Tastenkombination	Colab Enterprise öffnen
Tastenkombination	Notebook freigeben
Tastenkombination	Notebook hochladen
Tastenkombination	Notebook in Drive verschieben
Tastenkombination	Notebook in Google Drive markieren/Markierung aufheben
Tastenkombination	Notebook in Papierkorb verschieben
Ctrl+S	Notebook speichern
Ctrl+O	Notebook öffnen
Tastenkombination	Notebook-Einstellungen öffnen
Tastenkombination	Notebook-Quelle anzeigen
Tastenkombination	Notebook-Vergleich
Ctrl+M N	Nächste Zelle
Ctrl+Shift+]	Nächsten Tab hervorheben
Ctrl+G	Nächstes Element suchen
Tastenkombination	Ressourcen ansehen
Ctrl+Alt+N	Scratchpad-Codezelle öffnen
Tastenkombination	Seitenleiste für Kommentare
Tastenkombination	Sichtbarkeit des Headers aktivieren/deaktivieren
Tastenkombination	Sitzungen verwalten
Tastenkombination	Statusleiste einblenden/ausblenden
Tastenkombination	Tab in den nächsten Bereich verschieben
Tastenkombination	Tab in den vorherigen Bereich verschieben
Tastenkombination	Tabs in zwei Spalten anzeigen

Tastatureinstellungen

Tastenkombination	Dateibrowser anzeigen
Tastenkombination	Drive bereitstellen
Tastenkombination	Drive trennen
Tastenkombination	Editor-Einstellungen öffnen
Tastenkombination	Einstellungen öffnen
Shift+Tab	Einzug für aktuelle Zeile entfernen
Tastenkombination	Fokus mit Tabulatortaste verschieben
Tastenkombination	Formular hinzufügen
Ctrl+M F	Formularansicht ändern
Tastenkombination	Formularattribute bearbeiten
Tastenkombination	Formularfeld hinzufügen
Tastenkombination	Geplante Notebooks anzeigen
Ctrl+Enter	Hervorgehobene Zelle ausführen
Ctrl+Shift+S	Hervorgehobene Zelle auswählen
Tastenkombination	Hervorgehobene Zelle mit nächster Zelle zusammenführen
Tastenkombination	Hervorgehobene Zelle mit vorheriger Zelle zusammenführen
Esc	Hervorheben der Zelle aufheben
Tastenkombination	Im Playground-Modus öffnen
Tastenkombination	In Google Drive suchen
Tastenkombination	In Scratchpad-Zelle kopieren
Shift+Ctrl+H	In aktueller Zelle alle ersetzen
Tastenkombination	Informationen zu Notebook-Datei anzeigen
Tastenkombination	Inhaltsverzeichnis anzeigen
Tastenkombination	Tabs in zwei Zeilen anzeigen
Ctrl+M H	Tastenkombinationen anzeigen
Tastenkombination	Terminal anzeigen
Tastenkombination	Textzelle hinzufügen
Tastenkombination	Variablenübersicht anzeigen
Tastenkombination	Verbindung mit gehosteter Laufzeit herstellen
Tastenkombination	Verbindung trennen und Laufzeit löschen
Tastenkombination	Verlauf der Codeausführung anzeigen
Ctrl+M P	Vorherige Zelle
Ctrl+Shift+[Vorherigen Tab hervorheben
Ctrl+Shift+G	Vorheriges Element suchen
Tastenkombination	Wieder verbinden
Ctrl+M L	Zellennummern aktivieren/deaktivieren
Ctrl+Shift+Y	Zellaktion wiederholen
Ctrl+Click	Zellauswahl aktivieren/deaktivieren
Alt+Enter	Zelle ausführen und neue Zelle einfügen
Shift+Enter	Zelle ausführen und nächste Zelle auswählen
Tastenkombination	Zelle in Tab spiegeln
Tastenkombination	Zelle mit Abschnittsüberschrift hinzufügen
Tastenkombination	Zelle oder Auswahl ausschneiden

Tastatureinstellungen

Tastenkombination	JSON-Notebook ansehen
Ctrl+Alt+M	Kommentar hinzufügen
Tastenkombination	Kommentarbereich öffnen
Tastenkombination	Kopie als GitHub Gist speichern
Tastenkombination	Kopie in Drive speichern
Tastenkombination	Kopie in GitHub speichern
Tastenkombination	Zeile oder Auswahl kopieren
Ctrl+M D	Zelle/Auswahl löschen
Ctrl+F8	Zellen vor der aktuellen Zelle ausführen
Ctrl+M Y	Zu Codezelle konvertieren
Ctrl+M M	Zu Textzelle konvertieren
Tastenkombination	Zu einer konkreten Zelle
Ctrl+M S	Überarbeitung speichern und anpinnen
Tastenkombination	Überarbeitungsverlauf anzeigen

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.

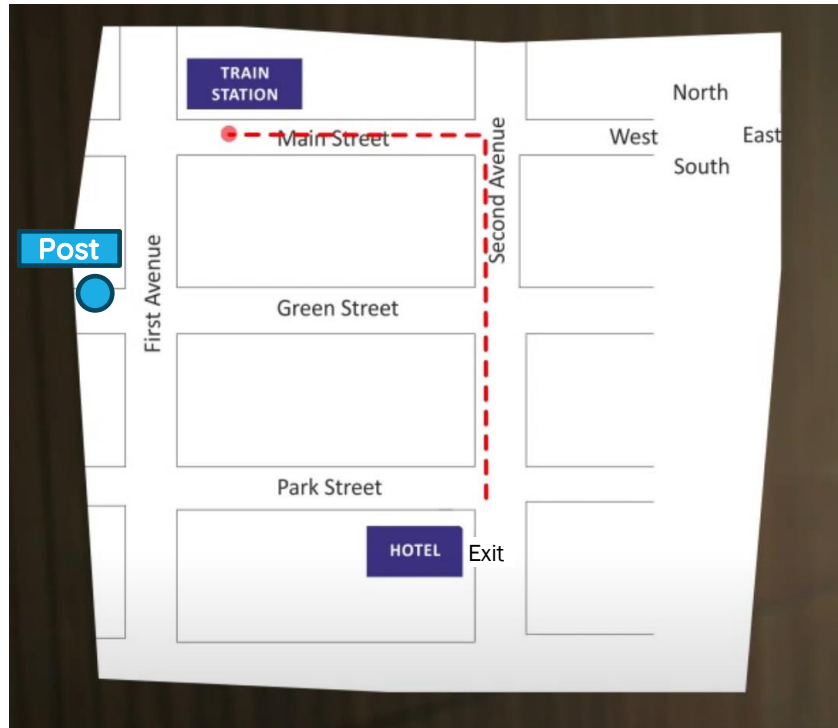
Quelle: Abgefragt 29.08.2021
[Allgemeine Python-FAQ – Python 3.9.7-Dokumentation](#)



GRUNDLAGEN PYTHON



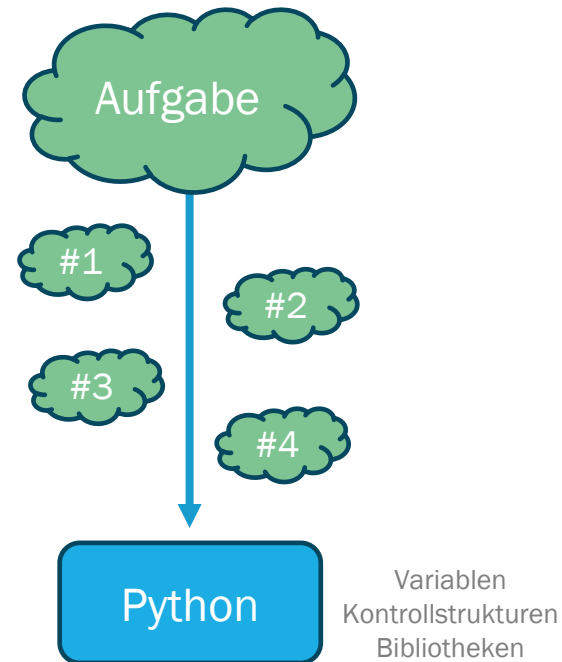
EINFACHE ABLÄUFE / SEQUENZEN



- 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](#)

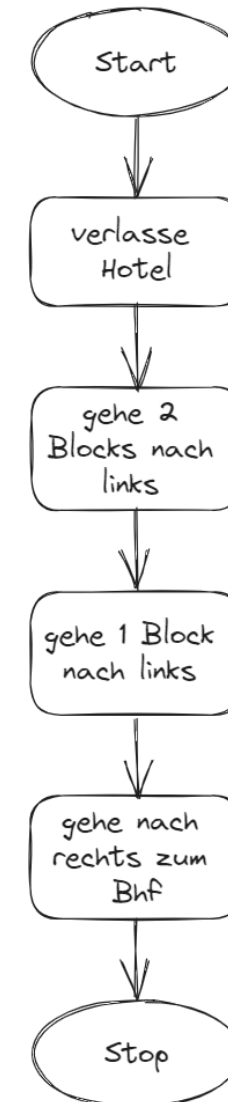
DER 1. SCHRITT: PROBLEMZERLEGUNG



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



PROGRAMMABLAUFPLAN

Bananen-Pfannkuchen

Zutaten:

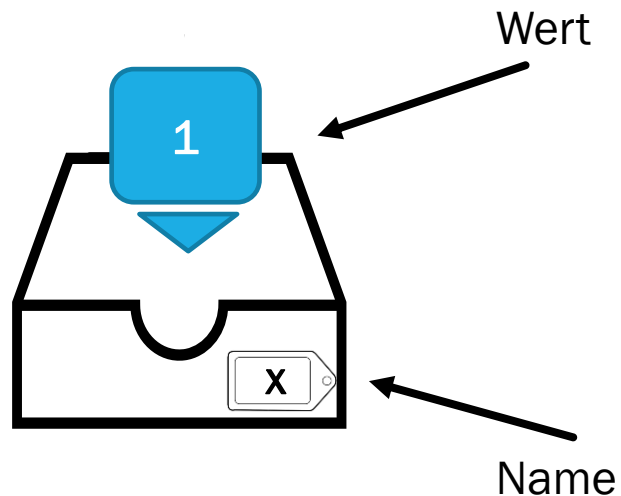
- 2 reife Bananen
- 2 Eier
- Ein wenig Butter oder Öl zum Braten

Zubereitung:

1. Die Bananen in einer Schüssel gut zerdrücken, bis eine breiige Masse entsteht.
2. Die Eier hinzufügen und alles gut vermischen, bis ein glatter Teig entsteht.
3. Eine Pfanne auf mittlerer Hitze erhitzen und ein wenig Butter oder Öl hinzufügen.
4. Kleine Mengen des Teigs in die Pfanne geben und 2-3 Minuten von jeder Seite braten, bis die Pfannkuchen goldbraun sind.
5. Die Pfannkuchen aus der Pfanne nehmen und heiß servieren.



ZUWEISUNGEN - VARIABLEN



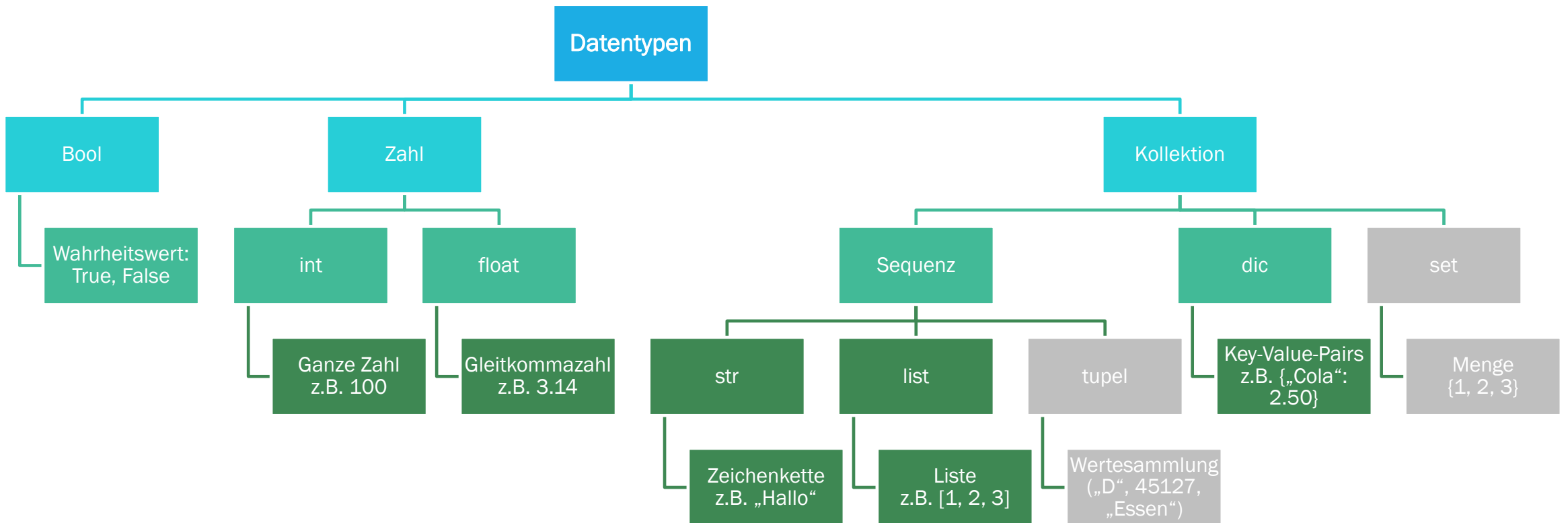
- Zuweisungen sind die häufigsten Befehle in einem Programm.
- Die einfachste Form besteht aus einem Namen gefolgt von einem Wert

Name ← Wert

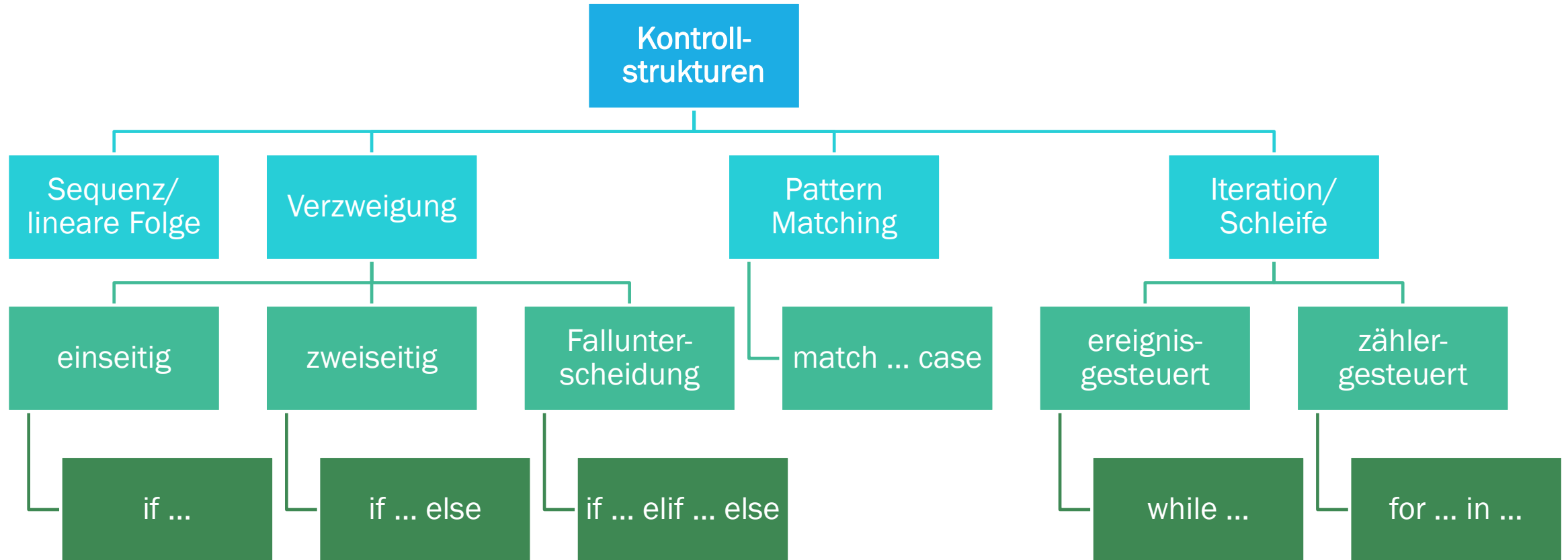
Beispiel: x ← 1

- 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.

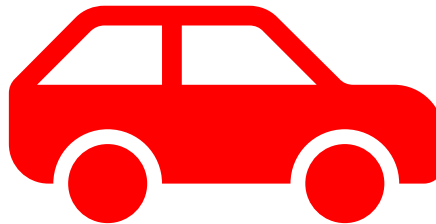
GRUNDBEGRIFFE OOP

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

Klasse: Auto

Methoden:

- Fahren
- Parken
- Tanken
- ...



Attribute:

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

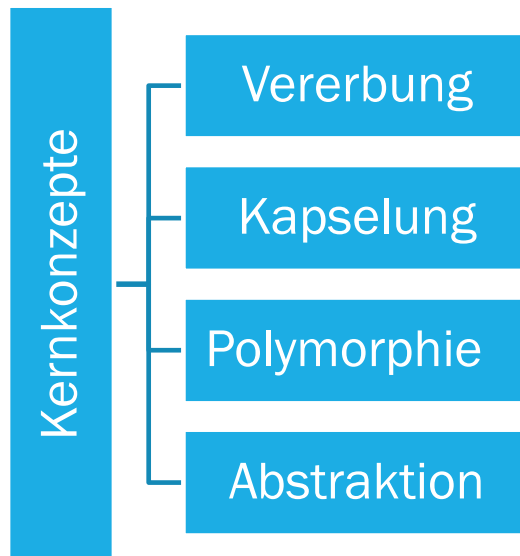
Methode – sind Fähigkeiten der Objekte. (~Funktion)

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



- **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.
- **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.

VORTEILE DER OBJEKTORIENTIERTE PROGRAMMIERUNG

- **Modularität:** Da der Code in Form von Objekten und Klassen organisiert ist, können Teile des Codes leicht wiederverwendet oder durch neue Implementierungen ersetzt werden, ohne andere Teile des Systems zu beeinflussen.
- **Wiederverwendbarkeit:** Klassen können in verschiedenen Projekten wiederverwendet werden, wodurch Entwicklungszeit gespart wird.
- **Erweiterbarkeit:** Durch Vererbung können neue Klassen leicht hinzugefügt werden, ohne den bestehenden Code zu beeinflussen.
- **Wartbarkeit:** Der kapselte und modulare Code ist oft einfacher zu lesen, zu verstehen und zu warten.

OOP ist jedoch nicht für alle Anwendungen oder Probleme geeignet. In manchen Fällen können andere Paradigmen, wie die funktionale Programmierung oder prozedurale Programmierung, besser geeignet sein.

__INIT__



```
1 class Auto:
2     def __init__(self, hersteller, modell, leistung):
3         self.hersteller = hersteller
4         self.modell = modell
5         self.leistung = leistung
6
7     def __str__(self):
8         return f"{self.hersteller}, {self.modell}, {self.leistung}"
```

- `__init__` eine spezielle Methode in objektorientierten Programmiersprachen, die automatisch aufgerufen wird, wenn ein Objekt/Instanz einer Klasse erstellt wird.
- Der Hauptzweck besteht darin, die Anfangswerte für die Attribute des Objekts festzulegen und jegliche erforderliche Initialisierungsaktivitäten durchzuführen.
- Es liefert damit den Bauplan, nach dem das Objekt bzw. die Instanz erstellt wird.

__STR__



```
1 class Auto:
2     def __init__(self, hersteller, modell, leistung):
3         self.hersteller = hersteller
4         self.modell = modell
5         self.leistung = leistung
6
7     def __str__(self):
8         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



```
1 class Auto:
2     def __init__(self, hersteller, modell, leistung):
3         self.hersteller = hersteller
4         self.modell = modell
5         self.leistung = leistung
6
7     def __str__(self):
8         return f"{self.hersteller}, {self.modell}, {self.leistung}"
```

- **self** ist ein Verweis auf das aktuelle Objekt/Instanz und wird verwendet, um auf Attribute oder Methoden dieses Objekts zuzugreifen.
- Es ist das erste Argument, das jeder Methode in einer Klasse in Python übergeben wird, einschließlich `__init__`.
- Es ermöglicht, zwischen Klassenattributen (die für alle Objekte der Klasse gemeinsam sind) und Instanzattributen (die für jedes Objekt einzigartig sind) zu unterscheiden.
- Ganz einfach: Mit der Klasse, die oben definiert wird, wird der allgemeine Bauplan für ein Objekt vorgegeben. Von dieser Klasse können wir aber beliebig viele Objekte/Instanzen erzeugen.
- Beim Aufruf der init-Methode wird die Referenz auf das aktuelle Objekt/Instanz 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:** Unabhängige Codeblöcke, die eine bestimmte Aufgabe ausführen, Parameter akzeptieren und optional einen Wert zurückgeben können.
- **Methoden:** Funktionen, die innerhalb einer Klasse definiert sind. Sie können, genau wie Funktionen, Werte zurückgeben und den Zustand eines Objekts verändern oder einfach Informationen darüber abrufen.

Beispiel:

```
[ ] 1 liste = ["apple", "banana", "kiwi", "grapefruit"]
```

```
[ ] 1 # Funktion
    2 liste_sortiert = sorted(liste, key=len)
    3 liste, liste_sortiert
```

```
(['apple', 'banana', 'kiwi', 'grapefruit'],
 ['kiwi', 'apple', 'banana', 'grapefruit'])
```

```
[ ] 1 # Methode
    2 liste.sort(key=len)
    3 liste
```

```
['kiwi', 'apple', 'banana', 'grapefruit']
```

VERGLEICH FUNKTION VS METHODE

Eigenschaften	Funktion	Methode
Definition	Ein unabhängiger Codeblock, der eine spezifische Aktion ausführt.	Eine Funktion, die zu einer Klasse gehört und auf deren Attribute zugreifen kann.
Aufruf	Unabhängig von einem Objekt.	Immer auf einem Objekt oder einer Klasse.
Erster Parameter	Keine festgelegten Konventionen.	Für Instanzmethoden ist es das Objekt selbst (self).
Definitionsart	Kann überall im Code definiert werden.	Wird innerhalb einer Klasse definiert.
Zugriff auf Objekt-Attribute	Hat keinen impliziten Zugriff auf das umgebende Objekt.	Hat Zugriff auf das Objekt und seine Attribute.



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 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.

INSTALL & IMPORT

Web



Install



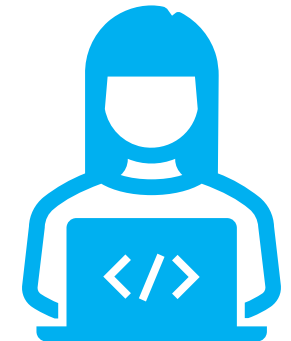
Entwicklungs-
umgebung



Import



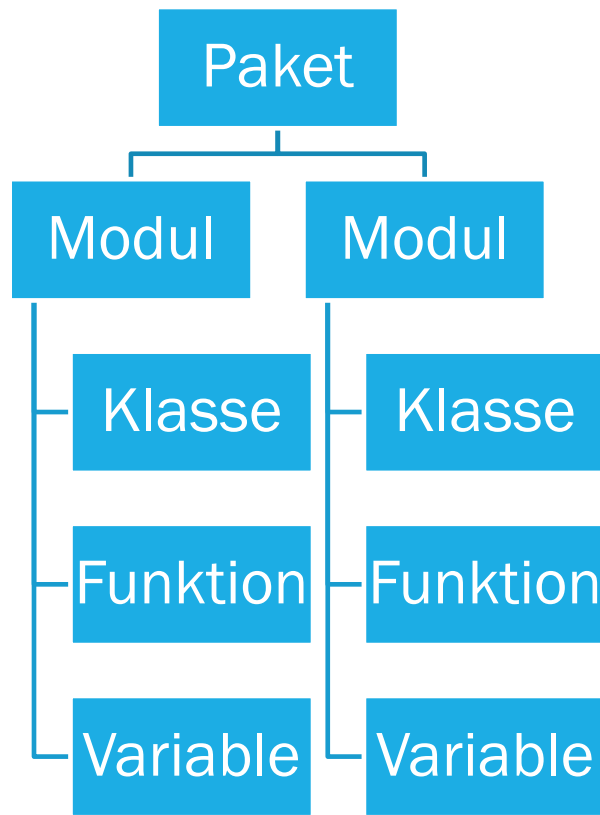
Programm



notwendig, wenn das Paket/Modul
noch nicht in der Entwicklungs-
umgebung, z.B. Colab, vorhanden ist
(!pip install modul)

notwendig, wenn das Modul im
Programm verwendet wird
(import modul)

PAKETE & MODULE



Verzeichnis

datei.py

Python-Code

Python-Code

Python-Code

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/>



CODING & KI – JUST GOOD FRIENDS



WAS IST KI – BEISPIEL CHATGPT?

- ChatGPT ist ein von OpenAI entwickeltes künstliches Intelligenzmodell.
- Es basiert auf der GPT-4 Architektur (Generative Pre-trained Transformer).
- Dieses Modell wurde mit Milliarden von Wörtern aus dem Internet trainiert und kann Texte generieren, Fragen beantworten und viele andere Aufgaben durchführen, die mit Sprache zu tun haben.
- Es nutzt tiefes Lernen und speziell entworfene neuronale Netze, um menschenähnliche Textantworten zu produzieren.
- ChatGPT kann in verschiedenen Anwendungen wie Chatbots, Textgenerierung und Informationsabruf verwendet werden.
- Es ist jedoch nur so gut wie die Daten, mit denen es trainiert wurde, und kann keine Emotionen oder Bewusstsein besitzen.
- Es ist ein fortschrittliches Werkzeug für Textverarbeitung und -generierung.

SUPPORT VON CHATGPT BEIM CODING (1/2)



1. **Code-Schnipsel:** Sie können KI bitten, Ihnen Beispiele oder kleine Code-Schnipsel zu geben, die bestimmte Funktionalitäten demonstrieren.
2. **Code-Verständnis:** Wenn Sie Schwierigkeiten haben, einen bestimmten Code-Schnipsel oder eine Funktion zu verstehen, können Sie KI um Erklärungen bitten.
3. **Fehlerbehebung:** Haben Sie einen Fehler in Ihrem Code, den Sie nicht beheben können? Sie können den Fehler und den relevanten Code an KI weitergeben, um Hinweise oder Lösungsvorschläge zu erhalten.
4. **Best Practices:** Fragen Sie nach bewährten Methoden oder üblichen Ansätzen für bestimmte Programmieraufgaben.
5. **Algorithmus-Erklärungen:** Wenn Sie einen bestimmten Algorithmus oder eine Datenstruktur nicht verstehen, kann KI Erklärungen oder Pseudocode bereitstellen.
6. **Tools und Libraries:** Fragen Sie nach Empfehlungen für Tools, Libraries oder Frameworks, die für Ihre Aufgabe geeignet sind.
7. **Lernressourcen:** Erhalten Sie Empfehlungen für Bücher, Online-Kurse oder Tutorials zu spezifischen Programmierthemen. KI kann auch direkt als interaktives Tutorial genutzt werden.
8. **Design-Muster:** Erhalten Sie Erklärungen und Beispiele zu verschiedenen Design-Mustern.

SUPPORT VON CHATGPT BEIM CODING (2/2)



9. **Sprachspezifische Fragen:** Fragen zu den Besonderheiten oder Syntax bestimmter Programmiersprachen.
10. **Optimierung:** Erhalten Sie Tipps zur Optimierung von Code in Bezug auf Geschwindigkeit, Speicherplatz oder andere Faktoren.
11. **Sicherheitsberatung:** Fragen Sie nach Best Practices, um Sicherheitslücken in Ihrem Code zu vermeiden.
12. **Integration und APIs:** Erhalten Sie Hilfe bei der Integration von Drittanbieter-APIs oder anderen externen Diensten in Ihr Projekt.
13. **Dokumentation:** Fragen Sie nach einer Kommentierung von Code-Snippets und einzelnen Codezeilen.
14. **Code-Interpretation:** Fordern Sie das Ausführen von Code-Snippets direkt in KI.
15. **Pair-Programming:** Legen Sie die Rolle von KI als „Pilot“ oder „Navigator“ fest.
16. **Code-Übersetzung:** Erfragen Sie die Neu-Codierung eines vorhandenen Codes in eine andere Sprache.

to be continued!

CODE-ÜBERSETZUNG



Die Fibonacci-Zahlen sind eine Zahlenfolge, bei der jede Zahl die Summe der zwei vorhergehenden Zahlen ist.



Python

```
def fibonacci_iterativ(n):  
    if n ≤ 1:  
        return n  
    a, b = 0, 1  
    for _ in range(2, n+1):  
        a, b = b, a+b  
    return b
```

JavaScript

```
function fibonacci(n) {  
    if (n ≤ 1) {  
        return n;  
    }  
  
    let a = 0;  
    let b = 1;  
    let temp;  
  
    for (let i = 2; i ≤ n; i++) {  
        temp = a + b;  
        a = b;  
        b = temp;  
    }  
  
    return b;  
}
```

FORTRAN

```
FUNCTION fibonacci(n) RESULT(res)  
    INTEGER, INTENT(IN) :: n  
    INTEGER :: res  
    INTEGER :: a, b, i  
  
    IF (n ≤ 1) THEN  
        res = n  
        RETURN  
    END IF  
  
    a = 0  
    b = 1  
    DO i = 2, n  
        res = a + b  
        a = b  
        b = res  
    END DO  
END FUNCTION fibonacci
```



INTEGRIERTE ENTWICKLUNGSUMGEBUNG



INTEGRIERTE ENTWICKLUNGSUMGEBUNG (IDE - INTEGRATED DEVELOPMENT ENVIRONMENT)

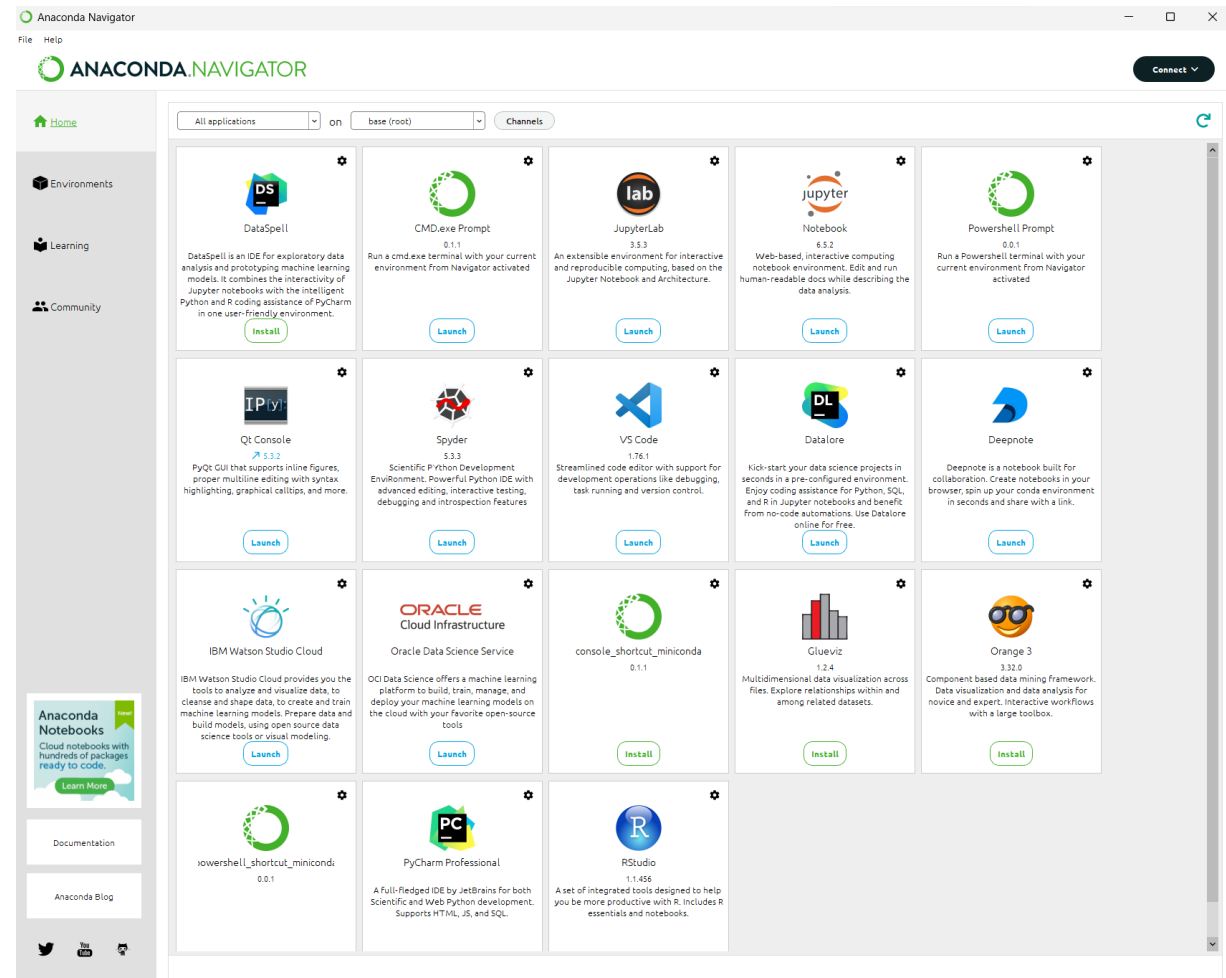
- Eine integrierte Entwicklungsumgebung ist eine Sammlung von Apps/Funktionen, mit denen die Aufgaben der Softwareentwicklung möglichst ohne Medienbrüche bearbeitet werden können.
- IDEs stellen hilfreiche Werkzeuge bereit, die Softwareentwicklern häufig wiederkehrende Aufgaben unterstützt, z.B. Arbeits(zwischen)ergebnisse 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-IDE'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

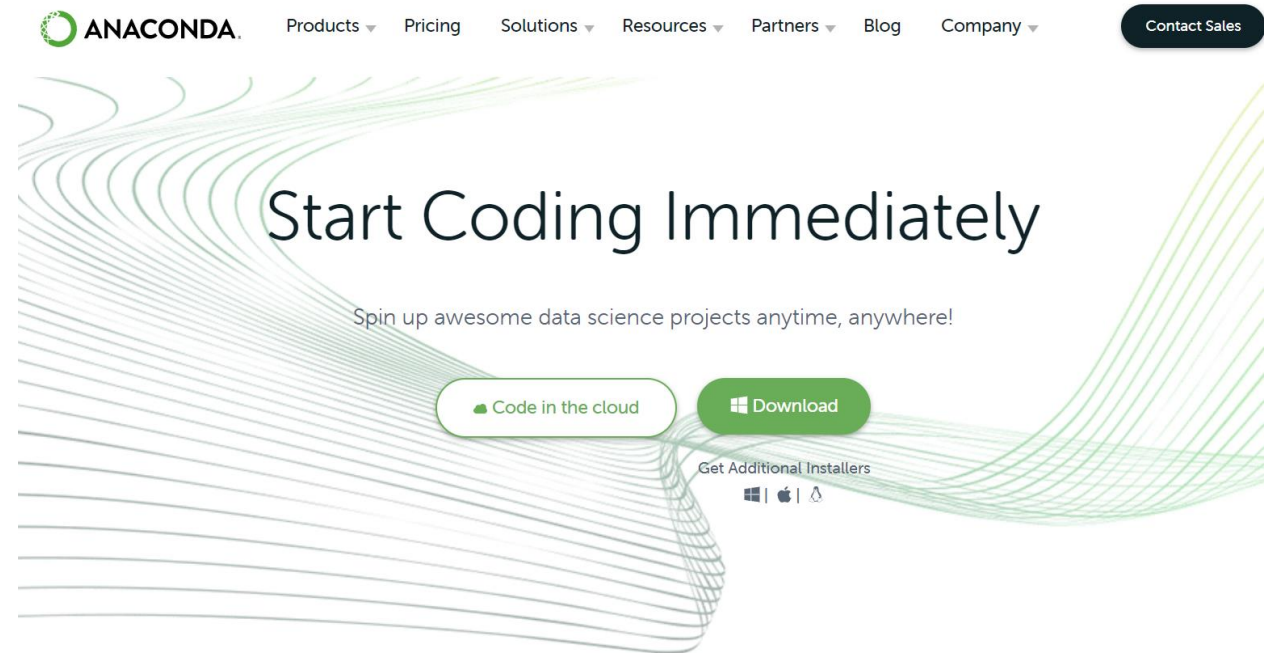
Anaconda ist eine Distribution für die Programmiersprachen Python und R, die unter anderem die Entwicklungsumgebung Visual Studio Code und Jupyter Notebook/Lab enthält.

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



ANACONDA

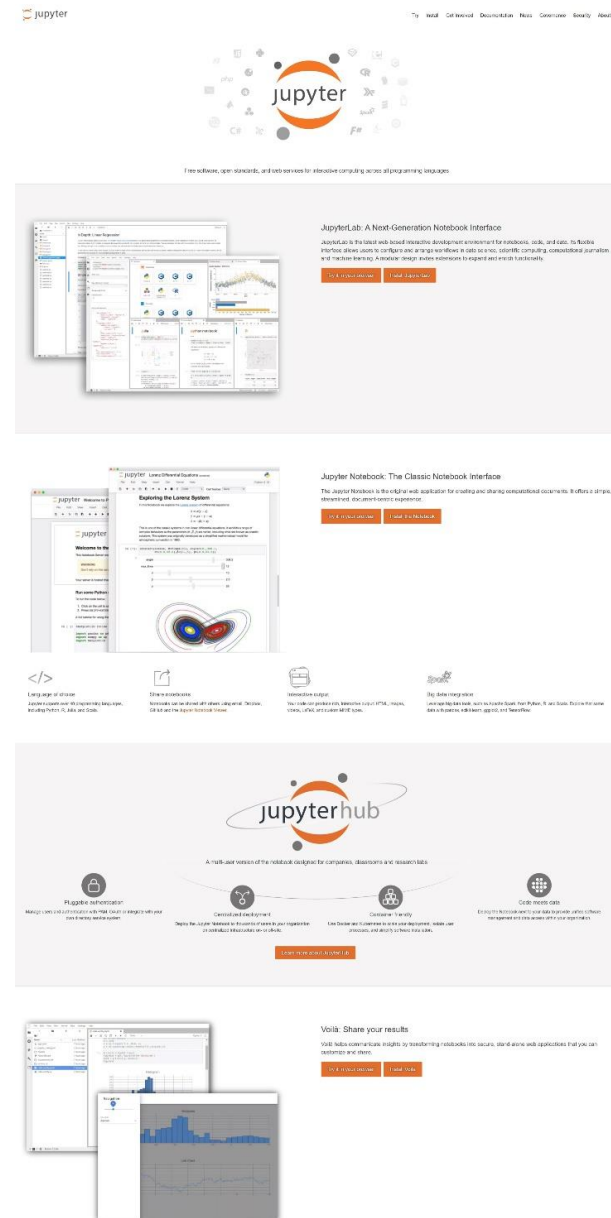
Anaconda bietet neben dem Navigator auch einen in der Cloud gehosteten Notebook-Service mit einer vollständig geladenen und schlüsselfertigen interaktiven Entwicklungsumgebung. Läuft auf jedem Browser, jedem System. 100 % Installations- und Konfigurationsfrei.



PROJEKT JUPYTER

Das Projekt Jupyter ist der Herausgeber von Softwareprodukten für interaktive wissenschaftliche Datenauswertung und wissenschaftliche Berechnungen. Das Projekt Jupyter hat die Produkte Jupyter Notebook, JupyterHub und JupyterLab entwickelt.

[Link](#)



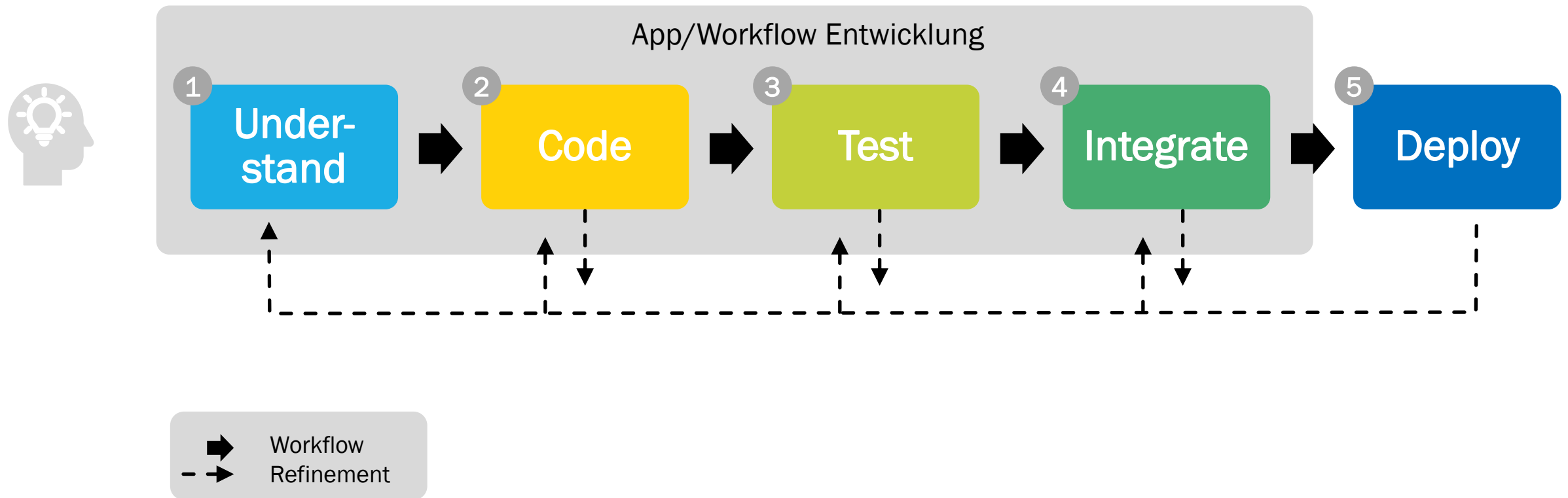
The screenshot shows the Jupyter website homepage. At the top, there's a navigation bar with links like 'Home', 'Get involved', 'Documentation', 'News', 'Governance', 'Security', and 'About'. Below the navigation bar is the Jupyter logo, which consists of a stylized orange circle with the word 'jupyter' in white. Underneath the logo is a tagline: 'Free software, open standards, and web services for interactive computing across all programming languages.' The main content area is divided into three sections. The first section is titled 'JupyterLab: A Next-Generation Notebook Interface' and describes it as 'the latest web-based interactive development environment for notebooks, code, and data'. It features a 'Get it now' button. The second section is titled 'Jupyter Notebook: The Classic Notebook Interface' and describes it as 'the original web application for creating and sharing computational documents'. It also features a 'Get it now' button. The third section is titled 'JupyterHub' and describes it as 'a multi-user version of the notebook designed for companies, classrooms and research labs'. It features a 'Get it now' button. At the bottom, there's a section titled 'Voilà: Share your results' which describes it as 'a web-based interface for transforming notebooks into secure, standalone web applications'. It also features a 'Get it now' button. The website uses a clean, modern design with a light blue and white color scheme.



ENTWICKLUNGSPROZESS



ENTWICKLUNGSPROZESS



RE-USE

Gute
Option

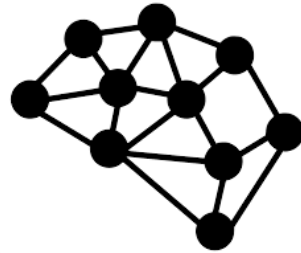


Bild von [Shirley Hirst](#) auf [Pixabay](#)

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

KI ALS SUPPORTER / ASSISTANT

Kann
helfen



- Code-Schnipsel: Sie können KI bitten, Ihnen Beispiele oder kleine Code-Schnipsel zu geben, die bestimmte Funktionalitäten demonstrieren.
- Code-Verständnis: Wenn Sie Schwierigkeiten haben, einen bestimmten Code-Schnipsel oder eine Funktion zu verstehen, können Sie KI um Erklärungen bitten.
- Fehlerbehebung: Haben Sie einen Fehler in Ihrem Code, den Sie nicht beheben können? Sie können den Fehler und den relevanten Code an KI weitergeben, um Hinweise oder Lösungsvorschläge zu erhalten.
- Algorithmus-Erklärungen: Wenn Sie einen bestimmten Algorithmus oder eine Datenstruktur nicht verstehen, kann KI Erklärungen oder Pseudocode bereitstellen.
- Tools und Libraries: Fragen Sie nach Empfehlungen für Tools, Libraries oder Frameworks, die für Ihre Aufgabe geeignet sind.
- Lernressourcen: Erhalten Sie Empfehlungen für Bücher, Online-Kurse oder Tutorials zu spezifischen Programmierthemen. KI kann auch direkt als interaktives Tutorial genutzt werden.
- ...

CODE-SNIPPETS

Gute
Option



Bild von [Hans](#) auf [Pixabay](#)

- Code-Snippets sind vorgefertigte Codefragmente, die in der Programmierung wiederverwendet werden können.
- Sie können in verschiedenen Entwicklungsumgebungen, Texteditoren oder IDEs (Integrated Development Environments) eingesetzt werden, um Entwicklungszeit zu sparen und die Effizienz zu steigern.
- Einsatzmöglichkeiten u.a.:
 - Wiederverwendung von Code: Häufig wiederkehrende Codeblöcke, die in verschiedenen Projekten oder sogar innerhalb desselben Projekts verwendet werden sollen. Das spart Zeit und vermeidet Tippfehler.
 - Schnelle Implementierung: Code-Snippets können helfen, häufig verwendete Funktionen oder Algorithmen schnell zu implementieren.
 - ...

CHECKLISTEN

Gute
Option



Bild von [Gerry](#) auf [Pixabay](#)

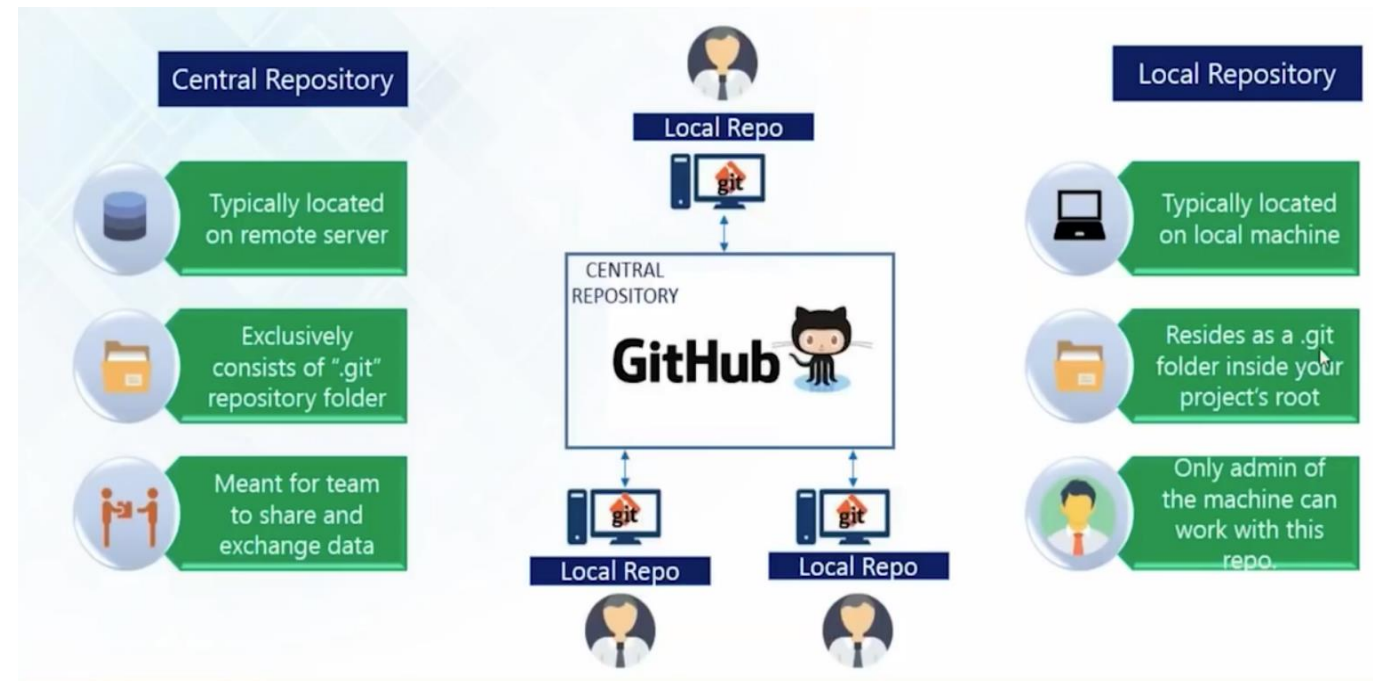
In der Fliegerei werden Checklisten verwendet, um sicherzustellen, dass wichtige Schritte und Verfahren ordnungsgemäß durchgeführt werden und keine wichtigen Details übersehen werden.

- **Sicherheit:** Checklisten helfen dabei, sicherheitsrelevante Aufgaben und Verfahren zu standardisieren und sicherzustellen, dass sie in einer sicheren und konsistenten Art und Weise durchgeführt werden. Dies minimiert das Risiko menschlicher Fehler und reduziert potenzielle Gefahren.
- **Komplexität:** Es gibt eine Vielzahl von Systemen, Verfahren und Protokollen, die beachtet werden müssen. Checklisten helfen den Piloten und dem Flugpersonal dabei, diese Komplexität zu bewältigen, indem sie sicherstellen, dass keine wichtigen Aufgaben oder Überprüfungen vergessen werden.
- **Standardisierung:** Checklisten ermöglichen eine Standardisierung der Abläufe und Verfahren. Dies ist besonders wichtig, wenn mehrere Piloten oder Crewmitglieder zusammenarbeiten, da es sicherstellt, dass alle nach dem gleichen Prozess arbeiten und keine wichtigen Schritte auslassen.
- **Vermeidung von Gedächtnisfehlern:** Durch das Arbeiten mit Checklisten wird das Risiko von Gedächtnisfehlern minimiert. Gerade in Stresssituationen oder unter Zeitdruck kann das Gedächtnis versagen oder wichtige Details vergessen werden. Checklisten bieten eine klare Anleitung und erinnern die Piloten und das Flugpersonal an alle notwendigen Schritte.

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.

 [Git and GitHub Tutorial for Beginners](#)



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

COMPILING UND EXE-UMWANDLUNG

Python EXE Datei erstellen



Dauer: 3:38

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

Compiling & Decompiling Python Scripts

Dauer: 9 Min

[\(49\) Compiling & Decompiling Python Scripts – YouTube](#)

Convert Python To Exe Files

Dauer: 8 Min

[\(49\) Convert Python To Exe Files - YouTube](#)



STYLE GUIDE



BENENNUNGSKONVENTIONEN FÜR VARIABLEN

Konvention	Beschreibung	Beispiel
PascalCase	Jedes Wort beginnt mit einem Großbuchstaben ohne Unterstriche.	MeineVariable
camelCase	Erstes Wort klein geschrieben, jedes folgende Wort groß beginnend.	meineVariable
snake_case	Wörter klein geschrieben, durch Unterstriche getrennt.	meine_variable
UPPER_SNAKE_CASE	Wörter groß geschrieben, durch Unterstriche getrennt.	MEINE_KONSTANTE

10 REGELN ZUR GESTALTUNG VON GUT LESBAREM/WARTBAREM CODE

1. Klare und konsistente Benennung: Verwende aussagekräftige Namen für Variablen, Funktionen und Klassen, die deren Zweck widerspiegeln. Halte dich an eine Namenskonvention (z.B. snake_case für Variablen und Funktionen in Python, PascalCase für Klassen).
2. Vermeidung langer Funktionen und Klassen: Zerlege deinen Code in kleinere, wiederverwendbare Funktionen und Klassen, die jeweils nur eine spezifische Aufgabe erfüllen. Dies macht den Code leichter zu verstehen und zu testen.
3. Kommentare und Dokumentation: Kommentiere deinen Code, wo es notwendig ist, um zu erklären, warum etwas auf eine bestimmte Weise gemacht wird. Verwende Docstrings (""" *Doc-String* """), um Funktionen, Klassen und Module in Python zu dokumentieren.
4. Einhalten von Code-Standards und Style-Guides: Folge den PEP 8-Richtlinien für Python, um Konsistenz im Code-Stil zu gewährleisten. Dies umfasst Empfehlungen zur Formatierung, zum Zeilenabstand und zu anderen Stilfragen.
5. Verwendung von Versionierungstools: Nutze Versionierungstools wie Git, um Änderungen am Code zu verfolgen. Dies erleichtert die Zusammenarbeit und hilft, den Überblick über verschiedene Versionen des Projekts zu behalten.

10 REGELN ZUR GESTALTUNG VON GUT LESBAREM/WARTBAREM CODE

6. Schreiben von Tests: Implementiere Unit-Tests, um die Korrektheit deines Codes zu überprüfen. Tests helfen dabei, Fehler frühzeitig zu entdecken und sicherzustellen, dass Änderungen am Code die bestehende Funktionalität nicht beeinträchtigen.
7. Vermeidung von globalen Variablen: Beschränke die Verwendung globaler Variablen, da sie den Code schwerer zu verstehen und zu debuggen machen können. Verwende stattdessen lokale Variablen oder übergib Variablen an Funktionen.
8. Vermeidung von magischen Zahlen und Strings: Definiere Konstanten für Zahlen und Strings, die eine spezifische Bedeutung haben, anstatt sie direkt in den Code einzufügen. Dies macht den Code lesbarer und erleichtert Änderungen an diesen Werten.
9. Beachtung der Prinzipien der Softwareentwicklung: Prinzipien wie DRY (Don't Repeat Yourself) und KISS (Keep It Simple, Stupid) können dabei helfen, Redundanzen zu vermeiden und die Komplexität des Codes zu minimieren.
10. Regelmäßige Code-Reviews: Lasse deinen Code regelmäßig von anderen überprüfen. Code-Reviews können helfen, Fehler, schlechte Praktiken und Verbesserungsmöglichkeiten zu identifizieren.

UNIT-TEST

```
1 import unittest
2
3 def addiere(a, b):
4     return a + b
5
6 class TestAddition(unittest.TestCase):
7     def test_addiere_positive_zahlen(self):
8         self.assertEqual(addiere(1, 2), 3)
9
10    def test_addiere_negative_zahlen(self):
11        self.assertEqual(addiere(-1, -2), -1)
12
13 if __name__ == '__main__':
14     unittest.main(argv=['first-arg-is-ignored'], exit=False)
```

F.
=====

FAIL: test_addiere_negative_zahlen (__main__.TestAddition)

Traceback (most recent call last):
 File "<ipython-input-6-2f9658d0ed86>", line 11, in test_addiere_negative_zahlen
 self.assertEqual(addiere(-1, -2), -1)
AssertionError: -3 != -1

Ran 2 tests in 0.157s

FAILED (failures=1)

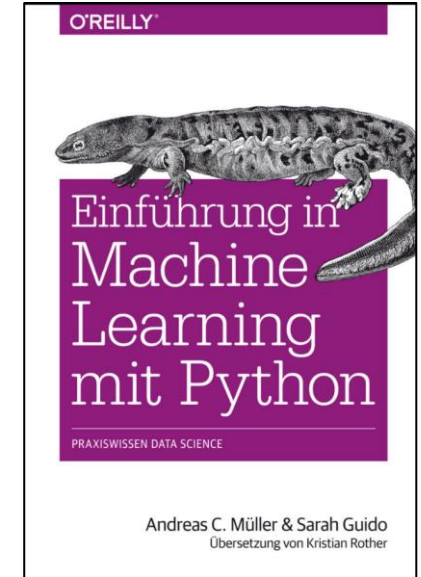
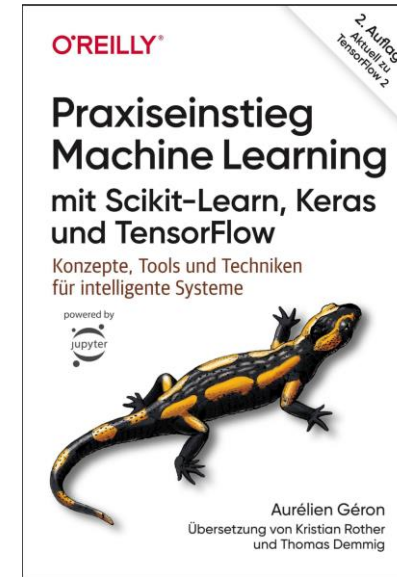
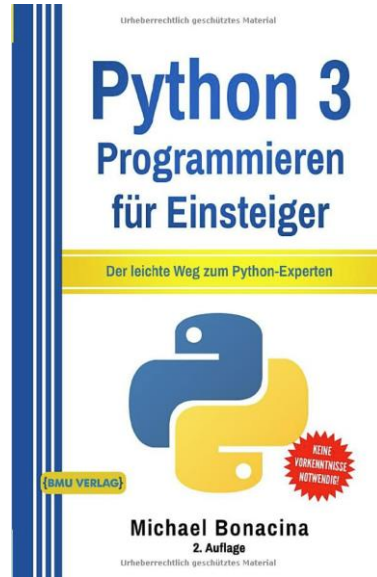
- Die Python-Bibliothek unittest wird verwendet, um Unit-Tests für Python-Programme zu schreiben und auszuführen. Unit-Tests sind Tests, die einzelne
- Komponenten oder Funktionen eines Programms überprüfen, um sicherzustellen, dass sie wie erwartet funktionieren. Unit-Tests helfen, Fehler im
- Code frühzeitig zu finden und zu beheben, die Codequalität zu verbessern und die Softwareentwicklung zu beschleunigen.
- Die unittest-Bibliothek bietet eine Reihe von Funktionen und Klassen, um Unit-Tests zu erstellen, zu organisieren und auszuführen. Zum Beispiel
 - Die Klasse `Testcase` repräsentiert eine einzelne Testeinheit, die eine oder mehrere Testmethoden enthält.
 - Die Klasse `Testsuite` repräsentiert eine Sammlung von Testfällen oder Testsuiten, die zusammen ausgeführt werden sollen.
 - Die Klasse `TestRunner` steuert die Ausführung der Tests und liefert das Ergebnis an den Benutzer zurück.
 - Die Funktion `assert` überprüft, ob eine Bedingung wahr ist, und löst eine Ausnahme aus, wenn sie falsch ist.



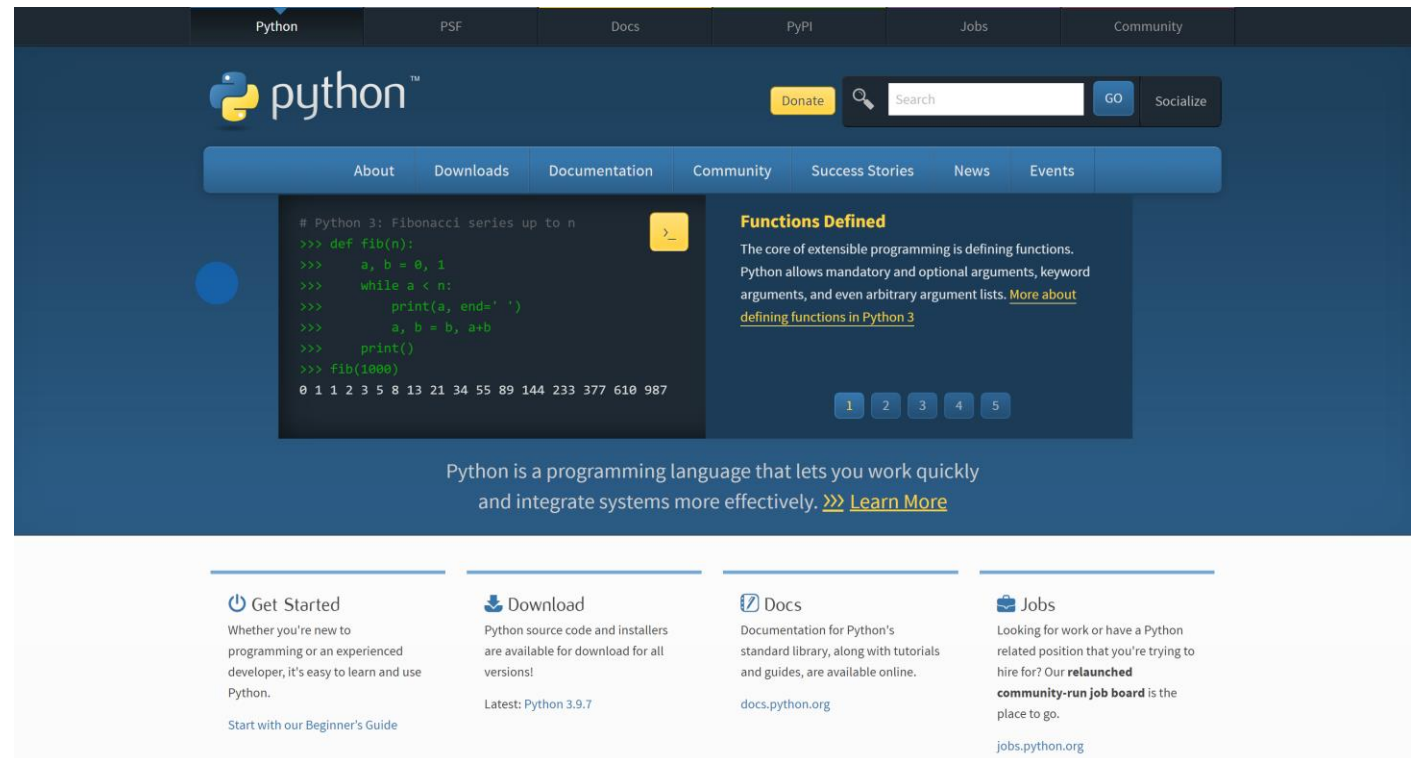
BÜCHER & LINKS



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 displays: '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 a small cube icon and 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
Google Colab	Willkommen bei Colaboratory - Colaboratory (google.com)
	Google Colab: Alles, was Sie wissen müssen - Geekflare
	Google Colab Tutorial for Beginners Get Started with Google Colab
	Google Colab Introduction. Colab Tutorial. Colab for Beginners. Colab Explained.
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
- 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)
Git/GitHub	(3) Git And GitHub in ~30 Minutes - YouTube
JupyterLab	
Debugger	(6) Visual Debugger for Jupyter Lab/IPython Notebooks Installation, Code Examples & Debugging - YouTube