Kapitel 1: Software, Tools und Programmiersprache

Einige dieser Aspekte haben wir bereits gemeinsam eingerichtet.

Lest euch die folgenden Inhalte dennoch noch einmal in Ruhe durch. Vor allem für den Fall, dass ihr zu Hause etwas für den Kurs erledigen oder programmieren möchtet, zum Beispiel in Vorbereitung auf die Leistungskontrollen, sind diese Hinweise hilfreich, damit ihr euren PC daheim entsprechend einrichten könnt. In Absprache mit einem Informatiklehrer ist es natürlich auch immer möglich, das Computerkabinett der Schule zu benutzen.

1.1. Programmierumgebung

Als **Programmierumgebung** (auch: IDE, aus dem Englischen: integrated development environment) nutzen wir **Visual Studio Code**. Dieser Editor steht für die meisten Betriebssysteme zur Verfügung. Heruntergeladen werden kann dieser Editor von folgender Website:

https://code.visualstudio.com/

Visual Studio Code kann - mit ein paar Einschränkungen im Funktionsumfang - auch im **Browser** genutzt werden, beispielsweise für den Fall, dass ihr zu Hause nur mit einem Tablet arbeiten wollt oder keine Programme installieren könnt.

https://vscode.dev/

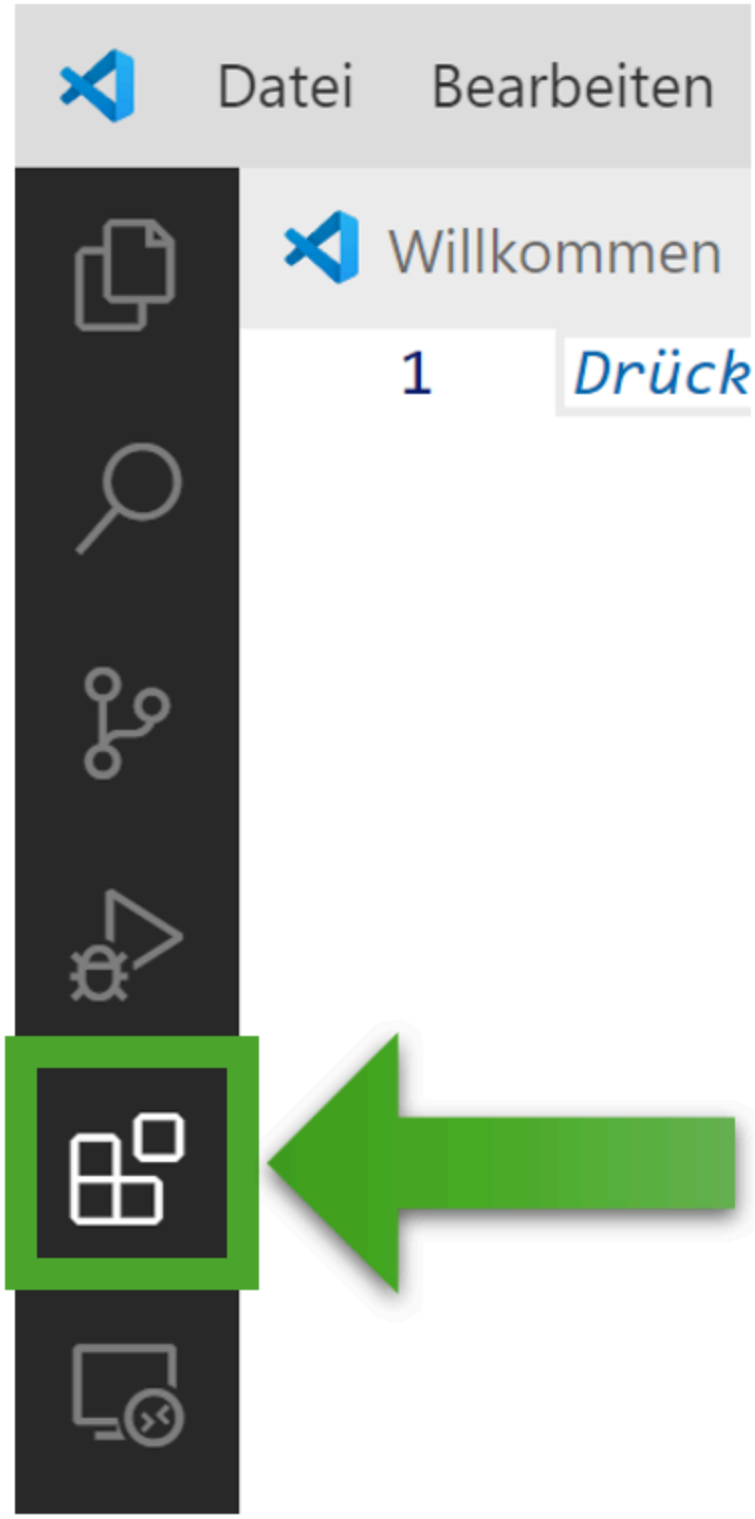
Die Software ist **kostenfrei** und **OpenSource**.

1.2. Plugins in Visual Studio Code

Visual Studio Code ist in der Basis-Variante nur ein **Texteditor** auf Steroide. Mit Plugins, kleinen Erweiterungen, die das Programm um diverse Funktionalitäten erweitern, können wir den Texteditor noch mächtiger machen. Für die Bearbeitung des Kurses benötigen wir **drei Plugins**.

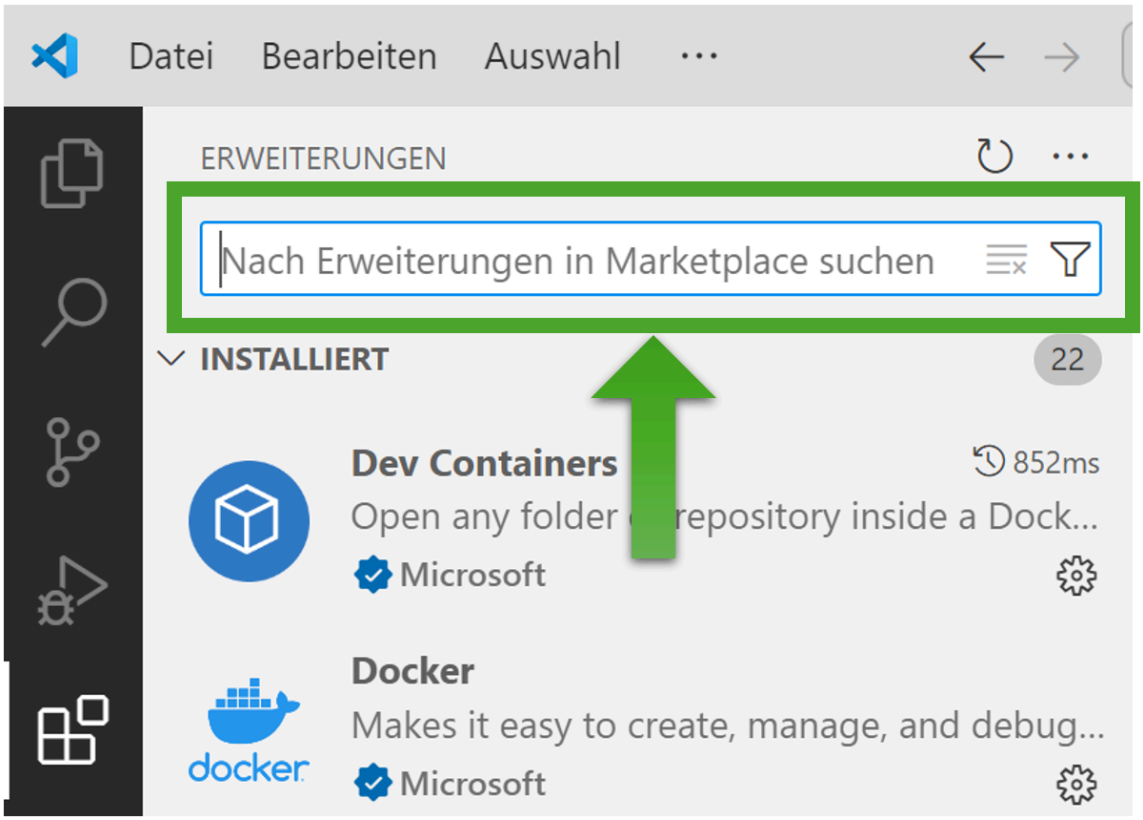
Installation von Plugins

Plugins können ganz einfach installiert werden: Klickt dazu in der linken Menüleiste auf das Icon für die Plugins. Das ist auf dem nachfolgenden Bild markiert.



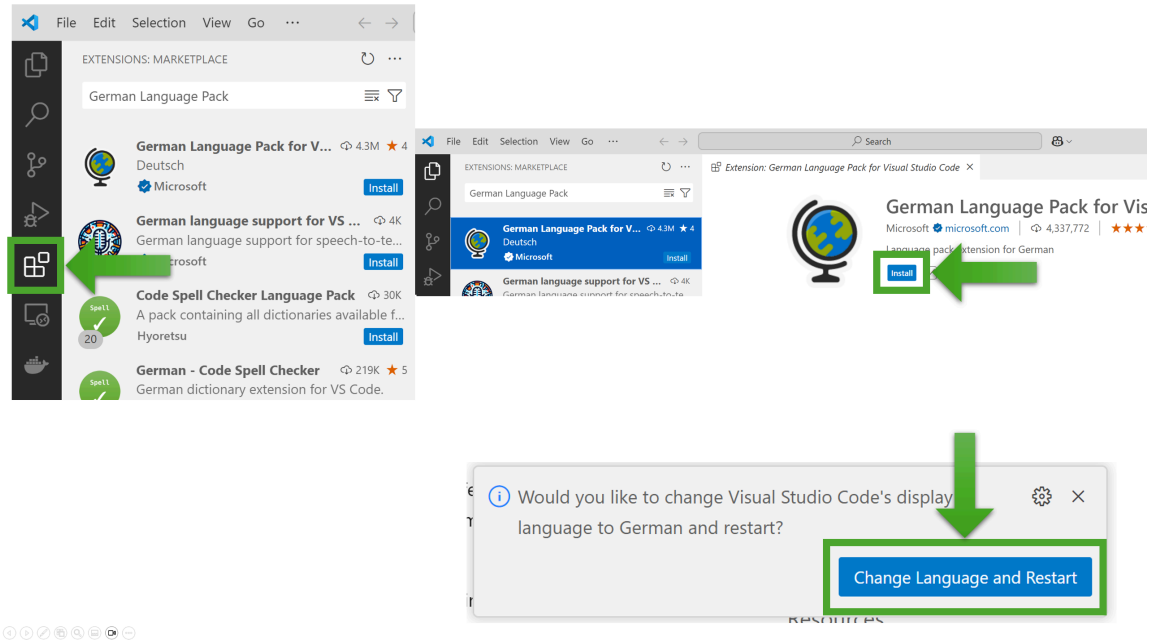
In der Suchleiste könnt ihr das Plugin anhand des Namens suchen. Habt man das Richtige gefunden, klickt man nur noch auf **Installieren** und wartet den Installationsprozess

vollständig ab.



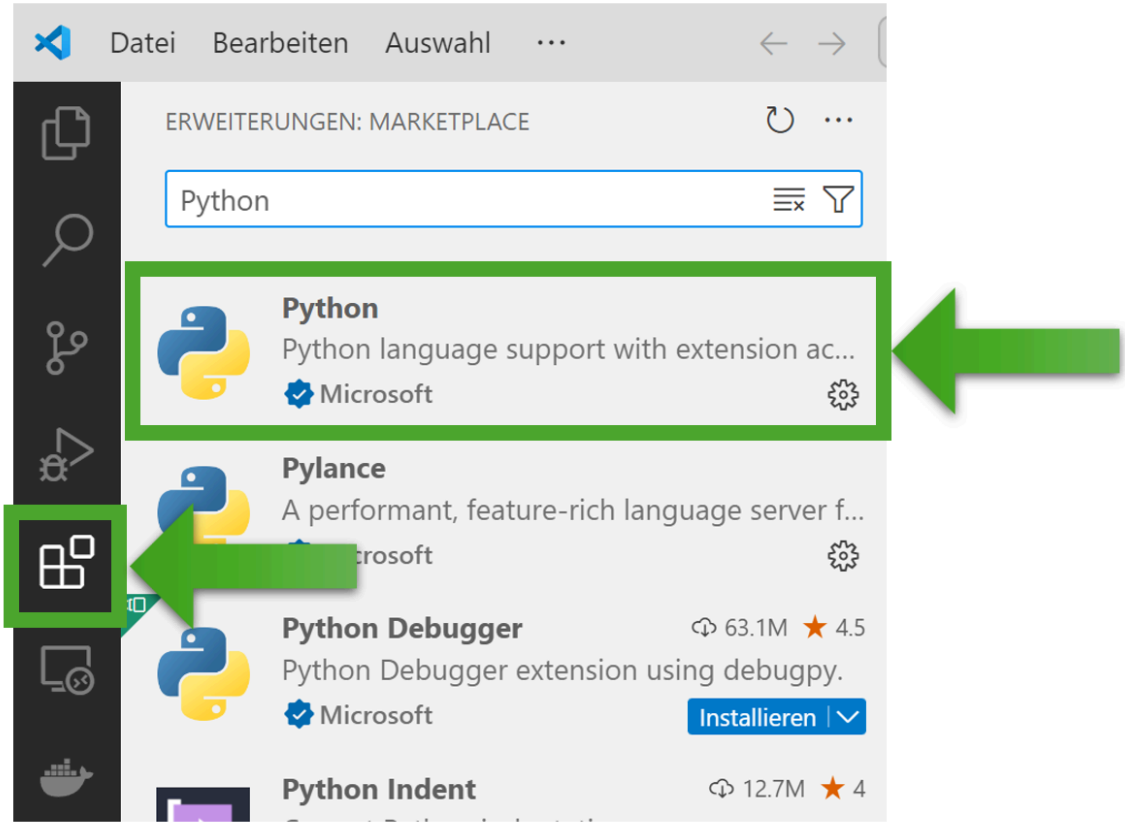
Plugin 1 - Deutsches Sprachpaket

Das erste Plugin heißt: `German Language Pack for Visual Studio Code`. Gebt dies in der Suche ein und installiert das Plugin. Im unteren rechten Bereich sollte ein Fenster auftauchen, in dem gefragt wird, ob ihr Visual Studio Code neustarten möchtet. Das müsst ihr tun. Nach dem Neustart ist die Oberfläche auf Deutsch verfügbar und nicht mehr auf Englisch.



Plugin 2 - Python

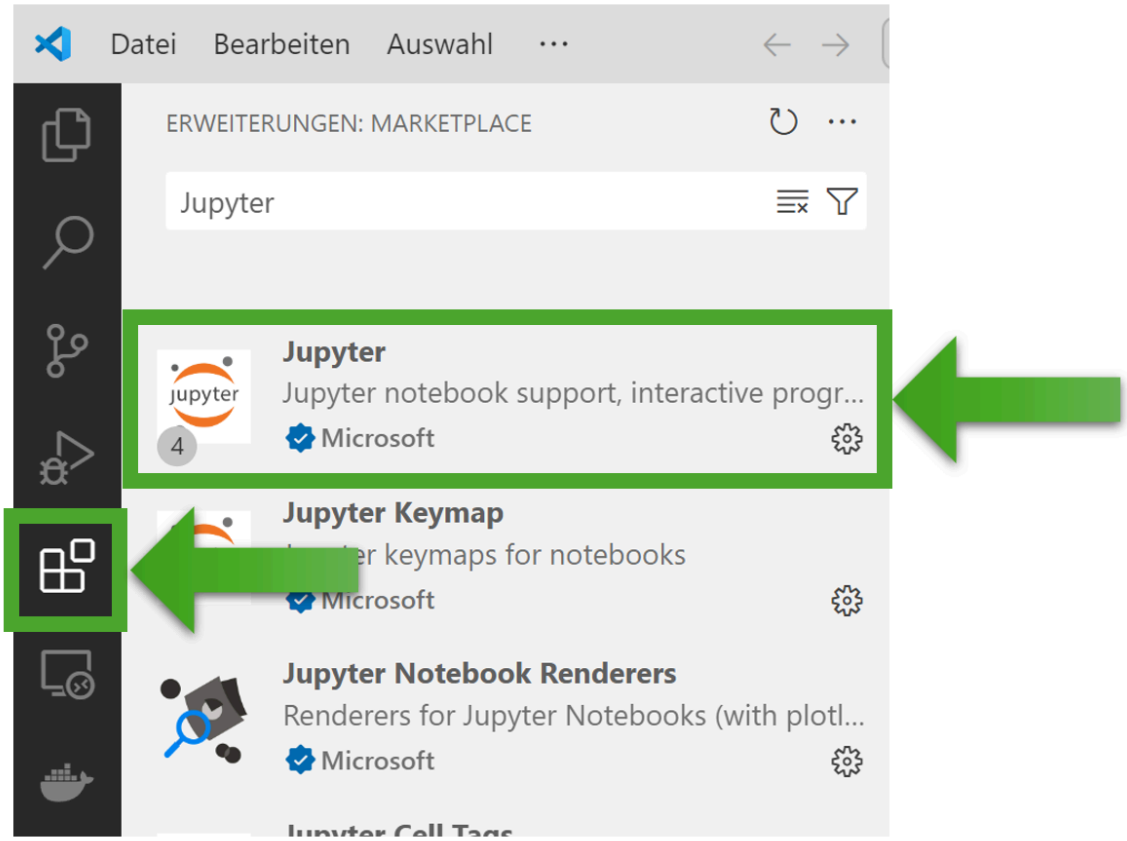
Um unsere Programme ausführen zu können, benötigen wir eine Unterstützung für unsere Programmiersprache. Wir verwenden Python als Programmiersprache. Das notwendige Plugin heißt ebenfalls `Python`:



Plugin 3 - Jupyter

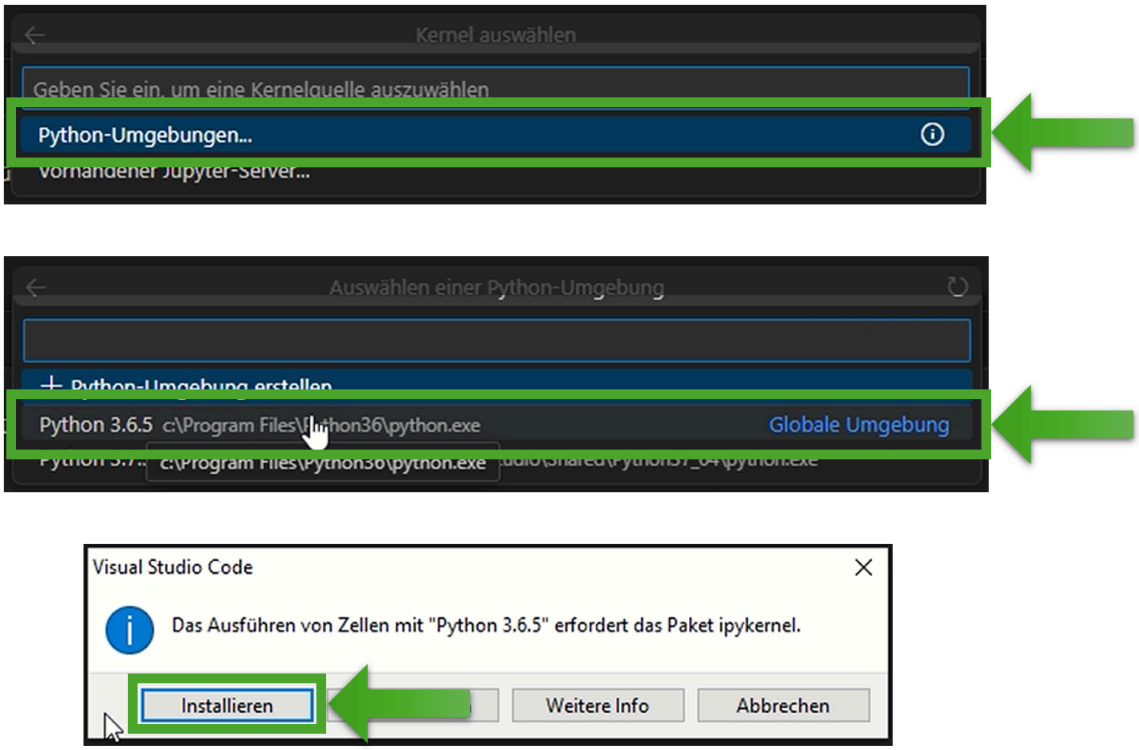
Die Dateien für diesen Selbstlernkurs sind Jupyter-Notebook-Dateien mit der Dateiendung `.ipynb`. **Visual Studio Code** kann diese Dateien verarbeiten und entsprechend darstellen. Solche Dateien ermöglichen es, erklärende Texte und Bilder direkt mit ausführbaren Code-Blöcken zu verbinden. Zu (nahezu) jeder Erklärung wirst du direkt darunter ein Beispiel finden, welches ausgeführt und getestet werden kann.

Um dieses Dateiformat vernünftig und mit allen Funktionen darstellen zu können, muss auch dafür ein Plugin installiert werden. Dieses heißt `Jupyter`.



1.3. Kernel

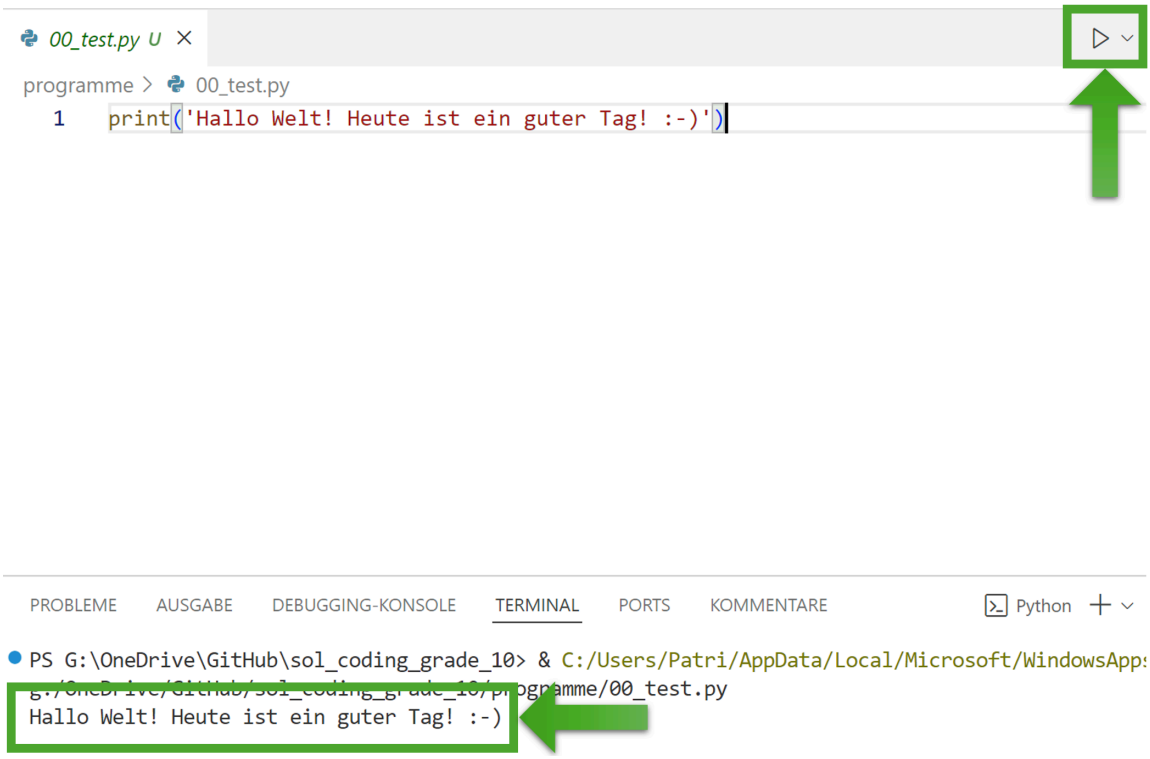
Der **Kernel** ist das Programm, das den Code in einem Jupyter-Notebook-Dokument ausführt. Ohne einen installierten Python-Kernel könnte Jupyter keinen Python-Code verarbeiten – es wäre nur ein Texteditor. Bevor wir Code-Zeilen ausführen können, müssen wir daher einen entsprechenden Kernel installieren. Bei der ersten Ausführung eines Code-Blocks erscheint im oberen Bereich ein Dialog zu dessen Installation. Der nachfolgende Screenshot zeigt euch, was ihr dabei jeweils anklicken müsst:



Die Installation dauert einen kurzen Moment, wundert euch nicht darüber, dass hat in der Regel alles seine Richtigkeit.

1.4. Testen

Im Ordner zum Kurs findet ihr die Datei `00_einrichtung.py` - eine erste kleine Python-Datei. Öffnet diese Datei! Im oberen rechten Bereich sollte ein Play-Button angezeigt werden. Mit diesem könnt ihr das Programm ausführen. Im unteren Bereich öffnet sich dann die Konsole. In dieser wird eine Ausgabe erzeugt. Zu lesen sollte dort `Hallo Welt! Heute ist ein guter Tag! :-)` sein. Probiert das aus! Der nachfolgende Screenshot zeigt nochmal, wo ihr klicken müsst:



Außerdem findet ihr eine Datei mit dem Namen `00_einrichtung.ipynb`. Das ist ein erstes Jupyter-Notebook-Dokument. Öffnet auch dieses und klickt neben dem Code-Block (auf der linken Seite) auf den Play-Button. Direkt unter dem Block sollte nun ebenfalls die Ausgabe generiert werden.

1.5. Warum Python?

Python ist eine **weit verbreitete, einfach zu erlernende und vielseitige Programmiersprache**. Sie wurde Anfang der 1990er-Jahren von Guido van Rossum entwickelt und erstmals 1991 veröffentlicht. Python zeichnet sich durch eine klare und gut lesbare Syntax aus, die es besonders für Einsteiger attraktiv macht.

Python gehört zu den sogenannten **höheren Programmiersprachen**, da es eine hohe Abstraktionsebene bietet und ermöglicht, Programme mit relativ wenig Code zu schreiben. Python verwendet eine klare und leicht verständliche Syntax, die sich an natürlicher Sprache orientiert. Es kann auf verschiedenen Betriebssystemen wie Windows, macOS und Linux ausgeführt werden. Python bietet eine umfangreiche Sammlung an Modulen und Bibliotheken für verschiedene Anwendungen: von Webentwicklung über Datenanalyse bis hin zu künstlicher Intelligenz.

Python wird in **vielen verschiedenen Bereichen** eingesetzt, darunter:

- Webentwicklung - leistungsfähige Webanwendungen
- Datenanalyse und wissenschaftliches Rechnen - Analyse großer Datenmengen
- Künstliche Intelligenz und maschinelles Lernen
- Automatisierung und Scripting - Python eignet sich hervorragend für kleine Skripte zur Automatisierung von Aufgaben
- Spieleentwicklung

Python ist nicht nur **besonders für Einsteiger geeignet**, sondern auch für professionelle Entwickler attraktiv, da es mächtige Werkzeuge für komplexe Anwendungen zur Verfügung stellt. Durch die große Community und die zahlreichen verfügbaren Ressourcen ist Python eine hervorragende Wahl für alle, die Programmieren lernen oder ihre Kenntnisse erweitern möchten.

Zusammenfassend lässt sich sagen, dass Python eine **vielseitige, benutzerfreundliche und leistungsfähige Programmiersprache** ist, die sich für zahlreiche Anwendungen eignet – von der einfachen Skripterstellung bis hin zur Entwicklung anspruchsvoller Softwareprojekte.

1.6. Geschichte von Python

Guido van Rossum schrieb 1996 über die Entstehung des Namens seiner Programmiersprache: "Vor über sechs Jahren, im Dezember 1989, suchte ich nach einem 'Hobby'-Programmier-Projekt, das mich über die Woche um Weihnachten beschäftigen konnte. Mein Büro ... war zwar geschlossen, aber ich hatte einen PC und sonst nichts vor. Ich beschloss mir einen Interpreter für die neue Skripting-Sprache zu schreiben, über die ich in der letzten Zeit nachgedacht hatte: ein Abkömmling von ABC, der UNIX/C-Hackern gefallen würde. Python hatte ich als Arbeitstitel für das Projekt gewählt, weil ich in einer leicht respektlosen Stimmung war (und ein großer Fan von Monty Python's Flying Circus)."

Guido van Rossum sagte in einem Interview: "Anfang der 80er Jahre habe ich an der CWI mit einem Team an der Sprache ABC gearbeitet. Ich hatte keine Ahnung wie ABC Python beeinflussen würde. Ich dachte an meine Erfahrungen und an den Frust mit ABC, und ich entschied mich, eine einfache Skriptsprache zu entwerfen. Sie sollte die Vorteile von ABC haben, nicht aber die Probleme/Nachteile. Es entstand eine einfache virtuelle Maschine, ein einfacher Parser, eine einfache Laufzeitumgebung und eine Syntax, die Einrückungen für die Gruppierung von Ausdrücken verwendet, und ein paar Datentypen: Dictionaries, Listen, Strings und Numbers/Integer."

Die folgenden **19** Regeln (PEP 20, auch: ZEN of Python) bilden die 20 Grundpfeiler der Programmiersprache Python:

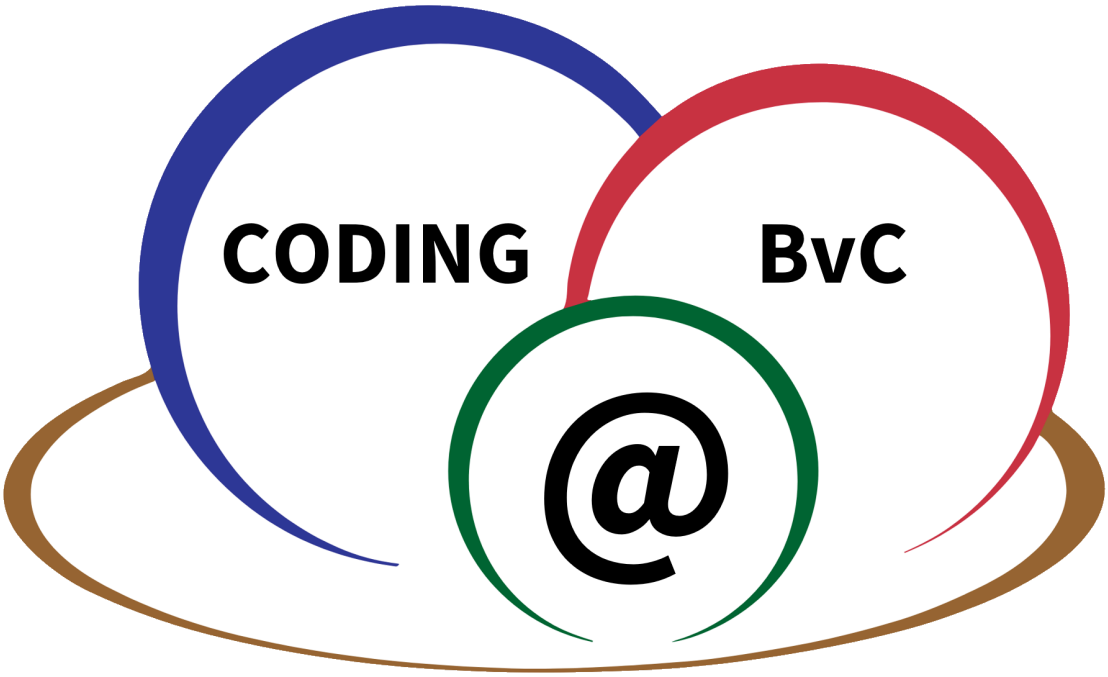
ZEN of Python! (original)

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one - and preferably only one - obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!

ZEN of Python! (übersetzt)

Schön ist besser als hässlich.
Explizit ist besser als implizit.
Einfach ist besser als komplex.
Komplex ist besser als kompliziert.
Flach ist besser als verschachtelt.
Sparsam ist besser als dicht.
Lesbarkeit zählt.
Sonderfälle sind nicht speziell genug, um die Regeln zu brechen.
Obwohl die Praktikabilität die Reinheit übertrifft.
Fehler sollten niemals schweigend vorübergehen. Es sei denn, sie werden ausdrücklich zum Schweigen gebracht.
Angesichts der Mehrdeutigkeit, widerstehen Sie der Versuchung zu raten.
Es sollte einen - und vorzugsweise nur einen - naheliegenden Weg geben, es zu tun.
Obwohl das anfangs nicht offensichtlich ist, es sei denn, man ist Holländer.
Jetzt ist besser als nie.
Obwohl nie oft besser ist als *gerade* jetzt.
Wenn die Umsetzung schwer zu erklären ist, ist es eine schlechte Idee.
Wenn die Umsetzung leicht zu erklären ist, kann es eine gute Idee sein.
Namespaces sind eine tolle Idee - lasst uns mehr davon machen!


CODING
BvC
@