(https://codingboss.de/)

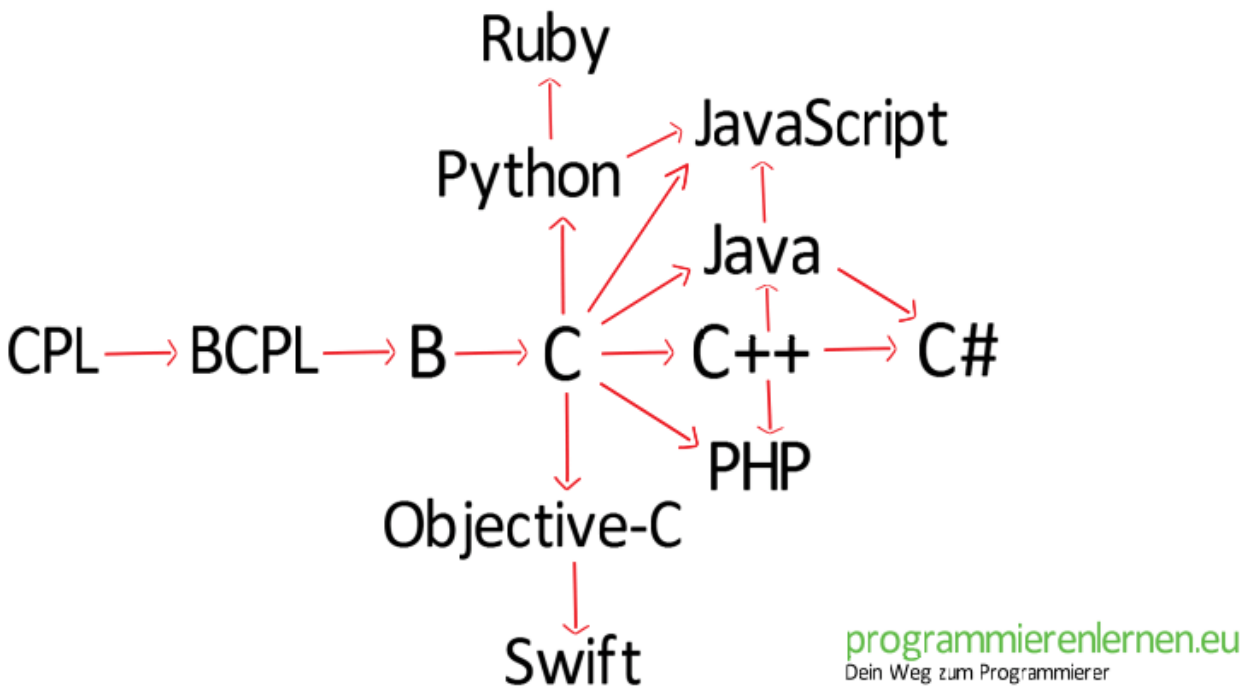
Pro Version (https://codingboss.de/pro)

Kurse (https://codingboss.de/programmierkurse/)

Einstiegsguide

(https://codingboss.de/programmieren-lernen/)

Programmiersprachen



Alle meist genutzten Programmiersprachen der Welt sind mit einander verwandt.

- Hier findest du Kurse zu den **12 beliebtesten Programmiersprachen** der Welt. Neben den Programmiersprachen findest du das Gebiet auf dem die Sprache hauptsächlich angewendet wird.
- Da alle Sprachen (bis auf SQL) mit einander verwandt sind, kannst du **problemlos alle lernen**.
- Wenn du neu in der Welt der Programmierung bist siehe dir unseren Einstiegsratgeber zum programmieren lernen (https://codingboss.de/programmieren-lernen/) an.

Ranking der Programmiersprachen

1.

Java

Kurs ansehen! (https://codingbos s.de/java/)

Java Anwendungsgebiete:

- Spiele für Windows und Consolen
- Android-Apps
- Webseiten
- Anfänger geeignet: Ja, die beliebteste Anfängersprache

2.

C

Kurs ansehen! (https://codingbos s.de/c/)

C **Anwendungsgebiete:**

- Windows-Software
- Anfänger geeignet: Nicht ideal

3.

Python

Kurs ansehen! (https://codingbos s.de/python/)

Python Anwendungsgebiete:

- Windows-Software
- Webseiten
- Anfänger geeignet: Ja

4.

C++

Kurs ansehen! (https://codingbos s.de/c-plus-plus/)

C++ Anwendungsgebiete:

- Windows-Software
- Windows-Spiele
- Anfänger geeignet: Nicht ideal

5.

C#

Kurs ansehen! (https://codingbos s.de/c-sharp/)

C# Anwendungsgebiete:

- Windows-Software
- Consolen-Spiele
- Anfänger geeignet: Nicht ideal

6.

Visual Basic .NET

Kurs folgt!

Visual Basic .NET Anwendungsgebiete:

- Windows-Software
- Webseite
- Anfänger geeignet: Nicht ideal

7.

JavaScript

Kurs ansehen! (https://codingbos s.de/javascript/)

JavaScript Anwendungsgebiete:

- Webseiten
- Anfänger geeignet: Ja, möglich

8.

SQL

Kurs folgt!

SQL Anwendungsgebiete:

- Datenbanken
- Anfänger geeignet: Nein

9.

PHP

Kurs ansehen! (https://codingboss.de/php/)

PHP Anwendungsgebiete:

- Webseiten
- Anfänger geeignet: Nicht ideal

10.

Objective-C

Kurs ansehen! (https://codingboss.de/objective-c/)

Objective-C Anwendungsgebiete:

- Apple-Produkte
- Anfänger geeignet: Nicht ideal

11.

Swift

Kurs ansehen! (https://codingbos s.de/swift/)

Swift Anwendungsgebiete:

- Apple Produkte
- Anfänger geeignet: Ja, möglich

12.

Ruby

Kurs ansehen! (https://codingbos s.de/ruby/)

Ruby Anwendungsgebiete:

- Windows-Software
- Webseiten
- Anfänger geeignet: Ja

7 Fragen die dich weiterbringen

Wenn du neu in der Welt der Programmierung bist, helfen dir die folgenden 7 Fragen bei der Orientierung.

Wie sehen die Programmiersprachen aus?

Jede Sprache schreibt sich etwas anders. Mit dem bekannten Hello-World-Programm (https://codingboss.de/blog/hello-world/) kannst du sehen wie sich die Programmiersprachen in ihrer Schreibweise unterscheiden.

Wie viele Programmiersprachen gibt es?

Es gibt weit über 1000 Programmiersprachen. Davon werden aber nur wenige häufig genutzt. Manche Sprachen werden speziell für eine einzige Sache entwickelt. Beispielsweise hat Facebook eine eigene Sprache entwickelt, die sich besonders gut zum programmieren der Facebook-Webseite eignet.

Wie schwer ist es diese Sprachen zu lernen?

Es ist nur bei der ersten schwer. Sobald du das Grundprinzip der Programmierung verstanden hast, fällt es dir viel leichter eine neue Sprache zu lernen. Du musst dich nur in eine Sprache reinarbeiten, danach lernst du die anderen sehr viel schneller.

Welche Programmiersprache ist die beste für Anfänger?

Java, Python oder Ruby. Java wird am häufigsten genutzt und ist zudem sehr stabil, d.h. du wirst nicht gleich bestraft, sobald du mal Mist eingibst.

Welche Programmiersprache soll ich lernen?

Das hängt davon ab, auf welches Gebiet du dich als Programmierer spezialisieren möchtest. Siehe: Einstiegsratgeber zum Programmieren lernen (https://codingboss.de/programmieren-lernen/).

Wie wird aus dem Code ein Programm?

Dies geschieht entweder mit einem **Compiler oder einem Interpreter**. Programme die mit einem Interpreter in Maschinensprache übersetzt werden, werden beim Ausführen des Programmes übersetzt.

Der Interpreter arbeitet also während das Programm läuft. Die Befehle werden nach für nach übersetzt und direkt ausgeführt. Z.B. wird die erste Codezeile eines Codes in Maschinensprache übersetzt und ausgeführt. Dann wird die zweite Zeile übersetzt und ausgeführt. So lange bis das Programm zu Ende ist.

Der Compiler dagegen übersetzt ein Quellcode komplett, speichert ihn und führt ihn erst dann aus. Ob ein Programm mit einem Interpreter oder mit einem Compiler übersetzt wird, hängt von der jeweiligen Programmiersprache ab in der der Quellcode geschrieben ist. Man spricht auch von interpretierten und kompilierten Programmiersprachen.

Vor- und Nachteile von Compilern bzw. Interpretern

Beide Varianten haben Vor- und Nachteile. Der Vorteil eines Interpreters ist, dass man Codefehler schneller finden kann. Bei einem Compiler muss man erst das ganze Programm übersetzten um es auf Fehler zu prüfen.

Den Interpreter kann man ab einer bestimmten Stelle im Quellcode starten lassen und entdeckt so direkt eventuelle Fehler. Auch kann man Fehler von interpretierten Programmiersprachen schneller beheben.

Man ändert einfach die betroffene Codezeile und führt das Programm erneut aus. Bei kompilierten Programmiersprachen muss der Quellcode bei jeder Änderung erneut vollständig übersetzt werden.

Der Vorteil von kompiliertem Code ist, dass er mit einer Übersetzung so oft ausgeführt werden kann wie man möchte. Der Code kann übersetzt werden und dann beliebig oft ausgeführt werden.

Ein Interpreter dagegen muss den Code bei jeder Ausführung erneut übersetzten. Auch kann ein kompilierter Code viel schneller ausgeführt werden, da der Prozessor nicht auf die Übersetzung warten muss.

Sicherlich denken Sie sich jetzt, dass es gut wäre eine Mischung aus einem Compiler und Interpreter zu verwenden, der alle Vorteile vereint. Und genau das gibt es für die meisten modernen Programmiersprachen (Java, C#, JavaScript, PHP, usw.).

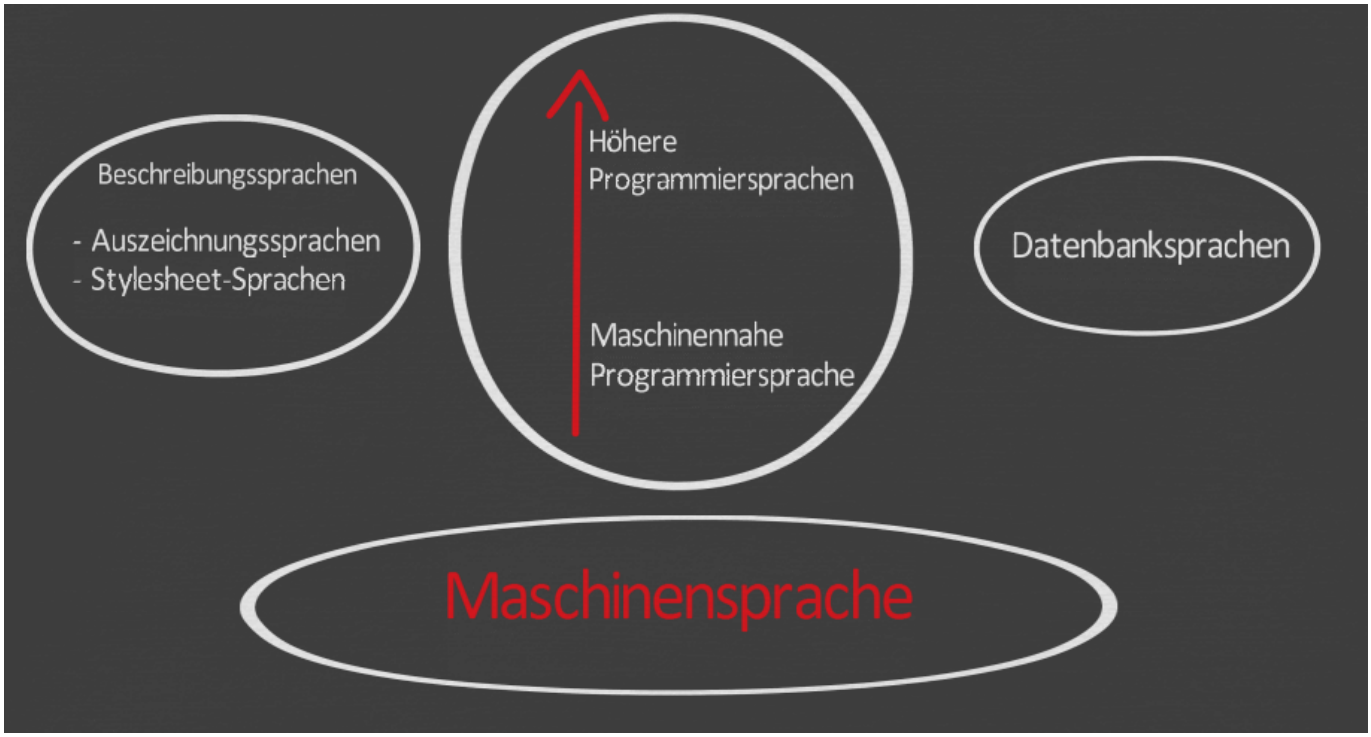
Heutzutage sind die meisten Programmiersprachen eine Mischung aus interpretierter und kompilierter Sprache.

Was sind Skriptsprachen?

Skriptsprachen bilden eine Untergruppe in den Programmiersprachen. Was Skriptsprachen genau sind erfährst du in einem kurzen Video.

IT-Definitionen: Skriptsprachen

Welche Programmiersprachen gibt es?



Das Bild zeigt, dass man die Programmiersprachen in 4 Gruppen einteilen kann.

Programmiersprachen dienen dazu dem Menschen die Möglichkeit zu geben einem Computer Befehle zu geben. Aber die einzige Sprach die ein Computer versteht ist Maschinensprache.

Maschinensprache besteht ausschließlich aus Nullen und Einsen. Natürlich wäre es sehr schwierig seine Befehle nur mit Hilfe von Nullen und Einsen auszudrücken, deshalb gibt es Programmiersprachen. Da der Computer den Code einer Programmiersprache aber nicht versteht, muss er zunächst übersetzt werden.

Dies geschieht mit einem Compiler oder mit einem Interpreter. Der Compiler oder der Interpreter übersetzt den für Menschen verständlichen Code einer Programmiersprache in Maschinensprache. (Den Unterschied zwischen Compilern und Interpretern sehen wir uns gleich an.)

Assembler - Die maschinennahe Programmiersprachen

Programmiersprachen die für Menschen verständlicher sind nennt man höhere Programmiersprachen. Programmiersprachen die näher an der Maschinensprache sind nennt man maschinennahe Sprachen oder hardwarenahe Sprachen.

Die Sprachen die am nächsten am Maschinencode sind, sind die Assemblersprachen. Die Assemblersprachen werden vom Assembler in Maschinensprache übersetzt. Es gibt auch sehr viele Assemblersprachen. Jeder Prozessor hat seine eigene Assemblersprache!

Um im Assembler programmieren zu können braucht es sehr viel Erfahrung, nicht nur mit höheren Programmiersprachen, auch in der allgemeinen Funktionsweise des Computers. Für Anfänger ist die Assemblerprogrammierung definitiv nichts. Es war nur wichtig, dass Sie schon Mal davon gehört haben und wissen was Assembler und Assemblersprache eigentlich ist.

Datenbanksprachen

Dann gibt es noch Datenbanksprachen. Auch die Datenbanksprachen kann man in verschiedene Untergruppen einteilen. So gibt es:

- Datenverarbeitungssprachen
- Datenbeschreibungssprachen
- Datenaufsichtssprachen

Ich möchte jetzt nicht weiter auf Datenbanken eingehen. Wenn Sie sich näher mit Datenbanken beschäftigen wollen, empfehle ich ihnen sich hier (https://www.eu-datenbank.de/fachartikel/die-wichtigsten-infos-zu-datenbanksprachen/) einmal umzusehen.

Als Anfänger könnte man auch eine Datenbank zu programmieren lernen. Mit ein bisschen Übung hat man die Grundlagen der Datenbanksprachen schnell drauf. Zum Einstieg macht es aber mehr Sinn eine Programmiersprache zu lernen, da man mit diesen viel mehr machen kann.

Höhere Programmiersprachen

Die Programmiersprache die am nächsten am menschlichen Verständnis ist, ist wahrscheinlich Python. Python eignet sich deshalb auch sehr gut für Anfänger.

Generell sind aber alle Objektorientierte Programmiersprachen auf das menschliche Verständnis ausgerichtet. Programme mit einer objektorientierten Programmiersprache zu entwickeln ist deutlich leichter als in anderen Programmiersprachen.

Auch zum Programmieren lernen eignen sich objektorientierte Sprachen sehr gut. Was genau man unter Objektorientierung versteht lernen Sie im Kurs der theoretischen Programmierung (unten auf der Seite).

Jetzt sein nur so viel gesagt, dass sich die Objekte aus der Objektorientierung auf die reale Welt beziehen. Objekte in einer Programmiersprache sind Objekte, die Dinge aus der realen Welt beschreiben. Z.B. könnte es ein Objekt namens „Hund“ geben und eines namens „Katze“.

Diese Objekte könnten dann (genauso wie in der realen Welt) miteinander agieren. Z.B. könnte der Hund die Katze jagen und wenn der Hund die Katze gefangen hat könnte die Katze „Miau“ sagen. Ein Objekt kann also die „Taten“ eines anderen Objektes auslösen.

Diese „Taten“ nennt man in der Objektorientierung Operationen. So kann eine Katze z.B. die Operationen laufen, springen, miauen, essen, fauchen, kratzen usw. anbieten. Der Hund müsste, um die Katze fangen zu können, die Operation jagen besitzen. Sie könnten ein Zufallsgenerator einbauen, der bestimmt ob der Hund die Katze bekommt oder ob die Katze es schafft zu entkommen.

Wenn Sie dieses Programm dann ausführen bekommen Sie ein „miau“ zurückgegeben, wenn der Hund die Katze bekommt oder kein miau wenn die Katze entkommt.

Heutzutage ist die Objektorientierung das am häufigste genutzte Programmierparadigma. Programmierparadigmen sind der Fachbegriff für den Programmierstil. So gibt es Sprachen in denen man objektorientiert Programmieren kann oder eben nicht.

Die Programmiersprachen die weltweit am häufigsten genutzt werden sind speziell auf die Objektorientierung ausgelegt, man nennt sie deshalb objektorientierte Programmiersprachen. Folgende Sprachen sind objektorientiert:

- Python
- Ruby
- Java
- PHP
- JavaScript

- C++
- C#
- Objective-C
- Swift
- Scratch

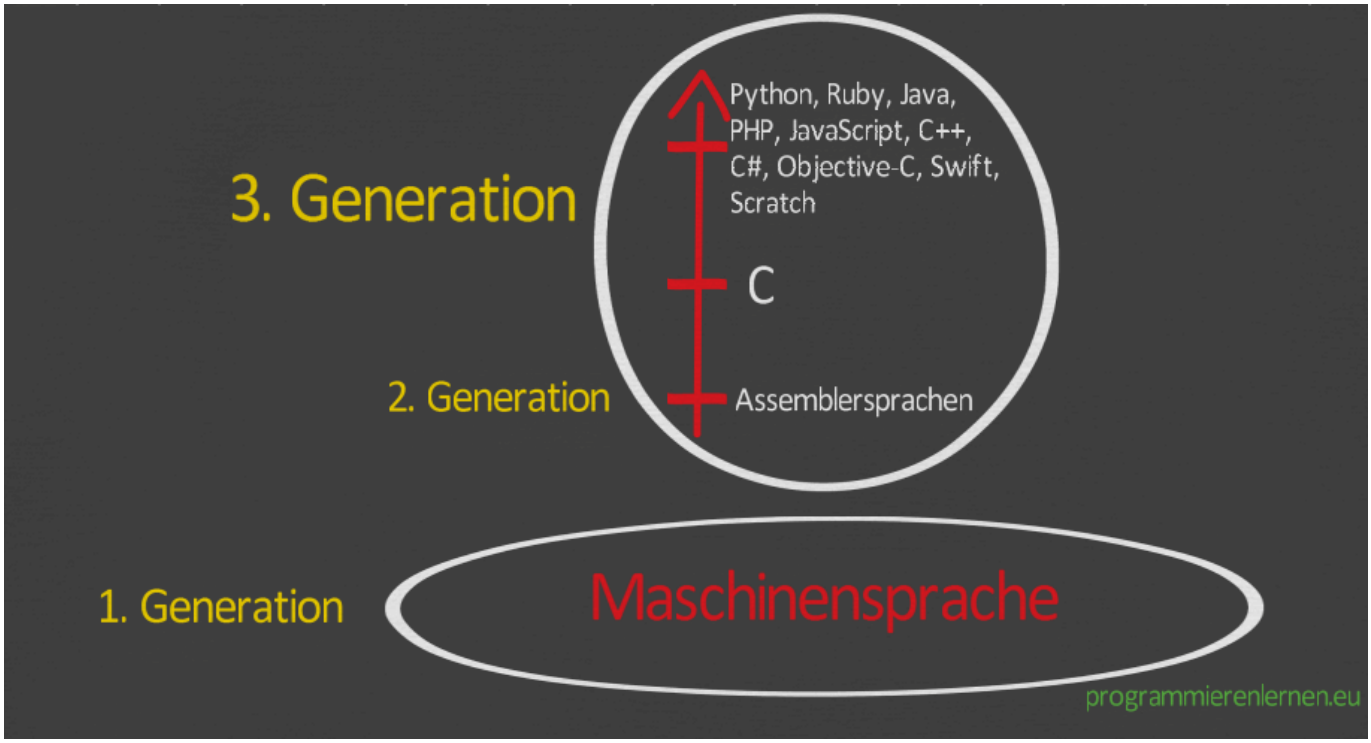
Es gibt natürlich noch weitere objektorientierte Programmiersprachen. Dies sind aber die am häufigsten Verwendeten. Wer Programmieren lernen will beginnt am besten mit objektorientierten Sprachen.

Zwar kann man auch C als Einstieg wählen, aber C wird immer schwerer je mehr man sich mit ihr beschäftigt. Mit ein bisschen Übung bekommen es aber auch Anfänger hin.

Wie Sie schon gelernt haben sind aus der Sprache C viel objektorientierte Sprachen hervorgegangen. C ist aber keine objektorientierte Programmiersprache und auch keine maschinennahe Programmiersprache.

Als C im Jahr 1972 entwickelt wurde gab es noch keine Objektorientierung. C machte den ersten Schritt in die Programmiersprachen der dritten Generation (siehe Bild). Und C war auf das menschliche Verständnis ausgelegt. C ist im Vergleich zu Assemblersprachen sehr verständlich, aber man kann auch sehr hardwarenahe in ihr programmieren.

Die Generationen der Programmiersprachen



Das Bild zeigt die 3 Generationen der Programmiersprachen

1. Generation

Die Programmiersprachen der ersten Generation bestanden nur aus Einsen und Nullen, eben genau den Elementen die ein Computer versteht. Die damaligen Programme waren sehr simple, da es extrem schwierig ist in Maschinensprache zu programmieren.

2. Generation

Die Zweite Generation sind die Assemblersprachen. Mit diesen konnte man Befehle schreiben (in Buchstaben) die dann in Maschinencode übersetzt wurden. Auf diese Weise konnten Entwickler anspruchsvollere Programme schreiben.

Man brauchte aber immer noch sehr viel Code um einen Befehl zu übermitteln und es war immer noch relativ kompliziert. Assemblersprachen werden sehr häufig zum Programmieren lernen genutzt. So gut wie jeder Informatikstudent muss früher oder später Übungen im Assembler programmieren.

Dadurch dass im Assembler sehr hardwarenahe programmiert wird, lernt man die Funktionsweise eines Computers viel besser kennen.

3. Generation

Die Sprachen der dritten Generation sind bis heute die meist genutzten der Welt. Eine der ersten war C. Diese Programmiersprachen kamen dem menschlichen Sprachen schon sehr viel näher.

In diesen Programmiersprachen war es sehr viel einfacher Algorithmen auszuführen, was dem menschlichen Verhalten sehr nahe kommt. (Algorithmen sind Schritt-für-Schritt Anleitungen, z.B. eine Bauanleitung, eine Rezept, eine Wegbeschreibung oder ein Betriebssystem für Computer. Auch Google ist ein Algorithmus.)

Um die Programmiersprachen der dritten Generation den menschlichen Sprachen noch näher zu bringen, entwickelte man die Objektorientierung. Objektorientierte Sprachen zählt man zu den Programmiersprachen der dritten Generation. Sprachen der dritten Generation nennt man auch Hochsprachen.