Programmiersprachen

- Mithilfe von Programmiersprachen gibt man dem Computer Anweisungen, welche Operationen er ausführen soll.
- Der Computer bzw. Prozessor kann mit dem Quellcode nichts anfangen, da er lediglich auf Bit/Byte-Ebene arbeitet. Der Quellcode muss daher in passenden Maschinencode umgewandelt werden, der zwar für den Menschen nicht (nur schwer) lesbar ist, dafür aber von der Prozessoreinheit verstanden wird.
- Die Übersetzung kann auf 2 Arten erfolgen:
  - Durch einen Compiler für den jeweiligen Computer- bzw. Prozessortyp
  - JIT (während der Ausführung wird der Code sozusagen "live" übersetzt)
  - oder meist Kombination aus beidem: Quelltext wird in Zwischencode übersetzt und anschließend zur Laufzeit innerhalb einer Laufzeitumgebung in Maschinencode übersetzt

- Was sollte ein Programmiersprache können?
  - Input / Output (Informationen /Daten verarbeiten (entgegennehmen <=> ausgeben))
  - Deklaration von Variablen zur Zwischenspeicherung von Daten
  - mathematische Grundoperationen
  - Grundfunktionen zur Zeichenkettenverarbeitung
  - Steueranweisungen (Conditions, Loops, Funktionen, Einbindung von Bibliotheken)

- Es gibt unterschiedliche Arten von Sprachen:
  - Maschinennähe:
    - Maschinennahe Sprachen (z.B. C, Assembler, …)
    - Höhere Sprachen (z.B. Python, JavaScript, …)
  - Kompilierung:
    - Scriptsprache (benötigt keine Compilierung, z.B. JavaScript, Python, …)
    - Kompilierte Sprachen (benötigen Compilierung, z.B. C, C++, …)
  - Zweck:
    - Programmiersprachen
    - Auszeichnungssprachen (z.B. HTML, XML)
    - Beschreibungssprachen (z.B. LaTex)
    - Stylesheet-Sprachen (z.B. CSS)

- Es gibt unterschiedliche Programmierparadigmen:
  - Imperativ / Prozedural:
    - Folge von Anweisungen, die vorgeben, in welcher Reihenfolge die Anweisungen ausgeführt werden
    - "Wie?"
    - z.B. C, C++, Java
  - Deklarativ:
    - Beschreibung des Problems steht im Vordergrund, Lösungsweg wird automatisch ermittelt
    - "Was?"
    - z.B. SQL, Lisp, Prolog
  - Funktional:
    - Funktionen werden nicht nur definiert und angewendet, sondern auch wie Daten miteinander verknüpft, als Parameter verwendet und als Funktionsergebnisse auftreten können
  - Objektorientiert
    - es werden Klassen/Objekte verwendet, die direkt die darunterliegenden Daten verwenden

        - im Gegensatz zum prozeduralen Ansatz, bei dem die Daten von den die Objekte verarbeitenden Routinen getrennt gehalten werden.
    - Logisch
    - Nebenläufig
- Es gibt unterschiedliche Typsysteme:
    - Der Computer muss in der Regel wissen, wie viel Speicherplatz für den Wert einer Variablen / Konstanten reserviert/belegt werden muss
    - Dazu gibt es 2 unterschiedliche Ansätze:
        - Typisierte Sprachen:
            - Der Inhalt einer Variable wird explizit durch einen Datentyp festgelegt
                - statische Typisierung: Typüberprüfung findet bei Übersetzung des Programms statt
                - dynamische Typisierung: Typüberprüfung findet zur Laufzeit des Programms statt
        - Typenlose Sprachen:
            - Verfügen über keine differenzierten Datentypen
            - Datentyp einer Variable wird erst zur Laufzeit festgelegt
            - Bei Zuweisung eines Wertes eines anderen Typs findet eine Umwandlung statt.