Einführung in das Programmieren

DEAF YOUNG CODE

Projektnummer: 2023-2-IT03-KA220-YOU-000179130

pictures: www.freepik.com

Co-funded by
the European Union

Video


Einführung in die Programmierung
https://youtu.be/EWX2ZH-QbPo?si=YoZiXzdZTP4AZQ6o

Was ist Programmierung?

DEAF YOUNG CODE

- Programmierung ist die **Grundlage moderner Technologie**.

- Ermöglicht die Entwicklung von Software,
  Anwendungen und Systemen,
  die viele Lebensbereiche verbessern.

- Dabei werden Anweisungen oder Code in einer
  Programmiersprache geschrieben,
  die ein Computer verstehen kann.

Sind Programmieren und Coding dasselbe?

DEAF YOUNG CODE

- Coden (= Anweisungen schreiben)
  ist nur ein Teil der Programmierung.

- Das Programmieren umfasst noch viel mehr.
  Es umfasst den gesamten Prozess der Erstellung
  eines digitalen Produkts.

Wichtige Begriffe

Algorithmen
Schritt-für-Schritt-Anleitungen zur Problemlösung, wie ein Kochrezept.

Variablen
Platzhalter zum Speichern von Daten,
die während der Programmausführung geändert werden können.

Datenstrukturen
Methoden zur Organisation und Speicherung von Daten, wie Listen oder Wörterbücher.

Kontrollstrukturen
Werkzeuge wie Schleifen und Konditionalen für Entscheidungsfindung und Wiederholung.

Funktionen
Wiederverwendbare Codeabschnitte, die bestimmte Aufgaben ausführen.

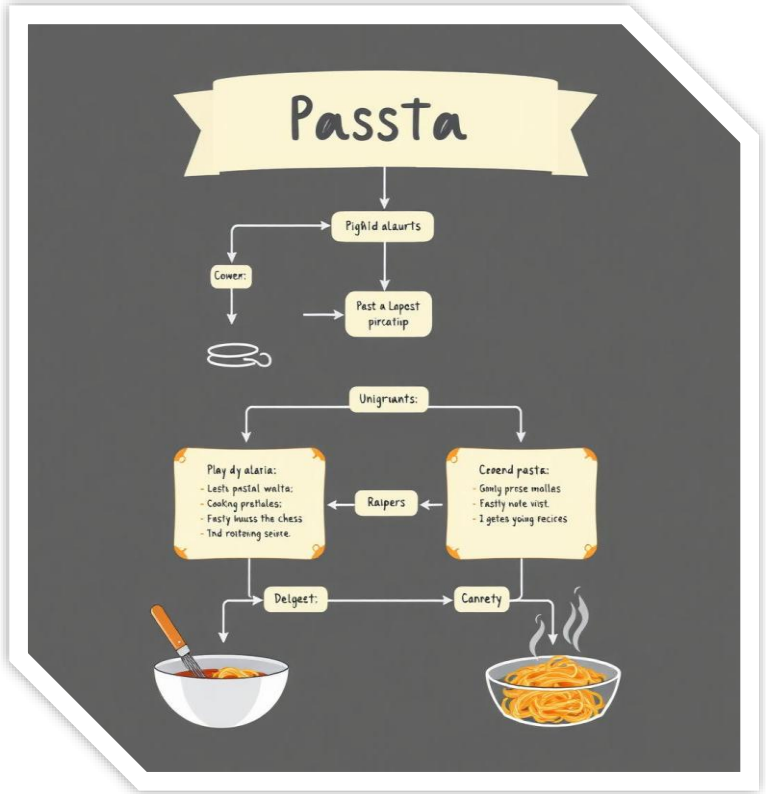
Wichtige Konzepte und Aufbau

Steuern des Programm-Verlaufs

DEAF YOUNG CODE

Algorithmus - was ist das?

- Eine Abfolge klar definierter Schritte,
  die dazu bestimmt ist,
  ein bestimmtes Problem zu lösen.
- Es ist wie eine **Schritt-für-Schritt-Anleitung**.
- Es ist wie ein Rezept, aber für Computer!

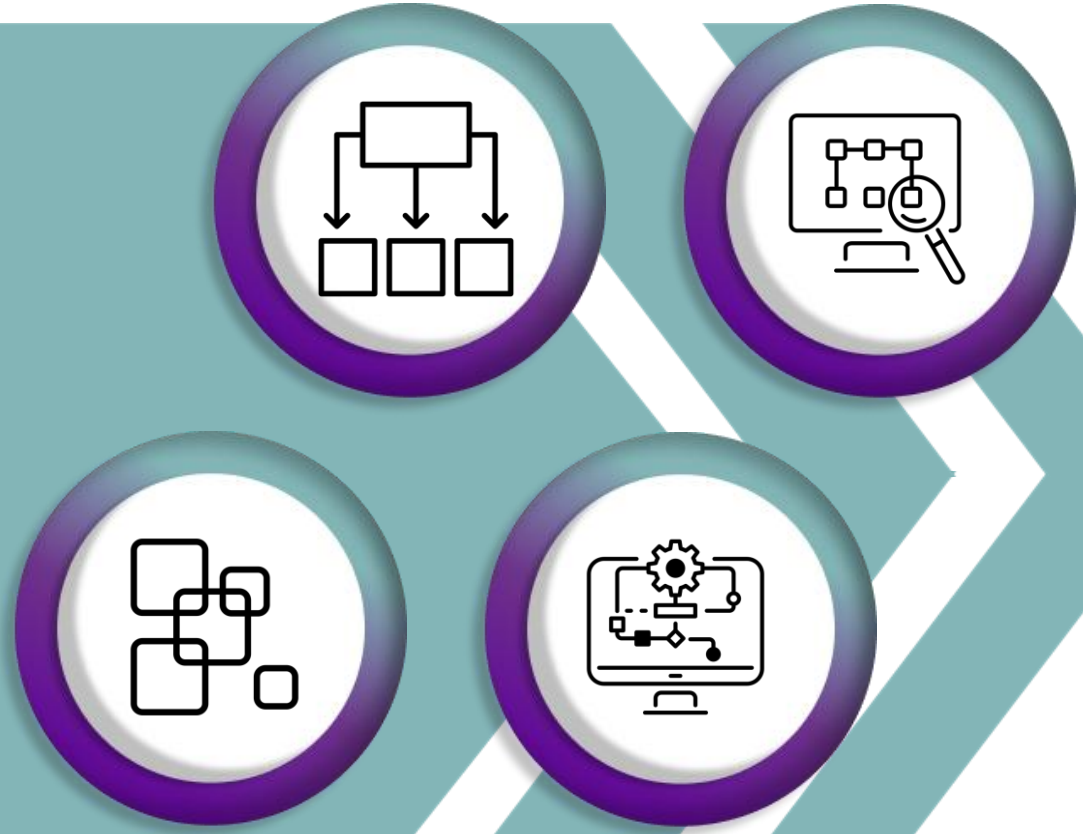


Video

Algorithms – Schritt für Schritt zum Ziel

CODE

1. Poblemlösung
2. Effizienz
3. Automation
4. Grundlage



Algorithmus – Schritt für Schritt zum Ziel

DEAF YOUNG CODE

Problemlösung
Zerlegen von komplexen Probleme in handhabbare Teile.

Effizienz
Ein guter Algorithmus löst ein Problem schnell mit minimalen Ressourcen.

Automatisierung
ermöglicht es Computern,
Aufgaben automatisch ohne menschliches Eingreifen auszuführen.

Grundlagen
Das Verständnis von Algorithmen bildet die Grundlage
für das Schreiben von effizientem und effektivem Code.

Activity

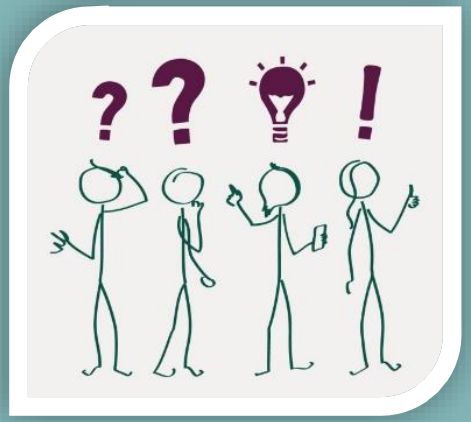
Algorithms

DEAF YOUNG CODE

Beispiele aus dem Alltag:

- Suchmaschinen im Internet
- GPS-Navigation

Was fällt euch noch ein?



Wichtige Konzepte und Aufbau
Steuern des Programm-Verlaufs

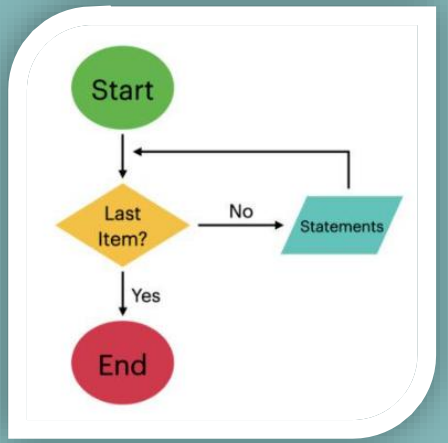
DEAF YOUNG CODE

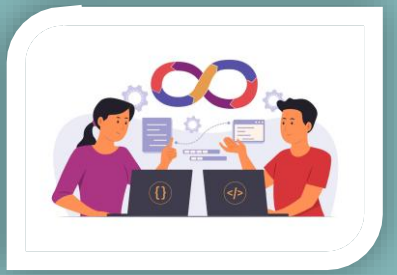
if-else (= wenn-dann)

“if" wenn das wahr ist > dann tue das.
"else" > wenn nicht, tue etwas anderes.

Loops (Schleifen)

Ermöglicht das Wiederholen von Anweisungen bis eine bestimmte Bedingung erfüllt ist.





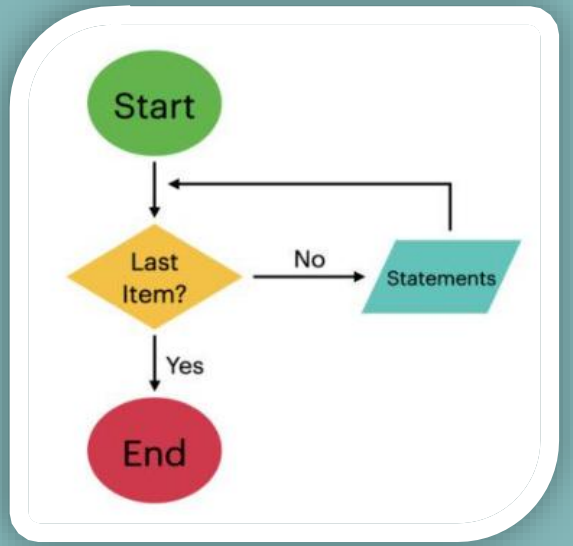
Wichtige Konzepte und Aufbau
Steuern des Programm-Verlaufs

DEAF YOUNG CODE

For-Loop
Wird verwendet, um eine bestimmte Anzahl von Malen zu iterieren, oft unter Verwaltung einer Zählervariable.

While-Loop
Wiederholt einen Codeblock,
solange eine bestimmte Bedingung wahr bleibt.

Do-While-Loop
Ähnlich wie die While-Schleife,
garantiert jedoch, dass der Code mindestens einmal ausgeführt wird,
wobei die Bedingung erst am Ende überprüft wird.



DEAF YOUNG CODE

Debugging

= Fehler (Bugs) in deinem Code zu finden und zu beheben.

Es ist eine sehr wichtige Fähigkeit für Programmierer*innen.

Debugging – Fehler im Programm lösen

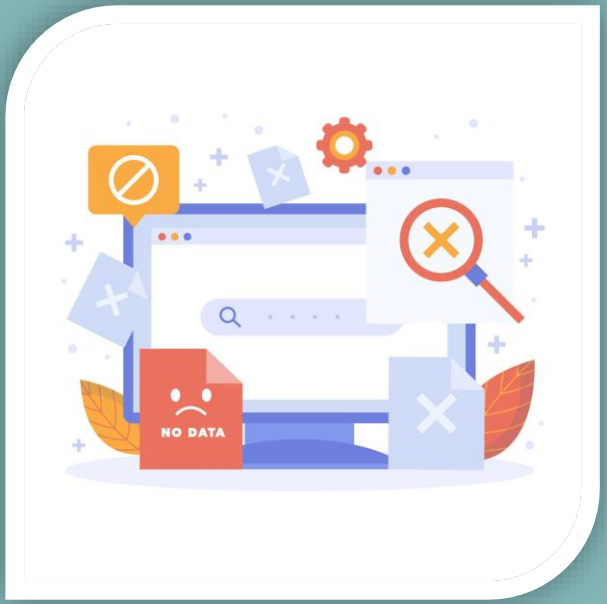
DEAF YOUNG CODE

Häufige Fehlerquellen

Syntaxfehler
Fehler in der Grammatik der Programmiersprache (z. B. Tippfehler, fehlende Satzzeichen).

Laufzeitfehler
Fehler, die während der Programmausführung auftreten
(z. B. Division durch null, Zugriff auf eine nicht existierende Datei).

Logikfehler
Das Programm läuft, liefert aber nicht die erwarteten Ergebnisse
(z. B. falsche Berechnungen, falsche Bedingungen).



Debugging – Finden und Lösen von Problemen

DEAF YOUNG CODE

Tipps:

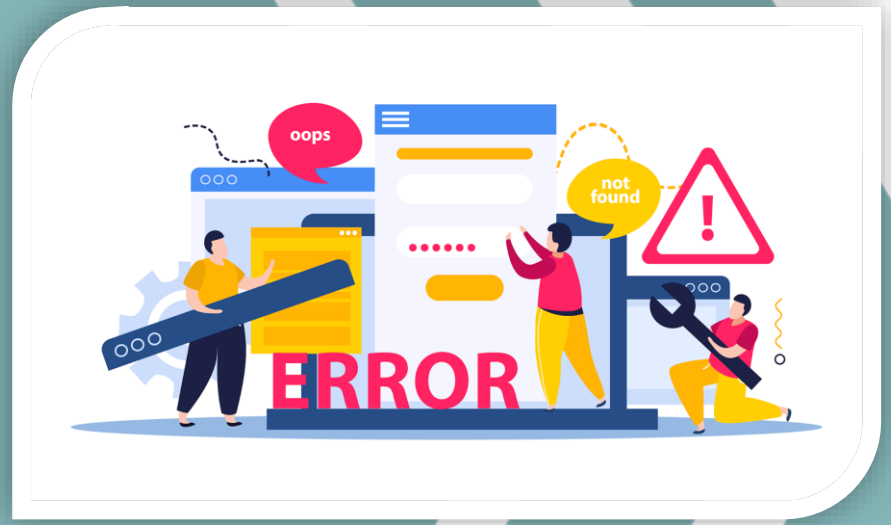
Fehlermeldungen lesen:
Achte genau darauf, was die Fehlermeldung sagt.
Sie liefert oft Hinweise auf das Problem.

Verwende einen Debugger:
Debugger sind Werkzeuge, die es dir ermöglichen, deinen Code Zeile für Zeile durchzugehen und die Werte von Variablen zu überprüfen.

Ausgabeanweisungen:
Füge eine Ausgabeanweisungen ein,
um die Werte von Variablen an verschiedenen Punkten in Ihrem Code anzuzeigen.

Teste deinen Code:
Teste deinen Code mit verschiedenen Eingaben, um Randfälle zu finden,
die möglicherweise Fehler verursachen.

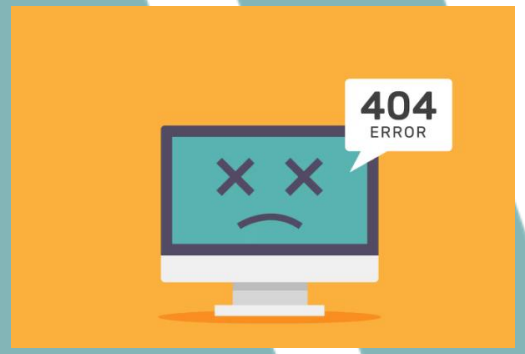
Rubber-Duck-Debugging:
Erkläre deinen Code Zeile für Zeile einer Gummiente
(oder einem beliebigen unbelebten Objekt). Dies kann Ihnen helfen, Fehler in Ihrer Logik zu erkennen.



Herausforderungen

Herausforderungen wie Programmierfehler,

Aufgabenkomplexität und die Lernkurve sind häufig.



Debugging,
Oder Code sorgfältig zu überprüfen, um Fehler zu finden, ist eine wichtige Fähigkeit.

Mit Geduld und Ausdauer,

Programmieren wird mit der Zeit einfacher.

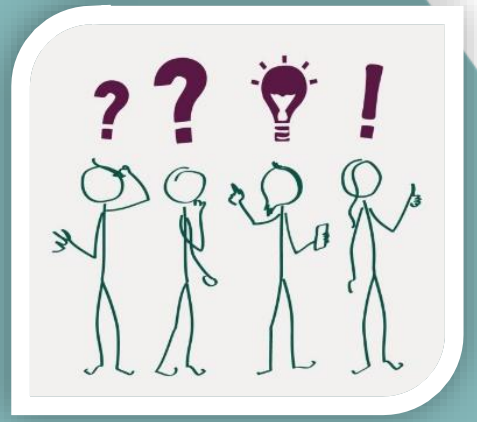
Activity

FlowChart - Flussdiagramm

DEAF YOUNG CODE

Beispiele aus dem Alltag:

1-Kaffee machen
2-Wasser aus dem Hahn trinken
3-Sich anziehen, bevor man rausgeht
4-Einen Bus nehmen
5-Zähne mit Zahnpasta putzen
6-Geschirr mit Spülwasser reinigen
7-Handy aufladen



Activity

Quiz

Vielen Dank für deine Aufmerksamkeit!

DEAF YOUNG CODE

… bleib dran für die nächste Präsentation!

Von der Europäischen Union finanziert. Die geäußerten Ansichten und Meinungen entsprechen jedoch ausschließlich denen des Autors bzw. der Autoren und spiegeln nicht zwingend die der Europäischen Union oder der Europäischen Exekutivagentur für Bildung und Kultur (EACEA) wider.
Weder die Europäische Union noch die EACEA können dafür verantwortlich gemacht werden.

Projektnummer : 2023-2-IT03-KA220-YOU-000179130

Co-funded by
the European Union