Co-funded by
the European Union

DEAF YOUNG
CODE

Modul 3

Einführung - Coding -
Einführung - Programmierung

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.

MODUL-PLAN

Modulplan	
Titel	**Einführung in das Coding**
Dauer	3 Stunden
Lerziele	• Grundlegende Konzepte der Programmierung und des Coding verstehen. • Verschiedene Programmiersprachen und ihre Verwendungszwecke erkennen. • Einfache Programme unter Verwendung grundlegender Programmierkonstrukte (z. B. Variablen, Schleifen, Bedingungen) schreiben. • Problemlösungsfähigkeiten durch Codierungsübungen entwickeln • Gesamtstrukturen eines Programmes erkennen • Bedeutung von Algorithmen und logischem Denken in der Programmierung verstehen • Lernen, wie man einfache Code-Fehler entdeckt • Erkunden von realten Anwendungsmöglichkeiten der Codierung in verschiedenen Branchen.

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.

Modulplan	
	• Fördern der Kreativität durch Programmier-Kenntnisse
Lehrplan	**Grundlagen des digitalen Designs und der digitalen Entwicklung** **Zielgruppe:** gehörlose Jugendliche **Dauer:** ~60 Minuten (3 Themen × 20 Minuten) **1. Was ist Coding und Programmierung (20 minutes)** **Ziel:** • Verstehen, was Programmierung und Coding im Alltag bedeutet **Hauptpunkte:** • Definition von Codierung und Programmierung • Unterschied zwischen Programmiersprachen und Beispiele (z. B. Python, JavaScript) • Beispiele für Codierungsanwendungen aus der Praxis (Apps, Websites, Spiele) **Aktivität:** • Diskussionsrunde & Brainstorming: Wo findest du Programmierung und Coding im Alltag? • Video: Inhalt: Bedeutung des Programmierens in der Technik

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.

Modulplan	
	Trainingsmethoden: • Vortrag mit visuellen Hilfsmitteln (Folien, Videos) • Interaktive Diskussion
	2. Grundbegriffe und Bausteine der Programmierung (20 Minuten) **Ziel:** • Introduce fundamental programming concepts like variables, data types, and basic control structures. • Einführung in grundlegende Programmierkonzepte wie Variablen, Datentypen und grundlegende Kontrollstrukturen. **Wichtige Themen:** • Variablen und Datentypen (Zahlen, Text) • Grundlegende Operatoren und Ausdrücke • Einfacher Kontrollfluss: if-else-Anweisungen und Schleifen (wenn-dann)

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.

Modulplan	

- Das Konzept von Algorithmen und Schritten zur Problemlösung

Aktivität:

- Praktische Übung: Erstellen einfacher Pseudocodes oder Flussdiagramme für alltägliche Aufgaben (z. B. ein Sandwich zubereiten, sich für die Schule fertig machen).

Lehrmethode:

- Demonstration mit Live-Coding/Beispielen
- Geführte Übungsaufgaben, Gruppenarbeit

3. Schreibe dein erstes Programm (20Minuten)

Ziel:

- Die Schüler sollen in die Lage versetzt werden, ihr erstes einfaches Programm zu schreiben und auszuführen.

Wichtigste Themen:

- Einführung in eine anfängerfreundliche Programmierumgebung (z. B. Scratch, Code.org, Python IDLE)
- Aufbau eines einfachen Programms (Ausgabeanweisung, Eingabe, grundlegende Logik)
- Behebung häufiger Fehler

Aktivität:

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.

Modulplan	
	• Schrittweises Programmieren: Die Schüler schreiben ein Programm, das sie nach ihrem Namen fragt und sie begrüßt. • Peer-Review und Übungen zur Fehlerbehebung **Methoden:** • Live-Demonstration des Programmierens • Angeleitete Übungen mit Unterstützung und Feedback • Zusammenarbeit mit Gleichaltrigen
Material	**1. Visuelle Hilfsmittel und Präsentationen:** • Folien oder PowerPoint-Präsentation mit Schlüsselkonzepten • Kurze Videos, die Codierung und Programmierung erklären (optional) **2. Computer und Software:** • Computer oder Laptops für jede*n Teilnehmer*in oder pro 2 Personen • Eine für Anfänger geeignete Programmierumgebung (z. B. Scratch, Python IDLE, Code.org, Blockly) **3. Handouts und Arbeitsblätter:** • Pseudocode- oder Flussdiagrammvorlagen für Aktivitäten • Arbeitsblätter mit Übungen zu Grundkonzepten und Problemlösung • Spickzettel für gängige Programmiersyntax und Befehle **4. Interaktive Tools:** • Online-Programmierplattformen oder Websites zum Üben (z. B. Code.org, Khan Academy, Trinket) • Debugging-Übungen oder spielerische Programmieraufgaben **5. Zusätzliche Materialien:**

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.

Modulplan	
	• Whiteboard und Marker für Erklärungen und Diskussionen • Ausgedruckte Beispiele für einfache Programme und Algorithmen • Notizbücher oder Papier für Entwürfe oder Brainstorming

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.

MODULINHALT

Inhalt	Beschreibung	Content
Inhalt Einheit	**Digitale Kompetenz**	**Digital literacy** means having the knowledge and skills to use digital tools like design software and programming languages to create apps, websites, and online content. It includes understanding how digital systems work and being able to build and design simple digital products. **Key Digital Terms:** • **Programming Language** – A language used to write instructions that a computer can understand. • **Code** – The set of instructions written in a programming language. • **Algorithm** – A step-by-step set of instructions to solve a problem. • **Variable** – A storage location in a program that holds data. • **Data Type** – The kind of data (e.g., integers, strings, floats) stored in a variable. • **Function / Method** – A block of code that performs a specific task and can be reused. • **Loop** – A sequence of instructions that repeats until a condition is met. • **Condition / If-Else** – A statement that performs different actions based on whether a condition is true or false. • **Debugging** – The process of finding and fixing errors or bugs in code.

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.

Inhalt	Beschreibung	Content
		• **Syntax** – The rules that define the structure of code in a programming language. • **Input** – Data that a program receives from the user or another source. • **Output** – Data produced by the program for the user. • **IDE (Integrated Development Environment)** – A software application that provides tools to write, test, and debug code. • **Bytecode** – An intermediate code that can be interpreted or compiled. • **Compilation** – The process of translating code into a machine language that a computer can execute. • **Machine Language** – The binary code that a computer's processor understands directly. • **Comments** – Notes written in code to explain what the code does; ignored by the computer. • **Syntax Error** – An error due to incorrect code that violates the language's rules. • **Logic Error** – An error where the code runs but produces incorrect results.

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.

Inhalt	Beschreibung	Content
Video Zusammenfassung	**Youtube Links**	1. An easy-to-understand explanation of what programming is and why it's important. 2. **"Introduction to Programming" – freeCodeCamp** https://www.youtube.com/watch?v=8oRjP8yj_w8 A beginner-friendly overview of programming fundamentals. 3. **"Learn Programming in 1 Hour" – Tech with Tim** https://www.youtube.com/watch?v=BoardJz_VRc A quick summary of core programming concepts designed for beginners. 4. **"What is a Programming Language?" – Computer Science Gaming** https://www.youtube.com/watch?v=7sA8vfQWWvg Short explanation of programming languages and how they work.

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.

Aktivität		**Gruppenaktivität** **Ziel:** ● Die TN dazu anregen, grundlegende Programmierkonzepte wie Sequenzen, Bedingungen und Schleifen anzuwenden, indem sie einen einfachen, schrittweisen Algorithmus für eine reale Aufgabe entwerfen. **Aktivität-Beschreibung:** 1. Teilen in kleine Gruppen ein (3-4 TN pro Gruppe). 2. Jede Gruppe wählt oder erhält eine einfache Alltagsaufgabe (z. B. ein Sandwich zubereiten, Zähne putzen, sich für die Schule fertig machen). 3. Die TN arbeiten gemeinsam daran, einen detaillierten Schritt-für-Schritt-Algorithmus oder ein Flussdiagramm zu erstellen, das die Aufgabe beschreibt, einschließlich Entscheidungspunkten (if-else-Bedingungen) und sich wiederholenden Schritten (Schleifen). 4. Jede Gruppe präsentiert ihren Algorithmus der Klasse und erklärt dessen Funktionsweise, wobei der Schwerpunkt auf den logischen Schritten und Entscheidungspunkten liegt. **Benötigte Materialien:** ● Große Blätter Papier oder Whiteboards, Marker oder ausgedruckte Vorlagen für Flussdiagramme.

		• Handouts mit grundlegenden Algorithmus-Symbolen und Richtlinien für Flussdiagramme (optional). **Lernziel:** • Die TN zeigen, dass sie die Programmierlogik verstehen und wissen, wie reale Aufgaben in codeähnliche Anweisungen zerlegt werden können. • Sie entwickeln Fähigkeiten zur Problemlösung und Zusammenarbeit.
		2. Rollenspiel **Ziel:** • Den TN helfen, Programmierkonzepte klar zu erklären und die reale Kommunikation beim Programmieren zu simulieren. Außerdem wird ihr Verständnis für Programmierschritte und Entscheidungsfindung gestärkt. **Aktivität-Beschreibung:** 1. Die TN bilden Paare; ein TN fungiert als **„Coding Coach"** und der andere als „Anfänger", der lernen möchte, wie man eine einfache Aufgabe programmiert (z. B. ein Sandwich zubereiten, Zähne putzen). 2. Der „Anfänger" bittet den „Coding Coach", ihm Schritt für Schritt zu erklären, wie er die Aufgabe mithilfe eines „Coding"-Prozesses erledigen kann.

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.

		3. Der „Coding Coach" muss den Prozess in einfachen, klaren Anweisungen beschreiben, einschließlich Entscheidungspunkten und Wiederholungen, so als würde er einem Anfänger beibringen, wie man wie ein Programmierer denkt. 4. Nach Abschluss der Erklärung werden die Rollen getauscht. 5. Nachdem beide Rollen gespielt wurden, diskutieren die Schüler die Bedeutung klarer Anweisungen und logischen Denkens beim Programmieren. **Material:** ● Keine, nur Raum für Rollenspiele. ● Geben Sie den Schülern optional Beispielaufgaben oder Schritte an die Hand, um ihnen den Einstieg zu erleichtern. **Lernziel:** ● Die TN üben, reale Aufgaben in logische, schrittweise Anweisungen zu übersetzen. ● Sie entwickeln Fähigkeiten, technische Konzepte klar zu erklären und aktiv zuzuhören.
		3. Beispiele aus dem wirklichen Leben **Ziel:**

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.

		• Den Schülern helfen zu verstehen, wie man eine tägliche Routine in klare, logische Schritte unterteilt, ähnlich wie beim Programmieren. Diese Aktivität fördert die Fähigkeiten zur Problemlösung, Reihenfolgeplanung und Entscheidungsfindung. **Aktivitätsbeschreibung**: 1. Bilde Paare; ein TN spielt den „Programmierer" (die Person, die die Anweisungen entwirft), und der andere ist der „Benutzer" (die Person, die die Anweisungen befolgt). 2. Der „Programmierer" soll eine typische Morgenroutine beschreiben (z. B. „Sich für die Schule fertig machen"), als würde er ein einfaches „Programm" oder eine Reihe von Anweisungen erstellen, die befolgt werden müssen. o Die TN sollten Schritte wie Aufwachen, Zähneputzen, Kleidung auswählen, Frühstücken usw. einbeziehen. o Sie sollten Entscheidungspunkte (z. B. „Wenn es kalt ist, zieh eine Jacke an") und Wiederholungen (z. B. „Zähne zweimal putzen") einbauen. 3. Der „Benutzer" hört zu und führt die Routine dann genau wie beschrieben aus.

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.

		4. Wechseln Sie die Rollen, damit beide Schüler üben können, Anweisungen zu geben und zu befolgen. 5. Abschlussbesprechung: Was war beim Verstehen und Befolgen der Anweisungen leicht oder schwierig war und inwiefern dies mit der Aufteilung von Problemen in einzelne Schritte beim Programmieren zusammenhängt. *Benötigte Materialien:* ● Es werden keine besonderen Materialien benötigt, lediglich Platz für Rollenspiele. ● Optional: Stellen Sie eine Checkliste oder ein Beispiel für eine Routine zur Verfügung, um den Schülern bei der Vorbereitung zu helfen. **Lernziel:** ● Die TN lernen, wie sie alltägliche Routinen in eine Abfolge von Anweisungen umsetzen können, ähnlich wie beim Programmieren. ● Sie verbessern ihre Kommunikationsfähigkeiten und ihr Verständnis für bedingte Logik und Abläufe.
Prüfung	**Quiz**	**Markiere die richtige Antwort:** **Was ist ein Programm?**

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.

		A) Eine Art von Hardware, die in Computern verwendet wird. B) Ein Spiel, das auf einem Computer gespielt wird. C) Eine Reihe von Anweisungen, die ein Computer befolgt, um eine Aufgabe auszuführen.✓ D) Ein Gerät, das Daten speichert. **2. Welches der folgenden Beispiele ist eine Variable in der Programmierung?** A) Ein Befehl zum Drucken von Text auf dem Bildschirm. B) Ein Speicherort, an dem Daten wie Zahlen oder Text gespeichert werden.✓ C) Eine Meldung, die erklärt, was der Code tut. D) Eine Art von Programmiersprache. **3. Was macht eine Schleife in einem Programm?** A) Sie prüft, ob eine Bedingung wahr oder falsch ist. B) Sie wiederholt eine Reihe von Anweisungen mehrmals. ✓ C) Sie speichert Daten für die spätere Verwendung. D) Sie sendet Daten an einen anderen Computer.

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.

		4. Warum ist Debugging beim Programmieren wichtig? A) Um dem Programm neue Funktionen hinzuzufügen. B) Um Fehler oder Bugs im Code zu identifizieren und zu beheben. ✓ C) Um das Programm schneller laufen zu lassen. D) Um Code in verschiedene Sprachen zu übersetzen.

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.