2.10 Programmieren: Unterstützung bei ausgewählten Aufgaben

Programmieren lernen, ohne dass die KI den Code schreibt

Fach: Programmieren · **Klassenstufe:** 9 (14–15 Jahre) · **Dauer:** 1 Doppelstunde (90 Min.) · **KI:** Unterstützung **MIT** KI

KI-bezogene Kompetenzen (DUAL.AI.TEACHer-Framework) – Lehrkräfteebene	**Fachspezifische Lernziele – Ebene der Schülerinnen und Schüler**
AF-TL-2b *(AI foundations and applications × Teaching & Learning, Stufe 2 – Reflektierte Umsetzung)*: „Lehrkräfte können Unterrichtselemente so gestalten, dass sie KI-Einsatz mit klaren pädagogischen Rollen und Strategien zur Überprüfung der Ausgabeelemente einbetten.“ **AF-FC-2b** *(AI foundations and applications × Facilitating Learners' (AI) Digital Competence, Stufe 2 – Reflektierte Umsetzung)*: „Lehrkräfte können Anleitungen umsetzen, die Lernenden helfen, praktische KI-Fähigkeiten zu entwickeln, wobei die Aufgaben an das Vorwissen der Lernenden und die verfügbaren KI-Werkzeuge angepasst werden.“	Am Ende der Stunde können die Schülerinnen und Schüler: • KI für wiederkehrende oder unterstützende Aufgaben beim Programmieren einsetzen, etwa zum Einrücken des Codes oder zum Schreiben von Kommentaren • ein KI-Modell um Verbesserungsvorschläge für den Code bitten, ohne den Code selbst erzeugen zu lassen, und so den eigenen Kenntnisstand selbst einschätzen • ein KI-Modell kritisch bewerten, wenn sie es fragen, wie es den vorgelegten Code optimieren würde

> **Kernbotschaft:** Wer die Ergebnisse eines Modells bewerten will, muss die Grundlagen des Programmierens kennen. Schülerinnen und Schüler, die mit dem Programmieren beginnen, sollten KI-Modelle daher für unterstützende und wiederkehrende Aufgaben rund um das Programmieren nutzen, nicht aber, um Code erzeugen zu lassen. Sie können das Modell außerdem nach möglichen Verbesserungen im Code fragen und so ihre eigenen Programmierfähigkeiten

> überprüfen.

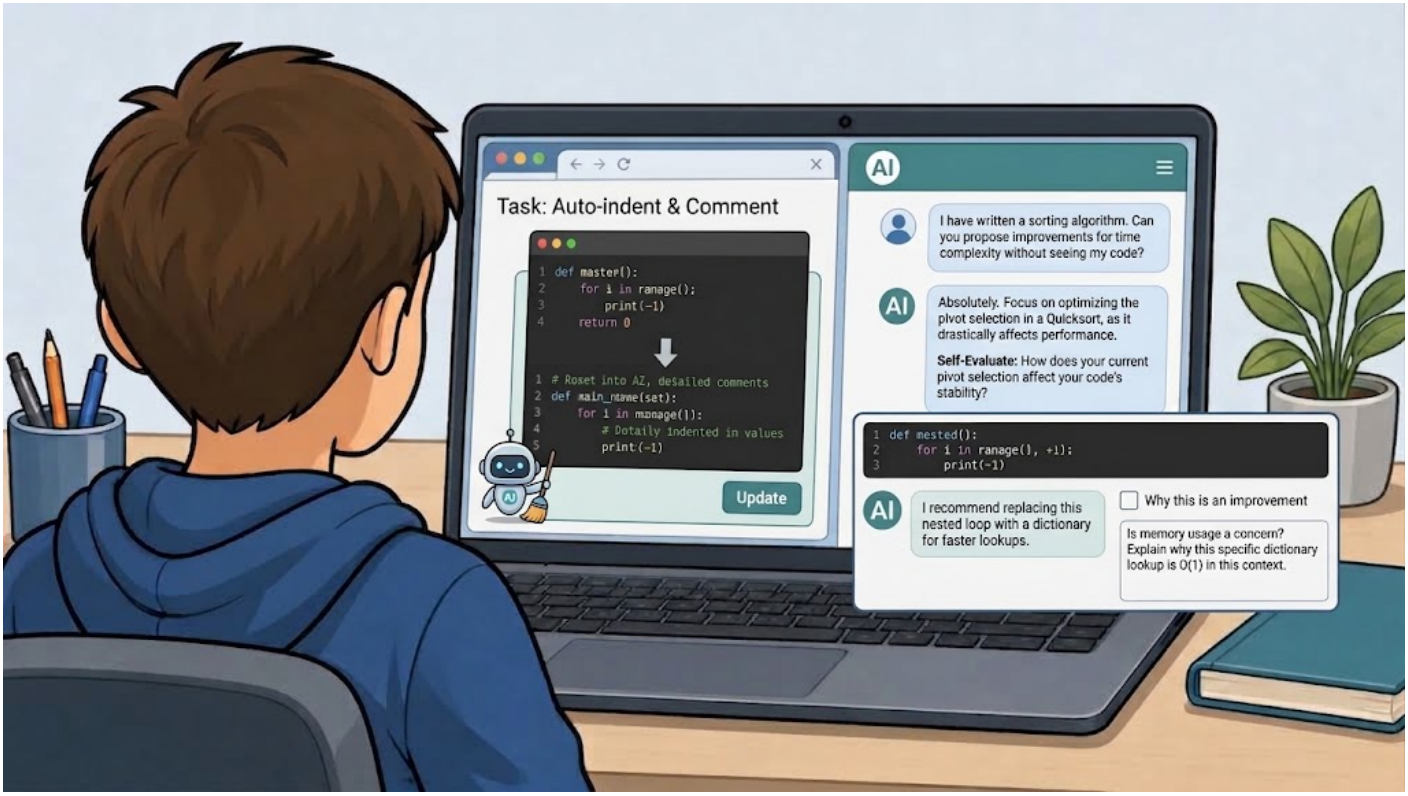

Abb. 1. Bild erzeugt mit Gemini (3.6 Thinking), 8. September 2026.

Inhalt

Der Programmierunterricht verändert sich in einem Umfeld, in dem KI-Systeme Code erzeugen, erklären und überprüfen können. Diese Werkzeuge eröffnen neue Lernmöglichkeiten, bringen aber auch eine wichtige pädagogische Herausforderung mit sich: Die Schülerinnen und Schüler brauchen ausreichende Programmierkenntnisse, um beurteilen zu können, ob ein KI-erzeugter Vorschlag korrekt, effizient oder für die gestellte Aufgabe überhaupt relevant ist. Ohne ein grundlegendes Verständnis von Programmierkonzepten übernehmen sie möglicherweise fehlerhafte Lösungen, übersehen Fehler oder erkennen bessere Alternativen nicht. Deshalb bleibt das Erlernen der Programmiergrundlagen unverzichtbar, auch wenn KI-Werkzeuge zur Verfügung stehen.

In einführenden Programmierkursen sollte KI als Lernunterstützung und nicht als Ersatzprogrammiererin positioniert werden. Die Schülerinnen und Schüler können KI sinnvoll für wiederkehrende oder einfache Aufgaben nutzen, die den kognitiven Aufwand des Programmierenlernens nicht verringern. Dazu gehören das Formatieren und Einrücken von Code, das Erzeugen von Dokumentationskommentaren, das Erklären von Fehlermeldungen, das Zusammenfassen von Syntaxregeln oder Hilfen dabei, den Zweck bestimmter Programmierkonstrukte zu verstehen. Diese Einsatzformen erlauben es ihnen, die Aufmerksamkeit auf die zugrunde liegende Logik, die Algorithmen und die Problemlöseprozesse zu richten, die für Programmierkompetenz zentral sind.

Eine weitere produktive Nutzung von KI ist das Code-Review. Statt ein KI-Modell um eine Lösung zu bitten, schreiben die Schülerinnen und Schüler zuerst ihren eigenen Code und lassen das Modell anschließend mögliche Verbesserungen benennen. Die KI schlägt vielleicht klarere Variablennamen, eine bessere Codestruktur, weniger Redundanz, bessere Lesbarkeit oder alternative Vorgehensweisen vor, die die Schülerinnen und Schüler eigenständig bewerten können. Damit wird die KI zu einer Feedback-Partnerin: Die Lernenden können ihre eigenen Überlegungen mit externen Vorschlägen vergleichen und Stärken und Schwächen ihrer Lösungen einschätzen.

KI-erzeugtes Feedback sollte jedoch nicht unkritisch übernommen werden. Modelle empfehlen mitunter Änderungen, die unnötig oder ineffizient sind oder auf falschen Annahmen über den Zweck des Codes beruhen. Ein zentrales Ziel dieser Stunde ist es deshalb, die Fähigkeit zu entwickeln, KI-Vorschläge kritisch zu bewerten. Die Schülerinnen und Schüler lernen, ein KI-Modell nicht nur danach zu fragen, welche Verbesserungen es vorschlägt, sondern auch danach, warum es diese Änderungen für sinnvoll hält. Indem sie die Begründung des Modells mit etablierten Programmierprinzipien vergleichen, können sie beurteilen, ob die vorgeschlagene Optimierung den Code tatsächlich verbessert oder lediglich eine andere Entwurfsentscheidung darstellt.

Diese Stunde behandelt KI als Werkzeug, das Reflexion, Selbsteinschätzung und die Verbesserung der Codequalität unterstützt – und nicht als Werkzeug zur Codeerzeugung. Die Schülerinnen und Schüler üben, eigene Programme zu schreiben, KI für unterstützende Aufgaben einzusetzen, Rückmeldung zu ihrer Arbeit einzuholen und die Qualität der erhaltenen Antworten kritisch zu analysieren. Das übergeordnete Ziel ist, dass sie sowohl bessere Programmiererinnen und Programmierer als auch informiertere Nutzerinnen und Nutzer von KI-Systemen werden.

Unterrichtsverlauf

Phase	Zeit	Aktivität
Einführung	10 Min.	Die Lehrkraft zeigt zwei kurze Codelösungen für dasselbe Problem: eine von einer Schülerin oder einem Schüler geschriebene und eine von einem KI-Modell erzeugte. Die Klasse diskutiert, welche Lösung klarer, effizienter oder leichter verständlich erscheint.
Input	10 Min.	Die Lehrkraft stellt angemessene Einsatzformen von KI im Programmierunterricht vor und betont die Unterscheidung zwischen unterstützenden Aufgaben, Code-Review und Codeerzeugung.
Erkundung	15 Min.	Die Schülerinnen und Schüler bearbeiten paarweise eine kleine Programmieraufgabe und bestimmen, welche Teilaufgaben sinnvoll an eine KI-Assistenz abgegeben werden können und welche in ihrer eigenen Verantwortung bleiben sollten.
Verbesserungszyklus	35 Min.	Die Schülerinnen und Schüler schreiben ihre eigene Lösung und durchlaufen 2 bis 3 Runden von schreiben → fragen → bewerten → überarbeiten, wobei sie KI nur für unterstützende Aufgaben und Verbesserungsvorschläge einsetzen.

Reflexion	20 Min.	Klassengespräch über Nutzen und Grenzen von KI-Feedback mit dem Schwerpunkt darauf, wie die Schülerinnen und Schüler entschieden haben, ob Vorschläge wertvoll waren oder nicht.

Leitfragen: Komplexität schrittweise aufbauen

Die Schülerinnen und Schüler bearbeiten während des 35-minütigen Verbesserungszyklus so viele Runden, wie die Zeit zulässt. Jede Runde folgt demselben Vier-Schritte-Ablauf: schreiben → fragen → bewerten → überarbeiten

1. **Runde 1 — KI für unterstützende Aufgaben nutzen**
   - Schreiben: Bearbeiten Sie eine kurze Programmieraufgabe eigenständig.
   - Fragen: Bitten Sie die KI um Unterstützung bei einer nicht-programmierenden Aufgabe rund um den Code, zum Beispiel:
     - die Einrückung verbessern,
     - Kommentare erzeugen,
     - eine Fehlermeldung erklären,
     - beschreiben, was eine Funktion tut.
   - Bewerten: Beschreibt die Erklärung der KI den Code zutreffend? Sind die erzeugten Kommentare nützlich und verständlich?
   - Überarbeiten: Übernehmen Sie hilfreiche Änderungen und stellen Sie dabei sicher, dass die Programmlogik vollständig Ihre eigene Arbeit bleibt.
2. **Runde 2 — Verbesserungsvorschläge einholen**
   - Schreiben: Sehen Sie Ihr fertiges Programm durch und markieren Sie Stellen, die sich verbessern ließen.
   - Fragen: Holen Sie Verbesserungsvorschläge ein, ohne die KI den Code neu schreiben zu lassen. Zum Beispiel:

     > „Nenne drei Möglichkeiten, die Lesbarkeit und Wartbarkeit dieses Programms zu verbessern, aber liefere keinen Ersatzcode.“

   - Bewerten: Welche Vorschläge erscheinen wertvoll? Welche hätten kaum Auswirkungen auf das Programm?
   - Überarbeiten: Setzen Sie die Verbesserungen um, denen Sie zustimmen, und testen Sie, ob sich das Programm weiterhin korrekt verhält.
2. **Runde 3 — Optimierungsratschläge kritisch bewerten**
   - Schreiben: Wählen Sie einen Abschnitt Ihres Codes aus, der eine wiederkehrende Aufgabe erledigt oder Schleifen verwendet.
   - Fragen: Holen Sie Optimierungsvorschläge ein und verlangen Sie von der KI eine Begründung ihrer Empfehlungen.

     Beispiel-Prompt:

„Erkläre, wie dieser Code optimiert werden könnte und warum jede vorgeschlagene Optimierung Leistung oder Lesbarkeit verbessern würde. Liefere den optimierten Code nicht.“

- Bewerten: Sind die vorgeschlagenen Optimierungen tatsächlich sinnvoll? Welche sind klar begründet? Welche wirken unnötig oder schwer zu rechtfertigen?
- Überarbeiten: Entscheiden Sie, welche Empfehlungen Sie umsetzen, und begründen Sie Ihre Entscheidung.

Reflexionsfragen:

- Welche KI-Vorschläge waren für die Verbesserung Ihres Codes am nützlichsten?
- Gab es Vorschläge der KI, denen Sie nicht zugestimmt haben? Warum?
- Woran haben Sie festgemacht, ob eine Optimierung lohnenswert war?
- Wann ist KI im Programmierprozess am hilfreichsten?
- Welche Programmieraufgaben sollten weiterhin von den Schülerinnen und Schülern selbst erledigt werden?
- Warum sind Programmierkenntnisse nötig, um KI-erzeugtes Feedback zu bewerten?
- Wie können Schülerinnen und Schüler vermeiden, beim Programmierenlernen zu abhängig von KI-Werkzeugen zu werden?

Abschlussfrage (für die Reflexionsphase):

Wie viel Programmierwissen ist nötig, um zu beurteilen, ob ein KI-Vorschlag ein Programm tatsächlich verbessert – und welche Risiken entstehen, wenn Nutzerinnen und Nutzer sich auf KI-erzeugte Ratschläge verlassen, ohne den Code selbst zu verstehen?

Materialien

- Ein Laptop pro Schülerpaar mit einer Programmierumgebung
- Zugang zu einer KI-Chat-Assistenz oder einem lokal betriebenen Sprachmodell
- Eine kurze Programmieraufgabe, die dem Kenntnisstand der Schülerinnen und Schüler entspricht
- Ein Arbeitsblatt zum Zyklus schreiben → fragen → bewerten → überarbeiten
- Reflexionsfragen mit Fokus auf KI-Kompetenz und kritische Bewertung
- Optional Codebeispiele, die gute und schlechte Programmierpraxis veranschaulichen

---

Revision #1
Created 2026-09-17 16:04:30 UTC by Alexander
Updated 2026-09-17 16:04:31 UTC by Alexander