2021

Praxisleitfaden
Ein Wissenschaftstransfer-Projekt
von Marcel Mösch & Raffael Meier

HAUSAUFGABEN INFORMATIK PROGRAMMIEREN

Warum dieses Faltblatt?

Hausaufgaben im Bereich „Programmieren“ können den Lernerfolg erhöhen, motivieren und zusätzliche zeitliche Ressourcen schaffen.

Wie aber sollen Lehrpersonen Hausaufgaben im Bereich „Programmieren“ gestalten? Diese Broschüre gibt Ihnen einen Überblick über Qualitätsmerkmale und Möglichkeiten.

Wissenschaftliche, ausführliche Informationen zum Thema und zu den vorliegenden Empfehlungen sowie Literaturangaben finden Sie unter

https://www.informatikhausaufgaben.ch

Herzlichst, Ihre Autoren
Marcel Mösch & Raffael Meier

Was sind Hausaufgaben? Warum sind sie sinnvoll?

Hausaufgaben sind eine Erweiterung des Unterrichts. Die Lernenden arbeiten dabei selbstständig (OHNE fremde Hilfe).

Ziele

- individuelle Auseinandersetzung mit einem Lerngegenstand
- Fach- und Selbstkompetenzen (z.B. Eigenverantwortung, Planung etc.) festigen/erweitern
- Erziehungsberechtigten einen Einblick in den Unterricht gewähren

Zeitpunkt von Hausaufgaben

- Vorbereitung des Unterrichts
  Hausaufgaben als Vorbereitung helfen den Lernenden in einem Themenfeld anzukommen. Dabei können einzelne Inhalte z.B. mit Lernvideos, Beispielen und Leitfragen bearbeitet werden. (Die Lernenden wissen dann schon einiges, wenn der Unterricht in der Klasse stattfindet.)
  ACHTUNG: Diese Methode funktioniert nicht automatisch, sondern muss strukturiert eingeführt und geübt werden.
- Nachbereitung des Unterrichts
  Hausaufgaben als Nachbereitung dienen dem Wiederholen, Üben oder Transfer der im Unterricht behandelten Inhalte.

Wann sind Hausaufgaben gut?

Hausaufgaben sollen das Interesse am Lerngegenstand wecken!

- *Allgemeine Aspekte*
  Name der Aufgabe, Ausgangslage, Auftrag / Vorgehen, Ziel / Erwartung, Zeitvorgabe, Benötigtes Material, Bemerkungen
- Bezug der Lebenswelt der Lernenden herstellen
- Orientierung an Kompetenzen und Lernzielen
- herausfordernd und trotzdem machbar
- Lernende brauchen individuelle und sachliche Feedbacks zu erledigten Hausaufgaben. Feedbacks wirken kompetenzfördernd, motivierend und stärken die Beziehung zwischen Lehrperson und Kind.
- Das Gehirn entlasten - Dos and Don'ts von Cognitive Load Theory:
  immer gleiches Layout verwenden;
  immer gleiche Auftragsstruktur verwenden;
  einfache Sprache verwenden;
  Zweideutigkeiten vermeiden;
  Lernschritte mit Hinweisen unterstützen (methodisches Vorgehen z.B. in Form eines phasenhaften Lernprozessdiagramms mit entsprechenden Lernschritten / fachliche Tipps z.B. in Form von Beispielen)
- differenzierte Hausaufgaben (Zugang + Schwierigkeit) = es müssen nicht alle Lnd zwingend das Gleiche machen
- kurz, dafür oft / regelmässig
- Umfang von Hausaufgaben (Dauer)
  Die Schuleinheit (z.B. Gemeinde, Kanton) definiert, wie viel Hausaufgaben pro Woche/Tag aufgegeben werden dürfen. Eine einzelne Aufgabe sollte folgende Zeiten nicht überschreiten:
  3.-4. Klasse: 10 min (inkl. Arbeitsvorbereitung)
  5.-6. Klasse: 20 min (inkl. Arbeitsvorbereitung)
- Gestaltungsmerkmale guter Hausaufgaben (mediale Präsentation):
  einfache und übersichtliche Gestaltung;
  Wichtiges markieren und Überflüssiges weglassen

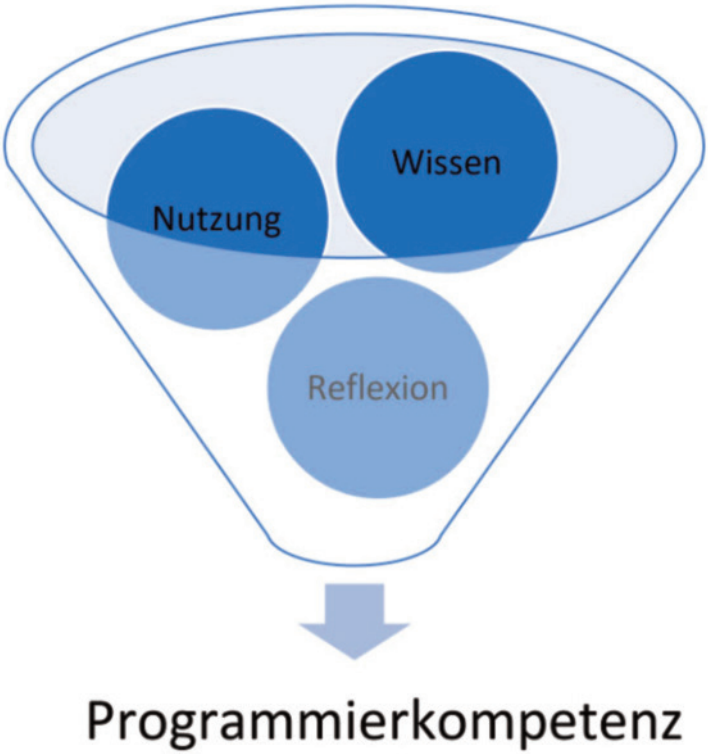
Was bedeutet Programmierkompetenz?

Programmieren ist ein Teil der Informatikkompetenz. Durch Programmieren werden Befehle in eine Sprache übersetzt, die ein Computer verstehen und ausführen kann.

Die verschiedenen Kompetenzstufen zu Programmieren sind im Lehrplan 21 unter MI.2.2 zu finden.

Programmierkompetenz umfasst folgende Bereiche:

- Wissen: Welche Begriffe gibt es? Wie funktionieren Programme?
- Nutzung: Was passiert in einem Programm? Wie können Programme entwickelt werden?
- Reflexion: Was bewirken Programme? Was können/wissen die Lernenden schon, was noch nicht?



Programmierkompetenz - Beispielaufgaben

Hausaufgaben sind eine Erweiterung des Unterrichts.
Die Lernenden arbeiten dabei selbstständig (OHNE fremde Hilfe).

Wissen

Über den QR-Code kommst du zu einem kurzen Erklärvideo.

? Schau das Video. Beantworte folgende Fragen. (maximal 3 Sätze pro Frage)
? Was macht ein Programm?
? Wer macht die Programme?
? Was ist eine Programmiersprache?

Nutzung

Du siehst hier ein kurzes Programm.

? Erkläre, was das Programm macht. (maximal 5 Sätze)

Reflexion

Wirkung
Computerprogramme kommen an vielen Orten zum Einsatz. Die Kasse im Supermarkt läuft zum Beispiel mit einem Programm.

? Beschreibe, was die Person an der Supermarktkasse alles machen müsste, wenn es keine Computerprogramme gäbe. (5 Sätze)

Eigenes Lernen
Du hast im Unterricht an einem Programm mit Schleifen gearbeitet.

? Beschreibe, was du verstanden hast. (3 Sätze)
? Beschreibe, wo du noch unsicher bist. (3 Sätze)

Was ist Computational Thinking (CT)?

«Computational Thinking ist der Gedankenprozess, der sowohl die Formulierung eines Problems als auch die Repräsentation der Problemlösung so darstellt, dass sie von Menschen oder durch Maschinen ausgeführt werden können.» (Jeannette Wing, Professorin für Informatik an der Carnegie Mellon University)

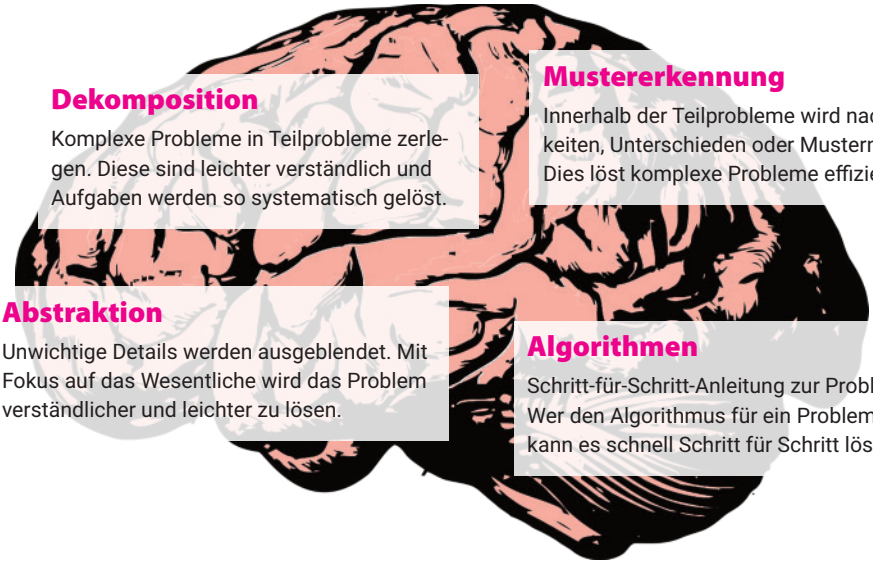
Wichtig ist in der heutigen Zeit, in der so viel von Copmutersystemen abhängt, (auch) so denken zu können, wie Informatiker, oder anders gesagt, Alltagsprobleme so zu formulieren und zu strukturieren, dass diese mit digitalen Maschinen bearbeitet werden könne - (selbst-)kritischer Umgang setzt grobes Verstehen voraus.

Computational Thinking hilft schon auf Primarschulebene den Schülerinnen und Schülern, Problemlösungsstrategien, die sie bereits unbewusst in ihrem Alltag anwenden, zu visualisieren und weiterzuentwickeln. Dies schafft eine wertvolle Basis, um später in der Berufswelt den heutigen Anforderungen gerecht zu werden.

«Computational Thinking» ist auch ein Denkstil, um Probleme zu lösen:

- Iterativ vorgehen: Abstraktion, Analyse, Automation

Die 4 Kern-Kompetenzen:



Dekomposition

Komplexe Probleme in Teilprobleme zerlegen. Diese sind leichter verständlich und Aufgaben werden so systematisch gelöst.

Mustererkennung

Innerhalb der Teilprobleme wird nach Ähnlichkeiten, Unterschieden oder Mustern gesucht. Dies löst komplexe Probleme effizienter.

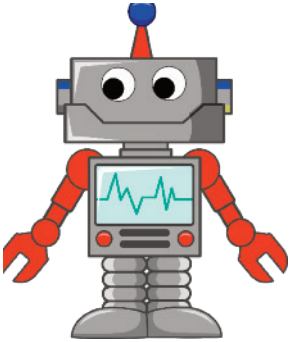
Abstraktion

Unwichtige Details werden ausgeblendet. Mit Fokus auf das Wesentliche wird das Problem verständlicher und leichter zu lösen.

Algorithmen

Schritt-für-Schritt-Anleitung zur Problemlösung. Wer den Algorithmus für ein Problem kennt, kann es schnell Schritt für Schritt lösen.

CT in der Praxis?



Sportunterricht einmal anders:

Ich lerne eine Sportart. Welche Ziele habe ich diese Saison? Wie erreiche ich sie?

Dekomposition

Coaching: Kenne ich die Regeln? Wo muss ich/mein Team mich verbessern? Wer ist in meinem Team? Welchen Stand will/mein Team ich erreichen?
Training: Was benötige ich für diesen Sport? Fitnesslevel (Geschwindigkeit, Ausdauer, Kraft, Beweglichkeit)? Technik? Taktik? Ausrüstung? Infrastruktur?
Wettkampf: Woraus besteht die Saison? Liga? Cup? Events? Wettkämpfe?

Mustererkennung

Coaching: Gute Technik und Bewegungsmuster/Abläufe entwickeln? Verschiedene Taktiken und Formationen entwickeln?
Training: Trainings strukturieren? Ausdauer, Erholung, Geschwindigkeit gleichmässig im Training verteilen/separieren um die Ausbeute zu maximieren?
Wettkampf: Den Gegner analysieren? Wer sind die Schlüsselspieler? Was können Sie besonders gut/schlecht? Welche Taktik nutzen sie? Mit welcher Taktik hat man gegen sie gewonnen? Wie habe ich mich vorbereitet, als ich besonders erfolgreich war?

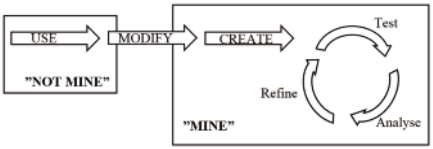
Abstraktion

Coaching: Erstelle einen Trainingsplan für das Jahr mit Trainingsphasen und wichtigen Wettkamp-/Eventdaten.
Training: Was ist mein langfristiger Trainingsplan? Ausserhalb der Saison: Fitness, Erholung? Vorsaison: Fitness aufbauen? Während der Saison: Fitness erhalten?
Wettkampf: Was sind meine Ziele? Liga: Aufstieg? Cup: Halbfinale erreichen? Persönliche Bestleistung /-zeit erreichen? In die erste Mannschaft wechseln?

Algorithmen

Coaching: Aufwärm-Ritual kreieren? Verschiedene Coachinginterventionen planen mit klarem Ziel: warm-up, Technik, Skills, Drills, Taktik und Ablauf?.
Training: Ein kurzfristiges wöchentliches Trainingsprogramm gestalten mit allen Übungen, Repetitionen und Kraft-/Ausdauerelementen.
Wettkampf: Alle Taktiken des Teams in ein Playbook schreiben? Eln Verzeichnis der Taktiken der wichtigsten Gegener erstellen? Einen Wettkampkalender schreiben? Eine Jahresagenda erstellen?

Warum «Use, Modify, Create»?



Das «Use-Modify-Create»-Framework für mehr Lernerfolg: 3 Stufen erhöhen das Engagement von Lernenden. Sie erfahren den Lerngegenstand, indem sie vollständig gelöste Arrangements nachvollziehen, verstehen, nutzen und deren Wert reflektieren. Zweitens wird Bestehendes modifiziert - der Lerngegenstand wird „zum eigenen". Drittens ist ein Lösen von Aufgaben „from Scratch" möglich, die kreative Eigenleistung wird angeregt und keine vorgegebene Lösung oder Teile davon sind vorgegeben.

Aufgabentyp 1 - «Use»

Programmieren lernen mit bereits bestehenden Programme - sie werden studiert, verwendet, beobachtet oder analysiert. Lernende sind Konsumenten.

(Mögliche) Aufgabentypen:

- ☐ Programme verwenden. z.B. Spiele spielen, Code "erleben", Zusammenhänge zwischen Code und Ergebnis herstellen.
- ☐ Programme nachbauen (Code zu Code). Z. B. mit einer Anleitung ein Programm nachprogrammieren oder "Listing abtippen".
- ☐ Blöcke erkunden (Code zu Text). Welche Funktion haben die einzelnen Blöcke? Eignet sich bei der Einführung von unbekannten Blöcken.
- ☐ Programme lesen (Code zu Text). Kann auch von einem Blatt gelesen werden. Die Ausführung des Programms überprüft Vermutungen.
- ☐ Programme lesen und Fehler finden (Code zu Code): Fehler im Programm werden gesucht und markiert.
- ☐ Programme testen (Code zu Text). Testen, ob bei Ausführung des Programms eintrifft, was erwartet wurde. Randfälle (Eingabe von 0 oder unerwartete Werte und Datentypen) können auf Fehler getestet werden.

Leitfragen: Was passiert? Wie funktioniert es?
Hilfsmittel: Oft sind Fragen oder Prompts hilfreich.

Beispiel «USE»

Schau Dir das Programm genau an (ohne Computer).

? Was passiert, wenn Du die Taste „d“ drückst?
? Was passiert, wenn Du die Taste „a“ drückst?
? Was passiert, wenn Du die Taste „w“ drückst?
? Was passiert, wenn Du die Taste „s“ drückst?
? Warum passiert das? Beschreibe in drei Sätzen.
? Welche Blöcke sind für die Taste „d“ und die Reaktion zuständig?
? Was musste die Programmiererin in den Blöcken anpassen, damit sie wie oben funktionieren (genau so liegen sie in Scratch nämlich nicht bereit)?
? Was passiert, wenn Du die Taste „x“ drückst?

oder

Führe das Programm in Scratch aus.

? Was passiert, wenn du die Tasten „d“, „a“, „w“ oder „s“ drückst?
? Warum sind die Blöcke in vier Gruppen getrennt und nicht alle miteinander verbunden?
? Warum ist der orange Block immer oberhalb des blauen?
? Was bedeutet die Zahl 10? Was bedeutet -10 (minus 10)?
? Kannst Du die Blöcke auf Deutsch übersetzen? Was heissen sie auf Deutsch?

Aufgabentyp 2 - «MODIFY»

Modify

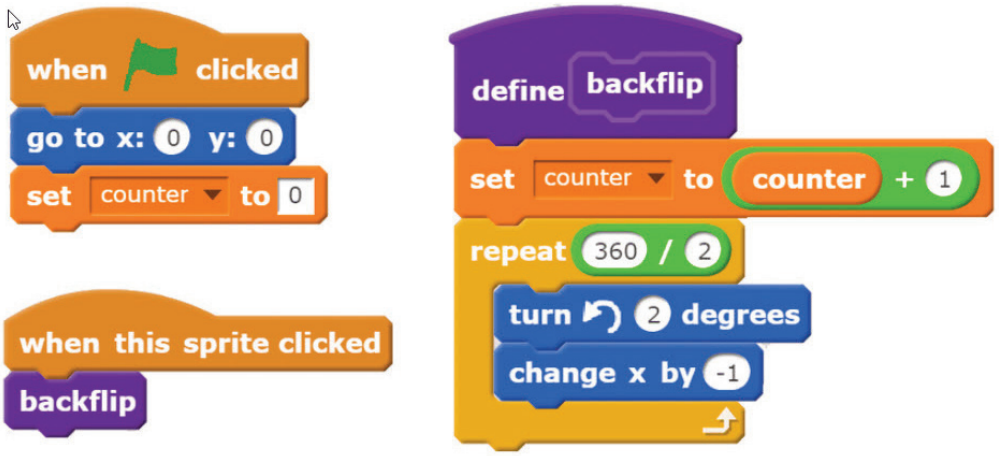
Lernende lernen programmieren mittels bereits bestehender Programme - nun, im zweiten Schritt in dem diese geändert werden. Die Lernenden sind “Editoren”.
Meist werden anfänglich triviale Aspekte wie Farben, Formen, Schrittweiten, etc. geändert; später auch ganze Abläufe, Elemente, Interaktionen, usw.

(Mögliche) Aufgabentypen:

- Programme debuggen (Code zu Text und Code). Fehler werden gesucht und das Programm korrigiert.
- Programme erweitern (Code zu Text und Code). Ausnahmen werden getestet und das Programm so optimiert, dass es in jedem Fall ausgeführt werden kann.

Leitfragen: Wo ändere ich es? Wie ändere ich es?
Hilfsmittel: Ermutigen, auszuprobieren. Neu starten, wenn das Programm „kaputt geht“. Motivation. Wertschätzung ermöglichen. Fortschritte betonen. Misserfolge ignorieren. Analogie Skateboarden: umfallen und wieder aufstehen.

Beispiel «MODIFY»



Führe das Programm in Scratch aus.

? Was passiert?

? Was passiert, wenn Du den Ball anklickst?

? Ändere die Zahlen 0 und 0 im blauen Block auf 50 und 100 ab. Was passiert? Was bedeuten die Zahlen?
Was bedeutet also x? Und was bedeutet y?

? Verändere einmal die Zahl 2 im Block rechts „turn 2 degrees“ auf 1 ab. Was passiert? Du kannst nun eine ANDERE Zahl anpassen, damit das gleiche passiert wie am Anfang. Findest Du heraus, welche Zahl und auf welchen Wert Du sie setzen musst?

? Was bewirkt das „change x by -1“? Probiere Werte aus.

? Kannst Du einbauen, dass beim Klicken auf den Ball ein Geräusch ertönt?

Aufgabentyp 3 - «CREATE»

Create

Lernende lernen programmieren anfänglich mittels bereits bestehender Programme - in dem diese programmatisch erweitert werden (nicht nur Parameter ändern). Später auch, in dem Vorgaben gemacht werden, die die Lernenden mittels eigenen Programmen umsetzen/erfüllen. Sie sind “Planer” und “Produzenten”.

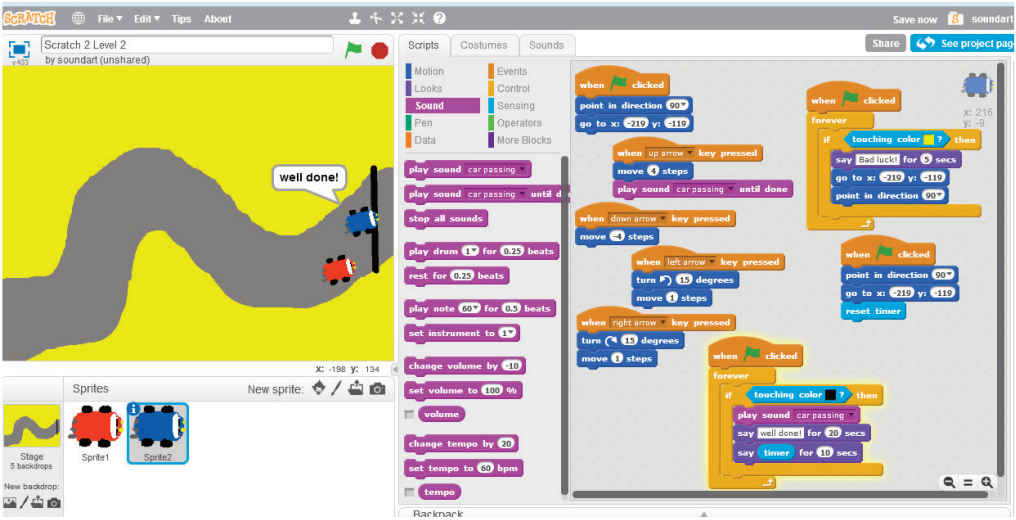
(Mögliche) Aufgabentypen:

- ☐ Programme erweitern, so dass neue Elemente (wie z.B. Sprites, Spielfiguren, etc.) oder auch Funktionen, Interaktionen, Features usw. hinzukommen. Die Vorgaben können in Bild-, Video- oder Textform vorliegen oder auch z.B. in Form von User-Stories (Kundenwünsche).
- ☐ Programmieren nach bildlichen Vorgaben (Bild zu Code). Das Produkt soll nach einer Bildergeschichte nachprogrammiert werden.
- ☐ Programmieren nach Vorgaben als Video (Video zu Code). Das Endprodukt wird als Video präsentiert. Die Schülerinnen und Schüler überlegen sich, wie sie die Vorgaben erfüllen können und erstellen das Programm dazu (Scratch-Challenges in der Klasse, über Klassen hinweg).
- ☐ Programmieren nach textlichen Vorgaben (Text zu Code). Die Schülerinnen und Schüler programmieren nach einer Text-Vorgabe. Bonus-Ziele können zur Differenzierung eingesetzt werden.
- ☐ Programmieren eigener Ideen nach textlichen Vorgaben (Text zu Code). Eine Grundidee wird vorgegeben, welche mit individuellen Umsetzungen erreicht werden kann.
- ☐ Programmieren eigener Ideen ohne Vorgaben. Eigene Ideen werden umgesetzt, Plots für Software entwickelt, Sprites und Hintergründe kreiert und ganze Spiele erfunden und umgesetzt.

Leitfragen: Was könnte ich noch einbauen? Wie bauen ich es ein? Wie stelle ich sicher, dass es in jedem Fall funktioniert?

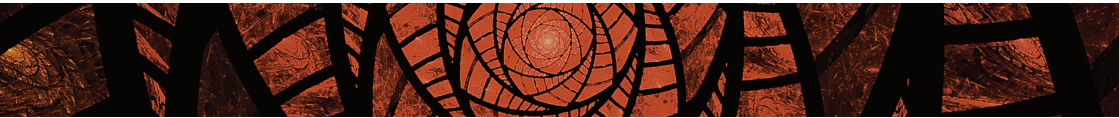
Hilfsmittel: Ein Abgleich mit den Vorgaben ist hilfreich, auch regelmässiges Feedback, Präsentationen, oder Paar-Programmierung. Alle Schwierigkeitsgrade ermöglichen. Nicht auf Scaffolding und formative Feedbacks verzichten

Beispiel «CREATE»



Führe das Programm in Scratch aus.

- ? Verändere die Spielmusik und die Geräusche, so dass sie Dir noch besser gefallen. Könnten alle Geräusche zu einem gleichen Thema passen (z.B. Safari)?
- ? Gestalte den Hintergrund der Rennbahn wie in der Arktis.
- ? Kannst Du ein drittes (grünes) Auto einbauen?
- ? Kannst du ein Auto so ändern, dass es ein Kamel ist und schneller reiten kann, als das andere Auto?
- ? Baue ein Zeitanzeige, die nach dem Start die Sekunden zählt, wie lange die Autos schon fahren. Die Zeit soll am Ziel stoppen.
- ? Hast Du eine Idee, wie man das Spiel noch spannender gestalten kann? Welche zusätzlichen Elemente könntest Du noch einbauen?
- ? Lass Dir eine Aufgabe von Deinem Banknachbarn stellen und setze sie um. Bist Du gespannt, ob Du es in einer Stunde schaffst? Stelle dafür einen Timer.



Wie werden Hausaufgaben geplant und reflektiert?

Prozessschritt	Leitfragen für die Lehrperson
Analysieren	*Fachliche und überfachliche Voraussetzungen* Was wissen die Lernenden bereits zum Thema? Welchen Lebensweltbezug haben die Lernenden? Welche Kompetenzstufen sollen erreicht werden? Welche Inhalte und Zusammenhänge können von den Lernenden selbstständig erarbeitet werden? Welche Arbeitsmethoden kennen die Lernenden bereits? Wie verhalten sich die Lernenden bei Unsicherheiten zu den Hausaufgaben während der Realisierung? Welche Hilfestellungen sind nötig? *Personale und soziale Voraussetzungen* Welche entwicklungsbedingten Voraussetzungen müssen bedacht werden? Welche Werte und Haltungen prägen den Lernprozess? In welcher Umgebung arbeiten die Lernenden? *Strukturelle Voraussetzungen* Welche Infrastruktur steht den Lernenden zur Verfügung? Welche Regeln müssen beachtet werden? Wie viel Zeit kann für Hausaufgaben eingesetzt werden? Welche Zeitfenster können genutzt werden?
Entscheiden	Welche Ziele sollen die Lernenden erreichen? Welche Differenzierung der Lernziele wird bei den Lernenden vorgenommen? Sind diese Ziele mit dem Lehrplan konform? Sind diese Ziele eine Herausforderung und gleichzeitig realistisch?
Entwerfen	Wo befinden sich die Lernenden im Lernprozess? Welche Lernaufgaben eignen sich für die Hausaufgabe? Welche Methoden eignen sich für die Hausaufgabe? Welche handlungsorientierten und/oder spielerischen Formen eignen sich für die Hausaufgabe? Auf welche Weise werden die Lernergebnisse überprüft?
Realisieren	Wie werden die Hausaufgaben begleitet? Auf welche Weise korrigiert / bespricht die Lehrperson die Hausaufgaben? Was hat gut funktioniert? Warum? Was hat nicht funktioniert? Warum? Welche Massnahmen unterstützen die Hausaufgabenpraxis?
Reflektieren	Was hat gut funktioniert? Warum? Was hat nicht funktioniert? Warum? Welche Massnahmen unterstützen die Hausaufgabenpraxis?

QR-Code zum Dossier der kompetenzorientierten Unterrichtsplanung (PH Schwyz)

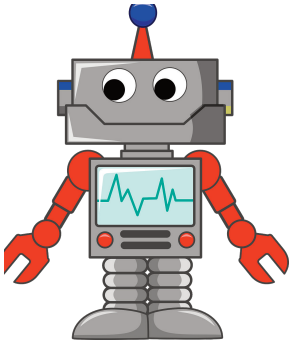
Wo stehen Lernende im Lernprozess? Welche Lernaufgaben eignen sich? Welche Fragen helfen sich zu orientieren?

PADUA-Modell	Lernaufgaben & Leitfragen für die Lernenden
Problemstellung	*Konfrontationsaufgaben* · *Wo finde ich Aspekte des Themas in meinem Leben?* · *Warum ist das Thema für mich interessant/wichtig/relevant?* · *Was weiss ich schon?* · *Was möchte ich noch wissen / können?*
Aufbau	*Erarbeitungsaufgaben* · *Wie funktioniert das? Welche Konzepte stehen dahinter?* · *Welche Begriffe brauche ich, um das Thema zu erklären?*
Durcharbeiten	*Vertiefungsaufgaben* · *Welche Gemeinsamkeit und Unterschiede bestehen zwischen verschiedenen Situationen?* · *Welche Verhaltensregeln lassen sich ableiten?*
Ueben	*Übungsaufgaben* · *Was muss ich machen, damit ein Teilaspekt funktioniert?* · *Welche Begriffe muss ich für eine Erklärung verwenden?*
Anwenden und Überprüfen	*Transfer- und Syntheseaufgaben* · Was muss ich genau machen, damit das Ganze funktioniert? *Formative und summative Beurteilungsaufgaben* · Wissen: Was weiss ich (noch nicht)? · Anwendung: Was kann ich (noch nicht)? · Vorgehen: Wie bin ich vorgegangen? Was eignet sich (nicht) für mich? Warum (nicht)?

QR-Code zu Erklärungen des PADUA-Modells (PH Luzern)

QR-Code zu den Lernaufgaben von Luthiger (PH Luzern)

Impressum



Autoren

Marcel Mösch
Raffael Meier

Dieses Faltblatt entstand als Leistungsnachweis
im Modul Wissenschaftstransfer
im Rahmen des Studiengangs
Master Fachdidaktik Medien & Informatik
an der Pädagogischen Hochschule Schwyz
im Juli 2021.
Dozent: Dr. Martin Hermida

www.informatikhausaufgaben.ch