WT_Faltblatt2_Faltblatt WT 01.08.2021 20:30 Seite 1

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

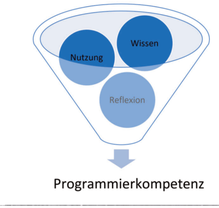
Was bedeutet Programmierkompetenz?

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)

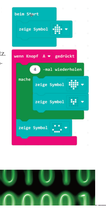
Reflexion

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)

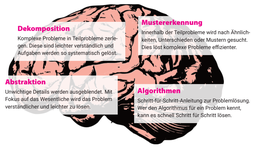
Wichtig ist in der heutigen Zeit, in der so viel von Computersystemen abhängt, (auch) so denken zu können, wie Informatiker, oder anders gesagt, Alltagsprobleme so zu formulieren und zu strukturieren, dass diese mit digitalen Maschinen bearbeitet werden können - (selbst-)kritischer Umgang setzt grobes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



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 Wettkämp-/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 / cmh erreichen? In die erste Mannschaft wechseln?

Algorithmen
Coaching: Aufwärm-Ritual kreieren? Verschiedene Coaching/interventionen 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? Ein Verzeichnis der Taktiken der wichtigsten Gegener erstellen? Einen Wettkampfkalender 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 „E" drückst?
? Was passiert, wenn Du die Taste „a" drückst?
? Was passiert, wenn Du die Taste „m" drückst?
? Was passiert, wenn Du die Taste „x" drückst?
? Warum passiert das? Beschreibe in drei Sätzen.
? Welche Blöcke sind für die Taste „E" 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 „f", „a", „w" oder „x" 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?