# Digitale Aufgaben für das selbstständige Lernen in Mathematik und Informatik

*Luise Stromeyer, Frank Burghardt, Andreas Zeiser*

**Abstract** *Lerngruppen mit einer großen fachbezogenen Heterogenität erfordern eine große Variabilität im Bereich der Lehr-Lern-Methoden, um den individuellen Lernbedürfnissen der Studierenden gerecht zu werden. Der vorliegende Artikel befasst sich in diesem Sinne am Beispiel einer einsemestrigen Informatikvorlesung im Studiengang Maschinenbau mit binnendifferenzierten, digitalen Übungsaufgaben, die den selbstständigen Lernprozess fördern sollen. Die qualitative Untersuchung ihrer Wirksamkeit mittels Think-Aloud-Interviews zeigt, dass die Struktur der Aufgaben effektiv ist und die niedrigschwelligen Einstiege, die ansteigenden Schwierigkeitsgrade sowie individuelle Rückmeldungen seitens des Tools die Motivation und den Lernerfolg der Studierenden steigern, wenn sie ein grundlegendes Verständnis der Grundvoraussetzungen der jeweiligen Aufgabe haben.*

*Learning groups with a high degree of subject heterogeneity require a wide variety of teaching and learning methods to meet the individual learning needs of students. In this context, this article exemplifies the use of differentiated digital exercises in a one-semester computer science course for mechanical engineering students, which promote independent learning processes. The qualitative investigation of their effectiveness through think-aloud interviews shows that the structure of the tasks is effective and that the low-threshold entry points, increasing levels of difficulty, and individualized feedback enhance student motivation and learning success, provided that the learners have a basic understanding of the prerequisites for each task.*

## 1. Einleitung und Motivation

Mathematik und Informatik sind für viele Studierende der Ingenieurwissenschaften Grundlagenfächer, die in der Regel zu Beginn des Bachelorstudiums auf dem Stundenplan stehen. Die Studierenden bilden zu diesem Zeitpunkt in Bezug auf die vorhandenen fachlichen Grundlagen und Vorkenntnisse eine sehr heterogene Gruppe. Gleichzeitig gilt für die Klausur am Ende eines jeden Semesters die sachliche Bezugsnorm (nach Ingenkamp & Lissmann, 2008), d.h. alle Studierenden müssen die gleichen Anforderungen erfüllen.

Weitere Herausforderungen für die Lehre sind die unterschiedlichen Lerntempi, die individuellen Lernvorlieben und unterschiedlich ausgeprägte Lernstrategien der Studierenden sowie der Umstand, dass eine Einzelbetreuung der Lernenden durch die Lehrenden aufgrund der Lerngruppengröße und des inhaltlichen Pensums nicht möglich ist.

Um den Anforderungen dennoch gerecht zu werden, muss die effektive Lernzeit der Studierenden mit dem Motto »from teaching to learning« erhöht werden. Durch die Nutzung unterschiedlicher Methoden können Lehr-Lern-Prozesse ganzheitlich gestaltet werden, um so unterschiedlichen Bedürfnissen im Lernen gerecht zu werden.

In diesem Artikel werden zum einen binnendifferenzierte digitale Aufgaben als ein Baustein für das ganzheitliche Lernen vorgestellt und zum anderen deren Wirksamkeit in Bezug auf die Förderung des selbstständigen Lernens durch die Durchführung und qualitative Inhaltsanalyse von lernbegleitenden Interviews mit der Think-Aloud-Methode (Bortz & Döring, 2014) untersucht. Diese Interviews sind Teil einer Begleitstudie zu den Materialien und Aufgaben, die an der Hochschule für Technik und Wirtschaft Berlin (HTW Berlin) im Rahmen des Hochschullehrentwicklungsprojekts »Curriculum Innovation Hub«, Schwerpunkt »Digitale Innovative MINT-Curricula«, entwickelt worden sind (siehe auch Stromeyer & Zeiser, 2023; Stromeyer & Burghardt, 2023).

## 2. Binnendifferenzierte digitale Aufgaben

Die Fächer Mathematik und Informatik bringen, egal auf welchem Niveau und zu welchem Zeitpunkt im Studium sie gelehrt werden, neben den rein fachlichen Inhalten einen hohen Anteil an Übungsphasen sowohl innerhalb einer Lehrveranstaltung als auch in Form von Hausaufgaben mit sich. Da die Übungsaufgaben im Hochschulkontext in der Regel komplex sind, of-

fenbaren sich hier sehr individuelle Probleme der Studierenden; die Hürden und Fehlerquellen Einzelner sind in Art und Niveau sehr vielfältig. Mithilfe binnendifferenzierter digitaler Übungsaufgaben soll dieser Vielfalt in den Übungsphasen der Lehrveranstaltungen Rechnung getragen werden. Eine Herausforderung hierbei ist und bleibt die Klausurorientierung am Ende der Lehrveranstaltung.

Die oben angesprochene sachliche Bezugsnorm hat zu der Entscheidung geführt, dass zu jedem zentralen Lerngegenstand ein Set von Aufgaben entwickelt wurde. Ein Set besteht in der Regel aus vier Aufgaben mit steigendem Niveau von 1 bis 4, die sich durch das Bearbeitungsniveau unterscheiden, jedoch immer den gleichen Lerngegenstand behandeln. Die aufgrund der Heterogenität der Studierenden notwendige Binnendifferenzierung erfolgt also nicht über den Lerngegenstand, sondern über die Komplexität und den Umfang der selbst zu erbringenden Leistung innerhalb einer Aufgabe sowie der zur Verfügung gestellten Hilfen. Die Lernenden wählen entsprechend ihrem Lernstand den Schwierigkeitsgrad selbstständig aus und können auch während der Bearbeitung wechseln.

Damit die Studierenden bei der Bearbeitung der Aufgaben auch einschätzen können, in welchem Verhältnis das Anforderungsniveau einer einzelnen Aufgabe aus einem Set zu den Klausuranforderungen am Ende des Semesters steht, erhalten sie die Information, welches Aufgabenniveau dem Klausurniveau entspricht. Bei den hier vorliegenden Aufgaben war dies für Studierende des ersten Semesters eines ingenieurwissenschaftlichen Studiengangs mit nur einer Informatikvorlesung im Studium das Niveau 3.

## 3. Aufbau der binnendifferenzierten Aufgaben am Beispiel der Informatik

### 3.1 Die Grundstruktur der Binnendifferenzierung

Im Rahmen einer einsemestrigen Informatikvorlesung im Studiengang Maschinenbau an der HTW Berlin wurden vier Aufgabensets mit Binnendifferenzierung zu vier Grundkonzepten des Programmierens in der Programmiersprache Python entwickelt:

- if-else-Verzweigung
- for-Schleife

- while-Schleife
- Funktionen in Python

Das zentrale Grundprinzip der Binnendifferenzierung liegt bei allen Sets im unterschiedlich großen Umfang des selbstständigen Programmierens und damit steigendem Schwierigkeitsgrad.

Die Struktur eines Sets sieht folgendermaßen aus:

### Beispiel Grundaufgabe: if-else-Verzweigung

Das vorliegende Programm soll Meter in Yard umrechnen. Folgende Anforderungen werden an das Programm gestellt:

- Es gibt eine Eingabemöglichkeit für eine beliebige Länge in Metern (Eingabe ohne Einheit).
- Längen unter 0, also negative Längen, können nicht eingegeben werden. Wird ein kleinerer Wert als 0 eingegeben, gibt das Programm die Fehlermeldung »Achtung, keine negativen Längen eingeben!« aus.

Es gibt eine Grundaufgabe für alle Niveaustufen und folgende Niveaustufen:

- Niveau 1 besteht aus der Grundaufgabe sowie mehreren Lösungsvorschlägen, von denen einer richtig ist. Die Lernenden ermitteln die richtige Lösung, im Idealfall durch systematische Analyse der gegebenen Lösungen.
- Niveau 2 stellt zusätzlich zur Grundaufgabe ein Grundgerüst des zu schreibenden Codes zur Verfügung. Die Lernenden ergänzen die fehlenden Teile des Codes, die sich in der Regel auf den aktuellen Lerngegenstand fokussieren.
- Niveau 3 gibt nur noch die Grundaufgabe vor, jedoch keinen Code mehr. Die Lernenden schreiben den gesamten Code allein.
- In Niveau 4 wird die Grundaufgabe derart erweitert, dass der Lerngegenstand vertieft wird. So muss z.B. keine einfache, sondern eine verschachtelte if-Anweisung oder eine doppelte for-Schleife programmiert werden.

Der Vorteil der immer gleichen Grundaufgabe für jeweils ein Aufgabenset besteht darin, dass die Lernenden sich nicht immer wieder in neue Kontexte hineindenken und diese in all ihrer Komplexität neu erfassen müssen. Haben sie einmal einen Kontext durchdrungen, kann der Fokus direkt auf dem Lerngegenstand liegen.

Alle vier Aufgabensets für die vier Grundkonzepte wurden nach dem gleichen Prinzip aufgebaut. Die Studierenden waren also nach der Arbeit mit dem ersten Set an den Aufbau gewöhnt.

### 3.2 Die technische Umsetzung

Die Aufgabensets wurden für das Lern-Management-System Moodle mit dem Plugin VPL, Virtual Programming Lab entwickelt. Dieses Plugin verfügt über eine IDE (integrierte Entwicklungsumgebung) mit einem Quellcode-Editor mit Syntaxhervorhebung, einer Run-Konfiguration zum Ausprobieren sowie einem Debugger und bedient damit alle Anforderungen an eine Software für das Programmieren.

Weiterhin gibt es die Möglichkeit, dass Lehrende Testfälle hinterlegen, die automatisch durchlaufen werden. So wird zusätzlich zu der eigenen Überprüfung durch die Lernenden hier eine zusätzliche Kontrollinstanz für die Auswertung und Rückmeldung an die Lernenden genutzt. Ein klassisches Beispiel wäre die Überprüfung, ob bei obiger Aufgabenstellung nur Zahlen eingegeben werden können oder der Algorithmus für vorgegebene Testfälle die vorgegebenen Ergebnisse liefert.

Die genaue Umsetzung der Aufgaben in Form einer Anleitung führt an dieser Stelle zu weit; für einen ersten Eindruck mit Bildern siehe Stromeyer & Burghardt (2023).

## 4. Die qualitativen Interviews

Mit viel Zeit und Aufwand wird Lehre vorbereitet und durchgeführt, werden Aufgaben entwickelt und eingesetzt, wird Lehre hinterfragt und weiterentwickelt, und immer steht die Frage im Raum: Lohnt sich dieser oder jener Aufwand, geht die Entwicklung in die richtige Richtung? In Bezug auf die hier vorgestellten Aufgabensets lautet die Forschungsfrage also: Inwieweit eignen sich diese binnendifferenzierten digitalen Aufgaben zum selbstständigen Lernen und Üben?

Dieser Frage wurde im Rahmen des Projekts Curriculum Innovation Hub mithilfe einer Interviewstudie nachgegangen. Das Design sowie die Ergebnisse dieser Begleitstudie werden im Folgenden dargestellt und diskutiert.

## 4.1 Studiendesign und Durchführung

Die oben beschriebenen Aufgabensets wurden in einer einsemestrigen Informatikveranstaltung für Maschinenbauingenieurwesen im 1. Semester eingesetzt. Die Lehrveranstaltung besteht aus 4 Semesterwochenstunden (SWS) seminaristischer Vorlesung und 2 SWS Übung. Die Übung wurde in zwei 14-tägig stattfindenden Gruppen mit je etwa 25 Studierenden durchgeführt.

Nachdem ein konkretes Programmierkonzept (vgl. Abschnitt 3.1) zentral in der Vorlesung für alle eingeführt wurde, standen den Studierenden in der folgenden Übung (jeder Gruppe) rund 30 Minuten Zeit zur Verfügung, um das jeweilige Konzept mithilfe der binnendifferenzierten Aufgaben selbstständig zu üben. In dieser Zeit haben die Studierenden selbstständig aus dem Set ausgewählt, welche Niveaustufe(n) sie bearbeiten wollen.

Während der Arbeitszeit standen unterschiedliche Hilfsmittel zur Verfügung: Es durfte sich mit Mitstudierenden ausgetauscht und der Dozent gefragt werden sowie Hilfsmittel wie ein Übersichtsblatt für das Programmieren in Python, im Folgenden Cheatsheet genannt, und das Internet genutzt werden. Das Cheatsheet stand auch in der abschließenden Klausur zur Verfügung.

Damit die technische Handhabung des Tools VPL während der Übungen vertraut ist, wurde im Vorfeld gemeinsam eine sehr einfache Programmieraufgabe gelöst, bei der alle Funktionen des Tools ausprobiert wurden. Die Aufgabensets standen anschließend den Studierenden bis zur Klausur zur freien Verfügung.

Für die Begleitstudie wurden innerhalb dieses Lehrsettings sechs Think-Aloud-Interviews durchgeführt, die mit einer einleitenden und einigen abschließenden Fragen ergänzt wurden. Da die Forschungsfrage die Nützlichkeit der entwickelten Aufgaben in den Fokus nimmt, haben wir uns im Rahmen des Projekts insbesondere dafür interessiert, was die Lernenden mit den Aufgaben während der Bearbeitung tun. Die Methode der Think-Aloud-Interviews erlaubt es, gerade diesen Teil sichtbar zu machen, denn die Lernenden sprechen während der Bearbeitung der Aufgaben alle Gedanken, die sie haben, laut aus. Eine anschließende Analyse der Interviews kann Aufschluss darüber geben, mit welchen Themen die Lernenden während der Bearbeitung beschäftigt waren.

Für die Think-Aloud-Interviews haben sechs Proband*innen parallel zu den Studierenden in der Übung in einem anderen Raum ebenfalls das Aufgabenset bearbeitet. Dabei galten die gleichen Bedingungen (Hilfsmittel, Wahl des Niveaus) wie für die anderen Studierenden, die Proband*innen

haben jedoch ihre Gedanken laut ausgesprochen. Die Interviews wurden als Audiodatei aufgenommen, transkribiert und anschließend einer qualitativen Inhaltsanalyse in Anlehnung an Kuckartz (Dresing & Pehl, 2018) unterzogen.

Initiierende Textarbeit sowie die folgenden Leitfragen, die sich an der Forschungsfrage orientieren, bilden die Grundlage für die Entwicklung des in Abschnitt 4.2 dargestellten Kategoriensystems:

- Ist die technische Umsetzung der Aufgaben für die Lösung der Aufgaben hinderlich?
- Wie gehen die Lernenden mit den Rückmeldungen des Tools um?
- Wie nutzen sie Hilfen?
- Was passiert, wenn Hürden auftauchen?
- Wie flexibel sind die Aufgaben in Bezug auf unterschiedliche Herangehensweisen und Lösungsstrategien?
- Werden die Niveaustufen entsprechend dem Wissen gewählt? Wird ggf. gewechselt, wenn ein zu schweres oder zu leichtes Niveau gewählt wurde?

## 4.2 Das Kategoriensystem der Analyse der Think-Aloud-Interviews

Unter Beachtung der Forschungsfrage wurden die Interviews analysiert und ein Kategoriensystem entwickelt. Dieses beinhaltet die folgenden vier Hauptkategorien, die jeweils durch Unterkategorien vertieft wurden (vgl. Anhang):

1. Einschätzung des eigenen Wissenstandes
2. Wahl der Niveaustufe
3. Bearbeitung der Aufgabe
4. Die Bedienung des Tools betreffend

Im Rahmen dieses Artikels wird die Kategorie 3 vertieft betrachtet, da diese sehr facettenreich ist. Ein Schwerpunkt liegt allein schon in der Frage, auf welche Art und Weise an eine Aufgabe herangegangen wird: inhaltlich-systematisch, durch technisch-systematisches Probieren oder planloses Probieren? Daran anschließend stellt sich immer die vertiefende Frage, wie Lernende mit der erhaltenen Rückmeldung und/oder dem Stand der Aufgabe umgehen, wenn sie einen ersten Schritt getan haben. Aber auch der Umgang mit der Lösung und deren Überprüfung ist ein interessantes weites Feld, ebenso die Frage, wie explizit Lernende ihr Vorgehen angeben und/oder reflektieren und was dies für den nächsten Arbeitsschritt bedeutet.

## 4.3 Zusammenfassende Auswertung der Interviews

Vier der sechs interviewten Personen haben ihr Können in Bezug auf das zu übende Thema als Anfänger*innen ohne Vorkenntnisse eingeschätzt, eine Person als mittelmäßig und eine Person als gut. Unabhängig von der Selbsteinschätzung haben alle sechs Personen immer mit Niveau 1 begonnen.

Die technische Seite der Aufgabenbearbeitung stellte bei keiner Person ein Problem dar. Es gab selten Fragen dazu und diese konnten schnell gelöst werden, auch bei Personen, die das erste Mal mit dem VPL-Tool gearbeitet haben.

Alle Personen begannen mit dem Lesen der Aufgabenstellung und verfolgten dann zuerst einen inhaltlich-systematischen Ansatz, jedoch mit unterschiedlichen Zielen.

Einige versuchten zuerst die Anzahl der Möglichkeiten zu reduzieren, indem sie Fehler fanden und dann einzelne Codes ausschlossen; andere suchten nach der richtigen Lösung.

Wurde keine Lösung auf diesem Weg gefunden, wurde zum systematischen Probieren übergegangen, wobei in der Regel mit der Version begonnen wurde, die als am wahrscheinlichsten eingestuft wurde, wie das folgende Zitat veranschaulicht (Namen und Geschlecht sind willkürlich gewählt):

> 04:13 Lili[1]
> […] Jetzt würde ich tatsächlich einfach ausprobieren, weil ich jetzt nicht weiterweiß. Aber ich würde auf jeden Fall den zweiten und den fünften Code erst mal auslassen, weil ich denke, dass die nicht richtig sind. […]

Auf diese Art und Weise haben alle Proband*innen irgendwann die Aufgabe auf dem Niveau 1 gelöst. Vier Proband*innen benötigten dabei jedoch trotz oben dargestellter Herangehensweise grundlegendere Unterstützung durch die Interviewerin (als Dozentin), da Probleme auftauchten, die das selbstständige Lösen massiv behindert haben. Auf diese Probleme wird weiter unten vertiefend eingegangen.

Der andere Teil der Proband*innen hat die Aufgaben (auch weitere Niveaustufen) unter Zuhilfenahme eigener Aufzeichnungen, des oben erwähnten Python Cheatsheets sowie gezielter Hilfeanfragen an die Interviewerin als Dozentin selbstständig gelöst.

---

1 Im Sinne der besseren Lesbarkeit wurden Interviewpassagen ggf. sprachlich geglättet, ohne sie inhaltlich zu verändern.

Festzuhalten ist an dieser Stelle, dass vier der sechs Personen schon bei Niveau 1 versucht haben, jede Codezeile nachzuvollziehen. Hierfür wurde der Code zeilenweise gelesen und ggf. Hilfe in Form von Mitschriften, Cheatsheet oder Fragen an die Interviewerin/Dozentin genutzt. Der folgende Interviewausschnitt illustriert dies stellvertretend:

> 19:12 Simon
> […] Okay, funktioniert. Sehr schön. Ich muss mir noch mal ganz kurz den Code angucken, damit ich mir das nochmal durch den Kopf gehen lasse und nachvollziehen kann. Also »Eingabeliste« ist das erste, was ich mache […] Jetzt muss ich ganz kurz überlegen, warum. […] Ist das quasi so eine Art Container?

Nicht immer wurde jede Zeile verstanden. Zwei Personen wollten nach dem Interview den Code noch einmal genau betrachten und nachvollziehen.

Im Folgenden werden die oben angesprochenen Probleme beschrieben, die nach unserer Beobachtung das selbstständige Lernen massiv beeinträchtigt haben. Einige dieser Probleme haben sich gleich bei der Bearbeitung von Niveau 1 gezeigt, einige andere aber auch erst bei Niveau 2 oder 3, da Niveau 1 auch nur über Probieren lösbar ist.

Problem 1: Um zu wissen, ob eine Aufgabe richtig gelöst ist, also ein Code funktioniert, wird dieser in der Informatik getestet. Dies erfolgt bei VPL über zwei Wege: Zum einen kann der Code direkt in der IDE über die run-Konfiguration selbst ausprobiert werden, zum anderen lässt man den Code mit den hinterlegten Testcases evaluieren. An dieser Stelle ist aufgefallen, dass einige Proband*innen das Konzept einer IDE und des Testens noch nicht verstanden hatten. Diese Personen konnten nicht selbstständig feststellen, ob ihre Lösung richtig ist, und somit die Aufgabe nicht selbstständig abschließen.

Ben hat zwar den run gestartet, das Programm jedoch nicht ausprobiert, sondern gewartet.

> 05:52 Ben
> Interviewerin: Dort geben Sie etwas ein.
> Ben: Ach so, soll ich ein Wort direkt eingeben?
> Interviewerin: Ja, sonst haben Sie doch gar nichts ausprobiert.

Gerd nutzte den run nicht, sondern nur das Evaluieren, las aber die Rückmeldungen der Evaluation nicht.

08:37 Gerd
[…] Ich war leider nicht richtig. (7)[2] Wieso nicht? (26) […]

Darüber hinaus kam die Bearbeitung der Aufgabe immer dann ins Stocken, wenn die Rück- und Fehlermeldungen zwar ausgelöst wurden, aber nicht klar war, dass es dort direkte Hinweise gab, wo z.B. ein Fehler im Code sein könnte.

26:10 Gerd
Interviewerin fragt, was mit der Rückmeldung des Testcases ist, da Proband nur den Titel, jedoch nicht den Rest davon liest.
Gerd: Aber so richtig viel kann ich damit immer nicht anfangen.
16:00 Lili
Interviewerin: Und die Fehlermeldung sagt auch, wo sie ein Problem hat. […] Der kleine Pfeil, der sagt genau wo.
Lili: (5) Krass. […]

Wurden diese Konzepte während des Interviews besprochen, wurden sie anschließend zunehmend genutzt.

Problem 2: Einige Probanden konnten einfache Syntaxfehler wie einen fehlenden Doppelpunkt nicht beheben, auch wenn die Rückmeldung den genauen Ort des Fehlers angegeben hat. Interessant daran ist, dass dies keine Logikfehler sind, sondern »nur« der Code richtig aus den Aufzeichnungen abgeschrieben werden muss, denn jeder Code hat eine feste Syntax, die genau eingehalten werden muss. Eine Vermutung ist, dass diesen Personen Aufbau und Funktionsweise eines Codes, also wie dieser durchlaufen wird, nicht klar ist.

03:19 Lili
Also ich sehe auf jeden Fall, dass hier zum Beispiel der Doppelpunkt hinter der Anzahl fehlt. […] Ich weiß jetzt nicht, ob es in dem Fall auch so ist. […]

Dieses Phänomen zeigte sich auch an anderen Stellen, z.B. beim Nachvollziehen eines Testcases. Die Interviewerin erarbeitete mit Simon, wie ein Testcase funktioniert. Dabei zeigte sich, dass Simon nicht klar war, dass ein Code in einer bestimmten Reihenfolge und nach festen Regeln durchlaufen wird. Die Rückmeldungen und Hilfen des Tools und der schriftlichen Hilfen haben in diesen Fällen nicht geholfen, sondern nur die direkte Interaktion mit der Interviewerin.

2 Die Zahlen in Klammern geben die Sekunden zwischen den Äußerungen an.

Problem 3: Aufzeichnungen, z.B. aus der Lehrveranstaltung, wurden nicht als Hilfe genutzt, auch wenn sich die Person gefragt hat, wie sie etwas genau aufschreiben soll.

> 11:07 Lili
> Ja, also, ich brauche auf jeden Fall ein i und ich brauche eine Range. Ähm, aber wie? In welcher Reihenfolge? Und wie wird das formatiert? (1) Da bin ich jetzt total überfragt.

Problem 4: Die Prozentrechnung in der einen Aufgabenstellung hat bei einer Person zu Problemen in der Bearbeitung der Aufgabe geführt. Die Hürde lag hier nicht in der Informatik, sondern in der Mathematik.

Neben diesen Problemfeldern konnten aber auch zielführende Schritte in der Bearbeitung der Aufgaben ausgemacht werden.

Die Proband*innen, die (in Teilen) die Aufgaben erfolgreich selbstständig gelöst haben, nutzten neben dem vorhandenen Wissen gezielt und planvoll Hilfen. Interessant ist, dass nicht alle Hilfen von allen gleichermaßen genutzt wurden.

Die beiden erfolgreichsten (im Sinne von: Aufgaben selbstständig lösen) Proband*innen nutzten sowohl die Hilfen des Tools als auch Mitschriften und/oder das Cheatsheet und als letzte Methode das Fragen der Interviewerin. Fehlermeldungen wurden sogar gezielt eingeholt:

> 19:00 Alex
> […] Wir machen einen Test, bevor ich jetzt hier »1 bis Anzahl« schreibe. […]

Das Cheatsheet wurde schon beim Codeschreiben genutzt und nicht erst bei einer Fehlermeldung:

> 06:25 Alex; schreibt einen Teil Code selbst
> […] War das mit…? (1) Ich gucke jetzt auf dieses Memento-Blatt (Anm.: Cheatsheet) und (2) suche die Syntax für die »Range«. (10) Okay, wird in runden Klammern angegeben.

Andere Probanden nutzten z.B. zwar die Fehlermeldung, nicht jedoch das Cheatsheet.

Besaßen die interviewten Personen Wissen in Bezug auf den Lerngegenstand, z.B. aus der Vorlesung, haben sich viele Ansatzpunkte zum selbstständigen Lösen der Aufgaben ergeben.

### Zusammenfassung der Antworten auf die abschließenden Fragen

Unabhängig davon, wie viel Hilfe und wie viel Zeit die interviewten Personen für das Lösen benötigten, haben alle sechs einen Mehrwert in den Übungen für sich gesehen.

Alle haben spontan etwas benennen können, was sie gerade gelernt hatten, und dies hatte immer mit dem Programmieren zu tun, wenn auch nicht immer direkt mit dem behandelten Lerngegenstand.

Auch die erfahrenere Person hat alle Niveaustufen genutzt. Niveau 1 wurde z.B. genutzt, um sich des eigenen Wissens rückzuversichern.

Der Aufbau mit der immer gleichen Grundaufgabe wurde als hilfreich empfunden, da man sich nicht immer in neue Konzepte einarbeiten musste.

Die Niveaustufen wurden als angenehm und motivierend wahrgenommen, insbesondere von schwächeren Lernenden. Es wurde als angenehm empfunden, dass man in die Aufgabe schrittweise hineinwachsen konnte und erst nach und nach immer mehr selbst entwickeln musste.

### Weitere Beobachtungen

Ob man sich ohne Vorkenntnisse eines der vier hier behandelten Programmierkonzepte mithilfe der Aufgaben komplett selbstständig erarbeiten könnte, konnte nicht beobachtet werden, da die Konzepte in diesem Setting immer direkt vorher in der Vorlesung behandelt wurden und die Studierenden daher mit Grundkenntnissen die Übungsaufgaben bearbeitet haben. Interessant ist, dass keiner der Proband*innen die bereitgestellte Musterlösung genutzt hat. Weiterhin konnte beobachtet werden, dass die stückweise Bearbeitung aller Niveaustufen dazu führen konnte, dass Niveau 2 und 3 teilweise durch Erinnerungen von Niveau 1 gelöst wurden. Eine Person hat dies explizit erwähnt. Der Code konnte auf diese Weise jedoch nie vollständig reproduziert werden, ein Eigenanteil war immer notwendig.

## 4.4 Diskussion der Ergebnisse

Auch wenn die Befragung aufgrund der relativ geringen Anzahl von sechs Interviews nicht repräsentativ ist, können aus den Beobachtungen einige Ansatzpunkte in Bezug auf die Fragestellung abgeleitet werden.

Mindestvoraussetzung für eine erfolgreiche Bearbeitung der Aufgaben ist, dass die Lernenden die Grundkonzepte des Programmierens beherrschen. Ihnen muss klar sein, wie die Funktionsweise eines Codes aussieht und wie Fehlermeldungen genutzt und gelesen werden.

Ist der Lerngegenstand aus einem anderen Kontext bekannt, können mit den Aufgabensets, wie sie hier vorgestellt wurden, die Inhalte geübt werden. Dabei bieten die unterschiedlichen Niveaustufen einen niedrigschwelligen und motivierenden Einstieg. Dies ist auch in Bezug auf die angesprochene Heterogenität der Lerngruppe lernfördernd: Sowohl starke als auch schwache Lernende waren motiviert, die Aufgaben in Angriff zu nehmen und »ihren Weg zu finden«. Da nicht alle Lernenden im gleichen Tempo die Aufgaben bearbeiten müssen und erst am Ende das Ergebnis überprüft wird, kann die Lernzeit individuell sinnvoll genutzt werden. Die sehr unterschiedliche Zeit, die von den Proband*innen zum Lösen einer Aufgabe benötigt wurde, stellt in dieser Form des Lernarrangements kein Problem dar. Niemand muss warten oder Lernschritte überspringen, die effektive Lernzeit kann für alle Lernenden auf diesem Weg erhöht werden.

Unabdingbare Notwendigkeit für die erfolgreiche Bearbeitung der Programmierübungen ist das Wissen um die Funktionsweise von Fehlerrückmeldungen in einer IDE. Dies muss in der Lehre unbedingt berücksichtigt werden, denn ist dieses Wissen nicht vorhanden, können die Aufgaben nicht selbstständig gelöst werden. Hier bietet sich eine sehr gute Schnittstelle innerhalb von Blended-Learning-Szenarien zwischen klassischer Instruktion und dem individuellen, digital gestützten Lernen an.

Einschränkend muss in Bezug auf die Durchführung der Interviews erwähnt werden, dass in Momenten, in denen anscheinend Ratlosigkeit bei den interviewten Personen herrschte, die Interviewerin manchmal zu schnell interveniert und nachgefragt hat. So wurde unter Umständen manch ein Lösungsansatz, der von den Probandenvielleicht noch gekommen wäre, unterbunden.

Gleichzeitig stellt sich durch die durchgeführten Interviews und Beobachtungen die Frage, wie die Übungsphase noch angereichert werden kann, damit für noch mehr Lernende ein selbstständigeres Lernen möglich wird. Eine Idee ist die aktive Einbindung von Künstlicher Intelligenz, denn diese könnte den Studierenden bei der Fehlersuche helfen oder auch eine bestimmte Codezeile erklären, wenn die Lernenden sich diese nicht selbst erschließen können.

Keine Aussage kann zu der Frage getroffen werden, ob die Wahl der Niveaustufe immer dem Lernstand entsprochen hat. Hierfür ist die gewählte Methode nicht geeignet.

In Bezug auf die Flexibilität der Aufgaben bezüglich unterschiedlicher Strategien und Herangehensweisen lässt sich durch diese sechs Interviews ableiten, dass zumindest für die hier dargestellten Strategien keine Einschränkungen vonseiten der Aufgaben und des Tools zu verzeichnen waren. Die Aufgaben konnten z.B. über die Fehlersuche, das Verstehen des gegebenen Codes oder eigene Lösungsansätze bearbeitet werden.

Abschließend lässt sich festhalten, dass alle Lernenden die Aufgaben sehr gut angenommen haben und während der Bearbeitung einen Lernzuwachs für sich verzeichnet haben.

Der Aufwand, solche Aufgabensets zu erstellen, ist sehr hoch, lohnt sich jedoch für die Studierenden, die systematisch ihr Wissen ausbauen möchten. Gerade die Reduzierung auf jeweils eine Grundaufgabe hat dazu beigetragen, dass die Lernenden zügig von einer Niveaustufe zur nächsten gegangen sind. Sie mussten sich nicht neu motivieren und in einen neuen Kontext einarbeiten, sondern konnten direkt in das Programmieren auf einer höheren Niveaustufe einsteigen. Die Lernzeit wurde effektiv genutzt, immer jedoch unter der Bedingung, dass logische Grundkonzepte des Programmierens inklusive Testens von Code zumindest in Ansätzen verstanden wurden.

Auch für die Lehrenden lohnt sich der Aufwand, wenn die Aufgaben in weiteren Kursen wiederverwendet werden.

## 5. Fazit

In diesem Artikel wurde ein Konzept einer Binnendifferenzierung in Informatikvorlesungen für Ingenieure erarbeitet, durchgeführt und evaluiert. Durch die steigenden Schwierigkeitsgrade konnten alle Studierenden die Aufgaben mit ihrem individuellen Wissenstand bearbeiten und waren dadurch motivierter. Durch Think-Aloud-Interviews konnten Schwierigkeiten identifiziert werden, die in folgenden Durchgängen von den Lehrenden bzw. durch verbesserte Fragestellungen korrigiert werden können.

Eine offene Frage ist, ob und wie man eine persönliche Unterstützung der Lernenden z.B. durch KI automatisieren kann, sodass die Lehrenden auch bei großen Gruppen auf individuelle, komplexere Fragen der Studierenden eingehen und diese fördern können.

## Literaturverzeichnis

Bortz, J. & Döring, N. (2014). *Forschungsmethoden und Evaluation in den Sozial- und Humanwissenschaften* (5. Aufl.). Springer.

Dresing, T. & Pehl, T. (2018). *Praxisbuch Interview, Transkription & Analyse. Anleitungen und Regelsysteme für qualitativ Forschende* (8. Aufl.). Eigenverlag.

Ingenkamp, K. & Lissmann, U. (2008). *Lehrbuch der pädagogischen Diagnostik* (6. Aufl.). Beltz.

Stromeyer, L. & Burghardt, F. (2023): Informatische Bildung zwischen Nebenfach und Grundlagenbildung. *Informatik 2023, Designing Futures: Zukünfte gestalten; 26.-29. September 2023, Berlin*. Gesellschaft für Informatik (GI) (Vol. 337). https://dl.gi.de/items/b3d4e724-9a56-4975-b77f-66ad4f9e0a97

Stromeyer, L. & Zeiser, A. (2023): Von den mathematischen Fertigkeiten zur Anwendung: digitale Aufgaben und digitale Labore. In E. Liebscher, R. Hübl, J. Mecker & B. Wacker (Hg.), *Digitale Lehre im Rahmen der Grundausbildung in MINT-Fächern an Hochschulen: Didaktische Integration von digitalen Medien und E-Learningsystemen in Lehrveranstaltungen. Tagungsband 2022* (S. 159–177). Hochschulverlag Merseburg. http://dx.doi.org/10.25673/103431.2

## Anhang

Dieser Anhang beinhaltet das gesamte Kategoriensystem, das anhand der Leitfragen aus Abschnitt 4.1 entwickelt wurde.

Einschätzung des eigenen Wissenstandes
- mit Begründung
- ohne Begründung

Wahl der Niveaustufe
- mit Begründung
- ohne Begründung

Bearbeiten der Aufgabe
- inhaltlich planvoll/systematisch

Strategie explizit benennen
    Verständnis von Aufgabenstellung/gegebenen Code herstellen, nicht: von Fehler- und Rückmeldungen
    Lösung wird überprüft, nicht: Punkte durch Testcase

Rückmeldungen und Fehlermeldungen werden gezielt eingeholt
    als Abschlussüberprüfung
    um zu wissen, wo man in der Lösung der Aufgabe steht

Fehlermeldungen und Rückmeldungen verstehen wollen und ggf. Hilfen dafür nutzen
    eigene Lösung in Form von Code entwickeln, inkl. Auswahl bei Niveau 1

Hürden/Probleme die nicht gelöst werden (Rückmeldung nicht verstehen, Fehler nicht finden, keine Idee haben)
    bemerken
    nicht bemerken

technisch systematisches Probieren
    Strategie explizit benennen
    systematisches Probieren

planloses Probieren
    Bauchgefühl
    Die Bedienung des Tools betreffend