Unterricht konzipieren und reflektieren mit den 12 fachdidaktische Prinzipien des NCCE

Im Herbst 2021 erschien das Big Book of Computing Pedagogy der Raspberry Pi Foundation, das nach den 12 fachdidaktischen Prinzipien des National Centre for Computing Education (NCCE), England, strukturiert wurde. Im Folgenden werden die Prinzipien beschrieben und mit Links aus der Videoreihe „Informatikdidaktik kurz gefasst“ zur Vertiefung unterlegt – wobei einige Videos mehrere Prinzipien adressieren.

Die 12 fachdidaktischen Prinzipien

1 Fokussiere auf Konzepte (Lead with concepts)

Unterstütze die Schülerinnen und Schüler bei der Aneignung von Wissen durch die Verwendung von Schlüsselkonzepten, Begriffen und Fachvokabular und biete ihnen die Möglichkeit, ein gemeinsames und konsistentes Verständnis aufzubauen. Glossare, Concept Maps und Visualisierungen zusammen mit regelmäßiger Wiederholung können diesen Ansatz unterstützen.

Videos: Fundamentale Ideen der Informatik; Was ist Informatik?; Produkt- & Konzeptwissen im Informatikunterricht Dagstuhldreieck

2 Semantic Waves (Unplug, unpack, repack)

Unterrichte neue Konzepte, indem zunächst komplexe Begriffe und Ideen herausgelöst, diese Ideen unplugged oder in vertrauten Kontexten erkundet werden und dieses neue Verständnis dann wieder in das ursprüngliche Konzept eingebaut wird. Dieser Ansatz (Semantic wave / Semantische Welle) kann Schülerinnen und Schülern helfen, ein sicheres Verständnis komplexer Konzepte zu entwickeln.

Videos: Semantic Waves als Unterrichtskonzept; Computer Science Unplugged

3 Projekte (Create projects)

Nutze projektbasierte Lernaktivitäten, um den Schülerinnen und Schülern die Möglichkeit zu geben, ihr Wissen und Verständnis anzuwenden und zu festigen. Design ist ein wichtiger, oft übersehener Aspekt des Informatikunterrichts. Die Schülerinnen und Schüler können überlegen, wie sie ein Artefakt für einen bestimmten Zielgruppe oder eine bestimmte Funktion entwickeln und es anhand einer Reihe von Kriterien bewerten können.

Videos: Agile Methoden und Projekte; Projekte im Informatikunterricht

4 Schülervorstellungen (Challenge misconceptions)

Nutze unterrichtsbegleitende (formative) Befragungen, um Fehlvorstellungen aufzudecken, und passe den Unterricht an, um auf diese einzugehen, sobald sie auftreten. Das Bewusstsein für häufige Fehlvorstellungen in Verbindung mit Diskussionen, Concept Mapping, Peer-Instruction oder einfachen Quizfragen kann helfen, Fehlvorstellungen zu identifizieren.

Videos: Fehlvorstellungen und Modelle bei Variablen; Typische Fehlvorstellungen zu SQL; Notional Machines als Erklärmodell beim Programmieren; Concept Cartoons

Stand: 16.6.2023

5 Unterricht strukturieren (Structure lessons)

Verwende bei der Unterrichtsplanung unterstützende Rahmenkonzepte wie PRIMM (Predict, Run, Investigate, Modify, Make) und Use-Modify-Create. Diese Modelle basieren auf Forschungsergebnissen und stellen sicher, dass eine Differenzierung in verschiedenen Phasen des Unterrichts vorgenommen werden kann. Sie helfen auch, die kognitive Belastung zu reduzieren und stellen sicher, dass die Schülerinnen und Schüler in den richtigen Momenten gefördert und gefordert werden. Für inklusiven Informatikunterricht bieten sich Frameworks wie Universal Design for Learning an (UDL).

Videos: Unterstützungsstrategien beim Programmierenlernen & PRIMM; Aufgabentypen in Scratch mit Use-Modify-Create; Programmierenlernen mit TIPP & SEE-Strategie; Benutzen, Analysieren, Gestalten und Verankern im Informatikunterricht; Cognitive Load Theory; Inklusiver Informatikunterricht; Phasenschemata zu Informatik, Mensch und Gesellschaft

6 Team- und Partnerarbeit (Work together)

Fördere die Zusammenarbeit, insbesondere mit Hilfe von Pair-Programming und Peer-Instruction, aber auch mit strukturierten Gruppenaufgaben nach den Prinzipien des kooperativen Lernens. Die gemeinsame Arbeit fördert den Dialog in der Klasse, die Artikulation von Konzepten und die Entwicklung eines gemeinsamen Verständnisses.

Videos: Agile Methoden und Projekte; Unterstützungsstrategien beim Programmierenlernen (Pair Programming)

7 Demonstriere alles (Model everything)

Demonstriere sowohl den Prozess als auch wie das fertige Produkt bzw. Ergebnis aussehen soll – vom Debuggen von Code bis zur Umwandlung von Binärzahlen – mit Techniken wie ausgearbeiteten Beispielen (worked examples) und Live-Coding. Forschungsergebnisse zum Cognitive Apprenticeship legen nahe, dass das Demonstrieren besonders für Anfänger von Vorteil ist, da es gestufte Unterstützung ermöglicht, die nach und nach abgebaut werden kann. Außerdem profitieren Anfänger besonders von Prozess-orientierten Worked Examples, während fortgeschrittenere Lernende stärker von Produkt-orientierten Beispielen profitieren.

Videos:Unterstützungsstrategien beim Programmierenlernen & PRIMM

8 Variiere den Unterricht (Add variety)

Biete vielfältige Aktivitäten mit gestuften Hilfen an, die aktives Lernen fördern – von stark strukturiert bis zu eher explorativem Lernen. Informatik ist sowohl eine Strukturwissenschaft wie die Mathematik, hat aber auch ingenieurwissenschaftliche Anteile und starke gesellschaftliche Auswirkungen. Die Unterrichtsziele zu diesen unterschiedlichen Bereichen erfordern unterschiedliche Unterrichtsmethoden. Die Anpassung der Unterrichtsmethoden an unterschiedliche Unterrichtsziele hilft dabei, das Engagement der Schülerinnen und Schüler aufrechtzuerhalten und fördert deren Selbständigkeit.

Videos: Cognitive Load Theory; Inklusiver Informatikunterricht

9 Mache es konkret (Make concrete)

Viele der Ideen, mit denen die Schülerinnen und Schüler im Informatikunterricht konfrontiert werden, sind abstrakt und komplex und müssen anhand konkreter Beispiele und Aktivitäten veranschaulicht werden. Erwecke abstrakte Konzepte zum Leben, indem Beispiele aus der Lebenswelt der Schülerinnen und Schüler mit Bezug zu ansprechenden Kontexten und anderen Unterrichtsfächern aufgegriffen werden. Dies kann erreicht werden durch den Einsatz von Unplugged-Aktivitäten, das Anbieten von Analogien und das Erzählen von Geschichten rund um diese Konzepte.

Videos: Didaktische Reduktion & Rekonstruktion; Repräsentationsebenen: enaktiv, ikonisch, symbolisch

Stand: 16.6.2023

10 Code lesen und erkunden kommt zuerst (Read and explore code first)

Fokussiere beim Programmieren zuerst auf das „Lesen“ von Code, bevor die Schülerinnen und Schüler ihn schreiben. Sowohl bei der blockbasierten als auch bei der textbasierten Programmierung sollten die Schülerinnen und Schüler dazu ermutigt werden, Codeblöcke zu überprüfen und zu interpretieren. Die Forschung hat gezeigt, dass die Fähigkeit, Code zu lesen, nachzuvollziehen und zu erklären, die Fähigkeit der Schülerinnen und Schüler, Code zu schreiben, verbessert.

Videos: Unterstützungsstrategien beim Programmierenlernen & PRIMM

11 Arbeite praktisch (Get hands-on)

Nutze Physical Computing und Making-Aktivitäten, die taktile und sensorische Erfahrungen bieten, um das Lernen zu verbessern. Die Kombination von Elektronik und Programmierung mit Kunst und Werken (vor allem durch Projekte) bietet den Schülerinnen und Schülern einen kreativen, ansprechenden Rahmen zur Erkundung und Anwendung von Informatikkonzepten.

Videos: Physical Computing

12 Fördere Programmverständnis (Foster program comprehension)

Nutze eine Vielzahl von Aktivitäten zur Festigung von Wissen und Verständnis über die Funktion und Struktur von Programmen, einschließlich Debugging, Tracing und Parsons-Probleme. Regelmäßige Aktivitäten zum Programmverständnis helfen, Verstehen zu sichern und Verbindungen mit neuem Wissen aufzubauen.

Videos: Eine Notional Machine für Python - Der ,,Python Computer"; Eine Notional Machine für Python: Funktionsaufrufe

Stand: 16.6.2023