

LPU - Programmieren mit Unterstützung von ChatGPT & Co. - Vorüberlegung

Diese LPU soll Schülerinnen und Schüler dazu befähigen, mit Hilfe von fortschrittlichen Sprachmodellen (Large Language Model - LLM) die Erstellung von Programmen teilweise zu automatisieren. In der Breite verfügbare Tools wie ChatGPT, MistralAI, LLaMA & Co. sollen die Lernenden beim Entwickeln einer Lösung effizient und individuell unterstützen.

- [LPU - Programmieren mit Unterstützung von ChatGPT & Co. - Vorüberlegung](#)
 - [Stoffanalyse](#)
 - [Welche wichtigen Begriffe werden eingesetzt?](#)
 - [In welcher Relation stehen diese Begriffe zueinander?](#)
 - [Welche Begriffe haben eine zentrale Bedeutung?](#)
 - [Welche Begriffe sind Ihrer Klasse bereits bekannt?](#)
 - [Lernziele nach dem Zielebenenmodell](#)
 - [Leitidee](#)
 - [Dispositionsziele](#)
 - [Operationalisierte Lernziele](#)
- [Anhang](#)
 - [Inhaltliche Abgrenzung](#)
 - [Technisches Setup](#)
- [Quellen](#)

Die Hauptbeiträge der Unterrichtseinheit sind, dass die Lernenden

1. ihre Fähigkeiten stärken, fremde Programme zu lesen, zu analysieren und nach Bedarf zu korrigieren und weiterzuentwickeln.
2. Problemstellungen für Programmieraufgaben eindeutig beschreiben können.

Spezifische Informatik-Kompetenzen:

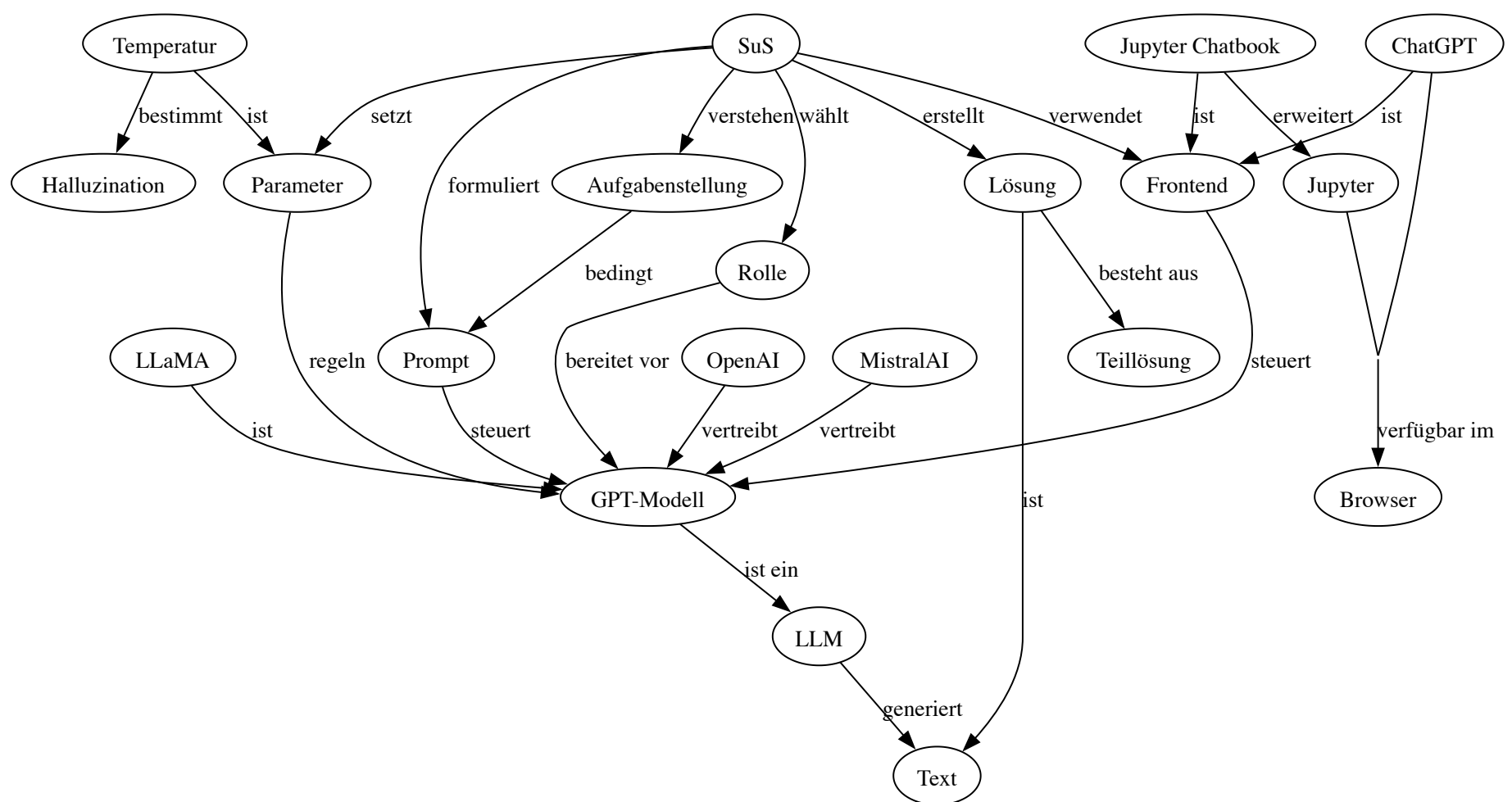
- Quellcode lesen und verstehen
- Funktionalität von Quellcode bestimmen
- Programme mittels Ausführung auf selbstgenerierten Eingaben testen
- Fehler in Quellcode finden und korrigieren
- Lösungswege/Algorithmen erklären und von LLM erstellte Erklärungen analysieren
- Teillösungen kombinieren

Stoffanalyse

Welche wichtigen Begriffe werden eingesetzt?

- Künstliche Intelligenz - KI
- Generative Pre-trained Transformer (GPT)
- LLM - Large Language Model
- ChatGPT, MistralAI, LLaMA
- Prompt
- Token
- Problem (im Sinne von Problem-Statement), Teilproblem
- Instanzen eines Problems
- Lösung (im Sinne von Lösungsstrategie/Lösungsweg/Algorithmus/Programm)
- Algorithmus
- Programm
- Jupyter
- Jupyter-Chatbook
- Programmiersprache
- Python
- Testen

In welcher Relation stehen diese Begriffe zueinander?



Fokusfrage: Wo liegen die Herausforderungen für die SuS beim Einsatz von LLM im Informatikunterricht?

Welche Begriffe haben eine zentrale Bedeutung?

- LLM
- GPT
- Modell
- Prompt
- Halluzinieren
- Aufgabenstellung
- Lösung/Algorithmus/Programm
- Jupyter-Notebook

Diese LPU geht nicht auf die Technologie hinter ChatGPT & Co. ein.

Welche Begriffe sind Ihrer Klasse bereits bekannt?

- **Algorithmus:** Ein Algorithmus ist eine endliche Reihe von präzisen Anweisungen, die schrittweise ausgeführt werden, um ein bestimmtes Problem zu lösen oder eine bestimmte Aufgabe zu erledigen. Ein Algorithmus ist nicht spezifisch an eine bestimmte Programmiersprache oder Umgebung gebunden und kann auf verschiedene Arten implementiert werden. Er beschreibt den Lösungsweg oder den Prozess, der befolgt werden muss, um ein bestimmtes Ergebnis zu erzielen. Ein übliches algorithmisches Problem besteht aus unendlich vielen Problemfällen (Instanzen). Ein Algorithmus ist eine Methode, die in allen Fällen das korrekte Resultat berechnet. Er kann sowohl in natürlicher Sprache als auch in Pseudocode oder in einer formellen Programmiersprache dargestellt werden.
- **Programm:** Programme enthalten Anweisungen in einer für den Computer verständlichen Form, die ausgeführt werden können. Ein Programm ist bspw. eine konkrete Umsetzung eines Algorithmus in einer bestimmten Programmiersprache.
- **Problem:** engl. Problem Statement ist eine (idealerweise eindeutige und präzise) Beschreibung eines Problems, einer Herausforderung oder eines Ziels, das gelöst oder erreicht werden soll.
- **Teilproblem:** Ein Problem lässt sich häufig in leichter lösbare Teilprobleme zerlegen. Vgl. Teile und Herrsche!
- **Lösung:** Wird hier verstanden im Sinne von Lösungsstrategie/Lösungsweg/Algorithmus/Programm für ein Problem und löst im Idealfall alle Instanzen des Problems.
- **Instanzen eines Problems:** Eine Instanz eines Problems bezieht sich auf eine spezifische Ausprägung oder ein spezifisches Beispiel eines allgemeinen Problems und meint damit oft eine bestimmte Eingabe oder Kombination von Werten, auf die eine Lösung angewendet werden kann.
- **Programmiersprache:** Mittels einer Programmiersprache, wie etwa Python, C++, Racket, Raku oder Rust, kommunizieren Nutzer mit dem Computer und erteilen diesem Aufträge.
- **Python:** Python ist eine Programmiersprache, die für Programmier-Einsteiger geeignet ist.

Die SuS haben schon mit Jupyter Notebooks in Jupyter Lab auf ihren privaten Computern gearbeitet - die Bedienung von Jupyter Lab im Webbrowser ist bekannt. Zudem sind die SuS mit den Grundlagen der Programmierung in Python und Funktionen sowie mit der Nutzung von `assert`-Statements zum einfachen Testen von Funktionen in Python vertraut.

Lernziele nach dem Zielebenenmodell

Wenn man sich die Concept-Map genauer anschaut, erkennt man, das die SuS vor vier Herausforderungen stehen:

1. Die Aufgabenstellung verstehen, um
2. einen geeigneten Chatbot zu erstellen,
3. um diesem dann das Problem mittels einer Anzahl geeigneter Prompts so zu beschreiben,
4. dass sie aus den vom Chatbot gelieferten Teillösungen eine Gesamtlösung konstruieren können.

Aus all diesen Vorüberlegungen ergibt sich die Leitidee dieser LPU:

Leitidee

Genau wie SuS den Umgang mit traditionelleren Quellen wie gedruckten Büchern oder Online-Dokumentationen lernen müssen, muss auch der Umgang mit Large Language Models (LLM) verstanden und geübt werden.

Diese LPU soll Schülerinnen und Schüler dazu befähigen, mit Hilfe von fortschrittlichen Sprachmodellen (Large Language Model - LLM) die Erstellung von Programmen teilweise zu automatisieren und die in den operationalisierten Lernzielen genannten Kompetenzen schulen.

Dispositionsziele

Die SuS

- haben ein auf eigener Erfahrung aufbauendes Verständnis von den Potentialen und Limitierungen von LLM.
- verstehen, wie ihnen LLM helfen können, schneller Programme für gegebene Problemstellungen zu generieren.
- interessieren sich für die kommenden Entwicklungen auf dem Gebiet der LLM.

Operationalisierte Lernziele

Sie SuS können

- die Funktionalität eines Programmcodes bestimmen.
- Lösungswege/Algorithmen anderen erklären.
- Teillösungen erstellen und kombinieren.
- syntaktische und logische Fehler in Programmcodes systematisch finden und korrigieren.
- Aufgabenstellungen abstrahieren, d.h. sie können sich von spezifischen Details lösen und allgemeine Konzepte herausarbeiten.

Anhang

Erläuterungen für Lehrperson

Inhaltliche Abgrenzung

Diese LPU dient nicht dazu, weiterführende Aspekte zu vertiefen, wie etwa:

- Wie funktionieren LLM grundsätzlich und worin unterscheiden sich die verschiedenen Ansätze?
- Wie nutzt man Testen systematisch zur Überprüfung von (AI-generierten) Lösungen? (Stichwort TDD)
- Wie und warum ist Benchmarking zum Vergleichen und Bewerten von (AI-generierten) Lösungen nützlich?
- Warum verbessert starke (bzw. graduelle) Typisierung die Spezifizierung, Entwicklung und Überprüfbarkeit von (AI-generierten) Lösungen?
- Welche Mittel bieten einfache Ansätze der Code-Verification, wie etwa die explizite Definition von Invarianten und Pre- und Postconditions, sowie deren automatische Überprüfung beim Kompilieren oder zur Laufzeit, für die Spezifizierung, Entwicklung und Überprüfbarkeit von (AI-generierten) Lösungen?
- Welche Problemklassen in der Informatik lassen sich derzeit gut mit Hilfe LLM lösen?
- Welche Problemklassen kommen hinzu, wenn Ansätze wie [ChatGPT+Wolfram](#) in der Breite verfügbar sind und wie müssen Aufgabenstellungen dann angepasst werden, um lernwirksam zu bleiben?

Neben dieser inhaltlichen Abgrenzung stehen für mich derzeit viele äusserst interessante Fragestellungen im Raum, auf die auch nicht eingegangen wird:

- Können die Vorschläge, Erklärungen und Beispiele, die ein LLM den SuS liefert, ihr persönliches Lernen (als aktive Tätigkeit im Angebots-Nutzen-Modell) wirklich nachhaltig unterstützen?
- Oder schränken LLMs die Kreativität und Originalität der Lernenden letztendlich ein, da die SuS lediglich vorgegebene Muster oder Lösungen des Modells befolgen, anstatt eigene Ansätze zu erkunden? (Stichwort Navigationssysteme und Routenplanung)
- Wie müssen Übungsaufgaben formuliert sein, um die SuS anzuregen, mit Unterstützung eines LLM nach eigenen Lösungen zu suchen?
- Ist das zielführende Fragen (Prompten) zukünftig wirklich eine wertvolle Fähigkeit bzw. Kompetenz an sich?
- Wie werden weniger offensichtliche Fehler, bspw. in Bezug auf Laufzeit- und Speicherkomplexität, in den Vorschlägen des LLM durch die Lernenden (und Lehrenden - wenn es bspw. gilt, 24 verschiedene Lösungen zu bewerten) systematisch erkannt und behandelt?
- Aber auch: Welche (technische/gesellschaftliche/rechtliche) Gefahr geht von der (unabsichtlichen) Preisgabe sensibler Informationen aus, während SuS (im Unterricht) mit den Modellen interagieren?
- Wie geht man als Lehrperson mit diesen Gefahren auf organisatorischer und technischer Ebene um?

Technisches Setup

TIMTOWTDI - There Is More Than One Way To Do It

Jupyter als technische Basis wurde von mir gewählt, da es weit verbreitet ist, plattformunabhängig und viele Programmiersprachen unterstützt - nicht nur Julia, Python und R, wie der Name suggeriert. Das verweben von Erläuterungen und lauffähigem Code in einem Dokument unterstützt nachvollziehbare (Muster-)Lösungen.

Jupyter-Chatbook als Erweiterung von Jupyter wurde gewählt, da es einen unabhängigen, regelbaren Zugriff auf verschiedene Anbieter von LLM bietet. Somit entsteht kein Vendor-Lock-In bei mir als Lehrperson und die SuS können ausprobieren, wie sich die Ausgaben je nach Anbieter oder gewähltem Modell oder Temperatur unterscheiden. Die Temperatur eines LLM ist ein Modellparameter, der die "Kreativität" und somit das "Halluzinieren" steuert.

Derzeit bietet Jupyter-Chatbook ff. Anbindungen:

- OpenAI (Backend für z.B. ChatGPT)
- MistralAI (OSS)
- LLaMA (Meta, auch als off-line-Version verfügbar, siehe Quellen am Ende des Dokuments)
- PaLM (Google, ich befinde mich immer noch auf der Warteliste für einen API-Zugang zu PaLM2)
- Gemini (Google)

Mit Jupyter-Chatbook werden für alle Anbieter dedizierte Kommandozeilenprogramme mitgeliefert - sehr einfache Benutzerschnittstellen für den schnellen Einstieg. Als Backend können auch [Off-Line-Versionen](#) einiger Modelle verwendet werden. Somit kann die Übertragung und Speicherung aller von den Lernenden getätigten Eingaben auf Server ausserhalb der Schweiz, wie es im Online-Setup zu erwarten ist, erschwert werden.

Der folgende Befehl dient als Beispiel dafür, wie einfache sich diese Kommandozeilenprogramme verwenden lassen. Die gezeigten Ein-/Ausgaben werden des Weiteren zur beispielhaften Berechnung der Kosten verwendet. Die Ausgaben sind direkt und ungefiltert von den LMM übernommen worden, und spiegeln nicht die Meinung des Autors oder des ABZ wieder.

```
openai-playground --max-tokens=4096 'Warum muss man in der Schule programmieren lernen und welche Programiersprache ist dazu am besten geeignet?'
```

führt bspw. zu einer Ausgabe wie dieser:

Es gibt mehrere Gründe, warum es wichtig ist, **in** der Schule Programmieren zu lernen. Zum einen fördert das Programmieren das logische Denken und die Problemlösungsfähigkeiten der Schülerinnen und Schüler. Es hilft ihnen, komplexe Probleme zu analysieren und systematisch zu lösen. Darüber hinaus ist Programmieren eine Schlüsselkompetenz **in** der heutigen digitalen Welt und kann den Schülerinnen und Schülern berufliche Perspektiven **in** der IT-Branche eröffnen.

In Bezug auf die am besten geeignete Programmiersprache für den Unterricht **in** der Schule gibt es keine eindeutige Antwort. Es gibt jedoch einige Programmiersprachen, die besonders gut für Anfänger geeignet sind, da sie leicht zu erlernen und zu verstehen sind. Zu den beliebtesten Programmiersprachen für Anfänger gehören Python, Scratch und Java. Python ist eine einfach zu erlernende Sprache, die **in** vielen Bereichen wie Data Science, Webentwicklung und künstlicher Intelligenz eingesetzt wird. Scratch ist eine visuelle Programmiersprache, die speziell für Kinder entwickelt wurde und eine spielerische Einführung **in** das Programmieren bietet. Java ist eine weit verbreitete und vielseitige Sprache, die **in** vielen Bereichen der Softwareentwicklung eingesetzt wird. Letztendlich hängt die Wahl der Programmiersprache davon ab, welche Ziele der Unterricht verfolgt und welches Programmierkonzept vermittelt werden soll.

Ein anderer Anbieter oder ein anderes Model des gleichen Anbieters oder andere Modellparameter beim gleichen Modell führen bei identischer Eingabe i.d.R. zu anderen Ausgaben.

```
mistralai-playground --max-tokens=4096 'Warum muss man in der Schule programmieren lernen und welche Programiersprache ist dazu am besten geeignet?'
```

In der Schule Programmieren lernen ist wichtig, weil dieses Fach wichtige Fähigkeiten für die zukünftige Karriere und alltägliche Lebensbedingungen vermittelt. Programmieren ermöglicht es, automatisierte Lösungen für komplexe Probleme zu finden und Effizienz und Präzision in technologischen Projekten zu erreichen.

Die bestgeeignete Programmiersprache für Schüler abhängig von ihrem Alter und dem jeweiligen Unterrichtsziel. Allerdings sind folgende Programmiersprachen häufig an Empfehlungen der Experten beteiligt:

1. Scratch: Dies ist eine visuelle Programmiersprache, die für Kinder und Anfänger geeignet ist. Sie unterstützt den Lernprozess durch das Bearbeiten von Blocks, was intuitive und leicht verständlich macht.
2. Python: Python ist eine universell anerkannte, leicht lernbare und effiziente Sprache, die von Anfängern und Professionals gleichermaßen genutzt wird. Sie bietet eine große Gemeinschaft, die Unterstützung und Ressourcen bereitstellt.
3. JavaScript: JavaScript ist eine wichtige Sprache für Webentwicklung, die oft empfohlen wird, weil sie die Möglichkeit bietet, dynamische Webseiten zu erstellen.
4. Swift: Swift ist eine Programmiersprache für iOS und macOS, die von Apple entwickelt und empfohlen wird, um Apps für Apple-Geräte zu entwickeln.
5. Java: Java ist eine universell anerkannte Programmiersprache, die für verschiedene Anwendungen, wie Web- und Desktop-Anwendungen, nutzbar ist. Sie wird auch in der Industrie für groß angelegte Projekte verwendet.

Um die passende Programmiersprache für sich auszuwählen, ist es wichtig, die eigenen Ziele und Interessen zu berücksichtigen und mögliche Unterstützung und Unterrichtsmaterialien zu recherchieren.

Im Gegensatz zum Fire-and-Forget der simplen Kommandozeilen-Tools entstehen in Jupyter-Chatbook, genau wie in der Browserversion von z.B. ChatGPT, verknüpfte Dialoge. Das heisst, alle vorhergehenden Eingaben beeinflussen die aktuelle Ausgabe - ganz ähnlich wie in einem Gespräch zwischen Menschen. Dies erlaubt den SuS die schrittweise Lösung eines Problems.

Als weiteren Aspekt ist die Pseudonymisierung der Zugriffe zu nennen. Wenn eine ganze Schulklasse über einen gemeinsamen, von der Lehrperson verwalteten und bezahlten API-Key zugreift, müssen keine individuellen Accounts für die Schüler angelegt werden und die Frage des Datenschutzes ist entschärft. Die entstehenden Kosten sind abhängig von der Anzahl der Token (ca. 1 Token entspricht etwa 4 Zeichen) und generell in ihrer Höhe überschaubar.

		Input pro 1000 token	Output pro 1000 token
OpenAI	gpt-3.5-turbo	\$ 0.0005	\$ 0.0015
MistralAI	mistral-tiny	€ 0,00014	€ 0,00042

(Stand 12.2.2024)

Die obige Eingabe an OpenAI (gpt-3.5-turbo ist derzeit das Default-Model) im obigen Beispiel hat demnach ca. 0.0000135 Dollar und die Antwort 0.00051788 Dollar gekostet.^[1]

Es gibt sicherlich noch andere Möglichkeiten, diese LPU technisch umzusetzen. Das von mir gewählte, eher offene Setup ist nur eine unter vielen Möglichkeiten - auch nahezu jede bekannte IDE bietet in ihrer kommerziellen Versionen entsprechende Plug-Ins. In meinen Augen sind komplexe IDE's jedoch nicht geeignet für Programmierneinsteiger. Aber das ist wieder ein ganz anderes Thema ...

Quellen

Rainer Hattenhauer, 2024, ChatGPT & Co. - Wie du KI richtig nutzt - schreiben, recherchieren, Bilder erstellen, programmieren. Rheinwerk Verlag, Bonn, 1. Auflage, ISBN 978-3-8362-9733-2

Christian Spannagel, Rules for Tools , Pädagogische Hochschule Heidelberg, [link](#) ,

- C't: KI-Praxis - 2023 - Mit künstlicher Intelligenz produktiv arbeiten, [links](#)
- C't: [KI-Praxis](#), Jo Bager, Fehler der Sprach-KI umgehen, S. 49
- C't: [KI-Praxis](#), Dorothee Wiegand, Mit CHatGPT & Co. lernen und arbeiten, S. 142

C't: 2024 - Schlechte Code-Qualität durch die KI-Assistenten GitHub Copilot und ChatGPT, [link](#)

Denny, Paul et.al. - 2024 - Computing Education in the Era of Generative AI, CACM, [link](#)

Liffiton, Mark H. et.al. - 2023 - CodeHelp: Using Large Language Models with Guardrails for Scalable Support in Programming Classes, [link](#)

Solow, Daniel, 2014, Linear programming: an introduction to finite improvement algorithms, ISBN 9780486493763, Dover Publications, Inc.

Sascha Lobo, 2024, Fake News war gestern. Die Ära der Fake Reality beginnt, Spiegel Online, [link](#)

Problem-Sets:

- https://www.rosettacode.org/wiki/Category:Programming_Tasks
- <https://projecteuler.net/>
- <https://rosalind.info/problems/topics/>
- <https://soi.ch/>
- <https://exercism.org/>
- <https://adventofcode.com/>
- <https://code.golf/>
- <https://benchmarksgame-team.pages.debian.net/benchmarksgame/description/simple.html#simple>
- <https://programmieraufgaben.ch/>
- <https://theweeklychallenge.org>
- <https://icpcarchive.github.io/>
- <https://codeforces.com/>
- <https://www.spoj.com/>
- <https://github.com/shekhargulati/99-problems>
- <https://github.com/Raku/examples/tree/master/categories>

AI:

- [Dall°E](#) (Images from Text)
- [Midjourney](#) (Images from Text)
- [Suno](#) (Music from Text)
- [OpenAI Sora](#) (Video from Text)
- [PerplexityAI](#) (Search Engine)
- [Devin](#) (IT Projects Planing)
- [ChatGPT](#) (LLM)
- [MistralAI](#) (LLM)
- [Llama](#) (LLM)
- [PaLM](#) (LLM)
- [OpenAI Tuning](#) (Model tuning)

Sprachen & Tools:

- [Repo mit diesen Unterrichtsmaterialien und Setup](#)
- [Obsidian](#) (Markdown Editor für Dokumente dieser LPU)
- [NixOS](#)
- [Brew](#)
- [Python](#)
- [Jupyter](#)

Chat Beispiele:

- [Reshape matrix](#)
- [Monotonic array](#)
- [AssetStorm](#)
- [Processing Carlson Putin Interview/](#)

1. Berechnung der Kosten von 0.0000135 Dollar für die Frage: echo 'Warum muss man in der Schule programmieren lernen und welche Programiersprache ist dazu am Besten geeignet?' | raku -e 'say \$*IN.comb.elems / 4 / 1000 * 0.0005'

↩