TU WIEN Informatics

Programmieren lernen - Sticht KI die klassische Lernmethode?

ChatGPT vs E-Book:

eine Analyse der klassischen Lehrmethode E-Book und der aktuellen interaktiven Methode künstliche Intelligenz ChatGPT

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Visual Computing

eingereicht von

DI Mag. Christian Johannes Tomaschitz, BSc

Matrikelnummer 00051491

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Associate Prof. Dipl.-Ing.in Dr.in techn. Hilda Tellioglu

Donnerskirchen, 10.07.2025 Christian Johannes Tomaschitz

TU WIEN Informatics

Learning to Program - Is learning with an AI better than the classical approach?

ChatGPT vs E-Book:

An analysis of the traditional learning method E-Book and the modern interactive method artificial intelligence ChatGPT

DIPLOMA THESIS

for the academic title

Diplom-Ingenieur

in

Visual Computing

by

DI Mag. Christian Johannes Tomaschitz, BSc

Student number: 00051491

to the Faculty of Informatics

Technical University of Vienna

Supervisor: Associate Prof. Dipl.-Ing.in Dr.in techn. Hilda Tellioglu

Donnerskirchen, 10.07.2025 Christian Johannes Tomaschitz

Erklärung zur Verfassung der Arbeit

- “Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit – einschließlich Tabellen, Karten und Abbildungen –, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.”
- “Ich erkläre weiters, dass ich mich generativer KI-Tools lediglich als Hilfsmittel bedient habe und in der vorliegenden Arbeit mein gestalterischer Einfluss überwiegt. Im Anhang „Übersicht verwendeter Hilfsmittel“ habe ich alle generativen KI-Tools gelistet, die verwendet wurden, und angegeben, wo und wie sie verwendet wurden. Für Textpassagen, die ohne substantielle Änderungen übernommen wurden, habe ich jeweils die von mir formulierten Eingaben (Prompts) und die verwendete IT-Anwendung mit ihrem Produktnamen und Versionsnummer/Datum angegeben.”

Donnerskirchen, 10. 07. 2025

Christian Johannes Tomaschitz

Kurzfassung

Diese Masterarbeit präsentiert eine umfassende experimentelle Analyse, die darauf ausgerichtet ist, die Effektivität von ChatGPT im Vergleich zu herkömmlichen Lehrmethoden durch den Einsatz eines Lehrbuchs zu beurteilen. Die Untersuchung erfolgte im Kontext des Informatikunterrichts an einem Oberstufenrealgymnasium in Österreich, indem die Programmiersprache Python an Schülerinnen und Schüler der 10. und 11. Schulstufe im Alter von 16 bis 18 Jahren vermittelt wurde.

Über einen Zeitraum von acht Wochen wurden 35 Lernende, die zuvor keine Kenntnisse in Programmieren besaßen, in das Experiment einbezogen. Ihnen wurde die Aufgabe zugewiesen, Programmieren mit der Programmiersprache Python und der Entwicklungsplattform TigerJython zu entwickeln. Zwei Gruppen wurden konstituiert und die Zuteilung der Schülerinnen und Schüler zu den beiden Gruppen wurde zufällig durchgeführt. Gruppe A nutzte das Lehrbuch „Programmieren in Python“ von Tobias Kohn, während Gruppe B mit ChatGPT arbeitete. Zur Erfassung des Ausgangsniveaus der Teilnehmer wurde zu Beginn der Untersuchung ein Eingangstest durchgeführt, um ihr vorheriges Wissen zu bewerten.

Die Untersuchung beinhaltete am Ende zwei Tests und ein abschließendes Projekt, um den Wissensstand zu ermitteln und einen unmittelbaren Vergleich der Lernergebnisse zwischen den Gruppen zu ermöglichen. Ein spezieller Fokus wurde auf die Gewinnung von Programmierfähigkeiten gelegt, ob die Verwendung von ChatGPT zu einem vertieften Verständnis von Programmierkonzepten beitrug oder das traditionelle E-Book. Darüber hinaus analysierte diese Arbeit die Entwicklung und Anwendung von Selbstlernfähigkeiten der Schülerinnen und Schüler innerhalb ihrer jeweiligen Gruppen, was für die autonome Bewältigung neuer Herausforderungen und die Ausdehnung ihres Wissens unabdingbar ist. Die traditionelle Datenanalyse wurde durch Reflexionen und Protokolle erweitert, um Einblicke in ihre individuellen Lernerfahrungen, Präferenzen und wahrgenommenen Herausforderungen zu erlangen.

Die Ergebnisse dieser Studie liefern Einblicke in die Effektivität des durch künstliche Intelligenz unterstützten Lernens im Vergleich zu herkömmlichen Verfahren und unterstreichen das Erfordernis einer differenzierten Anwendung diverser Unterrichtsmethoden im Bereich der Informatik.

Beide Lernmethoden, künstliche Intelligenz und E-Book, hatten unterschiedliche Vorzüge:

- Schüler, die das Lehrbuch nutzten, erlangten ein vertieftes Verständnis für Programmierprinzipien, während diejenigen,
- die mit ChatGPT arbeiteten, verbesserte und beschleunigte Problemlösungskompetenzen, Strategien und einen umfangreicheren Überblick über die gestellten Aufgaben aufwiesen.

Diese Ergebnisse betonen die Wichtigkeit der Verschmelzung traditioneller und zeitgenössischer Unterrichtsmethoden, um den diversen Lernanforderungen der Schülerinnen und Schüler zu entsprechen und sie optimal auf zukünftige Herausforderungen in einer zunehmend digitalisierten Welt vorzubereiten.

Abstract

This master's thesis presents a comprehensive experimental analysis aimed at assessing the effectiveness of ChatGPT compared to traditional teaching methods using a textbook. The study was conducted in the context of computer science classes at a secondary school in Austria, where the programming language Python was taught to students in the 10th and 11th grades, aged 16 to 18 years.

Over a period of eight weeks, 35 learners who had no prior knowledge of programming were included in the experiment and assigned the task of enhancing their programming skills using Python and TigerJython as the development platform. Two groups were formed, and the allocation of students to the two groups was carried out randomly. Group A primarily used the textbook "Programming in Python" by Tobias Kohn, while Group B worked with ChatGPT. To assess the participants' baseline level, an entrance test was conducted at the beginning of the study to evaluate their prior knowledge.

The investigation ultimately included two tests and a final project to assess the level of knowledge and allow for an immediate comparison of learning outcomes between the groups. A special focus was placed on acquiring programming skills, whether the use of ChatGPT contributed to a deeper understanding of programming concepts compared to traditional E-Books. Furthermore, the study analyzed the development and application of self-learning skills among the students within their respective groups, which is essential for autonomously tackling new challenges and expanding their knowledge. The traditional data analysis was expanded through analysis of the learner’s diary of the final project after the investigation to gain insights into their individual learning experiences, preferences, and perceived challenges.

The findings of this study provide insights into the effectiveness of AI-supported learning compared to traditional methods and emphasize the need for a differentiated application of various teaching methods in the field of computer science.

Both learning methods had their own advantages:

- Students who used the textbook gained a deeper understanding of programming principles,
- while those who worked with ChatGPT demonstrated improved and accelerated problem-solving skills, strategies, and a broader overview of the tasks at hand.

These results emphasize the importance of merging traditional and contemporary teaching methods to meet the diverse learning needs of students and to optimally prepare them for future challenges in an increasingly digitized world.

Inhaltsverzeichnis

1. Einleitung

Die kontinuierliche Digitalisierung hat in den vergangenen Jahren einen signifikanten Einfluss auf den Bildungsbereich ausgeübt und mit der Entwicklung von Large Language Models (LLMs) wie ChatGPT einen neuen Höhepunkt erreicht (Jyothy et al., 2024). Es ist besonders hervorzuheben, wie solche Technologien die Methoden der Vermittlung und des Erwerbes von Programmierfähigkeiten revolutionieren können. In diesem dynamischen Umfeld nehmen maschinelles Lernen und künstliche Intelligenz (KI) in Lehr- und Lernumgebungen stetig an Relevanz zu (Shahid et al., 2023). ChatGPT, ein auf Künstlicher Intelligenz basierendes Instrument, hat sich als innovative Methode zur Vermittlung von Programmierfähigkeiten etabliert, da es ein interaktives und personalisiertes Lernerlebnis bereitstellt und somit ein bedeutendes Forschungsgebiet in der Bildungspraxis darstellt.

Traditionelle Unterrichtsmethoden, wie der Einsatz von Lehrmaterialien, haben über einen langen Zeitraum hinweg den Standard im Informatikunterricht dargestellt. Im Rahmen des gegenwärtigen digitalen Lernens sind diese Methoden jedoch aufgrund ihrer statischen Beschaffenheit und eingeschränkten Interaktivität zunehmend kontrovers (Akinwalere & Ivanov, 2022). Die beschleunigten technologischen Entwicklungen haben zu Lehrmethoden geführt, die nicht nur Wissen vermitteln, sondern sich auch an individuelle Lernanforderungen anpassen. Dies führt zu einer völlig neuen Förderung kritischen Denkens im Programmieren und der Entwicklung von Problemlösungskompetenzen durch LLMs (Sasikala & Ravichandran, 2024).

In Anbetracht der zunehmenden Relevanz der Informatikbildung in diversen Lebensbereichen zielt diese Masterarbeit darauf ab, einen KI-basierten Lehransatz, dargestellt durch den Einsatz von ChatGPT, mit konventionellen, lehrbuchbasierten Lehrmethoden zu kontrastieren. Der Fokus dieser Untersuchung liegt auf dem Erlernen der Programmiersprache Python durch Schülerinnen und Schüler der 6. und 7. Schulstufe eines Oberstufenrealgymnasiums in Österreich, was den 10. und 11. Schulstufen entspricht. Die methodisch fundierte Untersuchung zielt darauf ab, ein vertieftes Verständnis für die Effektivität dieser Unterrichtsmethoden zu erlangen.

Obwohl Künstliche Intelligenz im Bildungssektor zunehmend zum Einsatz kommt, existieren bislang nur wenige empirische Untersuchungen, die die Wirksamkeit von KI-unterstützten Lerntechnologien im direkten Vergleich mit herkömmlichen Lehrmethoden im Programmierunterricht untersuchen (Sedlbauer et al., 2024, Shahid et al., 2023). Obwohl die existierende Literatur die Vorzüge interaktiven, durch künstliche Intelligenz unterstützten Lernens hervorhebt, mangelt es an vergleichenden Untersuchungen, die diese zeitgenössischen Methoden unmittelbar mit traditionellen Ansätzen, wie Lehrbüchern, insbesondere im Rahmen des Erlernens von Programmiersprachen, gegenüberstellen.

Die vorliegende Masterarbeit strebt danach, das Verständnis von Lernmethoden zu vertiefen, indem sie eine komparative Untersuchung der Lerneffekte unter den Studierenden durchführt. Der Fokus liegt auf der Quantifizierung und Gegenüberstellung verschiedener Aspekte des Lernerfolgs, darunter Wissenserweiterung, Verstehen von Programmierkonzepten, Förderung von Problemlösungskompetenzen und Anwendung des Erlernten (Roll & Wylie, 2016) Darüber hinaus wird eine Analyse der Erfahrungen, Vorlieben und Schwierigkeiten der Schülerinnen und Schüler im Umgang mit beiden Methoden durchgeführt, um umfangreiche Erkenntnisse über die Wirksamkeit der jeweiligen Ansätze zu erlangen.

1.1 Forschungsfrage

Die zentrale Fragestellung dieser Untersuchung lautet: „Welche Lernmethode – Lehrbuch oder künstliche Intelligenz – ermöglicht es einer Schülerin oder einem Schüler, das Programmieren eigenständig und ohne die Intervention des Lehrers effizienter zu erlernen?“.

Das statische Lehrbuch ist eine strukturierte, schrittweise Methodik, die mit ChatGPT, einer interaktiven, flexiblen und freiformigen Lernumgebung verglichen wird.

1.2 Hypothese

Die Hypothese postuliert, dass eine strukturierte Herangehensweise mittels E-Books effizienter für das Erlernen der Programmiertechniken ist, da der flexible Ansatz mit ChatGPT zwar kreativer ist, jedoch weniger vorhersehbar bei der Vermittlung der Grundprinzipien der Programmierung ist.

1.3 Methodologie

Die Datensammlung für das Experiment wurde durch eine Kombination aus Aufgabenstellungen, Tests und Reflexionsgesprächen durchgeführt. Gespräche und Diskussionen haben subjektive Lernerfahrungen, Vorlieben und wahrgenommene Herausforderungen erfasst, um signifikante Unterschiede im Lernerfolg zwischen den Gruppen zu ermitteln.

1.4 Bedeutung der Untersuchung

In der Zukunft könnten KI-unterstützte Werkzeuge das herkömmliche Lernen optimieren, indem sie einen erhöhten Grad an Individualisierung und Interaktivität ermöglichen. Die Implementierung von ChatGPT in Verbindung mit traditionellen Lernmethoden könnte den Lernerfolg erheblich verbessern (Luckin et al., 2016). Diese Untersuchung fördert das Verständnis, wie Künstliche Intelligenz-unterstützte und herkömmliche Lernmethoden effizient in den Bildungsprozess eingebunden werden können.

2. Vergleich mit aktueller Forschungsliteratur

In diesem Teil der Arbeit wird die gegenwärtige Forschung zur Anwendung von künstlicher Intelligenz als Lernmethode analysiert und relevante Artikel und Bücher analysiert. Diverse wissenschaftliche Studien, die sich mit den Themen „Lernen und Künstliche Intelligenz“ oder „Programmieren und Künstliche Intelligenz“ auseinandersetzen, werden beschrieben und ihr Sinn für die Arbeit wird analysiert. Leider ist es nicht möglich, alle Publikationen zu diesem Thema vollständig zu umfassen. Deshalb wird eine Analyse und Bewertung der für diese Untersuchung relevantesten Literaturquellen durchgeführt. Die nachfolgenden wissenschaftlichen Untersuchungen widmen sich diesem Thema.

2.1 Artificial Intelligence Teaching and Learning in K-12 from 2019 to 2022: A Systematic Literature Review

Der Artikel „Artificial Intelligence Teaching and Learning in K-12 from 2019 to 2022: A Systematic Literature Review“, verfasst von Rizvi, Waite & Sentance (2023), liefert eine umfangreiche Darstellung der gegenwärtigen Forschung zur KI-Bildung im Bildungsbereich K-12 (Primar- bis Sekundarstufe) im Zeitraum von 2019 bis 2022. Die Verfasser führen eine methodische Literaturüberprüfung durch, um empirische Belege für die Anwendung von KI-basierten Bildungsstrategien bei Kindern und Jugendlichen zu ermitteln und zu analysieren. Der Artikel hebt die zunehmende Relevanz von Künstlicher Intelligenz in diversen Lebensbereichen hervor und betont die Erfordernisse, KI-Kompetenzen bereits im schulischen Kontext zu fördern (Rizvi, Waite & Sentance, 2023). Darüber hinaus wird betont, dass die Bildung in Künstlicher Intelligenz an K-12-Schulen zunehmend an Bedeutung gewinnt (Rizvi, Waite & Sentance, 2023).

Die Autoren haben fünf Forschungsdatenbanken systematisch untersucht und insgesamt 8.175 Artikel identifiziert. Nach einem strengen Auswahlverfahren wurden 28 Studien in die abschließende Untersuchung einbezogen. Die Auswahlkriterien umfassen Faktoren wie die Implementierung von Künstliche- Intelligenz- (KI) und Maschine-Learning-Konzepten (ML) im Unterricht, empirische Studien, den Status einer Peer-Review sowie das Veröffentlichungsdatum (zwischen 2019 und 2022) (Rizvi, Waite & Sentance, 2023). Dieser methodische Ansatz ist von besonderer Bedeutung, um einen umfassenden Überblick über ein rasch fortschreitendes Gebiet wie die KI-Bildung zu erlangen.

Die Autoren verwendeten das SEAME-Modell zur Klassifizierung des vermittelten theoretischen Wissens und identifizierten vier Ebenen des KI-Lernens: Sozial/Ethisch, Anwendung, Modell und Engine. Obwohl die Mehrheit der Studien sich auf die Ebenen „Modell“ und „Anwendung“ fokussierte, wurde die Relevanz ethischer Aspekte und sozialer Verantwortung in der Bildung von Künstlicher Intelligenz zunehmend erkannt (Rizvi, Waite & Sentance, 2023).

Der Artikel erörtert zudem die Schwierigkeiten und unterstreicht die Erfordernisse weiterer Untersuchungen zur Wirksamkeit verschiedener pädagogischer Ansätze und Unterrichtsmethoden in der KI-Bildung. Es wurde ermittelt, dass KI-Bildungsinitiativen häufig von Forschungsinstitutionen und nicht von Schulen selbst geleitet werden, was auf die Erfordernisse einer intensiveren Kooperation zwischen Pädagogen und Wissenschaftlern hindeutet (Rizvi, Waite & Sentance, 2023). Ein weiteres Hindernis ist die Abwesenheit von Normen zur Beurteilung des

Lernfortschritts in KI-Thematiken, was den Vergleich unterschiedlicher Studien erschwert (Rizvi, Waite & Sentance, 2023). Der Artikel stellt fest, dass die KI-Bildung im K-12-Sektor ein neues Forschungsgebiet darstellt und zusätzliche Untersuchungen notwendig sind, um wirksame Unterrichtsmethoden zu ermitteln. Vor allem werden die Einbeziehung ethischer Fragestellungen und die Entwicklung altersgerechter, interaktiver Lernumgebungen als zukünftige Forschungsgebiete betont (Rizvi, Waite & Sentance, 2023).

Dieser Artikel ist von besonderer Bedeutung für meine Untersuchung, da er vergleichbare Aspekte der Erforschung des durch künstliche Intelligenz unterstützten Lernens im Vergleich zu herkömmlichen Methoden thematisiert. Er stellt eine robuste Basis für die Untersuchung der Wirksamkeit des KI-basierten Lernens dar.

Ein zentraler Vergleichsfaktor ist die Effizienz des durch künstliche Intelligenz unterstützten Lernens im Vergleich zu herkömmlichen Verfahren. Rizvi, Waite & Sentance (2023) unterstreichen, dass sowohl künstliche Intelligenz-basierte als auch herkömmliche Lernmethoden ihre jeweiligen Vorzüge aufweisen und keine Methode für alle Lernenden optimal geeignet ist (Rizvi, Waite & Sentance 2023). In dieser Arbeit wird ein vergleichbares Muster festgestellt, da sowohl ChatGPT als auch E-Books abhängig vom Kontext und den Lernanforderungen der Schülerinnen und Schüler effiziente Lernmethoden repräsentierten.

Ein weiterer bedeutsamer Faktor ist die Motivation zum Lernen und die Interaktivität. Die Untersuchung von Rizvi, Waite & Sentance (2023) betont, dass interaktive Lernumgebungen, speziell KI-basierte Systeme, die Lernmotivation und das Engagement der Lernenden steigern, indem sie individualisiertes Feedback und Unterstützung bereitstellen (Rizvi, Waite & Sentance, 2023). Diese Einsicht manifestiert sich in meinen Beobachtungen, bei denen die Gruppe ChatGPT aufgrund der interaktiven Unterstützung motivierter scheint als die Gruppe E-Book.

Die Anwendung von projektbasiertem Lernen und praktischen Anwendungen wurde sowohl in meiner Studie als auch im Artikel betont. Rizvi, Waite & Sentance (2023) verdeutlichen, dass projektbasiertes Lernen eine wirksame Methode zur Vermittlung von KI- und Programmierkenntnissen darstellt, indem Lernenden die Fähigkeit vermittelt wird, theoretische Konzepte auf praxisbezogene Aufgaben zu übertragen (Rizvi, Waite & Sentance, 2023).

Bemerkenswerterweise wiesen die Untersuchungen in den Studien keine signifikanten Differenzen im Lernerfolg zwischen den unterschiedlichen Methoden auf. Trotz diverser Methoden gelang es den meisten Studien nicht, signifikante Differenzen in den Lernergebnissen zu identifizieren (Rizvi, Waite & Sentance, 2023).

Ein weiterer zentraler Faktor ist die Wichtigkeit affektiver Lernziele. Rizvi, Waite & Sentance (2023) unterstreichen die Bedeutung von Faktoren wie Motivation und Einstellung im Lernprozess, insbesondere bei komplexen Themen wie Künstlicher Intelligenz und Programmierung (Rizvi, Waite & Sentance, 2023).

Abschließend verdeutlicht der Artikel von Rizvi, Waite & Sentance (2023), dass KI-unterstützte Lernmethoden und herkömmliche Methoden unterschiedliche, jedoch effiziente Methoden darstellen, um Fähigkeiten zu erlangen. Beide Methoden begünstigen autonomes Lernen und berücksichtigen die Motivation sowie die individuellen Lernstile der Lernenden.

2.2 Teaching Students About Conversational AI Using CONVO, a Conversational Programming Agent

Das Paper „Teaching Students About Conversational AI Using CONVO, a Conversational Programming Agent“, verfasst von Zhu & Van Brummelen (2021), analysiert die Wirksamkeit des Werkzeugs CONVO bei der Vermittlung von Grundprinzipien der künstlichen Intelligenz (KI) an Schüler der Mittelstufe. Konversations-KI-Agenten, wie Amazons Alexa und Apples Siri, sind omnipräsent, doch ihre zugrundeliegenden Prinzipien bleiben für viele Menschen oft unverständlich (Zhu & Van Brummelen, 2021). Angesichts der zunehmenden Bedeutung von Künstlicher Intelligenz in unterschiedlichen Lebensbereichen ist es von zentraler Bedeutung, junge Schüler frühzeitig mit dem Thema vertraut zu machen. Die Studie unterstreicht, dass existierende Entwicklungsplattformen für KI-Agenten häufig schwer zugänglich sind und nicht für Bildungszwecke geeignet sind (Zhu & Van Brummelen, 2021).

Der Fokus der Untersuchung lag auf der Entwicklung eines leicht zugänglichen Werkzeugs (CONVO), das Schülern ermöglicht, KI-Agenten zu erstellen, um ein vertieftes Verständnis für KI-Technologien zu erlangen (Zhu & Van Brummelen, 2021). Die Verfasser strebten danach zu ermitteln, ob Schüler durch den Einsatz von CONVO ein tieferes Verständnis für KI-Konzepte erlangen und ob sich ihre Haltung gegenüber KI dadurch verändert.

Aus technischer Sicht erlaubt CONVO den Lernenden, KI-Agenten mittels natürlicher Sprache zu programmieren (Zhu & Van Brummelen, 2021) Zur Eignung des Tools für Bildungszwecke wurde eine grafische Benutzerschnittstelle (GUI) konzipiert, die den Lernenden die Möglichkeit bietet, Trainingsdaten einzugeben und einfache maschinelles Lernen Modelle zu erstellen. Die Eingaben der Schüler in Modelle konnten durch die Integration von Rasa, einem Framework für maschinelles Lernen, übersetzt werden. Darüber hinaus verwendete CONVO BERT ein vortrainiertes Modell für Sprachverständnis, um die Bedeutung von Benutzeranfragen zu ermitteln (Zhu & Van Brummelen, 2021).

Die Untersuchung erfolgte in einem dreitägigen Online-Workshop mit 15 Schülern der 7. und 8. Klasse (Zhu & Van Brummelen, 2021). Im Verlauf des Workshops wurden Befragungen durchgeführt, um die Veränderungen im Wissen, Verständnis und der Haltung der Schüler gegenüber Künstlicher Intelligenz zu ermitteln.

Die Befunde deuteten darauf hin, dass die Schüler nach dem Workshop ein signifikant tieferes Verständnis für Konzepte der Künstlichen Intelligenz erlangt hatten. Zahlreiche Schüler, die zuvor KI-Agenten als „Black Boxes“ erachteten, waren nun in der Lage, ausführliche technische Erläuterungen zu ihrer Funktionsweise zu liefern (Zhu & Van Brummelen, 2021). Die Lernenden stellten zudem fest, dass KI-Agenten weniger intelligent sind, als sie ursprünglich annahmen, und einen erheblichen Bedarf an menschlicher Programmierung und Daten benötigen (Zhu & Van Brummelen, 2021). Zusätzlich erhöhte sich das Selbstvertrauen der Schüler in ihre Fähigkeiten in der Programmierung (Zhu & Van Brummelen, 2021).

Der Workshop vermittelte den Lernenden grundlegende Konzepte der Künstlichen Intelligenz, einschließlich der Erkenntnis, dass eine erhöhte Menge an Trainingsdaten zu präziseren KI-Modellen führt (Zhu & Van Brummelen, 2021). Die Untersuchung offenbarte jedoch auch, dass einige Schüler Probleme hatten, den Unterschied zwischen „eingeschränkter“ und „uneingeschränkter“ natürlicher Sprache zu begreifen (Zhu & Van Brummelen, 2021).

Die Verfasser räumten einige Limitationen der Untersuchung ein, einschließlich der geringen Stichprobengröße (lediglich sieben Schüler nahmen an der abschließenden Befragung teil) und der Selbstauswahl der Teilnehmenden sowie der Online-Durchführung des Workshops. (Zhu & Van Brummelen, 2021).

Das Forschungspapier endet mit der Feststellung, dass CONVO ein vielversprechendes Instrument darstellt, um Schüler in die Grundprinzipien der Künstlichen Intelligenz einzuführen, und schlägt weitere Untersuchungen vor, um die langfristigen Effekte auf das Verständnis der Schüler für Künstliche Intelligenz zu analysieren (Zhu & Van Brummelen, 2021).

Es existieren signifikante Übereinstimmungen zwischen dieser Arbeit und der Studie von Zhu & Van Brummelen (2021), insbesondere hinsichtlich des autonomen Lernens und der Verwendung von KI-basierten Lerninstrumenten. Beide Untersuchungen unterstreichen das autonome Lernen (Zhu & Van Brummelen, 2021) und erforschen, wie KI-Technologien den Lernprozess fördern können (Zhu & Van Brummelen, 2021). Beide Forschungen verdeutlichen, dass KI-Instrumente den Lernprozess durch interaktive und personalisierte Unterstützung optimieren können (Zhu & Van Brummelen, 2021). Zudem unterstreichen sie, dass der Einsatz solcher Technologien das Wissen und das Selbstvertrauen der Lernenden in ihren technologischen Kompetenzen steigert (Zhu & Van Brummelen, 2021). Ein weiterer signifikanter Aspekt ist das Bewusstsein, dass KI-Agenten stark auf menschliche Daten und Programmierung angewiesen sind, was den Schülern durch die Arbeit mit CONVO bewusstwurde (Zhu & Van Brummelen, 2021).

Abschließend verdeutlichen diese Arbeit als auch das Forschungspapier von Zhu & Van Brummelen (2021), dass die Anwendung von Künstlicher Intelligenz bedeutsame Erkenntnisse für die Programmierung liefert und ein vertieftes Verständnis für komplexe Konzepte begünstigt. Beide Untersuchungen weisen darauf hin, dass KI-unterstützte Instrumente das Lernerlebnis signifikant optimieren können, indem sie eine effiziente Verbindung zwischen theoretischem Verständnis und der praktischen Anwendung herstellen.

2.3 Students' reflections on their experience with ChatGPT

„Students' reflections on their experience with ChatGPT“ ist ein Paper, das analysiert die Wahrnehmung und Reflexion von ChatGPT durch Studierende an Universitäten (Šedlbauer et al., 2024). Der wissenschaftliche Artikel beschreibt die Konsequenzen, Schwierigkeiten und möglichen Vorteile, die ChatGPT für Studierende mit sich bringt, und verdeutlicht, dass der Einsatz von generativen KI-Instrumenten im akademischen Kontext eine Vielzahl ethischer und praktischer Fragestellungen aufwirft (Šedlbauer et al., 2024). ChatGPT kann diverse Lernvorteile bieten, einschließlich der raschen Informationsgewinnung, einer verbesserten Kenntnis komplexer Themen, der Förderung von Problemlösungskompetenzen und personalisiertem Lernen. Zugleich besteht das Risiko der Verbreitung von Desinformationen, einer möglichen Beeinträchtigung des kritischen Denkens sowie potenziellen Bedenken hinsichtlich des Datenschutzes (Šedlbauer et al., 2024). Die Untersuchung beabsichtigte, ein vertieftes Verständnis für die Erfahrungen der Studierenden mit ChatGPT zu erlangen und die Schwierigkeiten zu untersuchen, die sich bei der Verwendung für akademische Zwecke ergeben (Šedlbauer et al., 2024).

Die Studie beinhaltete 25 Studierende, die an einem interdisziplinären Kurs im Bereich der Umweltwissenschaften an einer tschechischen Universität eingeschrieben waren. Diese

Studierenden verfassten akademische Essays unter Verwendung von ChatGPT und reflektierten über ihre Erfahrungen. Die Befunde der Untersuchung offenbarten, dass die Ansichten der Studierenden zu ChatGPT von Neugier bis hin zu Skepsis variierten. Obwohl ChatGPT von einigen als nützlich und relevant erachtet wurde, bewerteten andere die Antworten als oberflächlich oder schwer zu überprüfen (Šedlbauer et al., 2024). Ein Hauptresultat der Untersuchung war, dass ChatGPT das kritische Denken der Studierenden begünstigte, insbesondere wenn sie die Grenzen der Künstlichen Intelligenz erkannten (Šedlbauer et al., 2024). Zusätzlich betrachteten einige Studierende ChatGPT als eine Art Dialogpartner, was zu einer Personifizierung der Künstlichen Intelligenz führte - ein Phänomen, das ihr Verständnis der Natur von Künstlicher Intelligenz verzerren könnte (Šedlbauer et al., 2024).

Die Untersuchung endet mit der Beobachtung, dass ChatGPT bereits fest in der Bildungsinfrastruktur verankert ist und dies voraussichtlich auch weiterhin bleiben wird. Sie empfiehlt eine Verlagerung der Bildung hin zu einem erfahrungsorientierten, personalisierten Lernen, wobei ChatGPT als unterstützendes Instrument verwendet werden sollte (Šedlbauer et al., 2024).

In dieser Arbeit zum autonomen Erlernen von Python-Programmierung durch E-Books und ChatGPT stellten sich zahlreiche Parallelen zu den Befunden von Šedlbauer et al. (2024) heraus. Beide Untersuchungen legen nahe, dass ChatGPT ein wirksames Instrument für das autonome Lernen darstellt und das kritische Denken der Lernenden stimulieren kann. In beiden Situationen wurde festgestellt, dass ChatGPT den Lernprozess individualisierte und die Güte der Lernergebnisse erhöhte (Šedlbauer et al., 2024).

2.4 A SWOT analysis of ChatGPT: Implications for educational practice and research

Ein weiteres wissenschaftliches Werk, „A SWOT analysis of ChatGPT: Implications for educational practice and research“, von Farrokhnia et al. (2023), liefert eine umfassende Analyse der Stärken, Schwächen, Möglichkeiten und Risiken, die mit der Anwendung von ChatGPT im Bildungsbereich assoziiert sind. Das Forschungspapier liefert durch den Einsatz des etablierten SWOT-Analysemodells eine umfangreiche Beurteilung der Effekte von ChatGPT auf Bildungsprozesse (Farrokhnia et al., 2023).

Farrokhnia et al. (2023) unterstreichen die Vorzüge von ChatGPT als KI-basiertes Instrument, einschließlich der Fähigkeit, menschenähnliche und kontextuell relevante Antworten zu erzeugen und die Anpassungsfähigkeit an spezifische Lernanforderungen. Das Tool ermöglicht einen schnellen und unkomplizierten Zugang zu Informationen und stellt somit ein wirksames Werkzeug für den Bildungsbereich dar (Farrokhnia et al., 2023, S. 5-7). Das Forschungspapier hebt jedoch gleichzeitig Schwächen hervor, wie das Fehlen eines umfassenden Verständnisses, das Risiko der Reproduktion gesellschaftlicher Vorurteile und die Schwierigkeit, die Qualität der Antworten zu beurteilen (Farrokhnia et al., 2023, S. 8-11).

Das Forschungspapier betont zudem, dass ChatGPT diverse Möglichkeiten im Bildungsbereich eröffnet, wie beispielsweise die Individualisierung des Lernprozesses und die Unterstützung bei der Entwicklung kritischer Denkkompetenzen. Zugleich existieren Risiken, einschließlich des Risikos, dass Lernende eine Abhängigkeit von der Künstlichen Intelligenz entwickeln, und potenzieller Konsequenzen für die akademische Integrität (Farrokhnia et al., 2023). Die Autoren befürworten schließlich eine sorgfältige und bewusste Einbindung von ChatGPT in

Bildungsprozesse und fordern, sowohl Lernende als auch Lehrende in der verantwortungsbewussten Handhabung von KI-Instrumenten zu schulen (Farrokhnia et al., 2023).

Die Übereinstimmungen zwischen dieser Untersuchung und dem wissenschaftlichen Werk von Farrokhnia et al. (2023) verdeutlichen, dass ChatGPT als KI-basiertes Lerninstrument das autonome Lernen, die Förderung kritischer Denkkompetenzen und die Qualität der Lernergebnisse unterstützen kann. Es ist jedoch erforderlich, es sorgfältig einzusetzen, um potenzielle Risiken zu reduzieren.

2.5 Comparative Analysis: ChatGPT vs Traditional Teaching Methods

Der wissenschaftliche Beitrag von Shahid et al. (2023) analysiert die Anwendung von ChatGPT im Vergleich zu herkömmlichen Lehrmethoden beim Erlernen von Englisch als Fremdsprache. Die Verfasser unterstreichen, dass ChatGPT als KI-Instrument an Relevanz gewinnt und das Potenzial hat, die Lehr- und Lernmethoden von Fremdsprachen grundlegend zu modifizieren. In einer dynamisch fortschreitenden Bildungslandschaft gewinnt die Anwendung von Technologie zunehmend an Bedeutung, was neue Perspektiven für die Pädagogik eröffnet. Die Untersuchung zielte darauf ab, die Effektivität, Vorzüge und Nachteile der Anwendung von ChatGPT im Vergleich zu konventionellen Lehrmethoden zu untersuchen und somit zur Debatte über die zukünftige Nutzung von KI-basierten Werkzeugen in der Sprachbildung beizutragen (Shahid et al., 2023).

In ihrer Studie formulierten Shahid et al (2023) diverse spezifische Forschungsziele. In erster Linie beabsichtigten sie, die Effekte von ChatGPT und herkömmlichen Lehrmethoden auf die Sprachkompetenz von Lernenden zu kontrastieren. Als zweiten Schritt analysierten sie das Engagement und die aktive Beteiligung der Schüler bei der Anwendung von ChatGPT im Vergleich zu konventionellen Verfahren. Letztendlich untersuchten sie, inwiefern ChatGPT effektiv an die spezifischen Lernanforderungen der Schüler angepasst werden konnte.

Die Literaturanalyse der Untersuchung untersucht den gegenwärtigen Forschungsstand hinsichtlich ChatGPT und seiner Funktion im Sprachunterricht. Er unterstreicht die Vorzüge von ChatGPT, speziell in Bezug auf Echtzeit-Feedback, Interaktivität und die Möglichkeit, personalisiertes Lernen zu ermöglichen. Darüber hinaus werden Untersuchungen angeführt, die aufzeigen, wie ChatGPT zur Verbesserung von Konversationskompetenzen sowie Grammatik und Wortschatz beitragen kann, was eine Basis für seine Anwendung im Unterricht der englischen Sprache darstellt. Zugleich erörtern Shahid et al. (2023) die Vorzüge und Nachteile herkömmlicher Unterrichtsmethoden, die auf etablierten pädagogischen Konzepten gründen.

Die Methodologie der Untersuchung basiert auf einem Mixed-Methods-Ansatz, der quantitative und qualitative Methoden kombiniert, um ein ganzheitliches Verständnis der Wirksamkeit von ChatGPT und herkömmlichen Unterrichtsmethoden zu liefern. Die Stichprobe umfasste etwa 200 Schüler der englischen Sprache, 20 Lehrkräfte und 10 Experten im Bereich der Künstlichen Intelligenz. Die Datensammlung wurde durch Befragungen, Fragebögen, Interviews und Beobachtungen durchgeführt, um Informationen über Sprachfähigkeit, Engagement, ethische Aspekte und Anpassungsfähigkeit zu erheben. Die nachfolgende Datenauswertung beinhaltete sowohl statistische Methoden als auch thematische Untersuchungen (Shahid et al., 2023).

Die Befunde der Untersuchung offenbarten signifikante Differenzen zwischen den beiden Methoden. Lernende, die ChatGPT einsetzten, zeigten signifikante Fortschritte in ihren Schreibkompetenzen ($p < 0,05$) im Vergleich zu jenen, die konventionelle Methoden angewandt

haben. Ungefähr 60 Prozent der Studienteilnehmer berichteten, dass ChatGPT ihre Schreibfähigkeiten verbessert hat (Shahid et al., 2023). Zudem verzeichneten Lernende, die ChatGPT verwendeten, signifikante Verbesserungen in ihren Kommunikationsfähigkeiten ($p < 0,01$), wobei 60 % der Befragten berichteten, dass sich ihr gesprochenes Englisch verbessert habe. Allerdings wurde ermittelt, dass herkömmliche Methoden bei der Erweiterung des Wortschatzes effizienter waren, da 60 % der Studierenden berichteten, dass ChatGPT in diesem Bereich weniger vorteilhaft war.

In Bezug auf das Engagement und die Beteiligung der Lernenden wurde ermittelt, dass diese aktiver und motivierter waren, wenn sie ChatGPT verwendeten. Ungefähr 70 % der Befragten berichteten, dass ChatGPT zu einer gesteigerten Beteiligung an Sprachlernaktivitäten führte (Shahid et al., 2023). Allerdings wurden Bedenken hinsichtlich ethischer Aspekte formuliert, insbesondere in Bezug auf den Datenschutz und das Risiko, dass Lernende die Antworten von ChatGPT ohne kritische Beurteilung übernehmen können.

Die Untersuchung erörtert die Vorzüge von ChatGPT, insbesondere hinsichtlich der Verbesserung der Schreib- und Konversationskompetenzen von Lernenden der englischen Sprache. Sie betont gleichzeitig die Limitationen des Tools, insbesondere bei der Entwicklung des Wortschatzes (Shahid et al., 2023). Die Verfasser unterstreichen, dass die Adaptivität und Interaktivität von ChatGPT es zu einem wirksamen Instrument im Sprachunterricht machen, jedoch auch ethische Auswirkungen sorgfältig in Betracht gezogen werden müssen.

Zum Abschluss deutet die Untersuchung darauf hin, dass ChatGPT ein wertvolles komplementäres Instrument im Unterricht der Englischen als Fremdsprache darstellen kann, insbesondere bei der Verbesserung der Schreib- und Konversationskompetenzen. Es ist jedoch von Bedeutung, Aspekte der Datensicherheit, des ethischen Einsatzes und des Plagiatsrisikos zu adressieren, um eine verantwortungsvolle Verwendung des Tools zu sichern. Shahid et al. (2023) raten zukünftig dazu, die Funktion von ChatGPT bei der Wortschatzentwicklung zu verbessern und die effiziente Integration von KI-Tools in den Sprachunterricht zu gewährleisten.

Es existieren zahlreiche Übereinstimmungen zwischen dieser Untersuchung und der Arbeit hier, die den Vergleich zwischen autonomem Lernen mit ChatGPT und einem E-Book beim Erlernen von Python analysiert. Beide Studien betonen, dass ChatGPT als effiziente Ergänzung zu herkömmlichen Lernmethoden das Engagement der Lernenden und die Lernergebnisse optimieren kann (Shahid et al., 2023). Die Resultate beider Untersuchungen betonen das Potenzial von ChatGPT als multifunktionales Instrument in der Bildung, welches die Interaktivität und Anpassungsfähigkeit des Lernprozesses erhöht.

2.6 ChatGPT for Good? On Opportunities and Challenges of Large Language Models for Education

Der Artikel „ChatGPT for Good?“ Kasneci et al. (2023) analysiert die Möglichkeiten und Schwierigkeiten, die mit der Anwendung großer Sprachmodelle wie ChatGPT in der Bildung verbunden sind. Er liefert einen umfangreichen Überblick über die gegenwärtigen Potenziale, die durch den Einsatz von KI-Technologien im Bildungssektor eröffnet werden, und betont die Risiken und Limitationen, die mit deren Anwendung einhergehen.

Das wissenschaftliche Werk beginnt mit einer Darstellung der Entwicklung großer Sprachmodelle, insbesondere GPT-3 und ChatGPT, und hebt deren Fähigkeit hervor, menschenähnlichen Text zu erzeugen und diverse Sprachaufgaben mit hoher Präzision zu bewältigen. Zentrale Konzepte wie die Transformer-Architektur und Self-Attention-Mechanismen werden dargelegt, um den Erfolg dieser Modelle in der natürlichen Sprachverarbeitung zu illustrieren. Die Wichtigkeit des Vortrainings auf umfangreichen Datensätzen wird betont, da dies entscheidend zur Performance von ChatGPT bei diversen Sprachaufgaben beiträgt (Kasneci et al., 2023).

Das Forschungspapier stellt eine Vielzahl von Optionen auf, wie umfangreiche Sprachmodelle den Lernprozess auf unterschiedlichen Bildungsstufen optimieren können. In der Primarstufe kann ChatGPT als Lernbegleiter fungieren, der Lese- und Schreibkompetenzen fördert und das Grammatikverständnis verbessert. In weiterführenden Bildungseinrichtungen fördert es die Entwicklung von Verständnis und Problemlösungsfähigkeiten in Disziplinen wie Mathematik, Naturwissenschaften und Sprachen, indem es detaillierte Erläuterungen bereitstellt. Auf universitärer Ebene leistet ChatGPT Unterstützung bei Forschungs- und Schreibaufgaben, indem es Zusammenfassungen generiert und Inhalte strukturiert. Zusätzlich hebt die Studie die Vorteile von ChatGPT im kollaborativen Lernen hervor, indem sie das kritische Denken fördert, individualisierte Lernwege unterstützt und Schüler mit Behinderungen einbezieht (Kasneci et al., 2023).

ChatGPT bietet auch für Pädagogen zahlreiche Vorteile. Es kann unterstützen, individualisierte Materialien zu erstellen und individuelle Rückmeldungen an Schüler zu liefern. Darüber hinaus trägt es zur Unterrichtsplanung bei, indem es bei der Konzeption von Lehrplänen, Aufgaben und interaktiven Übungen assistiert. Die automatisierte Untersuchung von Schülertexten ermöglicht eine effizientere Rückmeldung, was den Lehrkräften mehr Zeit für die unmittelbare Interaktion mit den Schülern bereitstellt (Kasneci et al., 2023).

Die Studie behandelt jedoch auch die Schwierigkeiten und Risiken, die sich aus der Anwendung großer Sprachmodelle in der Bildung ergeben. Dies beinhaltet Bedenken hinsichtlich des Urheberrechts, das Risiko der Verstärkung von Vorurteilen und Ungleichheiten durch nicht repräsentative Trainingsdaten sowie die Möglichkeit, dass sowohl Schüler als auch Lehrkräfte übermäßig auf die Modelle angewiesen sein könnten, was dazu führen könnte, dass kritische Denkfähigkeiten und kreatives Problemlösen vernachlässigt werden. Datenschutzprobleme und der erhebliche Energieverbrauch, der für den Betrieb umfangreicher Sprachmodelle notwendig ist, stellen ebenfalls bedeutsame Herausforderungen dar (Kasneci et al., 2023).

Die Studie präsentiert diverse Strategien zur Bewältigung dieser Herausforderungen, einschließlich der Anwendung diverser Daten zur Verzerrungsreduktion, der Ausbildung von Fähigkeiten für Pädagogen bei der effizienten Anwendung von KI-Tools, der Integration ethischer Überlegungen in den Unterricht sowie der Implementierung von Datenschutzbestimmungen und nachhaltigen Nutzungskonzepten (Kasneci et al., 2023).

Im Kontext meiner Masterarbeit, die die Anwendung von ChatGPT im Vergleich zum autonomen Lernen mit einem E-Book im Python-Programmieren analysiert, existieren diverse Parallelen zu den Erkenntnissen dieses wissenschaftlichen Werkes. Beide Studien unterstreichen die Fähigkeit von ChatGPT, den Lernprozess durch die Berücksichtigung individueller Fragen und Schwierigkeiten der Lernenden zu individualisieren. Meine Forschung und das vorliegende wissenschaftliche Werk legen nahe, dass ChatGPT das autonome Lernen begünstigt, indem es den Lernenden die Möglichkeit bietet, eigenständig Fragen zu stellen, Herausforderungen zu bewältigen und ihren Lernfortschritt zu steuern.

In beiden Studien agiert ChatGPT als Lernpartner, der nicht nur Fragen beantwortet, sondern auch Anleitungen und Erläuterungen bereitstellt, die den Lernprozess erleichtern. Sowohl meine Studie als auch das wissenschaftliche Papier erkennen die Schwierigkeiten bei der Nutzung von ChatGPT, insbesondere das Risiko unpräziser oder irreführender Antworten und die Notwendigkeit, diese kritisch zu hinterfragen.

Ein weiterer bedeutsamer Faktor ist die Erörterung ethischer Fragestellungen und die potenzielle Abhängigkeit von ChatGPT. Die vorliegende Untersuchung und das wissenschaftliche Werk unterstreichen, dass die Verwendung von ChatGPT in der Bildung verantwortungsvoll konzipiert werden muss, um zu gewährleisten, dass es nicht als Substitut für autonomes Denken fungiert. Abschließend betonen die Arbeiten, dass ChatGPT als komplementäres Instrument und nicht als vollständige Substitution für herkömmliche Lernmethoden angesehen werden sollte, da es effizient individualisierte Lernoptionen erweitert und Lehrkräfte unterstützt.

Die Gesamtanalyse von Kasneci et al. (2023) verdeutlicht, dass ChatGPT signifikante Vorteile im Bildungssektor bieten kann, jedoch mit einer klaren Strategie und einem bewussten Ansatz angewendet werden muss, um die damit einhergehenden Risiken zu reduzieren. Dies bestätigt und erweitert die Befunde meiner eigenen Untersuchung.

2.7 Struktur des Kurses

Im Rahmen des Lehrplans für das Oberstufen-Realgymnasium in Österreich erhalten die Schülerinnen und Schüler grundlegende Programmierkonzepte. In Anbetracht des umfangreichen Lehrplans sind effiziente Unterrichtsmethoden von besonderer Relevanz (Pellegrino & Hilton, 2012).

Verschiedene Programmiersprachen und Materialien wurden in der Vergangenheit zur Vermittlung von Programmierkenntnissen eingesetzt, wobei Python sich als besonders geeignet erwiesen hat (Lye & Koh, 2014). In Anbetracht gegenwärtiger Fortschritte mit Large Language Models (LLMs) wie ChatGPT wurde die Frage aufgeworfen, ob diese Modelle den Programmierprozess für Anfänger unterstützen und ein effizientes, individuelles Lernerlebnis bereitstellen können.

Dieses Experiment stellt einen Vergleich zwischen der herkömmlichen Lernmethode (E-Book) und der modernen, interaktiven Methode (ChatGPT) in einem spezifisch konzipierten Einführungskurs in die Programmierung dar. Die Lehrveranstaltung präsentiert eine Einführung in die Programmiersprache Python, die die TigerJython-Entwicklungsumgebung verwendet. Das Ziel besteht darin, den Lernenden eine robuste Basis in elementaren Programmierkonzepten zu vermitteln, wobei der Fokus auf der praktischen Anwendung liegt (Grover & Pea, 2013, Kohn 2019).

Der Kurs wurde so konzipiert, dass die Lernenden autonom lernen konnten, entweder durch den Einsatz eines elektronischen Buches oder durch die Interaktion mit ChatGPT. Jeder Schüler wurde mit identischen Übungen ausgestattet, die gemäß dem Prinzip „learning by doing“ bearbeitet werden sollten, mit dem Ziel, sämtliche Aufgaben autonom zu bewältigen (Papert, 1980). Der Kurs umfasste acht hundertminütige Module, die darauf abgestimmt waren, grundlegende Programmierkompetenzen zu vermitteln und das autonome Arbeiten zu unterstützen.

Zu Beginn jeder Einheit wurden Übungen zur Code-Analyse und zum Debugging ausgeführt, um das Verständnis für geschriebenen Code zu intensivieren und die Fähigkeiten zur Problemlösung zu verbessern (Wing, 2006). Es wurde ein Überblick über den Inhalt der jeweiligen Einheit

gegeben, der durch kurze Reflexionen der Lernenden ergänzt wurde, die dazu dienten, ihre Lernerfahrungen aufzuzeichnen.

Der Ablauf des Kurses erfolgte stufenweise: anfänglich wurden grundlegende Programmierprinzipien wie Variablen, Schleifen und Funktionen erläutert, bevor zu erweiterten Themen wie objektorientierter Programmierung übergegangen wurden. Eine ausführliche Sequenz der Einheiten ist im Anhang aufgeführt.

Zur Untersuchung der Wirksamkeit verschiedener Lernmethoden wurden die Schülerinnen und Schüler in zwei Gruppen unterteilt: eine Gruppe erwarb Kenntnisse mittels ChatGPT, während die andere Gruppe das E-Book als primäre und einzige Lernressource verwendete. Beide Gruppen führten identische Übungen durch, wobei der didaktische Ansatz auf eine praktische Anwendung ausgerichtet war (Grover & Basu,2017).

Abbildung 1:Logo von TigerJython

TigerJython ist eine vereinfachte Entwicklungsumgebung (IDE), die speziell für Bildungszwecke im Python-Bereich konzipiert ist. Sie stellt Anfängern, insbesondere Schülern, eine intuitive Benutzerschnittstelle bereit, eine einfache Syntax-Erläuterung und eine Vielzahl integrierter Beispiele und Bibliotheken zur Verfügung. Zudem kann Python-Code ohne umfangreiche Installation ausgeführt werden, was TigerJython optimal für den Einsatz im Lehrkontext geeignet macht.

Nach Abschluss des Kurses wurden zwei Prüfungen durchgeführt, um das erlangte Wissen und den individuellen Lernfortschritt zu bewerten. Der initiale Prüfprozess bestand aus einem Closed-Book-Test ohne Hilfsmittel, gefolgt von einem Open-Book-Test, bei dem sämtliche Ressourcen – einschließlich ChatGPT und Internetzugang – zugänglich waren. Diese Prüfungen dienten der Bewertung des erlangten Wissens und der entwickelten Programmierkompetenzen.

Zum Abschluss entwickelten die Schülerinnen und Schüler individuelle Projekte und dokumentierten ihren Lösungsprozess in einem Lerntagebuch. Die Projekte boten ihnen die Möglichkeit, die erworbenen Konzepte auf reale Problemstellungen anzuwenden. Sie hatten die Option, aus Aufgaben mit verschiedenen Schwierigkeitsgraden (leicht, mittel, schwer) zu wählen und hatten die Möglichkeit, das Projekt als vollständigen Code, Pseudocode oder als schriftliche Anleitung darzustellen (Resnick et al., 2009). Zugleich wurden ihre Erlebnisse und Gefühle während des Programmierkurses dokumentiert.

Dieser Kurs präsentierte somit eine anwendungsorientierte, schülerorientierte Einführung in die Programmierung mit Python und ermöglichte einen unmittelbaren Vergleich zwischen traditionellen und zeitgenössischen Lernmethoden in der Programmierlehre.

3. Aufbau des Experiments

Die Untersuchung in dieser Arbeit implementiert einen explorativen, qualitativen Ansatz nach einem quasi-experimentellen Design, um die Lernprozesse der Oberstufenschüler im Kontext eines Python-Programmierkurses mittels ChatGPT oder E-Book zu analysieren (Creswell, 2014). Da das Hauptaugenmerk nicht auf der statistischen Repräsentativität liegt, sondern auf dem Verständnis und der Erfassung von Lernprozessen und -strategien, wurde bewusst eine qualitative Methodik verwendet. Diese Methodik bietet umfassende Erkenntnisse über die Erfahrungen der Lernenden, insbesondere hinsichtlich ihrer Interaktion mit diversen Lernmaterialien, ihrer Reaktion auf Herausforderungen und ihrer Entwicklung ihrer Programmierkompetenzen im Verlauf des Kurses (Merriam & Tisdell, 2016).

Der Kurs basierte auf Lehrmaterialien, die sowohl eigenständig konzipiert wurden als auch aus dem E-Book von Tobias Kohn „Programmieren lernen mit Python“ bezogen wurden. Ergänzt wurde es am Ende mit einem Buch für Einsteiger in Python „Python: Programmieren für Dummies“. Das E-Book von Kohn vermittelte grundlegende Programmierkonzepte der E-Book-Gruppe, und ChatGPT, ein KI-basiertes Werkzeug, agierte als interaktiver Lernpartner (Kohn, 2019). Alle Schülerinnen und Schüler arbeiteten an identischen Übungen, die entweder aus Tobias Kohns Werk bezogen oder selbst konzipiert wurden.

Die Schülerinnen und Schüler wurden, ungeachtet ihres vorherigen Wissens, zufällig in zwei Gruppen eingeteilt, wo wie zuvor erwähnt, eine Gruppe das E-Book als Lernquelle verwendete, während die andere Gruppe mit der Unterstützung von ChatGPT arbeitete. Diese zufällige Verteilung erlaubte eine authentische Analyse des Lernprozesses in einem realen Bildungsumfeld, wie es von Yin (2018) in den Methoden der Bildungsforschung empfohlen wird. Diese zufällige Verteilung gewährleistete, dass Differenzen in den Lernresultaten auf die angewandten Lernmethoden und nicht auf die individuellen Fähigkeiten der Schülerinnen und Schüler zurückzuführen sind (Patton, 2002).

Die Methodik des Selbstlernens wurde angewandt, um zu ergründen, wie die Lernenden mit den bereitgestellten Lernmaterialien interagieren und welche Lernstrategien sie entwickeln. Diese explorative Beschaffenheit ermöglichte eine Vielzahl von Einblicken in individuelle Lernprozesse, Schwierigkeiten und Erfolge der Lernenden (Yin, 2018).

Die Datensammlung wurde vornehmlich durch mündliche Reflexionen mit einer zufällig ausgewählten Gruppe von Schülerinnen und Schüler aus beiden Gruppen, schriftliche Lerntagebucheinträge und Beobachtungen des Lernverhaltens während des Kurses durchgeführt. In bestimmten Phasen reflektierten die Lernenden über ihre Lernerfahrungen, Schwierigkeiten und Fortschritte und lieferten somit qualitative Erkenntnisse über ihre individuellen Lernprozesse (Lincoln & Guba, 1985). Weiters wurden abschließend zwei Tests, Open-Book und Closed-Book, durchgeführt und ein Projekt in verschiedenen Schwierigkeitsstufen implementiert.

Die Methodik zielte darauf ab, ein profundes Verständnis für die Erfahrungen und Lernprozesse der Lernenden zu erlangen, wie unterschiedliche Lernmaterialien und -methoden die Entwicklung von Programmierfähigkeiten beeinflussen (Merriam & Tisdell, 2016).und sowie KI-gestützte Werkzeuge wie ChatGPT effektiv im herkömmlichen Lernprozess eingebunden werden können.

3.1 Ablauf des Experiments

Der Programmierkurs wurde konzipiert, um den Lernenden in einem kurzen Zeitraum grundlegende Programmierkompetenzen effektiv zu vermitteln und gleichzeitig diverse

Lernmethoden zu vergleichen. Die Schülerinnen und Schüler arbeiteten an 8 Lerneinheiten, hatten kurze Wissenskontrollen verknüpft mit mündlichem Feedback und absolvierten am Ende des Experiments Tests.

3.1.1 Vortest zur Beurteilung des Vorwissens

Ein Vortest wurde zu Beginn des Kurses durchgeführt, um das vorhandene Programmierwissen der Lernenden zu evaluieren. Die Schüler hatten 25 Minuten Zeit alle Aufgaben im Test zu erledigen. Dieser umfasste praktische Programmieraufgaben sowie theoretische Fragestellungen. Das Ziel bestand darin, das Verständnis der Lernenden und ihre Vertrautheit mit elementaren Programmierkonzepten zu evaluieren (Patton, 2002). Die Teilnehmenden hatten die Möglichkeit, ihre Antworten als authentischen Code, Pseudocode oder eine schriftliche Darstellung der Lösung zu präsentieren, um den diversen Wissensständen und Methoden gerecht zu werden.

Folgende Fragen waren Teil des Vortests:

- Erstellen Sie ein Programm, das einen Text ausgibt.
- Schreiben Sie ein Programm, das zwei Zahlen addiert und das Ergebnis anzeigt.
- Schreiben Sie ein Programm, das überprüft, ob eine Zahl gerade oder ungerade ist.
- Schreiben Sie ein Programm, das alle Zahlen von 1 bis 10 ausgibt, indem eine Schleife verwendet wird.
- Definieren Sie eine Funktion, die zwei Zahlen als Argumente akzeptiert und deren Multiplikation zurückgibt.
- Schreiben Sie ein Programm, das eine Liste von Zahlen durchläuft und jede Zahl verdoppelt.
- Was ist der Unterschied zwischen einer if-Bedingung und einer switch-Anweisung in der Programmierung, und wann würden Sie welche verwenden?
- Erklären Sie den Unterschied zwischen einer Funktion und einer Methode in der Programmierung.
- Was ist der Zweck einer repeat-Schleife, und wie unterscheidet sie sich von einer while-Schleife? Geben Sie ein Beispiel, wann Sie welche Schleife anwenden würden.

Das Ergebnis des Vortests erlaubte eine Beurteilung des Programmierwissens der Lernenden. Die diversen Antwortformen – Code, Pseudocode oder beschreibende Lösungen – lieferten bedeutsame Erkenntnisse über die methodischen Vorgehensweisen und das Verständnis der Teilnehmenden.

3.1.2 Aufbau der Einheiten

Nach dem Abschluss des Vortests wurde der eigentliche Programmierkurs eingeleitet. Zur kontinuierlichen Überwachung der Programmierkompetenzen der Schülerinnen und Schüler wurde jede Einheit, mit Ausnahme der ersten Einheit, mit einer kurzen Wiederholung gestartet. Diese Wiederholungen dienten nicht der Beurteilung, sondern der Verfolgung des Lernfortschritts.

Sie beinhalteten diverse Aufgaben wie:

- **Analyse von Code** - die Lernenden analysieren Code und beschreiben, was der Code macht.
- **Vervollständigung des Codes** - die Lernenden waren verpflichtet, unvollständigen Code zu ergänzen, um die Funktionalität des Programms zu gewährleisten.
- **Fehlerbehebung** - die Lernenden erkannten und korrigierten Fehler im vorhandenen Code, um ihre Problemlösungskompetenzen zu verfeinern.
- **Anordnung des Codes** - die Lernenden waren dazu aufgefordert, durcheinander geratene Codezeilen in die korrekte Sequenz zu bringen.
- **Theoretische Fragestellungen** - diese prüften das Verständnis der bisher erörterten Konzepte.
- **Fragen zur Implementierung** - die Lernenden wurden dazu aufgefordert, spezifische Programmierprobleme zu bewältigen.

Danach gab es eine Einführung in die Einheit und die Teilnehmer des Kurses erfuhren, welche Inhalte in dieser Einheit vorgesehen sind. Danach waren die Studierenden dazu angehalten, sowohl ihr praktisches als auch ihr theoretisches Programmierwissen passend zur Einheit zu vertiefen. Sie beschäftigten sich gründlich mit diversen Aspekten der Programmierung, wobei der Schwerpunkt auf autonomem und problemlösungsorientiertem Lernen lag.

Die beiden Gruppen arbeiteten auf unterschiedliche Weise, um die Inhalte der Einheit zu erlernen (E-Book und ChatGPT). Um Vergleichbarkeit zwischen den Gruppen zu haben, lernten beide Gruppen dieselben Inhalte und beide Gruppen bekamen dieselben Aufgaben und Übungen

Die Einheiten wurden so erstellt, dass sie den Lernenden ein praxisorientiertes, anwendungsorientiertes Verständnis der Programmierlogik boten. Im Anhang können die detaillierten Einheiten eingesehen werden.

3.2 Einheit 1: Einführung in Python und TigerJython

Die erste Einheit des Programmierkurses fungierte als fundamentale Einführung in die Welt der Programmierung, wobei der Schwerpunkt auf der Programmiersprache Python und der speziell für Bildungszwecke konzipierten Entwicklungsumgebung TigerJython lag. Das Ziel dieser Einheit bestand darin, den Lernenden eine Einführung zu geben.

In der Einleitung der Einheit wurde den Lernenden wurde dargelegt, wie Programmieren in diversen Bereichen des Alltags angewendet wird und warum es von Bedeutung ist, diese Kompetenzen zu erwerben (Wing, 2006).

Danach sollten die Lernenden ein grundlegendes Verständnis von TigerJython bekommen. Sie beschäftigten sich mit der Installation und Nutzung. Sie installierten und konfigurierten TigerJython auf ihren Computern. Darüber hinaus erstellten die Schülerinnen und Schüler der ChatGPT-Gruppe ein gratis Konto auf openai.com, um die künstliche Intelligenz in Form von ChatGPT nutzen zu können.

Als nächstes setzten sich die Schülerinnen und Schüler mit ersten Programmieraufgaben in TigerJython auseinander. Sie wurden mit der elementaren Syntax von Python vertraut gemacht und erlangten ein Verständnis für die Funktionsweise einer integrierten Entwicklungsumgebung (IDE) (Kohn, 2019). Die ersten sehr einfachen, praktischen Übungen motivierten die Studierenden,

sich intensiver mit der Programmierung auseinanderzusetzen und ein Interesse für das Thema zu entwickeln.

Die praktische Tätigkeit begann mit der Entwicklung eines Programms namens „Hello, World!“. Dieses traditionelle Beispiel erlaubte den Lernenden, den vollständigen Prozess des Schreibens, Durchführens und Debuggens von Code zu erfahren (Lister et al., 2004). Diese Übung vermittelte ihnen ein grundlegendes Verständnis der Python-Syntax und ermöglichte ihnen eine direkte Erfahrung mit der Schreib- und Interpretation von Programmcode.

Nach dem Erfolg der ersten Übung wurden die Lernenden in die grundlegenden Konzepte von der grafischen Ausgabe von TigerJython eingeführt. Sie erlernten die ersten Befehle zur Steuerung der Turtle und beschäftigten sich in den erweiterten Aufgaben und Übungen mit Variablen und die Verarbeitung von Daten in einem Programm. Sie kamen mit diversen Datentypen in Kontakt, darunter Ganzzahlen, Gleitkommazahlen und Zeichenketten. Diese praxisorientierte Einführung bildete das Fundament für den weiteren Verlauf des Kurses.

3.3 Einheit 2: Vertiefung des Verständnisses

In der zweiten Einheit des Kurses konzentrierten sich die Lernenden auf die Vertiefung ihres Verständnisses für Variablen in Python und deren Anwendung in mathematischen Kontexten. Es ist von Bedeutung, dass die Lernenden die Verwendung von Variablen zur Speicherung, Verarbeitung und Anwendung von Daten in Berechnungen erlernen.

Das Konzept der Variablen wurde zu Beginn der Einheit wiederholt. Die Lernenden lernten die Fähigkeit, unterschiedliche Datentypen wie Ganzzahlen, Gleitkommazahlen und Zeichenketten kennen. Der Fokus lag auf der Anwendung dieser Variablen in mathematischen Berechnungen zur Bewältigung diverser Aufgabenstellungen. Die Schülerinnen und Schüler erwarben Kenntnisse in der Manipulation und Verarbeitung von Variablenwerten. (Robins, Rountree & Rountree, 2003).

Die Lernenden standen vor einer Vielzahl praktischer Aufgaben, bei denen sie mathematische Operationen mit Variablen ausführen mussten. Sie verwendeten elementare Rechenoperationen wie Addition, Subtraktion, Multiplikation und Division zur Vertiefung ihrer Kompetenzen im Umgang mit Variablen. Diese Aufgaben unterstützten sie dabei, die Anpassungsfähigkeit von Variablen in diversen mathematischen Szenarien und die Relevanz von Variablen im Programmierprozess zu begreifen.

Anschließend beschäftigten sich die Lernenden mit Befehlen und Parametern, da sie bereits grafische Funktionen von TigerJython einsetzten und an das Konzept von Funktionen herangeführt werden sollten.

Als Nächstes wurden die Lernenden in das Konzept der Iteration eingeführt. Sie erwarben ein grundlegendes Verständnis dafür, wie wiederholte Aufgaben in einem Programm implementiert werden können. Diese einführenden Übungen waren der Grundstein für komplexere Schleifen und Kontrollstrukturen in zukünftigen Einheiten. (Lahtinen, Ala-Mutka & Järvinen, 2005).

In diesem Zusammenhang bearbeiteten die Lernenden visuelle Übungen. Das förderte das Verstehen grundlegender Konzepte wie Schleifen und Variablen. Sie nutzten beispielsweise Schleifen zur Erzeugung von Mustern und Formen, was sie dazu befähigte, die Verbindung zwischen wiederholenden Strukturen und grafischen Ergebnissen zu identifizieren (Lye & Koh, 2014).

3.4 Einheit 3: Boolesche Werte, logische Operatoren und Bedingungen

Die dritte Einheit des Programmierkurses basierte auf den zuvor erworbenen Konzepten und initiierte die Lernenden in die Welt der Booleschen Logik und bedingten Anweisungen in Python (Kohn 2019, Robins, Rountree & Rountree, 2003) Das Ziel dieser Einheit bestand darin, das Verständnis der Lernenden für die Kontrolle des Programmverlaufs zu intensivieren und ihnen die Gelegenheit zu bieten, dynamischere und interaktivere Programme zu konzipieren.

Der Hauptteil dieser Einheit beinhaltete die Einführung in boolesche Werte (True/False) und logische Operatoren (AND, OR, NOT). Die Lernenden erlernten die Formulierung von Bedingungen und deren Integration in ihre Programme (Winslow, 1996). Indem sie einfache Beispiele und praktische Aufgaben verwendeten, konnten sie die Funktionsweise logischer Verknüpfungen erkennen und den Prozess eines Programms steuern.

Im Anschluss an die Einführung in die Boolesche Logik wurden die if-, else- und elif-Anweisungen implementiert, um den Programmablauf zu regulieren. Die Lernenden konzipierten Programme, die auf diverse Situationen reagieren und unterschiedliche Aktionen ausführen konnten, je nach erfüllter Bedingung. Sie erwarben Kenntnisse darüber, wie man diese Direktiven effizient implementiert, um ihre Programme dynamischer und flexibler zu gestalten (Grover & Basu, 2017). Diese Übungen unterstützten die Lernenden dabei, ein vertieftes Verständnis dafür zu erlangen, wie Bedingungen genutzt werden, um Programme auf Ereignisse und Eingaben zu reagieren.

Ein weiterer Fokus der Einheit konzentrierte sich auf die Handhabung komplexerer Benutzereingaben. Die Lernenden erlernten die Verarbeitung von Benutzereingaben und die Darstellung von Ausgaben in logischen Formaten. Sie vertieften ihr Verständnis für die Interaktion mit Nutzern und erlangten ein Verständnis dafür, wie Daten auf klare Weise in der Ausgabe präsentiert werden (Ben-Ari, 2001). Durch praktische Übungen waren die Schüler in der Lage, diese Kompetenzen unmittelbar zu nutzen und ihr Verständnis für die Entwicklung interaktiver Programme zu vertiefen.

3.5 Einheit 4: Schleifen und Listen

Zu Beginn der Einheit wiederholten die Schüler die zuvor erworbenen Fähigkeiten. Sie vertieften ihr Wissen durch die Lösung diverser praktischer Aufgaben. Dies ermöglichte ihnen, die effektive Anwendung von Schleifen auf diverse Problemstellungen zu erlernen (Lahtinen, Ala-Mutka & Järvinen, 2005).

Die Einheit des Programmierkurses fokussierte sich auf die Einführung und Vertiefung der Konzepte von Schleifen und booleschen Operatoren in Python. Hierzu wurde für die E-Book-Gruppe ergänzend das E-Book „Python programmieren lernen für Dummies“ zur Verfügung gestellt. Der Grund dafür war es, dass das Buch von Kohn den Fokus auf repeat-Schleifen legte, aber while und for-Schleifen auch von der E-Book-Gruppe erlernt werden sollten.

Die Schüler lernten in grundlegende Methoden zur Automatisierung wiederkehrender Aufgaben kennen und ein fundiertes Verständnis für den Umgang mit Datenstrukturen, wie Listen, wurde vermittelt. Diese beiden Aspekte sind zentrale Elemente der Programmierpraxis und von zentraler Bedeutung für die Entwicklung effizienter und gut strukturierter Programme (Lahtinen et al. 2005).

Die Lernenden erkannten, dass Schleifen nicht nur die Codekomprimierung fördern, sondern auch seine Verständlichkeit und die Lesbarkeit des Codes verbessern. Dies stellt eine zentrale

Fähigkeit dar, um wiederholende Prozesse in der Programmierung zu verstehen (Lahtinen et al., 2005).

Die Lernenden wurden in dieser Einheit sukzessive in diverse Formen von Schleifen eingeführt:

- Repeat-Schleifen: zu Beginn starteten die Schülerinnen und Schüler mit der bekannten Schleifenform und wiederholten die Kenntnisse über den Einsatz.
- Die For-Schleife wurde implementiert, um zu demonstrieren, wie über eine festgelegte Anzahl von Elementen iterieren werden kann. Dies führte zu ersten Erfahrungen der Schüler im Umgang mit Datenstrukturen wie Listen.
- While-Schleifen: Die Schüler erwarben Kenntnisse über die While-Schleife, um Situationen zu begreifen, in denen Wiederholungen fortgesetzt werden, bis eine spezifische Bedingung erfüllt ist. In diesem Zusammenhang lernten sie auch Endlosschleifen.

Die Integration der diversen Schleifentypen ermöglichte es den Lernenden, ihre Anwendungsgebiete zu vergleichen und die geeignete Schleife für diverse Problemstellungen auszuwählen.

Parallel zur Einführung der Schleifen wurden den Lernenden die Grundprinzipien der Listenverarbeitung in Python vermittelt. Listen stellen eine der elementaren Datenstrukturen in Python dar und erlauben die Speicherung mehrerer Werte innerhalb einer einzigen Variable. Die Lernenden erwarben Kenntnisse darüber, wie Listen generiert, Elemente hinzugefügt, entfernt und modifiziert werden. Darüber hinaus beschäftigten sie sich mit der Ansprache einzelner Listenelemente und die Manipulation von Daten innerhalb von Listen (Winslow, 1996).

Die Übungen beinhalteten:

- Die Generierung eigener Listen mit diversen Datentypen.
- Die Flexibilität von Listen durch das Hinzufügen und Entfernen von Elementen.
- Die Ansprache und Modifikation spezifischer Listenelemente.
- Die Integration von Schleifen und Listenabschnitten

Nachdem die Lernenden die Grundprinzipien von Schleifen und Listen beherrschten, konzentrierte sich die Aufmerksamkeit auf die Integration beider Konzepte. Dies umfasste das Iterieren über Listen mit Schleifen, was den Schülern ermöglichte, Daten in einer Liste systematisch zu verarbeiten und zu modifizieren (Lahtinen, Ala-Mutka & Järvinen, 2005).

Die Lernenden wurden auch in komplexere Datenstrukturen wie mehrdimensionale Arrays und Dictionaries eingewiesen. Diese erweiterten Listenstrukturen ermöglichten es ihnen, komplexere Aufgaben zu bewältigen und die Multifunktionalität von Python als Programmiersprache zu erfahren.

Um das erworbene Wissen zu konsolidieren, wurden den Schülern diverse praktische Aufgaben gestellt, die sie eigenständig bewältigen sollten. Diese Aufgaben begünstigten die Beschäftigung mit einfachen und komplexen Listenstrukturen. Diese Unterrichtseinheit ermöglichte es den Schülerinnen und Schülern nicht nur, ein vertieftes Verständnis für die Anwendung von Listen und Schleifen in Python zu erlangen, sondern förderte auch die Fähigkeit, repetitive Aufgaben effektiv zu automatisieren. Die Wichtigkeit der korrekten Verwendung von Schleifen und Datenstrukturen in der Programmierung wurde erkannt und die Erstellung flexibler und leistungsstarker Programme wurde gezeigt. (Grover & Basu, 2017).

3.6 Einheit 5: Funktionen (Befehle) und Gültigkeitsbereiche von Variablen

Die fünfte Einheit konzentrierte sich auf die Vermittlung der Bedeutung der Strukturierung und Wiederverwendung von Code durch den Einsatz von Funktionen. Die Schüler haben in Einheit 2 bereits Befehle und Parameter kennengelernt. Hier soll nochmals der Fokus auf Funktionen gelegt werden und die Anwendung auf die Inhalte der vorangegangenen Einheiten. Neben der Definition einer Funktion wurden Parametern und Rückgabewerten besprochen. Der Fokus konzentrierte sich auf die Vorzüge von Funktionen, insbesondere ihrer Fähigkeit, Code zu strukturieren und wiederverwendbar zu machen. Die Lernenden konzipierten individuelle Funktionen für diverse Aufgabenstellungen, um ihr Verständnis für die Entwicklung und Anwendung von Funktionen zu stärken (Robins, Rountree & Rountree, 2003). Diese Methodik erlaubte es ihnen, schrittweise ein vertieftes Verständnis dafür zu erlangen, wie Code effektiv strukturiert wird.

Die Lernenden erkannten durch praktische Beispiele, wann der Einsatz von Funktionen zweckdienlich ist und wie sie dazu beitragen, effizientere und klarer strukturierte Programme zu erstellen. Dies illustrierte die Vorzüge der Anwendung von Funktionen bei der Problemlösung und unterstützte die Schüler dabei, komplexe Probleme in kleinere, leichter handhabbare Segmente zu zerlegen (Wing, 2006; Grover und Pea (2013).

Das Konzept der Gültigkeitsbereichen von Variablen wurde zusätzlich zu den Funktionen implementiert. Die Lernenden erkannten die Unterschiede zwischen globalen und lokalen Variablen und deren Auswirkungen auf den Programmablauf. Durch praktische Übungen analysierten sie die Funktionsweise von Variablen innerhalb und außerhalb von Funktionen und die Konsequenzen des Variablenbereichs für die Funktionsweise eines Programms (Robins et al., 2003). Dies erweiterte ihr Verständnis für die Verwaltung und Strukturierung von Daten innerhalb eines Programms.

Zusätzlich haben die Schüler an der Steuerung von Schleifen in Python gearbeitet, wobei besonderes Augenmerk auf die „break“- und „continue“-Anweisungen gelegt wurde. Diese Konzepte ermöglichten es ihnen, unter spezifischen Bedingungen Schleifen zu beenden oder zu überspringen. Durch die Anwendung dieser Methoden in diversen Übungen erwarben sie die Fähigkeit, Schleifen mit größerer Genauigkeit zu steuern und wie diese Kontrollstrukturen zur Effizienz eines Programms beitragen.

3.7 Einheit 6: Anwendung der zuvor erworbenen Konzepte

In der sechsten Einheit hatten die Lernenden die Gelegenheit, ihr in den vorangegangenen Programmierstunden erworbenes Wissen zu konsolidieren und zu vertiefen. Diese Einheit fungierte als praktische Sitzung, während der die Lernenden diverse Programmieraufgaben lösten, um ihre Kompetenzen zu verfeinern und potenzielle Verständnisdefizite zu beheben. Die Struktur der Übungen beinhaltete Aufgaben zur Fehlerbehebung, Übungen zur Code-Vervollständigung und die Organisation von Codeabschnitten. Alle Aufgaben wurden konzipiert, um den Lernenden ein tiefgehendes Verständnis der Programmierlogik und -struktur zu vermitteln.

3.7.1 Übungen zum Debugging

Die Schüler wurden mit vorsätzlichen Fehlern in Programmieraufgaben konfrontiert. Ihre Aufgabe bestand darin, diese Fehler zu erkennen und zu beheben. Diese Übungen haben nicht nur ihre Kompetenz zur Erkennung von Syntax- und Logikfehlern verbessert, sondern auch dazu beigetragen, ein vertieftes Verständnis dafür zu erlangen, wie Code funktioniert. Gemäß Fitzgerald et al. (2008) erweisen sich Debugging-Aufgaben als besonders wirksam zur Verbesserung der Problemlösungskompetenzen und zur Vertiefung des Lernprozesses. Beispielsweise waren die Schüler dazu aufgefordert, die korrekte Anwendung von Schleifen und Variablen in einem fehlerhaften Programm zu erkennen und zu korrigieren, was ihnen ermöglichte, das Gesamtergebnis eines Programms zu begreifen.

3.7.2 Vervollständigung des Codes

In dieser Übung wurden den Schülern unvollständige Codepassagen bereitgestellt, die sie eigenständig ergänzen mussten, um spezifische Problemstellungen zu bewältigen. Diese Übungen intensivierten ihr Verstehen der logischen Struktur von Programmabläufen und ermöglichten es ihnen, ihr erworbenes Wissen in neuen Kontexten zu implementieren. Eine typische Aufgabe könnte die Vervollständigung einer Funktion umfassen, die die Summe von Zahlen in einer Liste ermittelt, wobei die Lernenden die korrekte Verwendung von Variablen, Schleifen und Bedingungen erlernen mussten. Diese Form der Übung begünstigte die flexible und kreative Anwendung der bereits erworbenen Programmierkonzepte durch die Schülerinnen und Schüler (Lahtinen, Ala-Mutka & Järvinen, 2005).

3.7.3 Strukturierung von Codesegmenten

In dieser Aufgabe wurden die Lernenden dazu angehalten, diverse Codeabschnitte in der korrekten Sequenz zu positionieren, um ein funktionsfähiges Programm zu konstruieren. Diese Übung förderte ihr Verständnis für die logische Struktur des Programms und die korrekte Sequenz von Direktiven. Durch das Erlernen der Zusammenarbeit zwischen einzelnen Codezeilen in einer spezifischen Sequenz erweiterten die Schüler ihr Verständnis für Programmierstrukturen und -logik. Gemäß Lister (2004) fördert diese Form des „Code-Tracings“ das Verständnis für die Funktionsweise von Algorithmen und stärkt die Fähigkeit, dem Prozess eines Programms nachzuvollziehen.

3.7.4 Anwendung auf praktische Herausforderungen

Diese Einheit erlaubte den Lernenden, ihre Kompetenzen in einem praktischen Kontext zu evaluieren und das erworbene Wissen zu konsolidieren. Der Schwerpunkt lag auf der eigenständigen Problemlösung.

3.8 Einheit 7: Einführung in die objektorientierte Programmierung (OOP)

Die siebte Einheit des Kurses konzentrierte sich auf die Einführung in die objektorientierte Programmierung (OOP), ein zentrales Prinzip der Softwareentwicklung. Das Ziel dieser Einheit bestand darin, den Lernenden ein fundamentales Verständnis dafür zu vermitteln, wie Klassen und Objekte in Python generiert und genutzt werden. Die Lernenden erlernten die Grundprinzipien der Objektorientierten-Programmierung und erkannten, wie diese Methode Programme modularer und strukturierter gestaltet.

Zu Beginn der Unterrichtseinheit befassen sich die Lernenden mit den zentralen Prinzipien der Objektorientierung, einschließlich der Konzepte von Klassen, Objekten, Attributen und Methoden:

- es wurden Klassen als Blaupausen für Objekte implementiert. Die Lernenden erkannten, dass eine Klasse als Muster dient, um bestimmte Charakteristika und Verhaltensmuster (Attribute und Methoden) für diverse Objekte festzulegen.
- Objekte wurden als spezifische Instanzen dieser Klassen definiert, die spezifische Merkmale aufweisen und spezifische Funktionen erfüllen können.

Ein zusätzlicher Fokus der Einheit lag auf der Einbindung von Funktionen innerhalb von Klassen. Die Lernenden erweiterten ihr Verständnis darüber, wie Methoden definiert und innerhalb einer Klasse angewendet werden. Sie stellten fest, dass das Einbetten von Funktionen als Verfahren innerhalb einer Klasse zur Strukturierung des Codes beiträgt und komplexe Aufgaben in kleinere, übersichtliche Module unterteilt. Die Kompetenz, Funktionen durch Methoden innerhalb einer Klasse zu implementieren, vermittelte den Lernenden die Möglichkeit, wiederverwendbaren und gut strukturierten Code zu generieren – ein zentraler Faktor für effizientes Programmieren (Wing, 2006).

Nach Abschluss der Einheit führten die Schüler eine Reflexion über ihre Erfahrungen durch und stellten fest, wie Objektorientierter-Programmierung dazu beiträgt, Programmierprojekte überschaubarer und wartbarer zu gestalten. Diese Kompetenzen stellen die Basis für anspruchsvolle Programmierprojekte dar und rüsten die Lernenden darauf vor, einen strukturierten Ansatz zur Bewältigung komplexer Aufgaben zu verfolgen.

3.9 Einheit 8: Konsolidierung und Anwendung von Programmierkonzepten

Die abschließende Einheit des Programmierkurses bot den Lernenden die Möglichkeit, das in den vorangegangenen Sitzungen erworbene Wissen in einem praktischen Projekt zu implementieren. Das Ziel dieser Einheit bestand darin, die Fähigkeiten der Schülerinnen und Schüler in der Programmierung zu stärken und ihnen eine umfangreiche, anwendungsorientierte Lernerfahrung bereitzustellen.

3.9.1 Wiederholung der Programmierkonzepte

Die Einheit wurde mit einer kurzen Wiederholung der zentralen Programmierkonzepte eingeleitet, die im Verlauf des Kurses behandelt wurden, einschließlich Variablen, Schleifen, Funktionen und den Grundprinzipien der objektorientierten Programmierung. Diese Wiederholung unterstützte die Lernenden dabei, ihr Verständnis zu entwickeln und sich auf die anstehende Projektarbeit vorzubereiten. Gemäß Lye & Koh (2014) fördert eine systematische Wiederholung nicht nur die dauerhafte Gedächtnisbildung, sondern stärkt auch das Selbstvertrauen der Schüler in ihre Programmierkompetenzen.

3.9.2 Konzeption und Ausarbeitung eines eigenen Projekts.

Die Schüler wurden dazu angehalten, ihr Programmierprojekt zu konzipieren und zu realisieren. Sie hatten die Möglichkeit, Themen auszuwählen, die mit ihren Interessen korrespondierten, was ihre Motivation und Kreativität begünstigten. Einige Schüler wählten die Programmierung eines einfachen Textabenteuers, während andere praktische Anwendungen wie einen Taschenrechner oder ein Bankkonto konzipierten. Die Option zur Auswahl ihres eigenen Projekts steigerte die intrinsische Motivation und das Engagement der Schüler (Grover & Basu, 2017).

3.9.3 Fehlerbehebung und Optimierung des Codes

Ein Aspekt dieser Einheit war die Fehlerbehebung. Die Lernenden erwarben die Fähigkeit, Fehler in ihrem Programmcode zu suchen und zu verbessern. Dies förderte ihre Fähigkeiten zur Problemlösung und vertiefte ihr Verständnis dafür, wie Programme funktionieren. Ben-Ari (2001) stellt Debugging-Kompetenzen als einen zentralen Aspekt des Programmierlernens dar, da sie das kritische Denken und die Fähigkeit stärken.

3.9.4 Präsentation und Reflexion

Nach Abschluss der Unterrichtseinheit präsentierten die Schüler ihre Projekte der gesamten Gruppe. Jede Präsentation beinhaltete eine Demonstration des Programms, eine Erläuterung der angewandten Programmierkonzepte und eine Reflexion über die während des Projekts entstandenen Herausforderungen und Lösungsansätze. Diese Phase der Präsentation erlaubte den Lernenden, ihre Kommunikationskompetenzen zu erweitern und ihre Projekte in einem umfangreicheren Kontext zu analysieren. Lister et al. (2004) unterstreichen, dass die Darstellung der eigenen Arbeit das Selbstvertrauen der Lernenden stärkt und ihnen dabei assistiert, komplexe Konzepte eindeutig zu kommunizieren. Im Verlauf der Präsentationsphase erhielten die Schüler konstruktive Rückmeldungen von ihren Klassenkameraden, was nicht nur ihr Verständnis für diverse Ansätze erweiterte, sondern auch ihre Kooperationsfähigkeit bei der Verfeinerung von Ideen verbesserte. Dieser Dialog stellte einen bedeutsamen Fortschritt dar, um zu erkennen, was in der gegenwärtigen Arbeitsumgebung von besonderer Bedeutung ist (Resnick et al., 2009).

Insgesamt erlaubte die achte und letzte Einheit den Lernenden, ihr im Verlauf des Kurses erlangtes Wissen anzuwenden und ein Projekt von der Planung bis zur Präsentation eigenständig zu realisieren. Die Integration von individuellem Lernen und kollaborativem Austausch hat dazu beigetragen, ihre Programmierkompetenzen zu erweitern und ein tieferes Verständnis für die praktische Anwendung von Programmierkenntnissen zu erlangen. Diese Einheit hat sie nicht nur auf komplexere Programmierprojekte vorbereitet, sondern auch ihr Selbstvertrauen in ihre Fähigkeit, komplexe Programmierprobleme eigenständig zu bewältigen. Die gegenwärtige Forschung, wie sie von Hmelo-Silver (2004) bestätigt wird, verdeutlicht, dass projektbasiertes Lernen eine wirksame Methode darstellt, um die Problemlösungskompetenzen der Lernenden zu steigern und ihr Selbstvertrauen dauerhaft zu stärken.

3.10 Closed-Book und Open-Book

Zum Abschluss des Programmierkurses wurden zwei Prüfungen durchgeführt: ein Closed-Book-Test und ein Open-Book-Test. Der Closed-Book-Test wurde konzipiert, um die Programmierkompetenzen der Schüler in Python und TigerJython ohne den Einsatz externer Ressourcen zu evaluieren und eine echte Bewertung ihrer Fähigkeiten und ihres Wissens zu gewährleisten.

3.10.1 Struktur und Inhalt des Closed-Book-Tests

Der Closed-Book-Test, der als „Grundlagen der Programmierung“ bezeichnet wurde, diente der gründlichen Evaluation des Wissens und der praktischen Kompetenzen der Schüler. Die Prüfung erstreckte sich über 45 Minuten und umfasste Aufgaben im Wert von insgesamt 100 Punkten. Sie bestand aus fünf zentralen Segmenten, die verschiedene Aspekte des Programmierwissens abdeckten: Programmierung, Fehlerbehebung, Codevervollständigung, Codeanordnung und Multiple-Choice-Fragen (Linn & Dalbey, 1989).

3.10.1.1 Programmierung (20 Punkte)

- **Aufgabe 1:** Die Lernenden sollten eine Funktion erstellen, die eine Liste von Tieren durchläuft und jedes Tier ausgibt. Für das Tier „Tiger“ sollte eine spezifische Mitteilung dargestellt werden. Diese Aufgabe zielte darauf ab, das Verständnis der Schüler für Schleifen und bedingte Anweisungen zu testen.

- **Aufgabe 2:** Hier sollten die Lernenden ein Programm entwickeln, das Benutzereingaben annimmt und auf Grundlage des eingegebenen Objektnamens eine spezifische Aktion ausführt (z. B. das Zeichnen eines Hauses). Studien zeigen, dass praktische Aufgaben wie diese das Problemlösungsverhalten der Schüler fördern und ihr Verständnis für die Anwendung von Programmierkonzepten vertiefen (Robins, Rountree & Rountree, 2003).

3.10.1.2 Fehlerbehebung (20 Punkte)

Die Lernenden mussten Syntaxfehler in bereitgestellten Codeabschnitten erkennen und korrigieren. Ziel dieser Aufgabe war die Bewertung der Fähigkeiten der Schüler im Bereich Fehleranalyse und -behebung. Laut Fitzgerald et al. (2008) ist Debugging eine zentrale Kompetenz, die das Verständnis für den Programmablauf und die Ursachen von Fehlern schärft.

3.10.1.3 Vervollständigung des Codes (20 Punkte)

Unvollständige Codeabschnitte sollten von den Lernenden so ergänzt werden, dass sie bestimmte Funktionen erfüllen. Zum Beispiel sollten sie verschachtelte Quadrate zeichnen oder eine Entscheidungslogik für das Zeichnen geometrischer Formen auf Grundlage von Zahlenlisten implementieren. Diese Aufgabenform unterstützt die Schülerinnen und Schüler dabei, unvollständige Algorithmen zu analysieren und zu vervollständigen (McCracken et al., 2001).

3.10.1.4 Anordnung des Codes (20 Punkte)

In diesem Segment mussten die Lernenden durcheinandergewürfelte Codezeilen in die richtige Reihenfolge bringen, um ein funktionsfähiges Programm zu erstellen. Diese Aufgabe diente der Überprüfung ihres Verständnisses von Programmstrukturen und logischen Prozessen. Diese Methode gilt als besonders effizient bei der Vertiefung des Verständnisses von Algorithmusstrukturen (Lister et al., 2004).

3.10.1.5 Multiple Choice (20 Punkte)

Der Test enthielt Multiple-Choice-Fragen, die verschiedene Aspekte der Python-Grundlagen abdeckten. Die Lernenden mussten die richtige Antwort aus einer Vielzahl von Möglichkeiten auswählen. Multiple-Choice-Tests sind ein anerkanntes Mittel zur Bewertung des Verständnisses grundlegender Programmierkonzepte (Tew & Guzdial, 2010).

Im Verlauf der Prüfung wurden die Lernenden dazu angehalten, eigenständig zu arbeiten, ohne Hilfsmittel zu nutzen. Diese Methode sollte autonomes Denken fördern (Hmelo-Silver, 2004).

Der Closed-Book-Test ermöglichte eine umfassende Bewertung der Programmierkompetenzen der Lernenden und stellte sicher, dass sie nicht nur theoretisches Wissen, sondern auch praktische Fertigkeiten im Umgang mit Python und TigerJython beherrschten. Der Test lieferte wertvolle Erkenntnisse über den individuellen Lernfortschritt und half dabei, Bereiche zu identifizieren, die zusätzliche Unterstützung und Verstärkung benötigten.

3.10.2 Struktur und Inhalt des Open-Book-Tests

Der Open-Book-Test erstreckte sich über 45 Minuten, 100 Punkte und bestand aus fünf zentralen Bereichen: Programmierung, Fehlerbehebung, Vervollständigung von Code, Anordnung von Code und Multiple-Choice-Fragen. Die Schüler hatten die Möglichkeit, auf alle Ressourcen zurückzugreifen, was die eine praktische Anwendung ihres Wissens erleichterte (Linn & Clancy, 1992).

3.10.2.1 Programmierung (20 Punkte)

In der Programmierung waren folgende Beispiele zu lösen:

- **Beispiel 1:** Die Lernenden waren dazu aufgefordert, ein Programm zu erstellen, das Schülernoten verwaltet, die in einer Liste gespeichert sind, ohne auf verschachtelte Strukturen zurückzugreifen. Es war erforderlich, spezifische Mitteilungen für spezifische Leistungen zu integrieren, um ihre Kompetenzen im Umgang mit Bedingungen und Listen zu evaluieren.
- **Beispiel 2:** Eine zusätzliche Aufgabe erforderte von den Lernenden die Entwicklung eines Programms, das vier zentrierte Quadrate unterschiedlicher Größe in aufsteigender Sequenz darstellt. Diese Aufgabe beurteilte ihre Kompetenz in der Anwendung von Schleifen und der Erstellung geometrischer Zeichnungen (Robins, Rountree & Rountree, 2003).

3.10.2.2 Fehlerbehebung (20 Punkte)

Die Lernenden waren aufgefordert, Fehler in zwei bereitgestellten Codebeispielen zu erkennen und zu beheben. Diese Beispiele konzentrierten sich auf geometrische Zeichnungen, die auf Benutzereingaben basieren, und verlangten, dass die Lernenden ihre Fähigkeiten im Bereich des Debuggings einsetzen. Forschungsarbeiten legen nahe, dass der Einsatz solcher Debugging-Aufgaben den Lernprozess im Programmieren signifikant verbessert und das Verständnis der Schülerinnen und Schüler für die Funktionsweise von Code intensiviert (Fitzgerald et al., 2008).

3.10.2.3 Vervollständigung des Codes (20 Punkte)

In diesem Segment wurde den Lernenden nahegelegt, den fehlenden Code in zwei Szenarien zu ergänzen. Das erste Szenario umfasste die Kalkulation und Darstellung des Volumens eines Quaders, während das zweite sich auf statistische Kalkulationen und Datenfilterung konzentrierte. Laut Lahtinen, Ala-Mutka & Järvinen (2005) fördert das Vervollständigen von Codefragmenten die Entwicklung analytischer Kompetenzen und unterstützt die Lernenden dabei, die Logik hinter spezifischen Programmiermethoden besser zu begreifen.

3.10.2.4 Anordnung des Codes (20 Punkte)

Die Lernenden wurden dazu angehalten, den durcheinandergebrachten Python-Code korrekt zu organisieren und zu strukturieren. Der Programmcode beinhaltete diverse Funktionen wie die Eingabe von Daten, Berechnungen und Ausgaben. Diese Aufgabe soll die Kompetenz der Schülerinnen und Schüler beurteilen, Code logisch zu strukturieren und zu sequenzieren (Lister et al., 2004).

3.10.2.5 Multiple-Choice (20 Punkte)

Der Multiple-Choice-Bereich beinhaltete Fragen zu diversen Konzepten in Python und bewertete das grundlegende Verständnis der Schüler für die Programmiersprache. Die Lernenden waren dazu aufgefordert, eine korrekte Antwort aus einer Vielzahl von Möglichkeiten auszuwählen. Dieses Format ist besonders geeignet zur Evaluierung des konzeptuellen Wissens und zur Erkennung von Verständnisdefiziten (Tew & Guzdial, 2010).

Das Ziel des Open-Book-Tests bestand darin, das Verständnis der Schülerinnen und Schüler für grundlegende Programmierkonzepte, ihre Fähigkeit zur Fehlerbehebung, ihre Anwendung von Logik bei der Codevervollständigung sowie ihre Fähigkeit zur Strukturierung und Sequenzierung von Python-Code zu evaluieren (Grover & Basu, 2017). Der Test erlaubte den Lernenden, ihr Wissen anzuwenden und zu demonstrieren, wie sie Programmieraufgaben mit Python und TigerJython bewältigen können.

3.11 Das Lerntagebuch als Reflexionsinstrument

In dieser Arbeit fungierte das Lerntagebuch als zentrales Werkzeug zur Dokumentation der individuellen Lernprozesse der Schüler nach den Tests. Die Schüler wählten ein Projekt, dass sie mit den erlernten Fähigkeiten umsetzen sollten. Sie hatten zwei Wochen Zeit, um dieses Projekt zu entwickeln und das Projekt inklusive eines Lerntagebuchs abzugeben. Das Lerntagebuch beinhaltete eine Beschreibung, wie sie das Projekt umgesetzt habe. Es ermöglichte den Lernenden, ihre Gedanken, Problemlösungsstrategien, Erkenntnisse und die Entwicklung ihres Verständnisses zu dokumentieren (Grover & Pea, 2013; Mohd Rum und Zolkepli, 2018).

Die primäre Zielsetzung des Lerntagebuchs in dieser Arbeit war es, ein Verständnis dafür zu erlangen, wie die Lernenden mit den unterschiedlichen Lernmaterialien – ChatGPT versus E-Books – interagierten, wie sie das Erlernte anwandten und welche Ressource sie als effizienter für ihren Lernprozess erachteten. Die Anwendung von Lerntagebüchern ermöglichte eine qualitative Untersuchung individueller Lernpfade, der Schwierigkeiten, denen die Lernenden

gegenüberstanden, und der Strategien, die sie entwickelten, um diese Schwierigkeiten zu bewältigen (Merriam & Tisdell, 2015).

Die Schülerinnen und Schüler hatten die Möglichkeit, alle verfügbaren Ressourcen zu nutzen, wodurch die Lerntagebücher ein authentisches und umfassendes Bild ihrer individuellen Lernerfahrungen und ihres Wissensstandes widerspiegelten. Diese Methodik förderte die primären Ziele der Arbeit, indem sie ein umfassendes Verständnis der Stärken und Schwächen beider Lernmedien ermöglichte und die Basis für fundierte Ratschläge zur Gestaltung effizienter Lernumgebungen legte.

Die Aufgaben für das Lerntagebuch waren in drei Niveaus der Komplexität klassifiziert. Sie können im Anhang detailliert eingesehen werden. Folgende Beispiele standen sinngemäß zur Auswahl:

- **Einfach: Grundrechenarten und Eingabe-/Ausgabeverfahren**
  Aufgabe: Entwickeln Sie ein Programm, das den Endpreis nach der Anwendung eines Rabatts errechnet. Das Programm sollte den initialen Preis eines Produkts sowie den Prozentsatz des Rabatts (beispielsweise 20 % Rabatt) vom Nutzer erfassen und den finalen Preis berechnen. Dokumentieren Sie Ihren Ansatz und erläutern Sie die Implementierung der Berechnung.
  Erwartetes Resultat: Die Lernenden erlernen die Handhabung von Benutzereingaben und die Anwendung mathematischer Kalkulationen in einem realen Kontext (Papert, 1980). Es ist erforderlich, die Prozesse zur Kalkulation des Rabatts zu dokumentieren und den Code klar zu strukturieren.
- **Mittel: Anwendung von Schleifen und Bedingungen**
  Aufgabe: Verfassen Sie ein Programm, das als unkomplizierter Temperaturrechner agiert. Das Programm sollte eine Aufstellung von Temperaturen in Celsius erfassen und diese in Fahrenheit konvertieren. Das Programm sollte „Gefrierpunkt erreicht“ für jede Temperatur unter dem Gefrierpunkt (0 °C) ausgeben. Nutzen Sie Schleifen und Bedingungen zur Durchführung der Umrechnung und Überprüfung.
  Erwartetes Ergebnis: Die Lernenden implementieren Schleifen und Bedingungen, um ein praktisches Problem zu bewältigen. Sie erlernen die Anwendung von Iterationen zur Listenverarbeitung und die Anwendung bedingter Anweisungen zur Ausgabe spezifischer Meldungen (Resnick et al., 2009).
- **Herausfordernd: Datenstrukturen und Algorithmen**
  Aufgabe: Erstellen Sie ein Programm für eine Bibliothek, um zu überprüfen, ob ein eingegebener Buchtitel ein Palindrom ist. Die Software sollte eine Aufstellung von Buchtiteln einlesen und jeden Titel auf das Palindrom-Kriterium überprüfen. Für jedes identifizierte Palindrom sollte der Titel sowie die Gesamtzahl aller in der Liste aufgeführten Palindrome angezeigt werden.
  Erwartetes Ergebnis: Die Lernenden arbeiten mit realen Datenstrukturen (Listen von Zeichenfolgen) und konzipieren einen Algorithmus zur Bewältigung eines spezifischen Problems. Sie erlernen die Verarbeitung von Zeichenfolgen, deren Überprüfung auf spezifische Kriterien und die Darstellung der Resultate in einem bedeutungsvollen Kontext (Wing, 2006).

3.12 Reflexion

Der Reflexionsprozess im Rahmen des Programmierkurses stellte einen unverzichtbaren Aspekt dieser Masterarbeit dar und wurde methodisch konzipiert, um das Bewusstsein und die

Selbsteinschätzung der Schülerinnen und Schüler zu erhöhen. Die Methodik beinhaltete sowohl eine Zwischenreflexion nach der Hälfte des Kurses als auch eine finale Reflexion am Ende. Eine systematische Methode wurde implementiert, um die Fähigkeit der Schülerinnen und Schüler zur Reflexion zu steigern, was laut Merriam & Tisdell (2015) ein zentraler Faktor für tiefes Lernen ist. Zur Gewährleistung einer repräsentativen und dennoch überschaubaren Datenauswahl wurden ein paar Schüler aus jeder Gruppe zufällig für die mündlichen Reflexionen ausgewählt. Diese methodische Herangehensweise ermöglichte eine konzentrierte Analyse individueller Lernprozesse und gewährleistete gleichzeitig eine Vielfalt an Perspektiven.

3.12.1 Zwischenreflexion

Die Zwischenreflexion wurde nach der vierten Unterrichtseinheit durchgeführt und zielte darauf ab, die Erfahrungen, den Lernfortschritt und die bisher auftretenden Schwierigkeiten der Schülerinnen und Schüler zu erkennen. Im Sinne wurden folgende Punkte erfragt:

- **Beurteilung des Lernfortschritts,** zum Beispiel: Wie würden Sie Ihren bisherigen Fortschritt im Programmieren mit Python beurteilen?
- **Verständnis der Konzepte,** zum Beispiel: Welche Konzepte oder Themen haben Sie bislang gut verstanden und bei welchen zeigen Sie noch Unsicherheit?
- **Schwierigkeiten,** zum Beispiel: Welche spezifischen Schwierigkeiten haben Sie in der ersten Hälfte des Kurses erlebt und wie haben Sie versucht, diese zu bewältigen?
- **Zielsetzung:** zum Beispiel: Welche Ziele setzt du dir für die zweite Hälfte des Kurses?

Diese Fragen ermöglichten eine eingehende Untersuchung des individuellen Lernprozesses und stellten eine Basis für zielorientiertes Lernen dar. Für Kolb (1984) als auch für Hattie & Timperley (2007) stellt Reflexion einen zentralen Aspekt des Lernprozesses dar, der den Lernenden die Möglichkeit bietet, aus ihren Erfahrungen zu lernen und ihre Kompetenzen weiter auszubauen.

3.12.2 Abschließende Reflexion

Die finale Reflexion wurde nach der achten Unterrichtseinheit durchgeführt und ermöglichte den Lernenden, ihren gesamten Lernprozess kritisch zu analysieren. Die Lernenden beurteilten nicht ausschließlich ihre Lernentwicklung, sondern auch die Anwendung des Erlernten auf zukünftige Projekte. Auch in diesem Fall wurden für jede Gruppe Schülerinnen und Schüler zufällig ausgewählt, um eine repräsentative Auswahl zu sichern. Folgende Punkte beinhaltete diese Reflexion:

- **Gesamteinschätzung,** zum Beispiel: Wie würden Sie Ihren gesamten Lernverlauf und Fortschritt im Kurs beurteilen?
- **Erreichte Ziele,** zum Beispiel: In welchem Ausmaß haben Sie ihre Ziele erreicht?
- **Relevante Lernerfahrungen,** zum Beispiel: Welche Lektionen oder Einsichten haben Sie aus dem Kurs am wertvollsten erworben?
- **Anwendung des Gelernten,** zum Beispiel: Wie beabsichtigen Sie, das erworbene Wissen in zukünftigen Projekten oder beruflichen Kontexten zu implementieren?

- **Selbstverbesserung,** zum Beispiel: Existieren Bereiche, in denen Sie sich weiterentwickeln möchten, und wie beabsichtigen Sie, diese zu realisieren?

Die Implementierung eines reflektierenden Verfahrens in der Untersuchung erlaubte es, sowohl die subjektiven Erfahrungen der Schüler als auch den objektiven Lernfortschritt zu erfassen. Eine repräsentative Darstellung der gesamten Lerngruppe wurde durch die gezielte Auswahl einer zufälligen Stichprobe von Teilnehmern erzielt, was die wissenschaftliche Validität der Resultate erhöhte (Yin, 2018). Diese Prozesse der Reflexion lieferten bedeutsame qualitative Daten, die zur Untersuchung der Programmierfähigkeiten, Selbstwahrnehmung und Lernmotivation der Schüler beitrugen (Robins et al., 2003).

4. Interpretation der Ergebnisse

Dieses Kapitel interpretiert die Ergebnisse der Untersuchung, die den Einfluss von ChatGPT als künstliche Intelligenz unterstütztes Lerninstrument im Vergleich zu einem E-Book auf den Erwerb von Programmierkompetenzen bei Schülerinnen und Schülern der Oberstufe analysiert. Die Einbindung von ChatGPT in Bildungsprozesse stellt ein häufig diskutiertes Thema in der gegenwärtigen Forschung dar (Rizvi et al., 2023), und diese Untersuchung liefert durch ihren praxisorientierten Ansatz einen Beitrag zu dieser Debatte.

Die Resultate stützen sich auf eine Kombination aus quantitativen und qualitativen Daten, die durch Open-Book- und Closed-Book-Tests sowie die von den Schülern erstellten Lerntagebücher und Reflexionsberichte erfasst wurden. Ein derartiger multidimensionaler Ansatz zur erlaubt es objektive Lernergebnisse als auch subjektive Erfahrungen der Schülerinnen und Schüler zu erfassen, was mit Creswells (2014) Konzept für eine umfangreiche Bildungsforschung korrespondiert. Diese Methodik korrespondiert mit Sung et al. (2020), die hervorheben, dass die Einbindung qualitativer Daten von zentraler Bedeutung ist, um Lernerfolge im digitalen Zeitalter zu dokumentieren.

Angesichts der Absicht dieser Untersuchung, einen explorativen Vergleich zwischen ChatGPT und dem E-Book als Lerninstrumente durchzuführen, wurde keine herkömmliche Kontrollgruppe herangezogen. Der Schwerpunkt lag auf der unmittelbaren Gegenüberstellung der beiden Lernmethoden hinsichtlich ihrer Wirksamkeit und ihrer Effekte auf den Lernfortschritt der Schüler. Diese Methodik orientiert sich an den Ratschlägen von Lai und Bower (2019), die hervorheben, dass technologische Werkzeuge, so schlussfolgernd auch künstliche Intelligenz, am effektivsten in realen Lernumgebungen untersucht werden.

Die Open-Book- und Closed-Book-Tests lieferten Informationen über die praktische Anwendung des erlangten Wissens. Obwohl die Closed-Book-Tests die Behaltensfähigkeit und das Verständnis der Schüler für Programmierkonzepte beurteilten, lieferten die Open-Book-Tests Einblicke in ihre Fähigkeit, erworbenes Wissen in realitätsnahen Programmieraufgaben zu implementieren. Beispielsweise konnten Schüler der ChatGPT-Gruppe Codierungsprobleme effektiver bewältigen, indem sie ChatGPT als unterstützendes Instrument einsetzten. Dies korrespondiert mit den Erkenntnissen von Zhu & Van Brummelen (2021), die die Funktion von Künstliche-Intelligenz-Werkzeugen als unterstützende Lernhilfen betonen.

Für Interpretation ist zu erwähnen, dass an dem Experiment 35 Schüler im Alter von 16 bis 17 Jahren teilnahmen, die in zwei Gruppen unterteilt waren: 16 Schülerinnen und Schüler in der E-Book-Gruppe und 19 Schüler in der Gruppe ChatGPT. Die Gruppe für E-Books setzte sich hauptsächlich aus männlichen Teilnehmern zusammen (11 männlich, 5 weiblich), während in der Gruppe für ChatGPT die weiblichen Teilnehmerinnen etwas stärker vertreten waren (11 männlich, 8 weiblich).

4.1 Interpretation des Vortests

Von den 35 Schülern hatten sechs bereits Erfahrung im Programmieren. Vier dieser Schüler mit fortgeschrittenen Kenntnissen haben fortgeschrittenes Wissen. Sie waren zufällig auf die traditionelle Lernmethode (E-Book) als auch auf die interaktive Plattform (ChatGPT) gleichverteilt. Sie waren in der Lage, komplexe Aufgaben wie Kontrollstrukturen oder Schleifen mit minimaler Unterstützung umzusetzen und theoretische Fragen zur Programmiersprache Python klar zu beantworten.

Die zwei Schüler mit mittleren Kenntnissen beherrschen grundlegende Konzepte und konnten einfache Aufgaben wie Textausgaben oder mathematische Operationen korrekt umsetzen. Bei komplexeren Themen, wie der Anwendung von Kontrollstrukturen und Schleifen, machten sie Fehler.

Die Mehrheit der Schülerinnen und Schüler hatte jedoch noch keine Programmiererfahrung. Sie lösten die Aufgaben (zum Beispiel Textausgabe, Zahlenaddition, Kontrollstrukturen) gar nicht. Bei den theoretischen Fragen (zum Beispiel Unterschiede zwischen if- und switch-Anweisungen oder Funktionen vs. Methoden) haben diese Schülerinnen und Schüler keine passenden Antworten geliefert.

4.2 Interpretation des Closed-Book-Tests

Am Closed-Book-Test nahmen 20 Schüler und Schülerinnen teil. Der Closed-Book-Test ergab eine durchschnittliche Punktzahl von 65,78 für die ChatGPT-Gruppe bei einer Standardabweichung von 20,91. Die E-Book-Gruppe erreichte eine ähnliche durchschnittliche Punktzahl von 68 Punkten mit einer Standardabweichung von 12,42. Diese nahezu identischen Mittelwerte deuten darauf hin, dass beide Gruppen vergleichbare Programmierfähigkeiten entwickelt haben. Solche Ergebnisse unterstützen die Erkenntnisse von Rizvi et al. (2023), die zeigten, dass traditionelle und künstliche-Intelligenz-gestützte Lernmethoden oft zu ähnlichen Lernergebnissen führen, insbesondere in einer kontrollierten Lernumgebung. Der Test auf Normalverteilung spielt für die Beurteilung der Hypothese, welche Lernmethode, ChatGPT gegenübergestellt mit E-Book, effektiver ist, eine wesentliche Rolle. Die Differenz in der Standardabweichung bedeutet, dass in der E-Book Ergebnisse konstanter um den Mittelwert angesiedelt sind und somit die Punktezahlen der Prüfungen näher an den 68 Punkten sind.

Der Shapiro-Wilk-Test wurde angewendet, um die Normalverteilung der Daten zu überprüfen. Beim Test der ChatGPT-Gruppe wurde eine Normalverteilung bestätigt. Bei der E-Book-Gruppe wurde ebenfalls eine Normalverteilung bestätigt. Die Normalverteilung beider Gruppenergebnisse ermöglichte die Anwendung eines t-Tests, der laut Creswell (2014) in solchen Kontexten statistisch angemessen ist. Der T-Test für zwei unabhängige Stichproben wurde angewendet. Die Nullhypothese ist, dass der Unterschied zwischen den zwei Gruppen nicht signifikant ist.

Der t-Test für zwei unabhängige Stichproben zeigte folgendes Ergebnis:

- ein 95%-Konfidenzintervall für den Mittelwertunterschied: [-9,054; 17,854].
- der beobachtete Unterschied betrug 4,400.
- der t-Wert betrug 0,693, mit einem kritischen Wert von ±2,120 bei 16 Freiheitsgraden (df).
- Der zweiseitige p-Wert betrug 0,498.

Da der p-Wert deutlich über 0,05 liegt, bestätigt dies die Nullhypothese, dass es keinen signifikanten Unterschied in der Leistung der beiden Gruppen gibt. Somit gibt es laut diesem Test keinen Unterschied, welche Methode effizienter ist.

4.2.1 Programmieren

Der erste Teil des Tests bestand darin, dass die Teilnehmenden Anforderungen in Code umsetzen sollten, was in den Programmieraufgaben (Beispiel 1 und 2) gemessen wurde. Die E-Book-Gruppe erzielte in Beispiel 1 eine durchschnittliche Punktzahl von 4,73 mit einer Standardabweichung von 3,72 und in Beispiel 2 eine durchschnittliche Punktzahl von 6,91 mit einer Standardabweichung von 3,14. Die ChatGPT-Gruppe erreichte in Beispiel 1 eine durchschnittliche Punktzahl von 4,89 mit einer Standardabweichung von 3,76 und in Beispiel 2 eine durchschnittliche Punktzahl von 6,44 mit einer Standardabweichung von 4,22.

Die durchschnittlichen Punktzahlen in beiden Gruppen zeigen, dass beide Schwierigkeiten hatten, die Programmieranforderungen erfolgreich umzusetzen. Die niedrigen Durchschnittswerte deuten darauf hin, dass die Aufgaben herausfordernd waren und die Teilnehmenden möglicherweise nicht ausreichend auf die Programmieraufgaben vorbereitet waren. Dies wird durch die hohen Standardabweichungen unterstützt, die auf eine große Streuung der Ergebnisse hinweisen. Das bedeutet, dass es in beiden Gruppen sowohl Teilnehmende gab, die die Aufgaben gut lösen konnten, als auch solche, die erhebliche Schwierigkeiten hatten.

Insgesamt zeigen die Ergebnisse, dass beide Gruppen Schwierigkeiten mit den Programmieraufgaben hatten. Dies deutet darauf hin, dass keine der beiden Methoden (E-Books oder ChatGPT) die Studierenden vollständig auf das eigenständige Programmieren vorbereitet hat. Laut Grover & Pea (2013) ist der Mangel an praktischer Anwendung in der Programmierung oft ein Faktor, der den Lernerfolg der Studierenden behindert, was hier ebenfalls der Fall sein könnte. Der Einsatz beider Lernmethoden, E-Book und ChatGPT, könnte durch zusätzliche praxisnahe Übungen ergänzt werden, um die Programmierfähigkeiten der Lernenden zu verbessern.

4.2.2Fehlererkennung

Der zweite Teil des Tests befasste sich mit der Fehlererkennung, wobei die Teilnehmenden zwei Beispiele (Beispiel 1 und 2) bearbeiten mussten. Die E-Book-Gruppe erzielte in Beispiel 1 eine durchschnittliche Punktzahl von 5,27 mit einer Standardabweichung von 4,03 und in Beispiel 2 eine durchschnittliche Punktzahl von 5,09 mit einer Standardabweichung von 4,50. Die ChatGPT-Gruppe schnitt in Beispiel 1 mit einem Durchschnitt von 5,78 und einer Standardabweichung von 5,04 etwas besser ab und erreichte in Beispiel 2 eine durchschnittliche Punktzahl von 6,67 mit einer Standardabweichung von 5,00.

Diese Ergebnisse zeigen, dass die ChatGPT-Gruppe insgesamt bei der Fehlererkennung erfolgreicher war, insbesondere in Beispiel 2, wo sie sowohl einen höheren Durchschnitt als auch eine höhere Streuung der Ergebnisse im Vergleich zur E-Book-Gruppe aufwies. Die höhere Standardabweichung in der ChatGPT-Gruppe deutet darauf hin, dass es innerhalb dieser Gruppe größere Unterschiede in den Leistungen der Teilnehmenden gab, was darauf schließen lässt, dass einige Studierende deutlich von der künstliche-Intelligenz-gestützten Lernumgebung profitieren konnten, während andere möglicherweise weiterhin Schwierigkeiten hatten.

Die höheren Durchschnittswerte in der ChatGPT-Gruppe, insbesondere in Beispiel 2, lassen vermuten, dass die interaktive und dialogorientierte Natur der künstlichen Intelligenz beim Lernen des Debuggings unterstützend wirken kann. Die Ergebnisse bedeuten, aber nicht, dass ein guter Debugger gleich ein guter Programmierer ist. Das hat auch Fitzgerald et al. (2008) festgestellt.

4.2.3Vervollständigung von Code

Der dritte Teil des Tests konzentrierte sich auf die Vervollständigung des Codes, wobei die Teilnehmenden zwei Beispiele (Beispiel 1 und 2) bearbeiten mussten. Die E-Book-Gruppe erzielte in Beispiel 1 durchschnittlich 7,27 Punkte mit einer Standardabweichung von 4,03 und in Beispiel 2 durchschnittlich 5,64 Punkte mit einer Standardabweichung von 4,18. Die ChatGPT-Gruppe erzielte hingegen in Beispiel 1 durchschnittlich 6,44 Punkte mit einer Standardabweichung von 4,88 und in Beispiel 2 4,44 Punkte mit einer Standardabweichung von 5,27.

Diese Ergebnisse zeigen, dass die E-Book-Gruppe insgesamt bei der Vervollständigung des Codes besser abschnitt als die ChatGPT-Gruppe, insbesondere in Beispiel 1, wo sie einen signifikant höheren Durchschnittswert erreichte. Auch wenn die Standardabweichungen in beiden Gruppen hoch sind, was auf eine breite Streuung der Ergebnisse hinweist, deutet die höhere durchschnittliche Punktzahl der E-Book-Gruppe darauf hin, dass strukturierte Lernmaterialien wie E-Books in der Lage sind, den Lernenden ein klareres Verständnis der Vervollständigung von Codefragmenten zu vermitteln.

Diese Erkenntnisse stehen im Einklang mit den Arbeiten von Shahid et al. (2023), die gezeigt haben, dass strukturierte und lineare Lernmaterialien, wie sie typischerweise in E-Books zu finden sind, den Lernenden helfen können, tiefer in spezifische Aufgaben einzutauchen und sich ein fundiertes Verständnis für die Vervollständigung von Programmcode zu erarbeiten. Dies könnte darauf zurückzuführen sein, dass die Lernenden bei der Arbeit mit E-Books einem klar strukturierten, systematischen Lernansatz folgen, der sie bei der Anwendung solcher Aufgaben besser unterstützt als die interaktive, aber weniger strukturierte Unterstützung durch ChatGPT.

4.2.4 Anordnung von Code

Der vierte Teil des Tests befasste sich mit der Anordnung von Codefragmenten, wobei die Teilnehmenden zwei Beispiele (Beispiel 1 und 2) bearbeiten mussten. Die E-Book-Gruppe erzielte in Beispiel 1 eine durchschnittliche Punktzahl von 9,09 mit einer Standardabweichung von 1,38 und in Beispiel 2 eine durchschnittliche Punktzahl von 6,36 mit einer Standardabweichung von 3,07. Die ChatGPT-Gruppe erzielte in Beispiel 1 eine durchschnittliche Punktzahl von 7,11 mit einer Standardabweichung von 4,26 und in Beispiel 2 5,33 Punkte mit einer Standardabweichung von 5,10.

Diese Ergebnisse zeigen, dass die E-Book-Gruppe in dieser Kategorie deutlich besser abschnitt als die ChatGPT-Gruppe, insbesondere in Beispiel 2, wo die Unterschiede besonders groß sind. Die höheren Durchschnittswerte in beiden Beispielen lassen darauf schließen, dass die E-Book-Gruppe ein stärkeres Verständnis für die Strukturierung von Code hatte. Die geringeren Standardabweichungen der E-Book-Gruppe deuten zudem auf eine konsistentere Leistung innerhalb der Gruppe hin, was darauf hindeutet, dass die meisten Teilnehmenden in dieser Gruppe in etwa auf einem ähnlichen Niveau gearbeitet haben.

Diese Ergebnisse stimmen mit den Erkenntnissen von Paper von Robins, Rountree & Rountree (2003) überein, die feststellten, dass strukturierte Lernmaterialien wie E-Books beim Lehren von Programmiersequenzen und Logik klare Vorteile bieten. Da das Anordnen von Code eine gute Kenntnis der logischen Abfolge von Programmierbefehlen erfordert, scheint die strukturierte Darstellung von Informationen in E-Books den Lernenden geholfen zu haben, diese Fähigkeiten besser zu entwickeln.

4.2.5Multiple-Choice

Der fünfte Teil des Tests bestand aus Multiple-Choice-Fragen, die theoretische Programmierkonzepte abfragten. Die E-Book-Gruppe erzielte hierbei durchschnittlich 17,64 Punkte mit einer Standardabweichung von 3,56. Die ChatGPT-Gruppe erreichte eine durchschnittliche Punktzahl von 18,67 mit einer Standardabweichung von 2,65.

Diese Ergebnisse zeigen, dass beide Gruppen ein hohes Maß an theoretischem Verständnis der Programmierkonzepte erreicht haben. Die nahezu perfekten Ergebnisse der E-Book-Gruppe und die perfekten Ergebnisse der ChatGPT-Gruppe deuten darauf hin, dass sowohl das E-Book als auch ChatGPT effektiv dabei waren, grundlegende theoretische Programmierkonzepte zu vermitteln. Die perfekte Punktzahl der ChatGPT-Gruppe hebt jedoch hervor, dass KI-gestützte Lernmethoden, wie sie durch ChatGPT repräsentiert werden, besonders gut geeignet sind, theoretisches Wissen präzise und effizient zu festigen.

Diese Erkenntnisse bestätigen auch die Ergebnisse von Shahid et al. (2023), die aufzeigen, dass KI-gestützte Lernumgebungen in der Lage sind, theoretisches Wissen klar und verständlich zu vermitteln, was zu einer höheren Wissensverankerung bei den Lernenden führen kann.

Die Ergebnisse des Closed-Book-Tests zeigen, dass die E-Book-Gruppe in Aufgaben, die strukturiertes Wissen erfordern, besser abschnitt, während die ChatGPT-Gruppe Stärken in der Fehlererkennung zeigte. Die Studien von Shahid et al. (2023) und Sedlbauer et al. (2024) bestätigt, dass beide Methoden spezifische Vorteile haben und bestimmte Kompetenzen effektiv fördern können:

- **Stärken der E-Book-Gruppe**: Die E-Book-Gruppe konnte strukturiertes Wissen besser abrufen, was den pädagogischen Vorteil klar strukturierter Lernmaterialien hervorhebt.
- **Stärken der ChatGPT-Gruppe**: Die ChatGPT-Gruppe zeigte besondere Stärke in der Fehlererkennung, was die Vorteile interaktiver, KI-gestützter Lernmethoden beim Debugging hervorhebt

Diese Closed-Book-Testergebnisse legen nahe, dass die Kombination beider Methoden das Lernergebnis in der Programmierausbildung optimieren könnte, indem sowohl das strukturelle Verständnis als auch die Problemlösungsfähigkeiten der Studierenden gefördert werden.

4.3 Interpretation des Open-Book-Tests

Zwei Gruppen wurden im Open-Book-Test analysiert: die ChatGPT-Gruppe und die E-Book-Gruppe. Die deskriptiven Statistiken zeigen, dass die E-Book-Gruppe im Durchschnitt 81,85 Punkte erzielte, während die ChatGPT-Gruppe durchschnittlich 79,07 Punkte erreichte. Obwohl dieser Unterschied gering ist, deutet er auf eine leichte Tendenz zugunsten der E-Book-Gruppe hin, was darauf hindeutet, dass traditionelle Lernmethoden in diesem Testkontext möglicherweise effektiver waren. Diese Beobachtung steht im Einklang mit den Ergebnissen von Shahid et al. (2023), die feststellten, dass strukturierte Lernressourcen oft zu einem effektiveren Wissenstransfer beitragen. Ein Shapiro-Wilk-Test wurde durchgeführt, um die Normalverteilung der Daten der beiden Gruppen zu überprüfen:

Die ChatGPT-Gruppe hat Daten, die normalverteilt sind. Die E-Book-Gruppe hat ebenfalls eine Normalverteilung der Stichprobe. Da beide Gruppen eine Normalverteilung zeigen, erfüllen die Daten die Voraussetzungen für einen t-Test, wie von Creswell (2014) in der statistischen Methodologie empfohlen. Der t-Test für zwei unabhängige Stichproben ergibt:

Hier ist die angepasste Textpassage basierend auf den neuen Ergebnissen:

- Ein 95%-Konfidenzintervall für den Mittelwertunterschied: [-9,824; 6,572].
- Der beobachtete Unterschied betrug -1,626.
- Der t-Wert betrug -0,409, mit einem kritischen Wert von ±2,060 bei 25 Freiheitsgraden (df).
- Der zweiseitige p-Wert betrug 0,686, was deutlich über dem Signifikanzniveau von 0,05 liegt.

Da der p-Wert nicht signifikant ist, bestätigt dies die Nullhypothese (H0), dass kein signifikanter Unterschied zwischen den Leistungen der beiden Gruppen besteht. Diese Ergebnisse stehen im Einklang mit Studien von Shahid et al. (2023), die feststellten, dass sowohl KI-unterstützte als auch traditionelle Lernmethoden in bestimmten Kontexten vergleichbare Lernergebnisse erzielen können. Die Interpretation der einzelnen Bereiche des Tests ergeben folgendes Bild.

4.3.1 Programmierung

Die Programmierung war der erste Teilbereich der Prüfung und bestand aus zwei Beispielen (Beispiel 1 und 2). Die E-Book-Gruppe erzielte in Beispiel 1 durchschnittlich 9,86 Punkte mit einer Standardabweichung von 0,53 und in Beispiel 2 9,29 Punkte mit einer Standardabweichung von 1,27. Die ChatGPT-Gruppe erzielte in Beispiel 1 durchschnittlich 9,60 Punkte mit einer Standardabweichung von 0,83 und in Beispiel 2 7,20 Punkte mit einer Standardabweichung von 1,47.

Die Ergebnisse zeigen, dass die E-Book-Gruppe unabhängig von der Lernhilfe den Code besser schreiben konnte als die ChatGPT-Gruppe. Diese höhere Leistung könnte auf die strukturierte Natur des E-Books zurückzuführen sein, das den Lernenden klare Anweisungen und Beispiele bietet, was sie möglicherweise bei der Programmierlogik unterstützte. Diese Interpretation wird durch die Forschung von Sedlbauer et al. (2024) gestützt, die zeigte, dass strukturierte Lernmaterialien die Entwicklung der Programmierlogik fördern und den Lernenden helfen, Aufgaben systematisch zu lösen.

4.3.2Fehlererkennung

Der Teilbereich der Fehlererkennung umfasste zwei Beispiele (Beispiel 1 und 2). Die E-Book-Gruppe erzielte in Beispiel 1 durchschnittlich 7,29 Punkte mit einer Standardabweichung von 2,79 und in Beispiel 2 5,00 Punkte mit einer Standardabweichung von 4,49. Die ChatGPT-Gruppe erzielte in Beispiel 1 6,80 Punkte mit einer Standardabweichung von 3,53 und in Beispiel 2 8,27 Punkte mit einer Standardabweichung von 3,45.

Die Ergebnisse zeigen, dass die ChatGPT-Gruppe, insbesondere in Beispiel 2, bei der Fehlererkennung besser abschnitt als die E-Book-Gruppe. Dies deutet darauf hin, dass die interaktive Natur von KI-gestützten Lernumgebungen, wie sie ChatGPT bietet, den Lernenden dabei hilft, effektiver zu debuggen. Die Möglichkeit, Lösungen interaktiv zu erkunden und spezifisches Feedback zu erhalten, scheint den Lernenden ein tieferes Verständnis für die zugrunde liegenden Fehler vermittelt zu haben. Diese Ergebnisse stehen im Einklang mit den Erkenntnissen von Farrokhnia et al. (2023), die darauf hinweisen, dass KI-gestützte Werkzeuge besonders gut geeignet sind, um die Debugging-Fähigkeiten der Lernenden zu fördern und ihre Fähigkeit zur Fehlererkennung zu verbessern.

4.3.3Vervollständigung von Code

Der Teilbereich der Vervollständigung des Codes umfasste zwei Beispiele (Beispiel 1 und 2). Die E-Book-Gruppe erzielte in Beispiel 1 durchschnittlich 6,29 Punkte mit einer Standardabweichung von 4,14 und in Beispiel 2 6,71 Punkte mit einer Standardabweichung von 3,81. Die ChatGPT-Gruppe erzielte in Beispiel 1 durchschnittlich 6,67 Punkte mit einer Standardabweichung von 4,39 und in Beispiel 2 ebenfalls 6,67 Punkte mit einer Standardabweichung von 4,64.

Die Ähnlichkeit der Punktzahlen zwischen den beiden Gruppen zeigt, dass beide Lernmethoden – sowohl das E-Book als auch ChatGPT – effektiv dabei waren, das Verständnis von Code-Strukturen zu fördern. Trotz unterschiedlicher Ansätze bei der Wissensvermittlung haben die Teilnehmenden beider Gruppen vergleichbare Ergebnisse erzielt. Dies steht im Einklang mit den Ergebnissen von Sedlbauer et al. (2024), die feststellten, dass sowohl traditionelle als auch KI-unterstützte Lernmethoden in der Lage sind, die Fähigkeit zur Code-Vervollständigung in vergleichbarer Weise zu entwickeln und zu stärken.

4.3.4 Anordnung von Code

Der Teilbereich der Anordnung des Codes beinhaltete zwei Beispiele. Die E-Book-Gruppe erzielte in Beispiel 1 durchschnittlich 9,29 Punkte mit einer Standardabweichung von 1,27 und in Beispiel 2 9,00 Punkte mit einer Standardabweichung von 1,30. Die ChatGPT-Gruppe erzielte in Beispiel 1 durchschnittlich 8,40 Punkte mit einer Standardabweichung von 2,53 und in Beispiel 2 8,53 Punkte mit einer Standardabweichung von 2,33.

Die nahezu identischen Ergebnisse zwischen den beiden Gruppen deuten darauf hin, dass die Fähigkeit, Code zu arrangieren, unabhängig von der gewählten Lernmethode effektiv entwickelt wird. Beide Gruppen, sowohl mit traditionellem E-Book-Lernen als auch mit der Unterstützung durch ChatGPT, zeigten vergleichbare Ergebnisse bei der Strukturierung von Code. Dies stimmt mit den Ergebnissen von Farrokhnia et al. (2023) oder Shahid et al. (2023) überein, die zeigten, dass sowohl traditionelle als auch KI-unterstützte Methoden gleichermaßen in der Lage sind, die Fähigkeit zur Code-Strukturierung zu fördern und zu entwickeln.

4.3.5Multiple Choice

Im Bereich der Multiple-Choice-Fragen erzielte die E-Book-Gruppe durchschnittlich 19,14 Punkte mit einer Standardabweichung von 1,35, während die ChatGPT-Gruppe durchschnittlich 16,93 Punkte mit einer Standardabweichung von 5,32 erreichte.

Die deutlich besseren Ergebnisse der E-Book-Gruppe deuten darauf hin, dass das E-Book ein tieferes theoretisches Verständnis von Programmierkonzepten fördert. Die geringere Standardabweichung in dieser Gruppe zeigt zudem eine konsistentere Leistung unter den Teilnehmenden. Diese Ergebnisse werden von den Studien von Sedlbauer et al. (2024), Shahid et al. (2023) gestützt, die darauf hinweisen, dass traditionelle Lernmaterialien oft effektiver sind, wenn es um die Vermittlung von konzeptionellem Wissen geht. E-Books bieten eine strukturierte und lineare Darstellungsweise, die den Lernenden hilft, theoretische Inhalte besser zu verinnerlichen.

Die Ergebnisse des Open-Book-Tests zeigen, dass die E-Book-Gruppe in vielen Kategorien geringfügig besser abschnitt als die ChatGPT-Gruppe, insbesondere in der Programmierung und im theoretischen Wissen. Dies deutet darauf hin, dass das E-Book als traditionelles Lernwerkzeug effektiv bei der Vermittlung von Programmierkonzepten ist. Auf der anderen Seite zeigte die ChatGPT-Gruppe besondere Stärken, insbesondere bei der Fehlererkennung, was die interaktiven Vorteile von KI-unterstützten Lernwerkzeugen hervorhebt Shahid et al. (2023).

Folgende Interpretation lässt sich daraus ableiten:

- **Vorteile des E-Books**: Besonders effektiv beim Erlernen von Strukturen, Theorie und dem Schreiben von Programmcode.
- **Vorteile von ChatGPT**: Nützlich für praktische, anwendungsorientierte Aufgaben, insbesondere bei der Erkennung und Behebung von Fehlern.

Die Ergebnisse zeigen, dass beide Methoden erfolgreich das Lernen unterstützen, wobei die E-Book-Gruppe insgesamt etwas besser abschnitt. Die unterschiedlichen Stärken jeder Lernmethode legen jedoch nahe, dass eine Kombination beider Ansätze besonders vorteilhaft sein könnte, was im Einklang mit den Vorschlägen von Lye und Koh (2014) für eine hybride Lernumgebung steht.

Diese Erkenntnisse bieten wertvolle Anhaltspunkte, wie traditionelle und KI-gestützte Lernressourcen kombiniert werden können, um einen umfassenden Ansatz für die Programmierausbildung zu schaffen, der sowohl strukturierte als auch interaktive Lernmethoden nutzt.

4.4 Vergleich der Open-Book- und Closed-Book-Ergebnisse

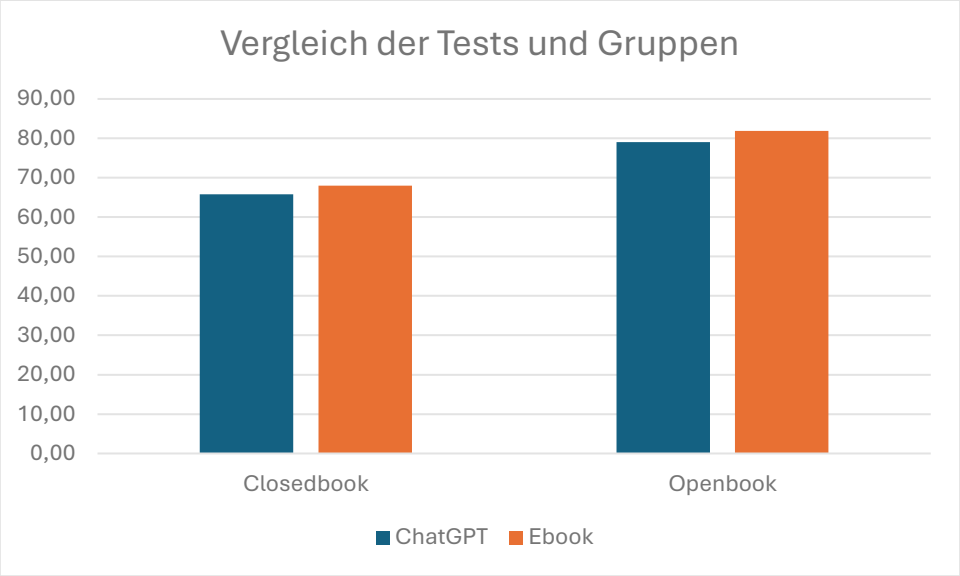

Abbildung 2:Vergleich der Tests und Gruppen

Der Vergleich zwischen den Ergebnissen des Open-Book- und des Closed-Book-Tests liefert wertvolle Einblicke in das Verständnis und die Anwendung von Programmierwissen der beiden Lernendengruppen. Ein Einblick in die Verteilung gibt die Grafik, Vergleich der Tests und Gruppen. Die Analyse zeigt, wie gut die Studierenden die erlernten Konzepte sowohl in einer Umgebung mit Hilfsmitteln als auch in einem Testumfeld ohne Ressourcen anwenden konnten.

Ein Vergleich der Open-Book- und Closed-Book-Test-Ergebnisse zeigt eine deutliche Veränderung der Leistung in beiden Gruppen. Die ChatGPT-Gruppe erzielte im Closed-Book-Test eine durchschnittliche Punktzahl von **65,78** mit einer Standardabweichung von **20,92**, während die E-Book-Gruppe **68,00** Punkte mit einer Standardabweichung von **12,43** erreichte. Im Open-Book-Test erzielte die ChatGPT-Gruppe durchschnittlich **79,07** Punkte mit einer Standardabweichung von **8,61**, während die E-Book-Gruppe **81,85** Punkte mit einer Standardabweichung von **12,12** erzielte.

Beide Gruppen verzeichneten einen deutlichen Leistungsanstieg im Open-Book-Test im Vergleich zum Closed-Book-Test, was darauf hinweist, dass die Studierenden im Closed-Book-Test ohne externe Hilfsmittel größere Schwierigkeiten hatten, das Gelernte anzuwenden. Dies ist erwartungsgemäß, da der Open-Book-Test den Lernenden ermöglicht, auf zusätzliche Ressourcen zurückzugreifen, wodurch sie die Programmieraufgaben besser lösen konnten.

Die E-Book-Gruppe erzielte in beiden Testformaten eine etwas höhere durchschnittliche Punktzahl und eine geringere Standardabweichung im Vergleich zur ChatGPT-Gruppe, was darauf hindeutet, dass die E-Book-Nutzer insgesamt konsistentere Leistungen erbrachten. Dies könnte auf die strukturierten Lernmaterialien des E-Books zurückzuführen sein, die den Studierenden möglicherweise geholfen haben, ein tieferes und besser verankertes Verständnis der Konzepte zu entwickeln, welches sich auch im Closed-Book-Test, der auf reinem Abruf von Wissen basiert, als vorteilhaft erwies.

Interessanterweise zeigt die höhere Standardabweichung im Closed-Book-Test beider Gruppen, dass es größere Leistungsunterschiede unter den Studierenden gab, wenn sie ohne Hilfsmittel arbeiten mussten. Dies könnte darauf hindeuten, dass einige Studierende stärker von den Lernmethoden profitierten als andere, insbesondere im Hinblick auf ihre Fähigkeit, selbstständig Wissen anzuwenden.

Insgesamt zeigen die Ergebnisse, dass die E-Book-Gruppe in beiden Testarten leichte Vorteile aufweisen, insbesondere im Hinblick auf die Konsistenz der Leistungen, was darauf hindeutet, dass die strukturierte Natur der E-Books den Studierenden möglicherweise eine bessere Vorbereitung auf den Abruf von Wissen im Closed-Book-Format bot.

4.4.1 Programmieren

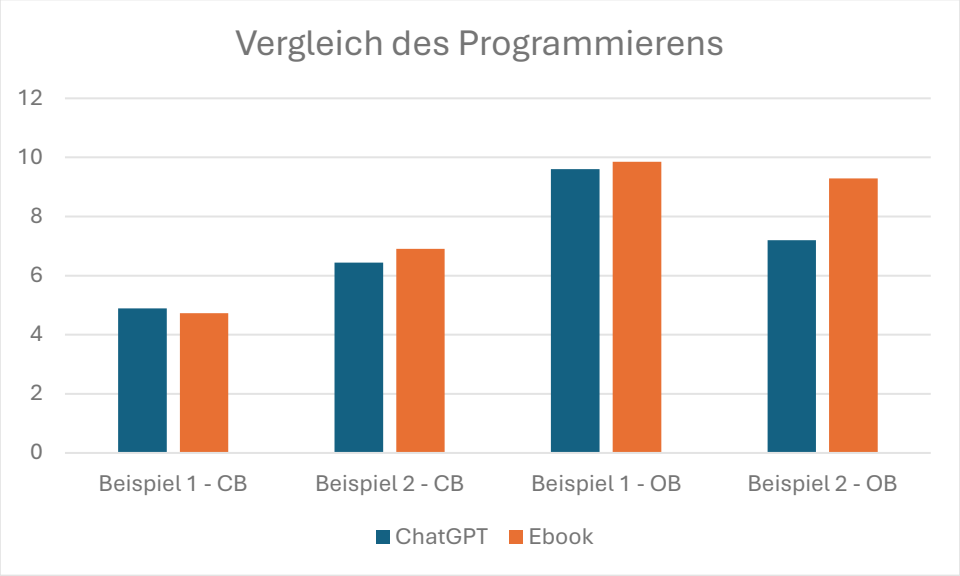

Abbildung 3: Vergelich des Programmierens

Die Ergebnisse zeigen, dass beide Gruppen im Closed-Book-Test einen signifikanten Leistungsrückgang bei den Programmieraufgaben verzeichneten. Dies deutet darauf hin, dass die Teilnehmenden Schwierigkeiten hatten, die Programmierkonzepte ohne externe Ressourcen anzuwenden, was den höheren Anforderungen an das Langzeitgedächtnis entspricht. Der Rückgang war jedoch bei der E-Book-Gruppe weniger ausgeprägt, was darauf hindeutet, dass diese Gruppe die Programmierkonzepte besser internalisiert hatte und sie im Closed-Book-Test erfolgreicher anwenden konnte.

Diese Ergebnisse unterstützen aktuelle Forschungen, die darauf hinweisen, dass traditionelle Lernmethoden, wie sie in E-Books verwendet werden, oft effektiver für die Förderung des Langzeitgedächtnisses sind. Die strukturierte und lineare Darstellung von Lerninhalten in E-Books scheint den Lernenden geholfen zu haben, die Konzepte nachhaltiger zu verinnerlichen, was ihnen im Closed-Book-Format, das auf die Abruffähigkeit von Wissen ohne externe Hilfsmittel angewiesen ist, einen Vorteil verschaffte (Sedlbauer et al. (2024)).

4.4.2 Fehlererkennung

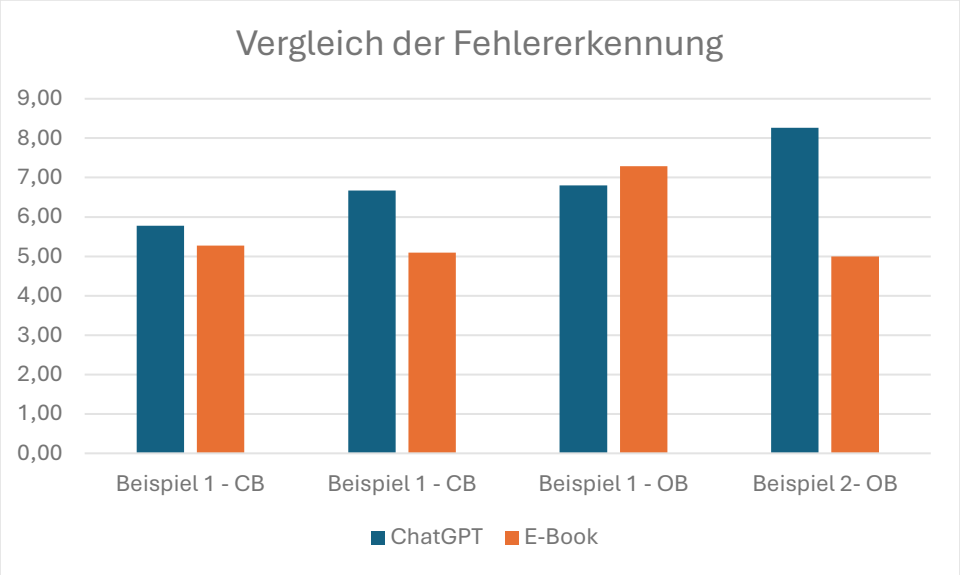

Abbildung 4: Vergleich der Fehlererkennung

Die Ergebnisse zeigen, dass die ChatGPT-Gruppe sowohl im Open-Book- als auch im Closed-Book-Test eine bessere Fähigkeit zur Fehlererkennung zeigte, insbesondere in Beispiel 2, wo sie konstant bessere Punktzahlen erzielte. Dies deutet darauf hin, dass der interaktive Ansatz von ChatGPT die Problemlösungsfähigkeiten der Studierenden verbessert, indem er ihnen ermöglicht, Fehler eigenständig zu erkennen und zu korrigieren. Die Ergebnisse stehen im Einklang mit den Erkenntnissen von Sedlbauer et al. (2024), Shahid et al. (2023), die darauf hinweisen, dass KI-gestützte Lernmethoden wie ChatGPT besonders effektiv beim Debugging sind, da sie eine dialogorientierte Lernumgebung bieten, die es den Studierenden erleichtert, Lösungen zu finden und ihre Fehlererkennungsfähigkeiten zu schärfen.

4.4.3 Vervollständigung von Code

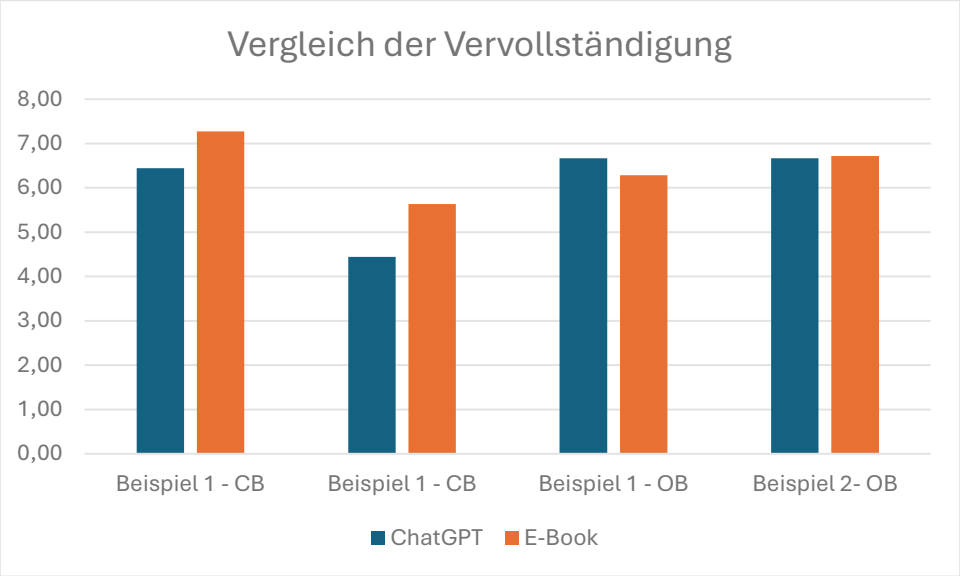

Abbildung 5: Vergleich der Vervollständigung

Die E-Book-Gruppe zeigte in dieser Kategorie eine stabilere Leistung im Vergleich zur ChatGPT-Gruppe, insbesondere im Closed-Book-Test. Dies deutet darauf hin, dass das E-Book den Lernenden half, die Prinzipien der Code-Vervollständigung effektiver zu internalisieren. Diese Ergebnisse stimmen mit den Forschungsergebnissen von Farrokhnia et al. (2023), Sedlbauer et al. (2024) überein, die aufzeigen, dass strukturierte Lernressourcen wie E-Books besonders effektiv sind, um die Syntax und Struktur von Code langfristig zu verinnerlichen, während interaktive, KI-gestützte Werkzeuge in dieser Hinsicht weniger wirkungsvoll sein können.

4.4.4 Anordnung von Code

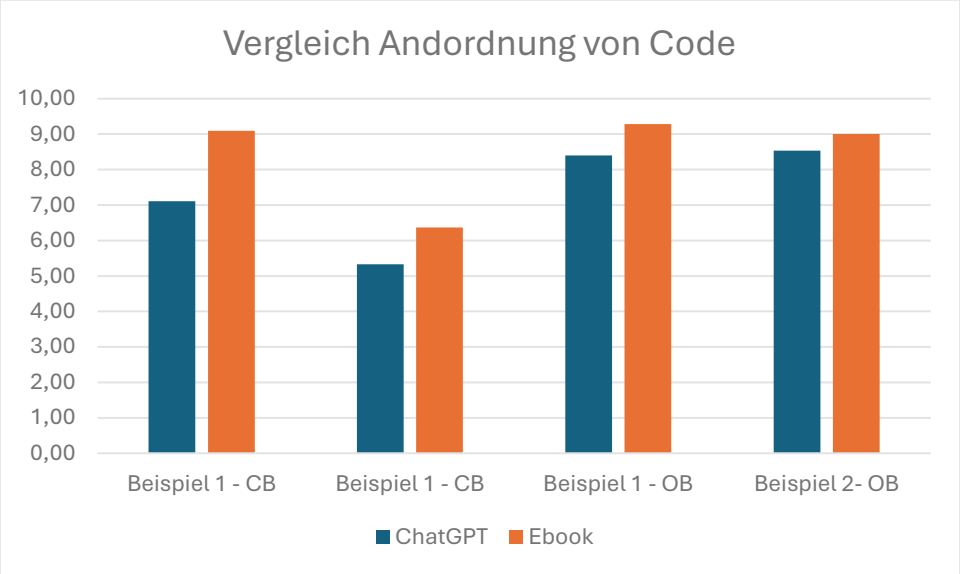

Abbildung 6: Vergelich Anordnung von Code

Die deutlich besseren Ergebnisse der E-Book-Gruppe im Closed-Book-Test zeigen, dass die Lernenden aus dieser Gruppe eine stärkere Fähigkeit entwickelten, Programmiercode korrekt zu strukturieren und die logische Reihenfolge der Codeelemente auch ohne externe Hilfsmittel zu rekonstruieren. Diese Leistung weist darauf hin, dass das E-Book wirksam Strategien zur Vermittlung der logischen Reihenfolge von Programmiercode und zur Entwicklung eines konzeptionellen Verständnisses der Code-Struktur förderte. Die Lernenden der E-Book-Gruppe konnten die Informationen offenbar besser verinnerlichen und im Closed-Book-Format abrufen, was auf die Vermittlung systematischer Denkmuster im Rahmen der strukturierten E-Book-Inhalte hinweist.

Im Gegensatz dazu schnitt die ChatGPT-Gruppe im Closed-Book-Test merklich schlechter ab, was darauf hindeutet, dass die interaktive Natur von KI-gestützten Lernmethoden möglicherweise nicht so effektiv darin war, den Lernenden die logische Strukturierung von Code auf langfristiger Basis zu vermitteln. Die starke Leistungsabnahme der ChatGPT-Gruppe von Open-Book zu Closed-Book zeigt, dass die Gruppe sich möglicherweise stärker auf die Unterstützung durch das Tool verlassen hat, anstatt die Konzepte vollständig zu verinnerlichen.

Diese Ergebnisse stehen im Einklang mit den Erkenntnissen von Kasneci et al. (2023), die darauf hinweisen, dass traditionelle Lernmethoden wie E-Books effektiver sind, um konzeptionelles Verständnis und logisches Denken zu fördern. Die strukturierte und sequentielle Darstellungsweise der E-Books bietet den Lernenden eine klar definierte Methode zur

Organisation und Anordnung von Code, die auch ohne zusätzliche Hilfsmittel erfolgreich angewendet werden kann.

4.4.5 Multiple-Choice

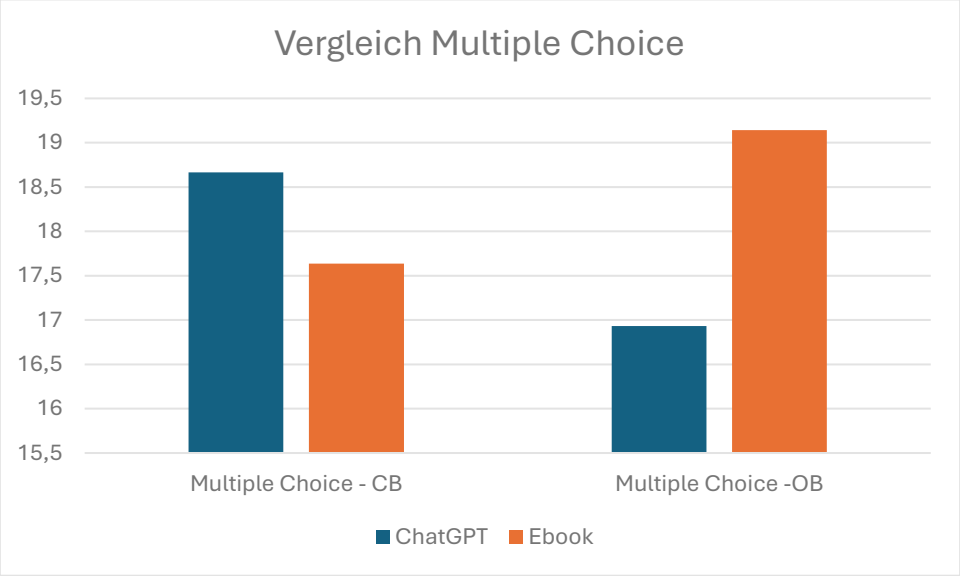

Abbildung 7: Vergleich von Multiple Choice

Im Bereich Multiple-Choice zeigten beide Gruppen sehr hohe Leistungen. Die ChatGPT-Gruppe erzielte im Closed-Book-Test 20 Punkte und im Open-Book-Test 16,93 Punkte mit einer Standardabweichung von 2,6. Die E-Book-Gruppe erreichte im Open-Book-Test 19,14 Punkte (Standardabweichung 1,35) und im Closed-Book-Test 19,64 Punkte (Standardabweichung 0,8).

Diese hohen Punktzahlen in beiden Gruppen deuten darauf hin, dass sowohl KI-unterstütztes Lernen durch ChatGPT als auch das Lernen mit E-Books gleichermaßen effektiv darin waren, den Studierenden grundlegendes theoretisches Faktenwissen zu vermitteln. Die nahezu perfekten Ergebnisse zeigen, dass die Lernenden in der Lage waren, theoretische Inhalte konsistent abzurufen, unabhängig davon, ob sie auf externe Hilfsmittel zugreifen konnten oder nicht.

Die konstanten 20 Punkte der ChatGPT-Gruppe unterstreichen die Fähigkeit von KI-gestützten Lernmethoden, theoretisches Wissen zu vermitteln und die Studierenden optimal auf Multiple-Choice-Fragen vorzubereiten. Gleichzeitig zeigt die starke Leistung der E-Book-Gruppe mit nur minimalen Abweichungen zwischen Open-Book und Closed-Book, dass E-Books ebenfalls eine effektive Vermittlung von Faktenwissen bieten.

Diese Ergebnisse stimmen mit den Erkenntnissen von Rizvi et al. (2023) überein, die nahelegen, dass sowohl KI-unterstütztes Lernen als auch traditionelle Lernmethoden wie E-Books gut geeignet sind, um theoretisches Wissen zu vermitteln und dieses auch in Testsituationen erfolgreich abzurufen.

4.4.6 Zusammenfassung

Die Open-Book-Tests zeigten, dass beide Gruppen mit ihren jeweiligen Lernmethoden erfolgreich waren. Der Closed-Book-Test hob jedoch hervor, dass die E-Book-Gruppe ihr Wissen besser

internalisierte, während die ChatGPT-Gruppe stärker auf interaktive Unterstützung angewiesen war.

Dieser Vergleich liefert wertvolle Erkenntnisse über die Stärken und Schwächen traditioneller versus KI-gestützter Lernmethoden. Die bessere Behaltensleistung der E-Book-Gruppe in Bezug auf strukturiertes Wissen stimmt mit Studien überein, die die Wirksamkeit traditioneller Materialien bei der Förderung des Langzeitgedächtnisses betonen, während die ChatGPT-Gruppe bei interaktiven Problemlösungsaufgaben herausragte und die Vorteile von KI bei der Verbesserung der Fehlererkennung und praktischen Anwendung zeigte.

Die Resultate von Closed-Book- und Open-Book-Bewertungen liefern Erkenntnisse über das Verständnis der Programmierkonzepte, die Behaltensleistung und die Fähigkeit der Schülerinnen und Schüler, ihr erworbenes Wissen praktisch zu implementieren. Die Befunde der Untersuchung weisen darauf hin, dass die ChatGPT-Gruppe in Problemlösungskompetenzen signifikant besser abschnitt, was mit den Befunden von Farrokhnia et al. (2023) übereinstimmt, die die Wirksamkeit von KI-basiertem Lernen bei der Entwicklung analytischer Fähigkeiten betonen. Schüler, die ChatGPT einsetzten, erzielten in Aufgaben, die kreative Problemlösungsstrategien erforderten, höhere Punktzahlen als ihre E-Book-Kolleginnen und Kollegen.

4.5 Interpretation der Abschlussprojekte und Lerntagebücher

Die Abschlussprojekte ermöglichten es den Studierenden, das im Kurs erworbene Programmierwissen in praktischen Kontexten anzuwenden. Aufgaben mit unterschiedlichen Schwierigkeitsgraden (leicht, mittel, schwer) wurden entwickelt, um den verschiedenen Fähigkeitsstufen der Studierenden gerecht zu werden (Grover & Basu, 2017). Begleitend dazu führten die Studierenden ein Reflexionstagebuch, das als Werkzeug zur Selbstreflexion diente und qualitative Daten zu ihrem individuellen Lernprozess, ihren Schwierigkeiten und Vorlieben lieferte. Es wurde beobachtet, dass die ChatGPT-Gruppe tendenziell eigenständiger arbeitete, während die E-Book-Gruppe ein tieferes Verständnis der Programmierkonzepte entwickelte, aber auch die künstliche Intelligenz zur Lösung einsetzte (Wing, 2006). Schülerinnen und Schüler der ChatGPT-Gruppe beschrieben ihre Erfahrungen häufig als motivierend und personalisiert, was durch die gegenwärtige Literatur zum KI-gestützten Lernen bestätigt wird (Šedlbauer et al., 2024).

Folgende Informationen wurden aus den Lerntagebüchern extrahiert:

- **Verständnis und Anwendung:** Das Lerntagebuch lieferte Einblicke, wie die Schülerinnen und Schüler Konzepte verstanden und anwendeten, welche Lernmaterialien ihnen halfen und wie sie theoretisches Wissen in praktische Aufgaben umsetzten (Robins, Rountree & Rountree, 2003).
- **Problemlösungsstrategien:** Die Schülerinnen und Schüler beschrieben ihre Strategien beim Bearbeiten von Programmieraufgaben und dem Verständnis komplexer Themen, was Aufschluss über ihre Problemlösungsfähigkeiten und ihren Umgang mit Herausforderungen gab (Robins, Rountree & Rountree, 2003).
- **Reflexion und Selbsteinschätzung:** Die Schülerinnen und Schüler reflektierten ihren Lernprozess, ihre Fortschritte, Schwierigkeiten und Erkenntnisse, wodurch sie ihre Fähigkeiten und Entwicklungen selbst einschätzen konnten (Merriam & Tisdell, 2015).

- **Vergleich und Präferenz:** Die Schülerinnen und Schüler verglichen ihre Erfahrungen mit ChatGPT und E-Books und bewerteten die Vor- und Nachteile beider Lernressourcen. Dies lieferte Einblicke in ihre Präferenzen und die wahrgenommene Effektivität der Lernmedien (Ryan & Deci, 2000).
- **Engagement und Motivation:** Das Lerntagebuch ermöglichte Beobachtungen zum Engagement und zur Motivation der Schülerinnen und Schüler. Die Häufigkeit und Tiefe der Einträge gaben Aufschluss über das Interesse der Schülerinnen und Schüler an jeder Lernressource (Ryan & Deci, 2000).

4.5.1 Leichtes Projekt

Die E-Book-Gruppe dokumentierte in ihren Tagebüchern sorgfältig, wie sie den Code strukturierten und die Berechnungen durchführten. Sie reflektierten über mögliche Fehlerquellen, was ihnen half, ein besseres Verständnis zu gewinnen. Sie berichteten, dass das eigenständige Durcharbeiten der Schritte ihr Verständnis der Programmierlogik vertiefte (Linn & Clancy, 1992). Die Anzahl der korrekten Lösungen lag bei 100 Prozent.

Die ChatGPT-Gruppe nutzte die künstliche Intelligenz in Form von ChatGPT, um schnell Antworten auf Syntax- und berechnungsbezogene Fragen zu erhalten. Ein Studierender sagte: „Ich war mir bei der Berechnung unsicher, also fragte ich ChatGPT nach der Methode, um Prozentsätze zu berechnen." Dies zeigt, dass sie sich auf die KI für sofortige Unterstützung verließen, was zu einer schnelleren Problemlösung führte. Auch hier war die Anzahl der korrekten Lösungen bei 100%.

Vergleichend entwickelte die E-Book-Gruppe ein tieferes Verständnis für die Strukturierung des Codes, während die ChatGPT-Gruppe von der sofortigen Unterstützung und schnelleren Klärung von Unsicherheiten profitierte.

4.5.2Mittelschweres Projekt

Einige der E-Book-Gruppe hatte Schwierigkeiten mit der Logik von Schleifen, fanden aber schließlich nach mehreren Versuchen eine Lösung. Sie dokumentierten ihren Prozess in ihren Tagebüchern und reflektierten ihre Lernerfahrungen. Dies bestätigt Studien, die zeigen, dass eigenständiges Arbeiten das Verständnis vertieft (Grover & Pea, 2013).

Die ChatGPT-Gruppe nutzten die KI, um ihre Logik für Schleifen zu überprüfen, so laut ihren Tagebüchern. Ein Studierender erwähnte: „ChatGPT zeigte mir ein Beispiel, das ich dann auf meine Aufgabe angepasst habe." Sie fanden schneller eine Lösung, aber ihr Verständnis war weniger tiefgehend.

Zusammenfassend zeigte die E-Book-Gruppe eine größere Bereitschaft, sich eigenständig mit Herausforderungen auseinanderzusetzen, während die ChatGPT-Gruppe von interaktiver Unterstützung profitierte.

4.5.3Herausforderndes Projekt

Die E-Book-Gruppe dokumentierte detailliert ihre Herausforderungen bei der Zeichenkettenmanipulation und der Verifizierung von Palindromen. Sie experimentierten mit

verschiedenen Ansätzen, was zu einem besseren Verständnis im Umgang mit Datenstrukturen führte (Ben-Ari, 2001).

Die ChatGPT-Gruppe nutzten ChatGPT, um Tipps und Ansätze zur Lösung des Problems zu erhalten. Sie erzielten schneller Ergebnisse, entwickelten jedoch weniger eigenständige Problemlösungsstrategien.

Die E-Book-Gruppe zeigte mehr Eigeninitiative und ein tieferes Verständnis von Datenstrukturen, während die ChatGPT-Gruppe von schnelleren Lösungen profitierte.

4.5.4 Aufteilung der Projekte

Die Verteilung der gewählten Projekte nach den Tests sieht wie folgt aus:

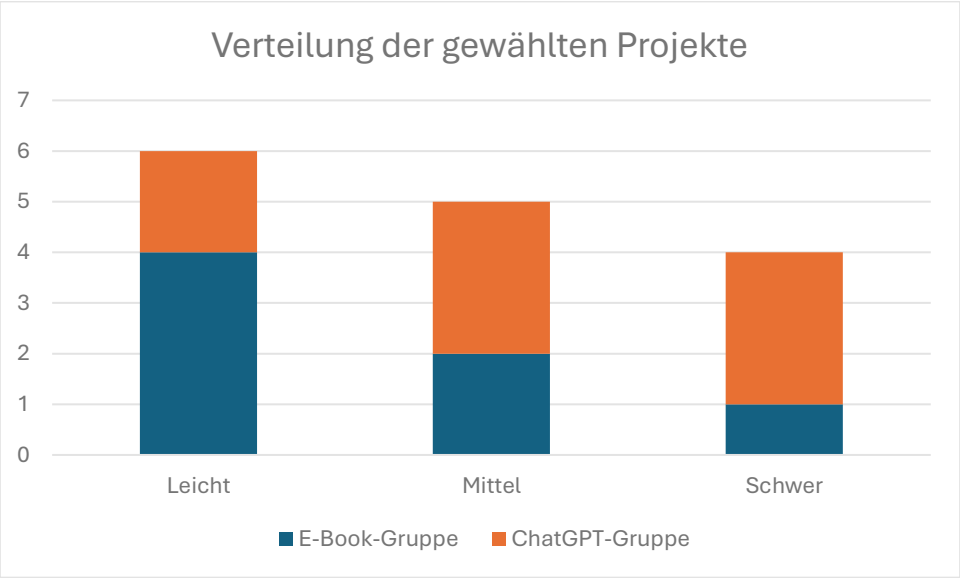

Abbildung 8: Aufteilung gewählter Projekte

Das Abschlussprojekt haben acht Personen aus der ChatGPT-Gruppe und sieben Personen aus der E-Book-Gruppe abgeschlossen. Tabellarisch sind die Schwierigkeitsgrade wie folgt gewählt:

Schwierigkeitsgrad	E-Book-Gruppe	ChatGPT-Gruppe
Leicht	4	2
Mittel	2	3
Schwer	1	3

Tabelle 1: Aufteilung der gewählten Projekte

Die Mehrheit der E-Book-Gruppe (4 von 7) entschied sich für das „leichte“ Projekt, was darauf hinweist, dass diese Studierenden es bevorzugten, ihr Programmierwissen in einem vertrauten Bereich anzuwenden. Dies deutet darauf hin, dass die E-Book-Gruppe vorsichtiger war und sich auf Aufgaben konzentrierte, bei denen sie sich sicher fühlten (Lister et al., 2004). Dies steht im

Einklang mit Forschungsergebnissen, die nahelegen, dass traditionelle Lernmethoden oft ein tieferes, aber langsameres Verständnis fördern (Papert, 1980).

Nur ein Studierender wählte die „schwere“ Aufgabe. Dies könnte auf geringeres Selbstvertrauen im Umgang mit komplexeren Programmierproblemen hindeuten oder darauf, dass sie sich zunächst auf den Aufbau eines soliden Grundlagenwissens konzentrieren wollten.

In der ChatGPT-Gruppe fiel die Wahl der Aufgaben gleichmäßiger über die Schwierigkeitsgrade verteilt aus, wobei jeweils drei Studierende die „mittelschweren“ und „schweren“ Aufgaben wählten. Dies deutet darauf hin, dass die ChatGPT-Gruppe insgesamt mehr Selbstvertrauen hatte oder durch die Interaktion mit der künstlichen Intelligenz geübt und motiviert war, größere Herausforderungen anzunehmen Farrokhnia et al. (2023), Sedlbauer et al. (2024).

Der schrittweise Ansatz zur Bearbeitung der Aufgaben deutet darauf hin, dass die Studierenden der E-Book-Gruppe ein gründlicheres Verständnis entwickelten, was langfristig Vorteile für die Internalisierung von Programmierkonzepten bieten könnte.

4.5.5Auswirkungen der Projekte

Das Lerntagebuch lieferte eine umfassende Perspektive auf die individuellen Lernerfahrungen und unterstützte die Bewertung der Effektivität der in dieser Studie untersuchten Lernressourcen (Siemens & Long, 2011). Folgende Schlussfolgerungen lassen sich davon für den Unterricht ableiten:

- **E-Book-Gruppe**, die traditionelle Lernmethode förderte die Entwicklung von Problemlösungsfähigkeiten und ein tieferes Verständnis. Diese Ergebnisse stimmen mit der Forschung überein, die zeigt, dass Studierende, die Lösungen eigenständig finden müssen, nachhaltiger lernen (Ben-Ari, 2001).
- **ChatGPT-Gruppe, künstliche Intelligenz**-unterstützte Methode erwies sich als effektiv, um Probleme schnell zu lösen und Fragen zu beantworten. Sie kann besonders hilfreich sein, um sofortige Unterstützung zu bieten, birgt jedoch das Risiko, eine Abhängigkeit zu fördern (Kasneci et al., 2023).

4.6 Interpretation der Reflexionen

Der strukturierte Reflexionsprozess im-Programmierkurs lieferte wertvolle Einblicke in die Entwicklung der Programmierfähigkeiten der Studierenden und zeigte auf, wie die beiden Lernmethoden ihren Fortschritt beeinflussten.

Die Studierenden der ChatGPT-Gruppe schätzten die Möglichkeit, sofortige Antworten zu erhalten, was zu einer schnelleren Lernkurve führte. Ein Studierender bemerkte: „Es fühlt sich an, als hätte ich einen persönlichen Tutor, der mir jederzeit helfen kann“ (Šedlbauer et al., 2024).

Die Studierenden der E-Book-Gruppe betonten, dass das eigenständige Arbeiten ihnen half, ein tieferes Verständnis der Programmierkonzepte zu entwickeln (Grover & Basu, 2017).

Der Vergleich zwischen den Open-Book- und Closed-Book-Tests, zusammen mit den Reflexionen der Studierenden, zeigt, dass beide Lernmethoden unterschiedliche Vorteile bieten. Während die E-Book-Methode eine stärkere Internalisierung des Wissens fördert, unterstützt ChatGPT die sofortige Fehlererkennung und -korrektur und bietet interaktive Hilfe bei der Herangehensweise

an Problemlösungen (Šedlbauer et al., 2024). Eine Kombination beider Ansätze könnte daher eine optimale Lernstrategie darstellen, um sowohl theoretisches Wissen als auch praktische Problemlösungsfähigkeiten zu verbessern.

Die Resultate der Reflexionsprozesse bestätigten die Hypothese, dass die Einbindung von Reflexion in den Lernprozess eine entscheidende Rolle bei der Entwicklung von Metakognition und Selbstregulierung spielt. Sie demonstrierten, dass die Lernenden ihre Lernstrategien durch kontinuierliche Reflexion besser begreifen und gezielt weiterentwickeln, was den Befunden von Wing (2006) entspricht..

4.6.1 Zusammenfassende Interpretation

Die Ergebnisse dieser Untersuchung legen nahe, dass ChatGPT als KI-basiertes Lerninstrument das Potenzial besitzt, den Erwerb von Programmierkompetenzen zu unterstützen, insbesondere durch die Bereitstellung personalisierter Unterstützung und Echtzeit-Feedback. Die Lernenden in der ChatGPT-Gruppe erlangten ein vertieftes Verständnis für Programmierstrategien und zeigten ein erhöhtes Engagement, was mit aktuellen Untersuchungen zur Wirksamkeit von künstlicher Intelligenz im Bildungssektor korrespondiert (Shahid et al., 2023). Die qualitative Untersuchung der Lerntagebücher hat ergeben, dass ChatGPT den Lernprozess bereichern kann, indem es den Lernenden die Möglichkeit bietet, in ihrem eigenen Tempo mit individuell angepasster Unterstützung zu lernen.

Die bedeutsamen Erkenntnisse über das Potenzial des KI-gestützten Lernens im schulischen Umfeld und deutet darauf hin, dass ChatGPT als komplementärer Ansatz zu herkömmlichen Lernmethoden verwendet werden kann, um die Programmierkompetenzen zu optimieren. Diese Untersuchung fördert somit ein vertieftes Verständnis der Einbindung von KI-Technologien in die Bildung und eröffnet neue Perspektiven für zukünftige Forschungsansätze.

4.7 Einflüsse auf die Ergebnisse und die Interpretation

Die Tatsache, dass einige Studierende während der Testphase abwesend waren, birgt das Risiko einer Stichprobenverzerrung. Eine unvollständige Teilnahme kann dazu führen, dass die Stichprobe die gesamte Studierendengruppe nicht vollständig repräsentiert. Es kann eine solche Verzerrung die externe Validität der Studie beeinträchtigen, da die Ergebnisse möglicherweise nicht das Leistungsniveau oder die Fähigkeiten der gesamten Population widerspiegeln. (Himme 2009) Dies bedeutet, dass die Zuverlässigkeit und Generalisierbarkeit der Ergebnisse eingeschränkt sein könnten, da individuelle Unterschiede oder Lernpräferenzen nicht berücksichtigt wurden.

Die abwesenden Studierenden könnten unterschiedliche Leistungsniveaus gehabt haben, was die Durchschnittsergebnisse beider Gruppen beeinflusst haben könnte. Robins et al. (2003) weisen darauf hin, dass solche Unterschiede die tatsächlichen Differenzen zwischen den beiden Gruppen verzerren können. Beispielsweise könnten fehlende leistungsstarke oder leistungsschwache Studierende die Gesamtergebnisse ihrer jeweiligen Gruppen verfälscht haben. Daher ist es denkbar, dass die tatsächlichen Leistungsunterschiede zwischen der E-Book- und der ChatGPT-Gruppe weniger ausgeprägt oder sogar signifikanter sein könnten als die aktuellen Testergebnisse vermuten lassen.

Bei der Interpretation der Testergebnisse ist es wichtig, diese potenziellen Verzerrungen zu berücksichtigen. Obwohl die aktuellen Ergebnisse möglicherweise nicht die gesamte Leistung aller Teilnehmer beider Gruppen vollständig widerspiegeln, bieten sie dennoch wichtige Einblicke in die Effektivität der jeweiligen Lernmethoden. Es ist wichtig, die Ergebnisse als Hinweise, aber nicht als abschließend zu betrachten. Wie Patton (2014) betont, können auch Studien mit begrenzter Stichprobengröße und bestimmten Einschränkungen wertvolle Hinweise darauf geben, wie verschiedene Lernmethoden in realen Bildungskontexten angewendet werden können. Trotz der Abwesenheit einiger Studierender bieten die gewonnenen Erkenntnisse dennoch bedeutungsvolle Informationen über die Stärken und Schwächen von künstliche Intelligenz-unterstütztem Lernen im Vergleich zu traditionellen Lernmethoden.

5. Diskussion

Die zentrale Frage der Studie lautete: „Welche Lernmethode – Lehrbuch oder künstliche Intelligenz – ermöglicht es einem Studierenden, Programmieren effektiver ohne Lehrer zu erlernen?“ Um diese Frage zu beantworten, wurde die Programmierleistung, Lernprozesse, Lerntagebücher und Erfahrungen von Studierenden verglichen, die entweder mit einem E-Book oder ChatGPT als Lernmethode arbeiteten.

Die Hypothese ist: „Es wird angenommen, dass der strukturierte Ansatz den Studierenden Programmieren effektiver beibringt, da die flexible Natur der Unterstützung durch ChatGPT weniger vorhersehbar und anpassbar für das Programmierenlernen ist.“

Die Ergebnisse der Arbeiten zeigen, dass beide Methoden – der strukturierte E-Book-Ansatz und der flexible ChatGPT-Ansatz – unterschiedliche Vorteile und Herausforderungen bieten. Eine eindeutige Antwort auf die Forschungsfrage ist daher nur teilweise möglich, aber klare Tendenzen hinsichtlich der Effektivität und der Flexibilität beim Lernen lassen sich erkennen.

5.1 Effektivität des Wissensaufbaus

Studierende in der E-Book-Gruppe schnitten sowohl im Open-Book- als auch im Closed-Book-Test etwas besser ab als die in der ChatGPT-Gruppe. Zum Beispiel erzielte die E-Book-Gruppe im Closed-Book-Test in den Aufgaben „Ordne den Code“ durchschnittlich 7,09 Punkte, während die ChatGPT-Gruppe nur 5,11 Punkte erreichte. Diese Ergebnisse deuten darauf hin, dass der strukturierte Ansatz des E-Books den Studierenden ein besseres Verständnis von Programmierkonzepten vermittelte und es ihnen ermöglichte, dieses Wissen effektiver ohne externe Unterstützung anzuwenden. (Šedlbauer et al. (2024), Grover & Basu (2017))

Dieses Ergebnis stützt die Hypothese, dass der strukturierte E-Book-Ansatz eine effektivere Methode für den Wissensaufbau im Programmieren sein kann. Die klare, lineare Struktur des E-Books ermöglichte es den Studierenden, Programmierprinzipien gründlich zu erlernen und zu verinnerlichen, was auch in anderen Studien zur Effektivität traditioneller Lernmethoden, wie Büchern, bestätigt wurde. (Papert, 1980).

5.1.1 Programmieren

In Aufgaben, wie dem Zeichnen eines Hauses mit Turtle-Grafiken, zeigte die E-Book-Gruppe eine solide Programmstruktur. Sie verwendeten separate Funktionen zum Zeichnen von Teilen des Hauses:

```
def zeichne_quadrat():
  for _ in range(4):
    forward(100)
    right(90)
def zeichne_dreieck()
...
def erstelle_haus()
```

...zeichne_quadrat()

...zeichne_dreiech()

... print(„Ein Haus wurde gezeichnet!“)

erstelle_haus()

Dieses Beispiel zeigt, dass die Studierenden in der Lage waren, modularen und wiederverwendbaren Code zu erstellen.

Im Gegensatz dazu verwendete ein Studierender aus der ChatGPT-Gruppe einen ineffizienten Ansatz zur Zeichenprüfung von Zeichenketten, wie zum Beispiel:

def zeichne_haus():

...forward(100)

...right(90)

...forward(100)

...right(90)

...forward(100)

...reight(90)

...forward(100)

...print("Ein Haus wurde gezeichnet")

Anstatt mehrere einfache Funktionen zu verwenden, wurde eine einzige Funktion zeichne_haus() erstellt. Dies zeigt, dass der ChatGPT-Ansatz zwar Lösungen bietet, aber auf effizienten Lösungen keinen Fokus legt, außer es wird der künstlichen Intelligenz explizit angewiesen. (Farrokhnia et al. (2023), Šedlbauer et al. (2024)).

5.1.2 Anordnung des Codes

Die E-Book-Gruppe schnitt bei den „Anordnung des Codes“-Aufgaben im Closed-Book-Test deutlich besser ab als die ChatGPT-Gruppe. Dies deutet darauf hin, dass der E-Book-Ansatz den Studierenden half, ein tieferes Verständnis der Codeorganisation und -struktur zu entwickeln, da sie mehr gefordert waren, die strukturellen Elemente eigenständig zu erarbeiten (Rizvi et al. (2023)).

5.1.3 Vervollständigung des Codes

Im Open-Book-Test erzielten beide Gruppen ähnliche Ergebnisse bei den „Vervollständige den Code“-Aufgaben (E-Book: 6,71 Punkte; ChatGPT: 6,67 Punkte). Im Closed-Book-Test schnitt die E-Book-Gruppe jedoch besser ab (5,64 Punkte) als die ChatGPT-Gruppe (4,44 Punkte). Dies zeigt, dass die E-Book-Gruppe durch wiederholte Auseinandersetzung mit den Konzepten ein stabileres Verständnis entwickelte, während die ChatGPT-Gruppe stärker auf die Unterstützung durch die künstliche Intelligenz angewiesen war. Dies bestätigt die Annahme, dass ein strukturierter

Lernansatz die Studierenden besser darauf vorbereitet, Programmierfähigkeiten ohne externe Hilfen anzuwenden.

5.2 Flexibilität und Problemlösungsfähigkeit

Die ChatGPT-Gruppe zeigte bemerkenswerte Flexibilität und Anpassungsfähigkeit beim Lösen von Programmierproblemen, insbesondere bei komplexeren Aufgaben. Die interaktive Natur von ChatGPT ermöglichte es den Studierenden, sofortiges Feedback zu erhalten und verschiedene Ansätze zu erkunden. Dies führte zu besseren Problemlösungsfähigkeiten, insbesondere bei Aufgaben, die mehrere Lösungen erforderten (Šedlbauer et al. (2024), Zhu & Van Brummelen, (2021)). Vermutlich wurde durch verschiedene Antworten von ChatGPT die unterschiedliche Antwortstrukturen und Programmierkonzepte beinhalteten die Flexibilität um Bereich Problemlösung und Schreiben von Code gefördert.

Ein Beispiel hierfür ist die Fähigkeit der ChatGPT-Gruppe, Fehler zu erkennen und zu beheben. Sie erzielten bessere durchschnittliche Ergebnisse in den Fehlersuche-Aufgaben als die E-Book-Gruppe, was zeigt, dass die interaktive Unterstützung durch ChatGPT die Debugging-Fähigkeiten besser förderte. Die ChatGPT-Gruppe musste oft die vorgeschlagenen Lösungen der künstlichen Intelligenz ausbessern oder adaptieren, was die Fähigkeiten zur Fehlererkennung förderte. Dies deutet darauf hin, dass ChatGPT ein effektives Werkzeug zur Entwicklung von Problemlösungsfähigkeiten ist, obwohl dies nicht immer zu einer langfristigen Wissensspeicherung führt. (Farrokhnia et al. (2023), Shahid et al. (2023))

5.2.1 Programmierung

Die Analyse der Programmierlösungen zeigt, dass Studierende, die mit dem E-Book arbeiteten, im Allgemeinen strukturiertere und effizientere Lösungen entwickelten. Dies zeigte sich insbesondere im Umgang mit grundlegenden Programmierkonzepten wie Schleifen, Bedingungen und Funktionen. Ein Studierender der E-Book-Gruppe verwendete beispielsweise effektiv eine for-Schleife zusammen mit einer if-else-Struktur:

```
def tiere_ausgeben(tiere):
  for tier in tiere:
    if tier == "Tiger":
      print("Der Tiger ist der König der Tiere!")
    else:
      print(tier)
```

In einem anderen Beispiel verwendete der Studierende die enumerate-Funktion, was auf ein tieferes Verständnis fortgeschrittener Python-Konzepte hindeutet:

```
def tiere_ausgeben(tiere):
```

```
for index, tier in enumerate(tiere):
    if tier == "Tiger":
        print("Der Tiger ist der König der Tiere!")
    else:
        print(f"{index + 1}. {tier}")
```

Im Gegensatz dazu zeigte die ChatGPT-Gruppe einen eher lösungsorientierten Ansatz, aber es fehlte ihnen an einem tiefen fundamentalen Verständnis. Ein Beispiel hierfür ist die Verwendung einer ineffizienten while-Schleife anstelle einer for-Schleife:

```
def tiere_ausgeben(tiere):
    i = 0
    while i < len(tiere):
        if tiere[i] == "Tiger":
            print("Der Tiger ist der König der Tiere!")
        else:
            print(tiere[i])
        i += 1
```

Dies zeigt, dass der Studierende zwar eine Lösung fand, aber nicht wusste, wie er die Funktion und Iteration effizienter gestalten könnte (Ben-Ari, 2001). Es ist anzunehmen, dass die Lernenden der E-Book-Gruppe sich durch das fokussierte Lesen und der Analyse von vorgegebenen Beispielen im E-Book intensiv mit der Struktur von Programmierkonzepten und Theorie auseinandersetzen und diese Inhalte intensiver wahrnehmen.

Ohne die Unterstützung von ChatGPT war die ChatGPT-Gruppe bei den „Vervollständige den Code“-Aufgaben deutlich weniger erfolgreich. Dies zeigt, dass die Flexibilität des künstlichen Intelligezmodells zwar kurzfristig effektiv ist, aber zu einer Abhängigkeit von der interaktiven Unterstützung führen kann, was die Fähigkeit der Studierenden einschränkt, eigenständig zu arbeiten (Patton, 2014). Die Schüler werden wahrscheinlich sich beim Generieren von Lösungen nicht auf den Inhalt der Generierung konzentriert haben, sondern einfach geprüft haben, ob die Lösung funktioniert. Somit fehlte Ihnen das Wissen über Code schreiben, was beim Vervollständigen wichtig ist.

Ein wesentlicher Unterschied zwischen den beiden Gruppen zeigte sich in Bezug auf Unabhängigkeit und Selbstvertrauen. Die E-Book-Gruppe entwickelte ein höheres Maß an Autonomie, da sie gezwungen war, Lösungen eigenständig zu finden. Ein Studierender der E-Book-Gruppe reflektierte in seinem Lerntagebuch: „Es war schwierig, aber ich habe gelernt, Probleme selbst zu lösen.“ Dies bestätigt Studien, die zeigen, dass strukturierte Lernansätze dazu

beitragen, die Selbstwirksamkeit und Unabhängigkeit von Studierenden zu fördern (Šedlbauer et al. (2024), Rizvi et al. (2023)). Im Gegensatz dazu zeigte die ChatGPT-Gruppe eine Tendenz, sich insbesondere bei komplexeren Aufgaben stärker auf die künstliche Intelligenz zu verlassen. Einige Studierende erwähnten, dass sie sich ohne die Unterstützung von ChatGPT unsicher fühlten. Dies deutet darauf hin, dass der strukturierte E-Book-Ansatz besser geeignet war, Unabhängigkeit zu fördern (Grover & Pea, 2013).

Der Vergleich der Open-Book- und Closed-Book-Testergebnisse zeigt, dass die E-Book-Methode zu einer besseren Internalisierung von Programmierkonzepten führte, insbesondere bei Aufgaben, die das Strukturieren und Vervollständigen von Code ohne externe Hilfe erforderten, da sie sich intensiv und fokussiert beim Erlernen des Programmierens auf das Buch konzentrieren mussten. Die ChatGPT-Gruppe hingegen zeichnete sich durch Flexibilität und Problemlösungsfähigkeit bei komplexeren Aufgaben aus und profitierte von Echtzeit-Feedback. Die Tendenz dieser Gruppe Antworten der künstlichen Intelligenz zu akzeptieren egal welche Aufgabe welcher Schwierigkeit gestellt wurde förderte das Selbstvertrauen komplexere Aufgaben zu lösen.

Beide Lernmethoden bieten einzigartige Stärken. Der strukturierte E-Book-Ansatz fördert ein tieferes Verständnis von Programmierkonzepten, während der ChatGPT-Ansatz Anpassungsfähigkeit und schnelle Problemlösung unterstützt. Eine Kombination beider Ansätze könnte die Programmierausbildung optimieren, indem sowohl ein solides Verständnis der Konzepte als auch die Fähigkeit zur flexiblen Problemlösung gefördert wird.

Zusammengefasst bedeutet das laut diesem Experiment:

- der strukturierte E-Book-Ansatz ist effektiver für den nachhaltigen Wissensaufbau, das Verständnis von Programmierkonzepten und die Förderung von Unabhängigkeit.
- der flexible Einsatz von ChatGPT verbessert Problemlösungsfähigkeiten und Kreativität, führt jedoch häufig zu einer Abhängigkeit von externer Unterstützung und einem weniger strukturierten Wissensaufbau.
- eine Kombination beider Methoden ist ideal, um den Studierenden sowohl ein solides Verständnis der Programmierkonzepte als auch die Fähigkeit zu vermitteln, flexibel auf verschiedene Herausforderungen zu reagieren. Solche hybriden Ansätze könnten die Vorteile beider Lernmethoden vereinen und die Studierenden umfassender und effektiver auf das Programmieren vorbereiten (Šedlbauer et al., 2024).

5.3 Diskussion der Reflexionen

Die während des Programmierkurses durchgeführten Status- und Befindlichkeitsabfragen beziehungsweise Reflexionen lieferten wertvolle Einblicke in die subjektiven Erfahrungen, Lernprozesse und Herausforderungen der Studierenden in beiden Gruppen. Diese Reflexionen halfen, die verschiedenen Auswirkungen der Lernmethoden – ChatGPT und E-Book – auf Motivation, Unabhängigkeit und das Verständnis von Programmierkonzepten besser zu verstehen.

Die Studierenden der ChatGPT-Gruppe hoben häufig die Vorteile der interaktiven Unterstützung durch die künstliche Intelligenz hervor. Viele berichteten, dass sie durch den direkten Zugang zu ChatGPT motivierter waren, insbesondere wenn sie bei einer Aufgabe auf Schwierigkeiten stießen. Ein Studierender bemerkte: „Wenn ich nicht wusste, was ich tun sollte, konnte ich

einfach ChatGPT fragen, und es half mir sofort. Das motivierte mich, weiterzumachen und nicht aufzugeben.“ Diese Erfahrung steht im Einklang mit anderen Studien, die zeigen, dass künstliche Intelligenz-unterstützte Lernwerkzeuge die Lernmotivation durch sofortiges Feedback und individuelle Problemlösungsmöglichkeiten steigern (Rizvi et al., 2023).

Die ChatGPT-Gruppe fand es besonders hilfreich, sofortige Antworten auf ihre Fragen zu erhalten und direkt während des Programmierens unterstützt zu werden. Dies führte zu einer erhöhten Beteiligung und einem größeren Interesse an den Aufgaben, was besonders vorteilhaft für Studierende war, die Schwierigkeiten hatten, komplexe Konzepte zu verstehen. Diese Interaktivität half, die Angst vor Programmierherausforderungen zu verringern und das Vertrauen in ihre Fähigkeit zu stärken, anspruchsvollere Aufgaben zu bewältigen (Šedlbauer et al., 2024).

Trotz dieser positiven Aspekte berichteten viele Studierende der ChatGPT-Gruppe von einer zunehmenden Abhängigkeit von der KI-Unterstützung im Laufe des Kurses. Einige erwähnten, dass sie zunächst versuchten, selbstständig zu arbeiten, aber schnell zu ChatGPT wechselten, sobald sie auf Schwierigkeiten stießen. Ein Studierender sagte: „Ich habe gemerkt, dass ich oft direkt zu ChatGPT gegangen bin, wenn ich die Antwort nicht wusste, anstatt selbst darüber nachzudenken.“ Diese Beobachtung deutet darauf hin, dass die interaktive Natur von ChatGPT zwar hilfreich ist, aber auch dazu führen kann, dass Studierende weniger Zeit mit der eigenständigen Problemlösung verbringen (Kasneci et al., 2023).

Diese Abhängigkeit von ChatGPT kann somit zu einem oberflächlicheren Verständnis im Vergleich zu einem strukturierten Ansatz führen. Ohne die Unterstützung der künstlichen Intelligenz fühlten sich einige Studierende weniger sicher in ihrer Fähigkeit, Probleme eigenständig zu lösen, was sich auch in ihrer Leistung im Closed-Book-Test widerspiegelte.

Die E-Book-Gruppe hob die klare Struktur und den linearen Lernprozess als Hauptvorteil ihrer Methode hervor. Viele Studierende berichteten, dass sie die schrittweisen Erklärungen im E-Book schätzten, da sie ihnen ermöglichten, in ihrem eigenen Tempo zu lernen. Ein Studierender sagte: „Das E-Book hat mir geholfen, die Grundlagen wirklich zu verstehen. Es war manchmal mühsam, aber am Ende wusste ich genau, warum etwas funktioniert.“ Diese Erkenntnis stützt andere Forschungsergebnisse, die zeigen, dass strukturiertes, schrittweises Lernen den Studierenden hilft, ein tieferes und nachhaltigeres Verständnis neuer Konzepte zu entwickeln (Kasneci (2023)).

Da die Studierenden der E-Book-Gruppe keine sofortige Unterstützung hatten, mussten sie sich intensiver mit dem Material auseinandersetzen und eigenständig nach Lösungen suchen. Dies führte zu einer besseren Internalisierung des Wissens und der Entwicklung eines höheren Maßes an Unabhängigkeit. Einige Studierende berichteten, dass ihnen das E-Book Geduld beigebracht habe und die Fähigkeit, verschiedene Ansätze auszuprobieren, bevor sie Hilfe von ihren Mitschülern oder Lehrern suchten. Diese Fähigkeit, eigenständig zu arbeiten, ist ein entscheidender Faktor für langfristigen Lernerfolg (Papert, 1980).

Die Reflexionen zeigen deutlich, dass beide Methoden – ChatGPT und E-Book – unterschiedliche, aber sich ergänzende Vorteile bieten:

- **ChatGPT** fungierte als motivierendes, interaktives Werkzeug, das die Studierenden ermutigte, sich komplexen Aufgaben zu stellen, und half, Frustration bei Programmierherausforderungen zu reduzieren. Es fungierte als „persönlicher Tutor“, das sofortige Hilfe bot und dadurch das Gefühl der Selbstwirksamkeit stärkte (Holmes et al., 2021). Das Risiko der Abhängigkeit führte jedoch dazu, dass das Lernen weniger nachhaltig war, wenn sich die Studierenden zu schnell auf externe Hilfe verließen.

- **Das E-Book** ermöglichte ein tieferes Verständnis von Programmierkonzepten und förderte die Unabhängigkeit. Es zeigte sich, dass strukturiertes, wiederholtes Arbeiten mit dem Material hilft, Programmierkonzepte langfristig zu festigen. Die Studierenden lernten, Probleme eigenständig anzugehen, und entwickelten ein stärkeres Selbstvertrauen in ihre Fähigkeiten.

Auch die aus der Reflexion gewonnenen Erkenntnisse aus den Reflexionen legen nahe, dass eine Kombination bei den Ansätzen ideal sein könnte. Während ChatGPT als Motivationswerkzeug dient, das die Studierenden unterstützt und ermutigt, Herausforderungen anzunehmen, kann der strukturierte E-Book-Ansatz dazu beitragen, ein tieferes Verständnis von Konzepten zu entwickeln und die Unabhängigkeit zu fördern. Durch die Integration beider Methoden könnten Lernprozesse optimiert und sowohl die Motivation als auch der langfristige Lernerfolg gesteigert werden (Farrokhnia et al., 2023).

5.4 Diskussion der Abschlussprojekte und Lerntagebücher

Die Analyse der Abschlussprojekte verdeutlichte die unterschiedlichen Herangehensweisen der beiden Gruppen – E-Book und ChatGPT – an die Aufgaben. Diese Unterschiede spiegeln wider, wie jede Lernmethode das Selbstvertrauen der Studierenden, ihre Problemlösungsstrategien und ihre Fähigkeit, selbstständig zu arbeiten, beeinflusste.

Die Studierenden der ChatGPT-Gruppe neigten dazu, häufiger herausforderndere Projekte zu wählen. Ein zentraler Grund dafür war die Möglichkeit, auf die Unterstützung der KI zurückzugreifen, was ihnen half, sich sicherer zu fühlen und bereit zu sein, sich komplexeren Aufgaben zu stellen. Zum Beispiel entschieden sich 3 von 8 Studierenden in der ChatGPT-Gruppe für die „schwere“ Aufgabe, was darauf hindeutet, dass die interaktive Unterstützung durch die KI sie ermutigte, komplexere Algorithmen zu entwickeln (Šedlbauer et al., 2024). Ein Studierender der ChatGPT-Gruppe erwähnte in seinem Tagebuch: „ChatGPT half mir, den Algorithmus zu verfeinern, indem es mir erklärte, wie ich den Code optimieren konnte“.

Im Gegensatz dazu konzentrierten sich die Studierenden der E-Book-Gruppe eher auf einfachere und mittelschwere Projekte, was darauf hindeutet, dass sie es bevorzugten, in einem vertrauteren Rahmen zu arbeiten. Von den 7 Studierenden wählten 3 die „leichte“ Aufgabe, und nur 1 entschieden sich für die „schwere“ Aufgabe. Die E-Book-Gruppe legte Wert darauf, ein tieferes Verständnis der Grundlagen zu erlangen, und ging die Aufgaben methodisch an. In ihren Tagebucheinträgen betonten die Studierenden, dass sie mehr Zeit investierten, um die Aufgabe schrittweise zu verstehen und Lösungen der künstlichen Intelligenz anzuwenden. Ein Studierender reflektierte: „Ich habe die Aufgabe in kleinere Teile zerlegt und jede Funktion Schritt für Schritt entwickelt, bis ich sicher war, dass sie korrekt war“ (Grover & Pea, 2013).

Studierende, der ChatGPT-Gruppe, profitierten von der Unterstützung durch die KI, da sie darin geübt waren die KI anzuwenden. Viele ihrer Lösungen waren innovativ und enthielten fortgeschrittene Algorithmen, die sie möglicherweise ohne die Hilfe der KI nicht hätten umsetzen können. Allerdings hatten einige Studierende Probleme beim Beschreiben ihrer Programme, weil sie die Abläufe nicht ganz verstanden, was auf eine starke Abhängigkeit von der KI hindeutet (Rizvi et al., 2023).

Die Analyse der Abschlussprojekte zeigt, dass beide Lernmethoden ihre spezifischen Vorteile haben: Die E-Book-Gruppe entwickelte ein tieferes Verständnis der Programmierkonzepte,

während die ChatGPT-Gruppe effizientere Problemlösungsstrategien durch interaktive Unterstützung erwarb. Eine Kombination beider Ansätze könnte das Lernpotenzial maximieren, indem sowohl ein tiefes Verständnis als auch effiziente Problemlösungsfähigkeiten gefördert werden.

5.5 Diskussion der Hypothese

Die Studie zeigt, dass beide Ansätze ChatGPT und E-Book ihre Stärken haben und den Lernprozess je nach Lernziel unterschiedlich beeinflussen.

Die Hypothese, dass ein strukturierter Ansatz effektiver ist, um Programmieren zu lernen, wurde weder bestätigt noch kann sie generell verneint werden. Die E-Book-Gruppe erzielte bessere Ergebnisse bei der Anwendung von Grundlagen und zeigte eine größere Fähigkeit, unabhängig zu arbeiten. Dies stützt die Annahme, dass strukturierte Wissensvermittlung durch ein E-Book den Studierenden hilft, ein nachhaltigeres Verständnis von Programmierkonzepten zu entwickeln (Lye & Koh, 2014).

Die Annahme, dass ChatGPT zu einem flexibleren, aber weniger vorhersehbaren Lernprozess führt, wurde ebenfalls bestätigt. ChatGPT ermöglichte es den Studierenden, komplexe Projekte unabhängig zu versuchen, und förderte kreatives Denken. Diese Methode war jedoch weniger effektiv beim Aufbau eines strukturierten, langfristigen Wissensfundaments und führte zu einer stärkeren Abhängigkeit von der Unterstützung durch die KI (Šedlbauer et al. (2024)).

Die Ergebnisse der Studie legen nahe, dass eine Kombination beider Ansätze optimal wäre. Eine strukturierte Ressource wie das E-Book kann verwendet werden, um eine solide Grundlage an Programmierwissen zu schaffen, während ChatGPT als ergänzendes Werkzeug dienen kann, um die Studierenden bei Schwierigkeiten zu unterstützen und kreatives Denken zu fördern.

Ein hybrides Lehrmodell, das sowohl strukturiertes Lernen als auch interaktive, KI-unterstützte Elemente integriert, könnte den Studierenden helfen, ein tiefes Verständnis der Programmiergrundlagen zu entwickeln und gleichzeitig ihre Problemlösungs- und kreativen Denkfähigkeiten zu verbessern. ChatGPT kann als ergänzendes Werkzeug betrachtet weden, das den Lernprozess bereichert, jedoch nicht als Ersatz für einen zentralen Lehransatz dient (Farrokhnia et al., 2023).

5.6 Einschränkungen und Validitätsbedrohungen

Diese Arbeit liefert wertvolle Einblicke in die Effektivität von ChatGPT im Vergleich zu traditionellen Lernressourcen wie E-Books beim Erlernen von Programmieren für Schüler. Mehrere Faktoren könnten jedoch die Validität und Übertragbarkeit der Ergebnisse beeinflussen. Es ist wichtig, diese Einschränkungen bei der Interpretation der Ergebnisse zu berücksichtigen.

Die Studie wurde mit 35 Studierenden durchgeführt, wobei 19 in der ChatGPT-Gruppe und 16 in der E-Book-Gruppe waren. Diese Gruppengrößen wurden durch organisatorische Zwänge bestimmt, was die Generalisierbarkeit der Ergebnisse einschränkt, da sie möglicherweise nicht repräsentativ für eine größere Schülerschaft ist (Linn & Clancy, 1992). Zudem könnte die ungleiche Gruppengröße Verzerrungen einführen, was die externe Validität der Studie beeinträchtigt (Yin, 2018).

Die zufällige Zuweisung der Gruppen führte zu natürlichen Unterschieden im Vorwissen, den Lernstilen und der Motivation der Studierenden (Robins et al., 2003). Diese Heterogenität könnte die Ergebnisse beeinflusst haben, da es schwer ist, die Effekte der Lernmethoden isoliert zu betrachten (Grover & Pea, 2013).

Der Einsatz von ChatGPT und E-Books im Unterricht könnte zudem durch technische Herausforderungen wie den Zugang zur Technologie oder Unterschiede in den Nutzungsmustern beeinflusst worden sein (Farrokhnia, 2023). Solche Probleme könnten Verzerrungen einführen und die interne Validität der Ergebnisse beeinträchtigen.

Da die Studierenden wussten, welche Lernressource sie verwendeten, könnte ein Erwartungseffekt entstanden sein, insbesondere in der ChatGPT-Gruppe, die sich möglicherweise motivierter fühlte (Holmes et al., 2021). Das Fehlen einer Verblindung könnte somit Motivationsunterschiede verursacht haben, die die Ergebnisse beeinflussten (Zhu & Van Brummelen, 2021).

Externe Faktoren wie der Einfluss von Lehrern, Schulstress oder das soziale Umfeld könnten ebenfalls die Ergebnisse beeinflusst haben, da die Studie in einer realen Schulumgebung stattfand (Hattie & Timperley, 2007). Diese Variablen könnten die interne Validität der Studie weiter beeinträchtigt haben.

Trotz der wertvollen Erkenntnisse weist die Studie mehrere Einschränkungen auf. Die kleine Stichprobengröße und die spezifische Schulumgebung begrenzen die Generalisierbarkeit der Ergebnisse. Zudem hing die Effektivität von ChatGPT stark davon ab, wie gut die Studierenden ihre Fragen formulierten. Zukünftige Forschung sollte daher eine größere Stichprobe und eine längere Studiendauer berücksichtigen, um die langfristigen Effekte der Methoden besser zu verstehen.

Die Studie verglich zwei aktive Interventionsgruppen (ChatGPT und E-Book), jedoch ohne eine Kontrollgruppe, was kausale Schlussfolgerungen erschwert (Šedlbauer et al. 2024). Qualitative Daten aus den Tagebucheinträgen der Studierenden boten zwar tiefe Einblicke, sind aber subjektiv und könnten von persönlichen Vorlieben beeinflusst worden sein (Creswell, 2014).

Die kurze Studiendauer machte es zudem schwierig, langfristige Lerneffekte zu beurteilen (Lye & Koh, 2014). Eine längere Studiendauer wäre erforderlich, um ein dauerhaftes Verständnis und die Nachhaltigkeit der erworbenen Programmierfähigkeiten zu bewerten.

6. Konklusion

Diese Studie untersuchte die Effektivität zweier Lernmethoden – eines traditionellen E-Books und einer auf künstlicher Intelligenz basierenden Lernplattform, ChatGPT – bei der Unterstützung von Schülern im eigenständigen Erwerb von Programmierkenntnissen. Die Hypothese war, dass der strukturierte Ansatz des E-Books effektiver sein würde, da ChatGPT aufgrund seiner Flexibilität und Unvorhersehbarkeit möglicherweise weniger geeignet ist, ein nachhaltiges Verständnis der Programmiergrundlagen zu vermitteln.

Um die Forschungsfrage zu beantworten, wurde ein Experiment durchgeführt, bei dem zwei Gruppen von Schülern über mehrere Unterrichtseinheiten hinweg entweder mit einem E-Book oder mit ChatGPT arbeiteten. Die Leistung der Schülerinnen und Schüler wurde durch Open-Book- und Closed-Book-Tests, Abschlussprojekte und Reflexionstagebücher bewertet.

Die Ergebnisse zeigten, dass der strukturierte Ansatz des E-Books zu einem tieferen Verständnis der Programmiergrundlagen führte, während die ChatGPT-Gruppe eine größere Flexibilität und Kreativität bei der Problemlösung zeigte. Diese Ergebnisse unterstreichen die Vor- und Nachteile beider Lernmethoden und legen nahe, dass eine Kombination beider Ansätze die effektivste Lösung sein könnte.

Die Studie bestätigt die Hypothese, dass der strukturierte E-Book-Ansatz die Schülerinnen und Schüler besser auf die Programmiergrundlagen vorbereitet hat, was sich besonders in ihrer Leistung bei den Closed-Book-Tests zeigte. Diese Ergebnisse stimmen mit der Lerntheorie von Sweller et al. (2011) überein, die betont, dass ein klar strukturierter Lernpfad das Verständnis und die Internalisierung von Kernkonzepten verbessert. Gleichzeitig zeigte die ChatGPT-Gruppe eine beeindruckende Fähigkeit zur kreativen Problemlösung und nahm oft komplexere Aufgaben in Angriff. Dies unterstreicht den Wert interaktiver und explorativer Lernmethoden.

Eine unerwartete Erkenntnis war, dass die ChatGPT-Gruppe trotz der Übernahme komplexerer Aufgaben häufig weniger effiziente Programmiermethoden verwendete. Dies deutet darauf hin, dass ChatGPTs exploratives Lernen zwar kreatives Problemlösen fördert, jedoch nicht immer zu einem optimalen Lernprozess führt.

Die Ergebnisse dieser Studie haben wichtige Implikationen für das Design von Lehrmaterialien und Lehrmethoden im Programmierunterricht. Die unterschiedlichen Stärken beider Ansätze legen nahe, dass eine hybride Lernmethode – die die Struktur eines E-Books mit der Flexibilität von ChatGPT kombiniert – der effektivste Weg sein könnte, um Programmierfähigkeiten zu vermitteln.

Lehrkräfte könnten das E-Book als primäre Lernressource nutzen, um die Programmiergrundlagen zu vermitteln, während ChatGPT als ergänzendes Werkzeug verwendet werden könnte, um eigenständiges Problemlösen zu fördern. Entscheidungsträger und Lehrplanentwickler sollten auch das Potenzial KI-gestützter Lernwerkzeuge in Betracht ziehen, um selbstgesteuertes Lernen zu verbessern und individuelle Lernbedürfnisse besser zu adressieren.

Zusammenfassend zeigt die Studie, dass sowohl der strukturierte Lernansatz des E-Books als auch der flexible Ansatz von ChatGPT spezifische Vorteile in der Programmierausbildung bieten. Während der E-Book-Ansatz ein tieferes und nachhaltigeres Verständnis der Programmiergrundlagen fördert, erleichtert ChatGPT kreatives Problemlösen und einen flexiblen Umgang mit Programmierherausforderungen.

Die Ergebnisse legen nahe, dass die optimale Lösung wahrscheinlich in der Kombination beider Ansätze liegt. Diese Studie leistet einen wichtigen Beitrag zum bestehenden Wissensstand, indem sie aufzeigt, wie unterschiedliche Lernmethoden die Entwicklung von Programmierfähigkeiten beeinflussen. Sie eröffnet neue Perspektiven für den Einsatz von KI-gestützten Lernwerkzeugen in der Bildung und legt eine solide Grundlage für zukünftige Forschung.

6.1 Zukünftige Forschung

Diese Studie dient als Ausgangspunkt für weitere Forschungen, insbesondere zur Integration von KI-basierten Werkzeugen in die Bildungslandschaft. Es bleibt noch viel zu erforschen, wie solche Technologien effektiv eingesetzt werden können, um verschiedene Lernstile und -bedürfnisse zu unterstützen. Zukünftige Forschungen könnten auf den hier gewonnenen Erkenntnissen aufbauen, um innovative und inklusive Ansätze für die Programmierausbildung zu entwickeln, die strukturierten Wissenserwerb mit der Förderung kreativer Problemlösungsfähigkeiten in Einklang bringen.

Da es sich bei dieser Studie um eine explorative Untersuchung handelt, gibt es mehrere Ansätze für zukünftige Forschung. Zunächst wäre es sinnvoll, die Studie mit einer größeren Stichprobengröße und über einen längeren Zeitraum zu wiederholen, um die Ergebnisse zu validieren und langfristige Effekte zu untersuchen. Darüber hinaus sollte der Einfluss eines hybriden Ansatzes – der sowohl E-Book als auch ChatGPT kombiniert – auf die Lernergebnisse untersucht werden.

Zukünftige Studien sollten auch untersuchen, wie Schüler mit unterschiedlichen Kompetenzniveaus von den beiden Ansätzen profitieren. Es wäre besonders interessant herauszufinden, ob fortgeschrittene Schüler mehr von ChatGPT profitieren, während Anfänger besser durch den strukturierten Ansatz unterstützt werden. Zudem wäre es wertvoll, die Nutzung von ChatGPT in verschiedenen Programmierumgebungen oder bei Schülern unterschiedlichen Alters zu erforschen.

7. Literaturverzeichnis

1. Rizvi, S., Waite, J., & Sentance, S. (2023). *Artificial Intelligence teaching and learning in K-12 from 2019 to 2022: A systematic literature review*. *Computers and Education: Artificial Intelligence, 4*, 100145. https://doi.org/10.1016/j.caeai.2023.100145.

2. Zhu, J., & Van Brummelen, J. (2021). *Teaching Students About Conversational AI Using CONVO, a Conversational Programming Agent*. IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC), 1-5. DOI: 10.1109/VL/HCC51201.2021.9576290

3. Šedlbauer, J., Činčera, J., Slavík, M., & Hartlová, A. (2024). Students' reflections on their experience with ChatGPT. *Journal of Computer Assisted Learning*, 1–9. https://doi.org/10.1111/jcal.12967

4. Farrokhnia, M., Banihashem, S. K., Noroozi, O., & Wals, A. (2023). A SWOT analysis of ChatGPT: Implications for educational practice and research. *Innovations in Education and Teaching International*. https://doi.org/10.1080/14703297.2023.2195846

5. Shahid, A., Hayat, K., Iqbal, Z., & Jabeen, I. (2023). *Comparative Analysis: ChatGPT vs Traditional Teaching Methods*. Pakistan Journal of Society, Education and Language (PJSEL), 9(2), 585-593.

6. Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., Krusche, S., Kutyniok, G., Michaeli, T., Nerdel, C., Pfeffer, J., Poquet, O., Sailer, M., Schmidt, A., Seidel, T., Stadler, M., Weller, J., Kuhn, J., & Kasneci, G. (2023). *ChatGPT for Good? On Opportunities and Challenges of Large Language Models for Education*. Retrieved from ResearchGate.

7. Pellegrino, J. W., & Hilton, M. L. (2012). *Education for Life and Work: Developing Transferable Knowledge and Skills in the 21st Century*. National Academies Press.

8. Lye, S. Y., & Koh, J. H. L. (2014). *Review on teaching and learning of computational thinking through programming: What is next for K-12?*. Computers in Human Behavior

9. Grover, S., & Pea, R. (2013). *Computational Thinking in K–12: A Review of the State of the Field*. Educational Researcher, 42(1), 38–43.
10. Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. Basic Books, Inc.
11. Wing, J. M. (2006). *Computational Thinking*. Communications of the ACM, 49(3), 33-35.
12. Grover, S., & Basu, S. (2017). *Measuring student learning in introductory block-based programming: Examining misconceptions of loops, conditionals, and Boolean logic*. Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education
13. Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., ... & Kafai, Y. (2009). *Scratch: Programming for all*. Communications of the ACM, 52(11), 60-67.
14. Creswell, J. W. (2014). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches*. SAGE Publications
15. Merriam, S. B., & Tisdell, E. J. (2015). *Qualitative Research: A Guide to Design and Implementation* (4. Aufl.). Jossey-Bass.

16. Kohn, T. (2019). *Programming in Python: A Brief Introduction with TigerJython and Turtle-Graphics*. Retrieved from http://jython.tobiaskohn.ch/
17. Yin, R. K. (2018). *Case Study Research and Applications: Design and Methods* (6th ed.). SAGE Publications.
18. Patton, M. Q. (2014). *Qualitative Research & Evaluation Methods: Integrating Theory and Practice* (4th ed.). SAGE Publications.
19. Robins, A., Rountree, J., & Rountree, N. (2003). *Learning and teaching programming: A review and discussion*. Computer Science Education, 13(2), 137-172.
20. Mannila, L., Peltomaki, M., & Salakoski, T. (2006). *What about a simple language? Analyzing the difficulty level of object-oriented programming*. Computer Science Education, 16(3), 211-227
21. Parsons, D., & Haden, P. (2006). *Parson's Programming Puzzles: A Fun and Effective Learning Tool for First Programming Courses*. In Proc. 8th Australasian Computing Education Conference (ACE 2006) (Vol. 52)
22. Winslow, L. E. (1996). *Programming pedagogy—a psychological overview*. ACM SIGCSE Bulletin, 28(3), 17-22
23. Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, 49(3), 33-35.
24. Lister, R., Adams, E. S., Fitzgerald, S., Fone, W., Hamer, J., Lindholm, M., ... & Seppälä, O. (2004). A multi-national study of reading and tracing skills in novice programmers. *ACM SIGCSE Bulletin*, 36(4), 119-150
25. Lahtinen, E., Ala-Mutka, K., & Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. *ACM SIGCSE Bulletin*, 37(3), 14-18
26. Ben-Ari, M. (2001). Constructivism in computer science education. *Journal of Computers in Mathematics and Science Teaching*, 20(1), 45-73
27. Linn, M. C., & Clancy, M. J. (1992). *The case for case studies of programming problems*. *Communications of the ACM*, 35(3), 121-132. https://doi.org/10.1145/129888.129894
28. Hmelo-Silver, C. E. (2004). *Problem-Based Learning: What and How Do Students Learn?* Educational Psychology Review, 16(3), 235-266. https://doi.org/10.1023/B:EDPR.0000034022.16470.f3
29. Fitzgerald, S., Simon, B., Thomas, L., & Webb, M. (2008). Debugging: Finding, Fixing and Flailing, A Multi-Institutional Study of Novices. *Computer Science Education*, 18(2), 93-116.
30. McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagenson, D., Kolikant, Y. B.-D., & Laxer, C. (2001). A Multi-National, Multi-Institutional Study of Assessment of Programming Skills of First-Year CS Students. *In Working Group Reports from ITiCSE on Innovation and Technology in Computer Science Education* (pp. 125-180)
31. Tew, A. E., & Guzdial, M. (2010). Developing a validated assessment of fundamental CS1 concepts. *Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (SIGCSE '10).
32. Ryan, R. M., & Deci, E. L. (2000). *Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being*. American Psychologist, 55(1), 68-78.
33. Siemens, G., & Long, P. (2011). *Penetrating the Fog: Analytics in Learning and Education*. EDUCAUSE Review, 46(5), 30–32.
34. Hattie, J., & Timperley, H. (2007). *The Power of Feedback*. Review of Educational Research, 77(1), 81-112.
35. Kolb, D. A. (1984). *Experiential Learning: Experience as the Source of Learning and Development*. Prentice-Hall.
36. Hattie, J. (2021). *Visible Learning: A Synthesis of Over 800 Meta-Analyses Relating to Achievement*. Routledge.

37. Zawacki-Richter, O., Marín, V. I., Bond, M., & Gouverneur, F. (2019). Systematic review of research on artificial intelligence applications in higher education. *International Journal of Educational Technology in Higher Education*, 16(1), 1-27.
38. Lai, J. W. M., & Bower, M. (2019). How is the use of technology in education evaluated? A systematic review. *Computers & Education*, 133, 27-42.
39. Holmes, W., Bialik, M., & Fadel, C. (2021). *Artificial Intelligence in Education: Promises and Implications for Teaching and Learning*. Center for Curriculum Redesign.
40. Sung, Y. T., Chang, K. E., & Liu, T. C. (2020). The effects of integrating mobile devices with teaching and learning on students' learning performance: A meta-analysis and research synthesis. *Computers & Education*, 94, 252-275.
41. Anderson, J., Rainie, L., & Luchsinger, A. (2021). *Artificial Intelligence and the Future of Humans*. Pew Research Center.
42. Shadish, W. R., Cook, T. D., & Campbell, D. T. (2002). *Experimental and Quasi-Experimental Designs for Generalized Causal Inference*. Houghton Mifflin.
43. Selwyn, N. (2014). *Digital Technology and the Contemporary University: Degrees of Digitization*. Routledge.
44. Sasikala, P., & Ravichandran, R. (2024). Study on the Impact of Artificial Intelligence on Student Learning Outcomes. *Journal of Digital Learning and Education*, 4(2), 145-155. https://doi.org/10.52562/jdle.v4i2.1234
45. Jyothy, S. N., Kolil, V. K., Raman, R., & Achuthan, K. (2024). *Exploring large language models as an integrated tool for learning, teaching, and research through the Fogg Behavior Model: a comprehensive mixed-methods analysis.* Cogent Engineering, 11(1), 2353494. https://doi.org/10.1080/23311916.2024.2353494
46. Roll, I., & Wylie, R. (2016). Evolution and Revolution in Artificial Intelligence in Education. *International Journal of Artificial Intelligence in Education, 26(2), 582-599*.
47. Ayeni, O. O., Al Hamad, N. M., Chisom, O. N., Osawaru, B., & Adewusi, O. E. (2024). AI in education: A review of personalized learning and educational technology. *GSC Advanced Research and Reviews, 18(02)*, 261-271. DOI: 10.30574/gscarr.2024.18.2.0062
48. Luckin, R., Holmes, W., Griffiths, M., & Forcier, L. B. (2016). *Intelligence Unleashed: An argument for AI in Education*. Pearson.
49. Akinwalere, S. N., & Ivanov, V. T. (2022). Artificial Intelligence in Higher Education: Challenges and Opportunities. Border Crossing, 12(1), 1-15. https://doi.org/10.33182/bc.v12i1.2015
50. Siti Nurulain Mohd Rum und Maslina Zolkepli (2018). Metacognitive Strategies in Teaching and Learning Computer Programming. *International Journal of Engineering & Technology*, 7(4.38), 788-794. https://doi.org/10.14419/ijet.v7i4.38.25084
51. Himme, A. (2009). Gütekriterien der Messung: Reliabilität, Validität und Generalisierbarkeit. In: Albers, S., Klapper, D., Konradt, U., Walter, A., Wolf, J. (eds) Methodik der empirischen Forschung. Gabler Verlag, Wiesbaden. https://doi.org/10.1007/978-3-322-96406-9_31

8. Abbildungsverzeichnis

9. Tabellenverzeichnis

10. Anhang

Im Anhang werden alle Mittel für die Beschreibung und Reproduktion der Studie benötigten Informationen angeführt.

10.1 Einheit 1

TigerJython ist eine spezielle Version von Jython (Java-basierte Python-Implementierung) mit einer integrierten Turtle-Graphics-Bibliothek, die speziell für den Einsatz im Unterricht entwickelt wurde. Es ist kostenlos und läuft auf Windows, macOS und Linux.

- Schritt 1: Laden Sie die neueste Version von TigerJython herunter. Gehen Sie zur offiziellen TigerJython-Website unter https://www.tigerjython.ch und laden Sie die neueste Version für Ihr Betriebssystem herunter.
- Schritt 2: Installieren Sie TigerJython
- Windows: Doppelklicken Sie auf die heruntergeladene Datei und folgen Sie den Anweisungen im Installationsassistenten.
- macOS: Öffnen Sie das heruntergeladene DMG-Paket und ziehen Sie das TigerJython-Programm in den Ordner "Anwendungen".
- Linux: Extrahieren Sie das Archiv und starten Sie das TigerJython-Programm aus dem extrahierten Ordner.
- Schritt 3: Starten Sie TigerJython - öffnen Sie TigerJython, indem Sie auf das Symbol im Startmenü (Windows), im Launchpad (macOS) oder in Ihrem Anwendungsverzeichnis (Linux) klicken.
- Schritt 4: Starten Sie die Turtle-Shell - in TigerJython öffnen Sie die Turtle-Shell, indem Sie auf den "Turtle"-Knopf im unteren Bereich des Fensters klicken.
- Schritt 5: Verwenden Sie die Turtle-Graphics-Bibliothek. Sie können jetzt die Turtle-Graphics-Bibliothek verwenden, um geometrische Formen auf der Turtle-Leinwand zu zeichnen. Verwenden Sie das from turtle import * Statement, um die Turtle-Graphics-Bibliothek zu importieren, und beginnen Sie dann mit dem Programmieren.

10.1.1Übungsaufgaben für SchülerInnen

Erstelle ein Programm, dass “Hello World!” in Tigerjython in der Textausgabe ausgibt.

10.1.1.1 Einfache Formen

AUFGABEN

1. Zeichne mit der Turtle ein regelmässiges Fünfeck (Pentagon) mit einer Seitenlänge von 150 Pixeln.

2. Zeichne mit der Turtle zwei Quadrate ineinander wie in Abbildung 2.1(a).

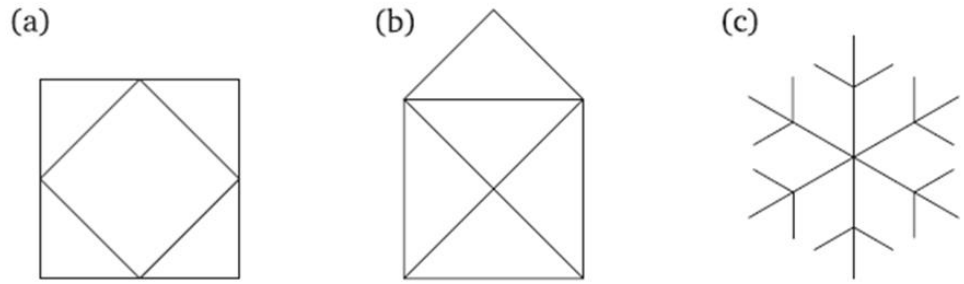

Abbildung 2.1: In der Mitte das «Haus von Nikolaus» und rechts eine einfache Schneeflocke.

3. Das «Haus vom Nikolaus» ist ein Zeichenspiel für Kinder. Ziel ist es, das besagte Haus (vgl. Abbildung 2.1(b)) in einem Linienzug aus genau 8 Strecken zu zeichnen, ohne dabei eine Strecke zweimal zu durchlaufen. Zeichne das Haus vom Nikolaus mithilfe der Turtle. Wähle dabei für die Seitenlänge des Quadrats 120 Pixel und nimm an, dass die beiden Dachflächen rechtwinklig aufeinander treffen.

4.* Lass die Turtle eine einfache Schneeflocke zeichnen wie in der Abbildung 2.1(c).

- Zeichne ein Quadrat: Schreibe ein Programm, das eine Turtle erstellt und diese bewegt, um ein Quadrat zu zeichnen. Experimentiere mit der Größe des Quadrats und den Winkeln, um es schräg oder geneigt zu machen.
- Zeichne ein Dreieck: Schreibe ein Programm, das eine Turtle erstellt und diese bewegt, um ein Dreieck zu zeichnen. Experimentiere mit der Größe und Form des Dreiecks und versuche, verschiedene Arten von Dreiecken zu zeichnen, wie beispielsweise gleichschenklige oder ungleichschenklige Dreiecke.
- Zeichne eine Spirale: Schreibe ein Programm, das eine Turtle erstellt und diese bewegt, um eine Spirale zu zeichnen. Experimentiere mit der Größe und Form der Spirale, indem du verschiedene Werte für die Linienlänge und Winkel verwendest.
- Zeichne ein Rechteck: Schreibe ein Programm, das eine Turtle erstellt und diese bewegt, um ein Rechteck zu zeichnen. Experimentiere mit der Größe und Form des Rechtecks, indem du die Länge und Breite des Rechtecks variierst.
- Zeichne ein Stern: Schreibe ein Programm, das eine Turtle erstellt und diese bewegt, um einen Stern zu zeichnen. Experimentiere mit der Anzahl der Ecken und der Größe des

Sterns, indem du die Länge der Linien und die Winkel zwischen ihnen anpasst.

AUFGABEN

5. Zeichne mit der Turtle ein regelmässiges Sechseck (Hexagon) und wähle für jede Seite eine andere Farbe.

6. Lass die Turtle einen «Regenbogen» zeichnen wie in der Abbildung 2.2 angedeutet. Ganz innen ist dieser Regenbogen *rot*, danach *orange*, *gelb*, *grün* und schliesslich aussen *blau* und *violett*. Es genügt allerdings auch, wenn du nur drei Bogen zeichnen lässt.

Abbildung 2.2: Ein (eckiger) Regenbogen.

AUFGABEN

7. Definiere einen Befehl für ein Quadrat, das auf der Spitze steht und zeichne damit die Figur in der Abbildung 2.3. Die fünf Quadrate berühren sich nicht, sondern haben einen kleinen Abstand und die Farben *blau*, *schwarz*, *rot* (oben) und *gelb*, *grün* (unten).

Abbildung 2.3: Olympische Quadrate.

10.1.1.2 Übung 1: Einführung in Variablen

- Erstelle ein Programm, das zwei Zahlen speichert und sie addiert. Verwende dazu Variablen.
- Berechne die Summe der beiden Zahlen und gib sie mit print() aus.
- Ändere die Werte der Variablen und probiere verschiedene Rechenoperationen wie Minus, Mal und Geteilt aus.

10.1.1.3 Übung 2: Arbeiten mit verschiedenen Datentypen

- Schreibe ein Programm, das deinen Namen, dein Alter und eine Punktzahl speichert.

- Gib alle Informationen mit print() aus.
- Ändere die Werte der Variablen und arbeite mit verschiedenen Datentypen (Ganzzahl, Gleitkommazahl, Zeichenkette).

10.1.1.4 Übung 3: Turtle-Grafik mit Variablen

- Schreibe ein Programm, in dem die Turtle eine Linie zeichnet. Die Länge der Linie wird durch eine Variable bestimmt.
- Ändere die Länge der Linie, indem du den Wert der Variable veränderst.
- Probiere aus, die Turtle in andere Richtungen zu drehen.

10.1.1.5 Übung 4: Ein Quadrat mit der Turtle

- Schreibe ein Programm, das ein Quadrat zeichnet. Die Seitenlänge wird durch eine Variable festgelegt.
- Verändere die Seitenlänge der Turtle, um unterschiedlich große Quadrate zu zeichnen.

10.1.1.6 Übung 5: Rechnen mit Variablen

- Schreibe ein Programm, das das Alter in „Hundejahren“ berechnet (1 Menschenjahr = 7 Hundejahre).
- Lasse den Benutzer sein Alter eingeben und berechne das Alter in Hundejahren mit input().

10.2 Einheit 2

10.2.1 Wiederholung

10.2.1.1 Code analysieren

Beispiel 1:

```
from gturtle import *

t = makeTurtle()

t.forward(100)
t.right(90)
t.forward(100)
t.right(90)
t.forward(100)
t.right(90)
t.forward(100)
```

Beispiel 2:

```
from gturtle import *

t = makeTurtle()

t.forward(100)
t.right(72)
t.forward(100)
t.right(72)
t.forward(100)
t.right(72)
t.forward(100)
t.right(72)
t.forward(100)
```

Beispiel 3:

```
from gturtle import *

t = makeTurtle()

t.forward(50)
t.right(90)
t.forward(50)
t.right(90)
t.forward(50)
t.right(90)
t.forward(50)
t.right(90)
t.right(10)

t.forward(50)
t.right(90)
t.forward(50)
t.right(90)
t.forward(50)
t.right(90)
t.forward(50)
t.right(90)
t.right(10)
```

10.2.1.2 Fehlersuche

Beispiel 1:

```
1  from gturtle import *
2
3  t = makeTurtle()
4
5  t.forward(100
6  t.right(90)
7  t.forward(100)
8  t.right(90)
9  t.forward(100)
10 t.right(90)
11 t.forward(100)
12
```

Beispiel 2:

```
1  import turtle
2
3  t = makeTurtle()
4
5  t.forward(100)
6  t.right(90)
7  t.forwar(100)
8  t.right(90)
9  t.forward(100)
10 t.right(90)
11 t.forward(100)
12
```

Beispiel 3:

```
1  import turtle
2
3  t = makeTurtle()
4
5  t.forward(-100)
6  t.right(90)
7  t.forward(100)
8  t.right(90)
9  t.forward(100)
10 t.right(90)
11 t.forward(100)
12
```

10.2.1.3 Ergänzen bzw Verbessern

Beispiel 1:

```
2  import turtle
3
4  t = turtle.Turtle()
5
6  t.forward(50)
7  # TODO: Schritt 2
8  t.forward(50)
9  t.right(90)
10 # TODO: Schritt 6
11 # TODO: Schritt 7
12 t.forward(50)
13 # TODO: Schritt 9
14 t.right(90)
15 # TODO: Schritt 11
16 # TODO: Schritt 12
17 # TODO: Schritt 13
18
19
```

10.2.2 Übungen

Befehle und Parameter

AUFGABEN

8. Definiere einen Befehl `jump(Distanz)`, mit dem die Turtle die angegebene Distanz überspringt. Der Befehl funktioniert also wie `forward()`, allerdings zeichnet die Turtle hier keine Spur.

9. Definiere einen Befehl `rechteck(grundseite)`, mit dem die Turtle ein Rechteck zeichnet, das doppelt so hoch wie breit ist.

10.* Definiere einen Befehl, der ein offenes Quadrat ⊓ zeichnet und verwende deinen Befehl, um ein Kreuz zu zeichnen, wobei du mit dem offenen Quadrat die vier Arme zeichnest.

AUFGABEN

8. Definiere einen Befehl `jump(Distanz)`, mit dem die Turtle die angegebene Distanz überspringt. Der Befehl funktioniert also wie `forward()`, allerdings zeichnet die Turtle hier keine Spur.

9. Definiere einen Befehl `rechteck(grundseite)`, mit dem die Turtle ein Rechteck zeichnet, das doppelt so hoch wie breit ist.

10.* Definiere einen Befehl, der ein offenes Quadrat ⊓ zeichnet und verwende deinen Befehl, um ein Kreuz zu zeichnen, wobei du mit dem offenen Quadrat die vier Arme zeichnest.

AUFGABEN

11. Die Abbildung 2.4 zeigt vier Figuren mit farbigen Flächen. Lass deine Turtle diese vier Figuren zeichnen (in vier verschiedenen Programmen). Die Farben kannst du dabei selbst wählen.

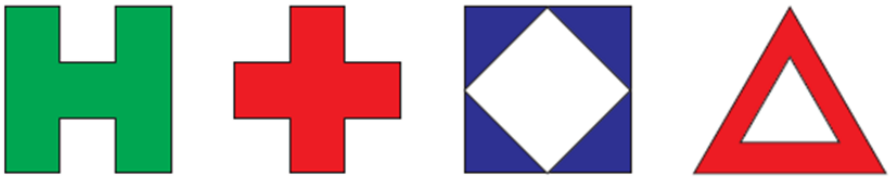

Abbildung 2.4: Figuren mit gefüllten Flächen.

12. Definiere einen neuen Befehl `fillQuadrat(seite)` um eine ausgefülltes Quadrat zu zeichnen. Verwende dann diesen neuen Befehl, um die ersten drei Figuren in der Abbildung 2.4 nochmals zu zeichnen.

AUFGABEN

16. Zeichne mit der Turtle einen Kreis mit einem Radius von $r = 20$ Pixeln.

17. Definiere einen eigenen Befehl `circle(s)`, um einen Kreis zu zeichnen und einen Befehl `fillcircle(s)`, um einen ausgefüllten Kreis zu zeichnen. Der Parameter s gibt dabei an, um wieviel die Turtle bei jedem Schritt vorwärts gehen soll.

18. Lass deine Turtle den PacMan (a), das Yin-Yang-Symbol (b) und ein beliebiges Smiley (c) zeichnen wie in der Abbildung 2.6. Bei den kleinen Punkten genügt es meist, ein kurzes, breites Linienstück zu zeichnen.

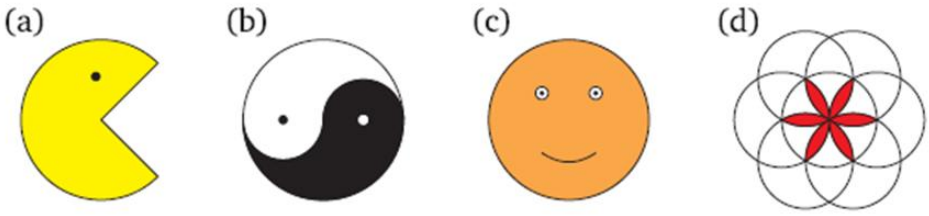

Abbildung 2.6: Kreisfiguren.

19. Die «Blume» in der Abbildung 2.6(d) setzt sich aus sieben gleich grossen Kreisen zusammen. Verwende einen eigenen Befehl `circle()` und zeichne damit diese Blume.

- Zeichne einen Kreis: Schreibe ein Programm, das eine Turtle erstellt und diese bewegt, um einen Kreis zu zeichnen. Experimentiere mit der Größe des Kreises und der Anzahl der Segmente, um einen glatteren oder stärker gestuften Kreis zu erzeugen
- Zeichne eine Blume: Schreibe ein Programm, das eine Turtle erstellt und diese bewegt, um eine Blume zu zeichnen. Experimentiere mit der Anzahl der Blütenblätter, ihrer Größe und ihrer Form, um eine einzigartige Blume zu erstellen
- Zeichne ein Haus: Schreibe ein Programm, das eine Turtle erstellt und diese bewegt, um ein Haus zu zeichnen. Experimentiere mit der Größe und Form des Hauses, indem du verschiedene Winkel und Linienlängen verwendest, um das Dach, die Wände und die Türen und Fenster zu zeichnen
- Zeichne einen Baum: Schreibe ein Programm, das eine Turtle erstellt und diese bewegt, um einen Baum zu zeichnen. Experimentiere mit der Größe und Form des Baums, indem du verschiedene Winkel und Linienlängen verwendest, um den Stamm, die Äste und das Laub zu zeichnen.
- Zeichne ein Tier: Schreibe ein Programm, das eine Turtle erstellt und diese bewegt, um ein Tier deiner Wahl zu zeichnen. Experimentiere mit der Größe und Form des Tiers, indem du verschiedene Linien und Winkel verwendest, um seine Form und Merkmale darzustellen.

- Erstelle ein geometrisches Muster: Verwende Turtle Graphics, um ein eigenes geometrisches Muster zu erstellen. Experimentiere mit verschiedenen Formen, Größen, Farben und Winkeln, um ein einzigartiges Muster zu erstellen.
- Erstelle ein abstraktes Kunstwerk: Verwende Turtle Graphics, um ein eigenes abstraktes Kunstwerk zu erstellen. Experimentiere mit verschiedenen Linien, Formen, Farben und Schattierungen, um ein einzigartiges und interessantes Kunstwerk zu erstellen
- Erstelle ein Mosaik: Verwende Turtle Graphics, um ein eigenes Mosaik zu erstellen. Experimentiere mit verschiedenen Formen, Farben und Anordnungen, um ein einzigartiges Mosaikbild zu erstellen.
- Erstelle eine Tierzeichnung: Verwende Turtle Graphics, um eine Zeichnung deines Lieblingstiers zu erstellen. Experimentiere mit verschiedenen Linien, Formen und Farben, um eine realistische oder abstrakte Darstellung des Tiers zu erstellen.
- Erstelle ein Logo: Verwende Turtle Graphics, um ein eigenes Logo für ein Unternehmen oder eine Organisation zu erstellen. Experimentiere mit verschiedenen Schriftarten, Formen und Farben, um ein einzigartiges und ansprechendes Logo zu erstellen.

10.2.3 Variablen und Rechenoperationen

Übung 1: Wiederholung von Variablen

- Erstelle ein Programm, das zwei Zahlen speichert und die Summe, Differenz, das Produkt und den Quotienten dieser beiden Zahlen berechnet.
- Gib die Ergebnisse mit passenden Texten aus.

Übung 2: Rechnen mit Ganzzahlen und Gleitkommazahlen

- Erstelle ein Programm, das zwei Zahlen speichert: eine Ganzzahl und eine Gleitkommazahl.
- Berechne die Summe und das Produkt der beiden Zahlen. Beobachte, wie Python mit den unterschiedlichen Datentypen umgeht.

Übung 3: Umfang eines Rechtecks berechnen

- Schreibe ein Programm, das die Länge und Breite eines Rechtecks in zwei Variablen speichert.
- Berechne den Umfang des Rechtecks und gib das Ergebnis aus.

Übung 4: Fläche eines Kreises berechnen

- Schreibe ein Programm, das den Radius eines Kreises in einer Variablen speichert.
- Berechne die Fläche des Kreises und gib das Ergebnis aus.

Übung 5: Durchschnitt von drei Zahlen

- Erstelle ein Programm, das drei Zahlen speichert.
- Berechne den Durchschnitt dieser drei Zahlen und gib das Ergebnis aus.

Übung 6: Temperaturumrechnung

- Schreibe ein Programm, das eine Temperatur in Grad Celsius speichert und diese in Fahrenheit umrechnet.
- Gib das Ergebnis aus.

Übung 7: Umwandlung von Sekunden in Stunden, Minuten und Sekunden

- Schreibe ein Programm, das eine Anzahl von Sekunden speichert.
- Berechne und gib aus, wie viele Stunden, Minuten und Sekunden dies sind.

Übung 8: Zeichenketten und Zahlen kombinieren

- Erstelle ein Programm, das deinen Namen und dein Alter in zwei Variablen speichert.
- Gib eine Nachricht aus, die deinen Namen und dein Alter in einem Satz kombiniert, z. B.: „Hallo, ich heiße [Name] und ich bin [Alter] Jahre alt.

AUFGABEN

6. Bei der Division durch 4 können im Prinzip die Reste $0, 1, 2, 3$ auftreten. Bei Quadratzahlen kommen aber nicht alle vier Möglichkeiten vor. Rechne mit einigen Quadratzahlen durch, welche der vier möglichen Reste bei der Division durch 4 auftreten.

7. Bei Geldautomaten gibst du einen Betrag ein. Der Automat muss dann ausrechnen, wie viele Noten von jedem Typ er dazu ausgeben soll. Der Automat in unserer Aufgabe kennt die Notentypen «200», «100» und «20». Schreibe ein Programm, das für den Gesamtgeldbetrag ausrechnet, wie viele Noten von jedem Typ ausgegeben werden sollen – dabei sollen möglichst grosse Noten verwendet werden. Das funktioniert natürlich nur für Beträge, die auch aufgehen, z. B. $480 \rightarrow 2 \cdot 200 + 4 \cdot 20$.

8.* Eine Herausforderung für Profis: Nimm beim Geldautomaten noch «50» als Notentyp hinzu und lass dein Programm auch für 210 die korrekte Antwort geben: $100 + 50 + 3 \cdot 20$.

9. Schreibe ein Programm, das eine dreistellige Zahl in Hunderter, Zehner und Einer zerlegt.

10.* Schreibe ein Programm, das natürliche Zahlen bis 255 ins Binärsystem umrechnet. Für 202 soll also beispielsweise `1 1 0 0 1 0 1 0` ausgegeben werden.

AUFGABEN

15. Schreibe ein Programm, das den Umfang und den Flächeninhalt eines Kreises mit vorgegebenem Radius berechnet. Verwende für den Radius eine Variable wie im Programm oben und lass das Programm untereinander den Flächeninhalt und das Volumen auf drei Stellen genau ausgeben.

16. Die Kreiszahl π lässt sich zwar nicht genau angeben. Es gibt aber eine Reihe von Brüchen und Wurzelausdrücken, um π anzunähern. Einige davon sind:

$$\pi \approx \quad \frac{22}{7}, \quad \frac{355}{113}, \quad \sqrt{2}+\sqrt{3}, \quad \sqrt{7+\sqrt{6+\sqrt{5}}}, \quad \frac{63\left(17+15\sqrt{5}\right)}{25\left(7+15\sqrt{5}\right)}$$

Berechne diese Näherungswerte mit Python und vergleiche sie mit π. Auf wie viele Stellen stimmen die Werte jeweils?

17. Der goldene Schnitt ist ein Verhältnis, das in der Kunst und Architektur gerne verwendet wird. Zeige numerisch, dass der goldene Schnitt $a : b = \frac{\sqrt{5}+1}{2}$ die Eigenschaft $b : a = a : b - 1$ erfüllt.

18. Nach dem Satz des Pythagoras gilt für die drei Seiten a, b und c eines rechtwinkligen Dreiecks $a^2 + b^2 = c^2$. Berechne mit Python für die zwei Katheten $a = 48$ und $b = 55$ die Hypotenuse c.

19.* Schreibe ein Programm, mit dem du Winkel aus dem Gradmass ins Bogenmass umrechnen kannst. (Genauigkeit: 3 Nachkommastellen)

Zur Erinnerung: Das Bogenmass entspricht der Bogenlänge des entsprechenden Sektors auf dem Einheitskreis.

AUFGABEN

20. Verwende $\star=$, um den Wert einer Variablen bei jedem Schritt zu verdoppeln und lass dir damit alle Zweierpotenzen $(2, 4, 8, \ldots)$ bis $2^{10} = 1024$ ausgeben.

21. Lass dir vom Computer die ersten 20 (a) geraden, (b) ungeraden Zahlen ausgeben.

22. Lass dir vom Computer alle natürlichen Zahlen von 1 bis 100 zusammenzählen und bestätige, dass das Resultat 5050 ist.

23. Schreibe ein Programm, das alle natürlichen Zahlen von 1 bis 10 multipliziert und das Resultat $3\,628\,800$ auf den Bildschirm schreibt.

24. Bilde mit dem Computer die Summe der ersten n Stammbrüche:

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \cdots + \frac{1}{n}$$

Probiere aus: Wie gross muss das n sein, damit die Summe grösser ist als 5?

25.* Berechne die Summe der ersten $10\,000$ Stammbrüche einmal von «vorne» und einmal von «hinten» (also $1+\frac{1}{2}+\ldots$ bzw. $\frac{1}{10000}+\frac{1}{9999}+\ldots$). Wie gross ist der Unterschied zwischen den beiden Summen?

10.2.4 Input und Output

- Schreibe ein Programm, das eine Turtle auf der Leinwand platzierst und sie dann um eine bestimmte Anzahl von Grad drehst. Der Grad wird vom Benutzer eingegeben
- Schreibe ein Programm, das eine Turtle auf der Leinwand platziert und sie dann eine bestimmte Länge vorwärts bewegt. Die Anzahl der Schritte wird vom Benutzer eingegeben
- Schreibe ein Programm, das eine Turtle auf der Leinwand platziert und sie dann eine bestimmte Anzahl von Schritten rückwärts bewegt. Die Anzahl der Schritte wird vom Benutzer eingegeben.
- Schreibe ein Programm, das eine Turtle auf der Leinwand platziert und sie dann einen Kreis zeichnen lässt. Die Größe des Kreises und die Farbe werden vom Benutzer eingegeben.
- Schreibe ein Programm, das eine Turtle auf der Leinwand platziert und sie dann eine Spirale zeichnen lässt. Die Größe der Spirale und die Farbe werden vom Benutzer eingegeben

- Schreibe ein Programm, das den Benutzer nach seinem Namen fragt und ihn dann auf der Konsole begrüßt. Verwende anschließend die Turtle-Grafik, um eine einfache Animation mit dem Namen des Benutzers zu erstellen.
- Schreibe ein Programm, das den Benutzer auffordert, eine Zahl einzugeben, und dann den doppelten Wert dieser Zahl auf der Konsole ausgibt. Verwende anschließend die Turtle-Grafik, um eine einfache Animation mit der eingegebenen Zahl zu erstellen.
- Schreibe ein Programm, das den Benutzer nach seinem Lieblingsfarbcode fragt und ihn dann auf der Konsole ausgibt. Verwende anschließend die Turtle-Grafik, um ein Quadrat mit der eingegebenen Farbe zu zeichnen.
- Schreibe ein Programm, das den Benutzer nach der Anzahl der Sterne in einer Reihe fragt und dann eine entsprechende Anzahl von Sternen auf der Konsole ausgibt. Verwende anschließend die Turtle-Grafik, um eine Reihe von Sternen mit der eingegebenen Anzahl zu zeichnen.
- Schreibe ein Programm, das den Benutzer nach einer Zahl von 1 bis 10 fragt und dann einen Countdown von dieser Zahl bis 1 auf der Konsole ausgibt. Verwende anschließend die Turtle-Grafik, um einen Countdown mit der eingegebenen Zahl zu animieren.

10.2.5 Repeat

AUFGABEN

13. Zeichne mit der Turtle die Treppenfigur (a) aus der Abbildung 2.5. Verwende dazu `repeat`.

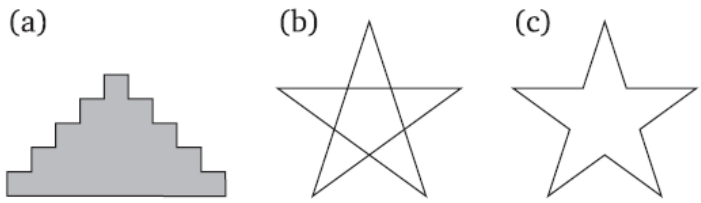

Abbildung 2.5: Treppen und Sterne

14. Zeichne den fünfzackigen Stern aus der Abbildung 2.5(b) oder (c). Dazu musst du zuerst die entsprechenden Winkel berechnen. Verwende wiederum `repeat` für alles, was sich wiederholen lässt.

15. Definiere einen Befehl `achteck(seite)`, um ein regelmässiges und ausgefülltes Achteck zeichnen zu lassen.

10.3 Einheit 3

10.3.1 Wiederholung

10.3.1.1 Code analysieren

Beispiel 1: Quadrate zeichnen

```
from gturtle import *

makeTurtle()

repeat 4:
  forward(100)
  right(90)
```

Beispiel 2: Dynamisches Quadrat mit Variablen

```
from gturtle import *

makeTurtle()

size = 50

repeat 4:
  forward(size)
  right(90)
```

Beispiel 3: Benutzerdefinierte Eingaben für Dreieck

```
from gturtle import *

makeTurtle()

size = input("Gib die Seitenlänge des Dreiecks ein: ")
size = int(size)

repeat 3:
  forward(size)
  right(120)
```

Beispiel 4: Fehlerhaftes Programm (Fehler erkennen)

```
from gturtle import *

makeTurtle()

size = 100
```

```
repeat 5:
  forward(size)
  right(90)
```

Beispiel 4: Sierpinski-Dreieck (Rekursion)

Schreibe ein Programm, das ein **Sierpinski-Dreieck** zeichnet. Dies ist ein rekursives Muster, bei dem in einem großen Dreieck immer kleinere Dreiecke gezeichnet werden.

Anleitung:

- Definiere eine Funktion sierpinski(size, depth), die das Sierpinski-Dreieck rekursiv zeichnet.
- Wenn depth == 0 ist, zeichne einfach ein Dreieck.
- Wenn die Tiefe größer als 0 ist, rufe die Funktion drei Mal auf, um kleinere Dreiecke innerhalb des Hauptdreiecks zu zeichnen.
- Die Größe der kleineren Dreiecke ist jeweils die Hälfte des aktuellen Dreiecks.

```
from gturtle import *

def draw_triangle(size):
  repeat 3:
    forward(size)
    right(120)

def sierpinski(size, depth):
  if depth == 0:
    draw_triangle(size)
  else:
    sierpinski(size / 2, depth - 1)
    forward(size / 2)
    sierpinski(size / 2, depth - 1)
    backward(size / 2)
    right(60)
    forward(size / 2)
    left(60)
    sierpinski(size / 2, depth - 1)
    left(60)
    backward(size / 2)
    right(60)

makeTurtle()
hideTurtle()
sierpinski(200, 4)  # Größe 200 und Rekursionstiefe 4
```

Beispiel 5: Sonnenblumenmuster (Fibonacci-Spirale)

Schreibe ein Programm, das ein **Sonnenblumenmuster** basierend auf der Fibonacci-Folge und einer logarithmischen Spirale zeichnet.

Anleitung:

- Verwende eine Schleife, um nacheinander Kreise an Positionen zu zeichnen, die einer logarithmischen Spirale folgen.
- Die Positionen basieren auf dem **goldenen Winkel** (137,5 Grad), um die typisch spiralförmige Anordnung der Samen einer Sonnenblume zu imitieren.
- Die Größe der Kreise variiert entsprechend der Fibonacci-Folge.

```
from gturtle import *

def fibonacci(n):
  if n == 0:
    return 0
  elif n == 1:
    return 1
  else:
    return fibonacci(n-1) + fibonacci(n-2)

def draw_flower(petals):
  golden_angle = 137.5  # Typisch für die Sonnenblumenanordnung
  for i in range(petals):
    size = fibonacci(i + 1) * 2  # Fibonacci-Größe der Kreise
    forward(i * 3)  # Der Abstand zwischen den Kreisen wächst
    right(golden_angle)
    dot(size)

makeTurtle()
hideTurtle()
draw_flower(20)  # Zeichne 20 Blütenblätter
```

10.3.2 Fehlersuche

Beispiel 1: Einfache Fehlersuche

Das folgende Programm soll ein Quadrat zeichnen, aber es funktioniert nicht richtig. Finde den Fehler.

```
from gturtle import *

makeTurtle()

repeat 5:  # Fehler: Es sollte nur 4 Wiederholungen geben
  forward(100)
  right(90)
```

Beispiel 2: Mittelschwere Fehlersuche

In diesem Programm soll die Turtle ein Dreieck zeichnen, aber der Code führt zu einem Fehler.

```
from gturtle import *

makeTurtle()

size = input("Gib die Seitenlänge des Dreiecks ein: ")

repeat 3:
  forward(size)  # Fehler: 'size' ist ein String, aber wir brauchen eine Zahl
  right(120)
```

Beispiel 3: Verschachtelte Schleifen und Koordinatenfehler

Dieses Programm soll ein **Gitternetz** aus Kreisen zeichnen, aber die Positionen der Kreise sind nicht korrekt ausgerichtet.

```
from gturtle import *

makeTurtle()

size = 50
rows = 5
cols = 5

repeat rows:
  repeat cols:
    dot(size)
    forward(100)
  backward(100 * cols)  # Fehler: Die Turtle wird in eine falsche Position zurückgesetzt
  right(90)
  forward(100)
  left(90)
```

10.3.3 Codevervollständigen

Aufgabe: Vervollständige den Code zum Zeichnen einer Schneeflocke mit Turtle Graphics

Eine Schneeflocke besteht aus mehreren gleichen Linien, die in einem Winkel voneinander abstehen. Deine Aufgabe ist es, den Code zu vervollständigen, um eine Schneeflocke zu zeichnen.

Unvollständiger Code:

```
from gturtle import *

makeTurtle()

def draw_branch(branch_length):
  forward(branch_length)
  backward(branch_length)

def draw_snowflake(arms):
```

```
    for i in range(arms):
        # TODO: Zeichne einen Arm der Schneeflocke
        _____
        right(360 / arms)

# TODO: Vervollständige den Funktionsaufruf für die Schneeflocke
_____

hideTurtle()
```

Anweisungen:

- Die Funktion draw_branch() zeichnet einen einzelnen Arm der Schneeflocke, indem sie eine Linie nach vorne und dann wieder zurück bewegt.
- In der Funktion draw_snowflake() wird die Turtle in einem Kreis bewegt und bei jedem Schritt ein Arm der Schneeflocke gezeichnet.
- Die Anzahl der Arme der Schneeflocke wird durch den Parameter arms bestimmt.
- Du musst den Funktionsaufruf vervollständigen und den Code so ergänzen, dass alle Arme der Schneeflocke gezeichnet werden.
- Denke daran, am Ende des Programms die Schneeflocke mit einer bestimmten Anzahl von Armen zu zeichnen.

Versuche, den fehlenden Code zu vervollständigen, um die Schneeflocke zu zeichnen!

10.3.4 Boolsche Werte – True und False

- Schreibe ein Programm, das prüft, ob eine Zahl gerade oder ungerade ist. Das Programm soll "gerade" ausgeben, wenn die Zahl durch 2 teilbar ist, und "ungerade", wenn nicht.
- Schreibe ein Programm, das den Benutzer nach seinem Alter fragt. Wenn das Alter größer oder gleich 18 ist, soll das Programm "Sie sind volljährig" ausgeben, ansonsten "Sie sind minderjährig".
- Schreibe ein Programm, das prüft, ob eine Zahl positiv oder negativ ist. Das Programm soll "positiv" ausgeben, wenn die Zahl größer als 0 ist, und "negativ", wenn sie kleiner als 0 ist. Wenn die Zahl 0 ist, soll das Programm "neutral" ausgeben.
- Schreibe ein Programm, das den Benutzer nach seinem Geschlecht fragt. Wenn der Benutzer "weiblich" eingibt, soll das Programm "Sie sind weiblich" ausgeben, ansonsten "Sie sind männlich".
- Schreibe ein Programm, das prüft, ob eine Zahl zwischen zwei Grenzwerten liegt. Das Programm soll den Benutzer nach der Zahl und den Grenzwerten fragen und dann ausgeben, ob die Zahl innerhalb des Bereichs liegt oder nicht.

10.3.5 Verknüpfung boolescher Werte

- Schreibe ein Programm, das die Turtle in einem Quadrat umherlaufen lässt, aber nur dann nach oben geht, wenn sie bereits nach rechts gegangen ist.
- Schreibe ein Programm, das die Turtle einen Zickzack-Kurs zeichnen lässt, aber nur dann nach oben geht, wenn sie nach rechts gegangen ist UND sich in einer geraden Spalte befindet.
- Schreibe ein Programm, das die Turtle einen Kreis zeichnen lässt, aber nur dann einen Punkt setzen lässt, wenn die Turtle sich auf der linken Seite des Kreises befindet.
- Schreibe ein Programm, das die Turtle einen Stern zeichnen lässt, aber nur dann einen Punkt setzt, wenn sie in einer ungeraden Ecke des Sterns steht.
- Schreibe ein Programm, das die Turtle ein Muster zeichnen lässt, das aus Quadraten in verschiedenen Farben besteht. Wenn die Turtle sich in einer geraden Spalte befindet, soll das Quadrat blau sein, wenn sie in einer ungeraden Spalte ist, soll das Quadrat grün sein. Wenn die Turtle in einer geraden Zeile ist, soll das Quadrat rot sein, wenn sie in einer ungeraden Zeile ist, soll das Quadrat gelb sein.

10.3.6 Bedingungen

- Schreibe ein Python-Programm, das den Benutzer nach einer Zahl fragt und dann eine Nachricht auf der Konsole ausgibt, die besagt, ob die Zahl gerade oder ungerade ist. Verwende anschließend die Turtle-Grafik, um ein Quadrat mit der Länge der eingegebenen Zahl zu zeichnen, wenn die Zahl gerade ist, oder ein Dreieck mit der Länge der eingegebenen Zahl zu zeichnen, wenn die Zahl ungerade ist.
- Schreibe ein Python-Programm, das den Benutzer nach einem Buchstaben fragt und dann eine Nachricht auf der Konsole ausgibt, die besagt, ob der Buchstabe ein Vokal oder ein Konsonant ist. Verwende anschließend die Turtle-Grafik, um einen Kreis mit dem Radius von 50 Pixeln zu zeichnen, wenn der Buchstabe ein Vokal ist, oder ein Quadrat mit der Länge von 50 Pixeln zu zeichnen, wenn der Buchstabe ein Konsonant ist.
- Schreibe ein Python-Programm, das den Benutzer nach zwei Zahlen fragt und dann eine Nachricht auf der Konsole ausgibt, die besagt, welche Zahl größer ist. Verwende anschließend die Turtle-Grafik, um ein Dreieck mit der Länge der größeren Zahl zu zeichnen.
- Schreibe ein Python-Programm, das den Benutzer nach einer Zahl fragt und dann eine Nachricht auf der Konsole ausgibt, die besagt, ob die Zahl positiv, negativ oder null ist. Verwende anschließend die Turtle-Grafik, um einen Kreis mit dem Radius von 50 Pixeln zu zeichnen, wenn die Zahl positiv ist, oder ein Quadrat mit der Länge von 50 Pixeln zu zeichnen, wenn die Zahl negativ ist, oder ein Dreieck mit der Länge von 50 Pixeln zu zeichnen, wenn die Zahl null ist.
- Schreibe ein Python-Programm, das den Benutzer nach einer Farbe fragt und dann eine Nachricht auf der Konsole ausgibt, die besagt, ob die Farbe warm oder kühl ist. Verwende anschließend die Turtle-Grafik, um ein Quadrat mit der Länge von 100 Pixeln in

der eingegebenen Farbe zu zeichnen, wenn die Farbe warm ist, oder ein Kreis mit dem Radius von 100 Pixeln in der eingegebenen Farbe zu zeichnen, wenn die Farbe kühl ist.

- Schreibe ein Python-Programm, das den Benutzer nach einem Jahr fragt und dann eine Nachricht auf der Konsole ausgibt, die besagt, ob das Jahr ein Schaltjahr ist oder nicht. Verwende anschließend die Turtle-Grafik, um ein Quadrat mit der Länge von 50 Pixeln in der Farbe Rot zu zeichnen, wenn das Jahr ein Schaltjahr ist, oder ein Dreieck mit der Länge von 50 Pixeln in der Farbe Grün zu zeichnen, wenn das Jahr kein Schaltjahr ist
- Schreibe ein Python-Programm, das den Benutzer nach einer Zahl fragt und dann eine Nachricht auf der Konsole ausgibt, die besagt, ob die Zahl eine Primzahl ist oder nicht. Verwende anschließend die Turtle-Grafik, um eine Spirale zu zeichnen, die aus der eingegebenen Zahl besteht. Wenn die Zahl eine Primzahl ist, sollte die Spirale in der Farbe Rot gezeichnet werden, andernfalls sollte sie in der Farbe Blau gezeichnet werden.

AUFGABEN

31. Schreibe ein einfaches Quiz: Dein Programm stellt also eine Frage und prüft dann, ob die Antwort richtig ist. Wenn ja, gibt es «Richtig!» aus, ansonsten «Falsch!».

32.* Du kannst die Osterformel von Gauss auch selbst programmieren. Hier sind die Formeln dafür (natürlich kannst du in Python nicht alles so kompakt schreiben wie hier):

```
k = Jahr // 100,   p = k // 3,   q = k // 4,
M = (15 + k - p - q) % 30,   N = (4 + k - q) % 7
a = Jahr % 19,   b = Jahr % 4,   c = Jahr % 7,
d = (19a + M) % 30,   e = (2b + 4c + 6d + N) % 7
Ostertag = 22 + d + e
```

10.4 Einheit 4

10.4.1 Wiederholung

10.4.1.1 Code analysieren

Beispiel 1: Einfache Bedingung (If-Bedingung)

Dieser Code verwendet eine einfache Bedingung, um zu überprüfen, ob eine eingegebene Zahl positiv ist. Analysiere, wie die Bedingung funktioniert und warum der Output so erfolgt.

```
num = int(input("Gib eine Zahl ein: "))

if num > 0:
  print("Die Zahl ist positiv.")
else:
  print("Die Zahl ist nicht positiv.")
```

Beispiel 2: Mehrere Bedingungen (if-elif-else)

Der folgende Code enthält mehrere Bedingungen und verwendet den elif-Block, um verschiedene Werte zu überprüfen. Analysiere, wie das Programm auf unterschiedliche Eingaben reagiert.

```
age = int(input("Wie alt bist du? "))

if age < 13:
  print("Du bist ein Kind.")
elif age < 18:
  print("Du bist ein Teenager.")
else:
  print("Du bist ein Erwachsener.")
```

Beispiel 3: Komplexe logische Operatoren (AND, OR, NOT)

Dieser Code kombiniert mehrere logische Operatoren, um komplexe Bedingungen zu erstellen. Analysiere, wie die Bedingungen evaluiert werden und was die Ergebnisse bedeuten.

```
weather = input("Ist es sonnig? (ja/nein): ").lower()
temperature = int(input("Wie warm ist es (in Grad Celsius)? "))

if weather == "ja" and temperature > 20:
  print("Es ist sonnig und warm draußen.")
elif weather == "ja" and temperature <= 20:
  print("Es ist sonnig, aber kühl.")
elif weather == "nein" or temperature < 10:
  print("Es ist nicht sonnig oder sehr kalt draußen.")
else:
  print("Das Wetter ist okay.")
```

Beispiel 4: Verschachtelte Bedingungen

Dieser Code verwendet verschachtelte Bedingungen, um eine komplexere Entscheidungsstruktur zu erstellen. Analysiere, wie die verschachtelten Bedingungen den Ablauf des Programms steuern.

```
score = int(input("Gib deine Punktzahl ein (0-100): "))

if score >= 90:
  print("Note: A")
elif score >= 80:
  if score >= 85:
    print("Note: B+")
  else:
    print("Note: B")
elif score >= 70:
  if score >= 75:
    print("Note: C+")
  else:
    print("Note: C")
else:
  print("Note: F")
```

Beispiel 5: Benutzerinteraktion und Validierung

Dieser Code zeigt, wie Benutzerinteraktionen mit booleschen Bedingungen validiert werden können. Analysiere, wie das Programm auf ungültige Eingaben reagiert und was es ausgibt.

```
answer = input("Möchtest du fortfahren? (ja/nein): ").lower()

if answer == "ja":
  print("Du hast fortgefahren.")
elif answer == "nein":
  print("Du hast abgebrochen.")
else:
  print("Ungültige Eingabe. Bitte 'ja' oder 'nein' eingeben.")
```

10.4.1.2 Fehlersuche

Aufgabe 1: Falsche Logik in einer Bedingung (Benutzerberechtigung prüfen)

Der folgende Code soll prüfen, ob ein Benutzer berechtigt ist, auf eine Webseite zuzugreifen, basierend auf seinem Alter und einem Aktivierungscode. Es gibt jedoch einen Fehler in der Logik, der dazu führt, dass das Programm die falschen Ergebnisse liefert.

```
age = int(input("Gib dein Alter ein: "))
activation_code = input("Gib deinen Aktivierungscode ein: ")

if age >= 18 or activation_code == "ABC123":
  print("Zugriff erlaubt.")
else:
```

```
  print("Zugriff verweigert.")
```

Aufgabe 2: Komplexe Logik mit verschachtelten Bedingungen (Zugang zu einer Veranstaltung)

Dieser Code soll entscheiden, ob eine Person zu einer Veranstaltung zugelassen wird. Es gibt jedoch einen Fehler in den verschachtelten Bedingungen, der zu einem falschen Ergebnis führen kann.

```
age = int(input("Gib dein Alter ein: "))
has_ticket = input("Hast du ein Ticket? (ja/nein): ").lower()

if age >= 18:
  if has_ticket == "ja" or has_ticket == "nein":
    print("Zugang zur Veranstaltung erlaubt.")
  else:
    print("Ungültige Eingabe.")
else:
  print("Zugang nicht erlaubt, da du zu jung bist.")
```

Aufgabe 3: Falsche Benutzung des NOT-Operators (Prüfen, ob eine Zahl in einem Bereich liegt)

Der folgende Code soll überprüfen, ob eine eingegebene Zahl zwischen 10 und 20 liegt. Es gibt jedoch einen Fehler in der Verwendung des not-Operators, wodurch der Code nicht korrekt funktioniert.

```
number = int(input("Gib eine Zahl ein: "))

if not (number >= 10 and number <= 20):
  print("Die Zahl liegt zwischen 10 und 20.")
else:
  print("Die Zahl liegt außerhalb des Bereichs.")
```

Aufgabe 4: Falsche Verwendung des and-Operators in einer if-Bedingung (Prüfen, ob eine Person zur Mitgliedschaft berechtigt ist)

Der folgende Code soll prüfen, ob eine Person zur Mitgliedschaft in einem Club berechtigt ist, basierend auf Alter und Wohnort. Es gibt jedoch einen Fehler in der Logik der Bedingung, der dazu führt, dass das Programm die falschen Ergebnisse liefert.

```
age = int(input("Gib dein Alter ein: "))
city = input("In welcher Stadt wohnst du? ").lower()

if age >= 18 and city == "berlin" or "hamburg":
  print("Du bist zur Mitgliedschaft berechtigt.")
else:
  print("Du bist nicht zur Mitgliedschaft berechtigt.")
```

10.4.1.3 Codevervollständigen

Aufgabe 1: Altersprüfung und Ticketpreisberechnung

Du sollst ein Programm schreiben, das das Alter des Benutzers überprüft und den Preis eines Tickets basierend auf dem Alter berechnet. Es gibt drei Kategorien:

- Kinder unter 12 Jahren: 5 Euro
- Jugendliche zwischen 12 und 17 Jahren: 8 Euro
- Erwachsene ab 18 Jahren: 12 Euro

Außerdem erhalten Senioren ab 65 Jahren einen Rabatt von 20% auf den Erwachsenenpreis. Vervollständige den Code.

Unvollständiger Code:

```
age = int(input("Gib dein Alter ein: "))

# TODO: Berechne den Preis basierend auf dem Alter
if age < 12:
  price = _____  # Preis für Kinder
elif age >= 12 and age <= 17:
  price = _____  # Preis für Jugendliche
elif age >= 18 and age < 65:
  price = _____  # Preis für Erwachsene
else:
  # TODO: Berechne den Rabatt für Senioren
  price = _____ * _____

# TODO: Ausgabe des Preises
print("Der Ticketpreis beträgt: ", price, "Euro.")
```

Aufgabe 2: Zugang zu einem System basierend auf mehreren Kriterien

Du sollst ein Programm schreiben, das überprüft, ob ein Benutzer Zugriff auf ein System erhält. Der Benutzer muss die folgenden Bedingungen erfüllen:

- Das Passwort muss korrekt sein.
- Der Benutzer muss entweder ein Administrator sein oder seit mehr als 2 Jahren registriert sein.
- Zusätzlich darf der Benutzer keine Sperrung haben.

Vervollständige den Code, um diese Bedingungen korrekt zu implementieren.

Unvollständiger Code:

```
password = input("Gib dein Passwort ein: ")
is_admin = input("Bist du ein Administrator? (ja/nein): ").lower() == "ja"
years_registered = int(input("Wie viele Jahre bist du registriert? "))
is_banned = input("Bist du gesperrt? (ja/nein): ").lower() == "ja"
```

```
# TODO: Überprüfe, ob der Zugriff erlaubt ist
if password == "geheim" and (_____ or _____) and not is_banned:
  print("Zugriff erlaubt.")
else:
  print("Zugriff verweigert.")
```

10.4.2 Übungen

10.4.2.1 Schleifen

- Erstelle eine for-Schleife, die eine Variable von 1 bis 10 zählt und jedes Mal eine Linie zeichnet, die um 10 Pixel länger ist als die vorherige.
- Erstelle eine for-Schleife, die eine Variable von 1 bis 5 zählt und jedes Mal einen Kreis zeichnet, dessen Durchmesser um 20 Pixel größer ist als der vorherige.
- Erstelle eine for-Schleife, die eine Variable von 10 bis 1 zählt und jedes Mal einen Text auf dem Bildschirm ausgibt, der die aktuelle Variable enthält.

AUFGABEN

1. Schreibe ein Programm, dass alle Quadratzahlen von 1^2 bis 20^2 in eine Liste schreibt und diese Liste am Ende ausgibt.

2. Du weisst bereits, wie du prüfst, ob eine Zahl eine Primzahl ist (vgl. `isPrime`, p. 103). Schreibe damit ein Programm, das alle Primzahlen bis 101 sucht und in eine Liste schreibt.

3.* Ändere das Programm aus der letzten Aufgabe so ab, dass es die ersten hundert Primzahlen in einer Liste sammelt.

4. Mit `randint` aus dem Modul `random` kannst eine Zufallszahl ziehen (vgl. p. 64). Schreibe ein Programm, das drei voneinander verschiedene Zufallszahlen zwischen 1 und 9 zieht und in einer Liste sammelt. Wenn also eine Zahl bereits in der Liste enthalten ist, dann musst du eine neue Zufallszahl ziehen, bis du drei verschiedene Zufallszahlen hast.

5. Erweitere das Programm aus der letzten Aufgabe und schreibe ein Programm, das eine zufällige *Permutation* (d. h. eine Anordnung) der Zahlen von 1 bis 9 erzeugt. Eine solche Permutation könnte sein:

```
[5, 9, 1, 4, 2, 3, 8, 6, 7]
```

AUFGABEN

6. Schreibe analog zur Summenfunktion oben eine Funktion `produkt`, die das Produkt aller Zahlen in einer Liste bildet (also alle Zahlen miteinander multipliziert).

7. Schreibe eine Funktion `last(liste)`, die das letzte Element in der Liste zurückgibt.

8. Schreibe eine Funktion `anzahl`, die die Elemente zählt, die in einer Liste sind. Auch hier verwendest du die gleiche Struktur wie bei der Summenfunktion oben.

9. Den Mittelwert (engl. «*average*») einer Zahlenliste a, b, c berechnest du, indem du alle Zahlen zusammenzählst und dann durch die Anzahl teilst: $M = \frac{a+b+c}{3}$. Schreibe eine Funktion `average`, die den Durschnitt aller Zahlen in einer Liste berechnet. Ergänze dazu die Summenfunktion so, dass sie nicht nur die Summe der Zahlen berechnet, sondern gleichzeitig auch noch zählt, wie viele Zahlen in der Liste sind.

10. Ergänze die Funktion `isPrime` zu einem Programm, das alle Primzahlen sucht, die kleiner sind als 1000. Beginne dafür mit der Liste `primzahlen = [2, 3]`. Das Programm geht dann alle Zahlen bis 999 durch (verwende dazu die klassische **`repeat`**-Schleife). Wenn eine Zahl eine Primzahl ist, wird sie mit `append` zur Liste der Primzahlen hinzugefügt.

AUFGABEN

11. Ergänze das Programm mit dem Häuschen so, dass es das vollständige Haus des Nikolaus zeichnet (siehe p. 11).

12. Ändere das Programm so ab, dass die Streckenlänge in `forward` fest 10 Pixel ist. Dafür kann der Winkel beliebig verändert werden. Lass die Turtle auch dann ein Häuschen zeichnen.

13.* Du kannst das Programm auch so anpassen, dass du sowohl den Winkel als auch die Streckenlänge frei angeben kannst. Dazu kombinierst du beide Werte zu einer einzelnen Zahl: Die «Tausender» geben dann die Streckenlänge an, der Rest den Winkel. Das Häuschen liesse sich dann mit der Liste `[100045, 71090, 71045, 100090, 100090]` zeichnen. Schreibe das Programm dazu.

14. Baue das Pixel-Programm so aus, dass du neben schwarz und weiss noch grau und einige Farben zur Verfügung hast. Dann könnte z. B. 1 für blau, 2 für rot, 3 für grün stehen, usw. Zeichne damit eigene kleine Bilder!

15.* Schreibe ein Programm für Graustufenbilder. Für einen beliebigen Wert `x` zwischen 0.0 und 1.0 (das `.0` ist hier wichtig) erzeugst du das «Grau» dazu über `makeColor(x, x, x)`. Eine Pixelliste könnte dann so aussehen: `[0.0, 0.75, 1.0, 0.4]`

- Schleife durch Liste: Schreibe ein Programm, das eine Liste von Zahlen erstellt und eine For-Schleife verwendet, um die Summe aller Zahlen in der Liste zu berechnen.
- Zufällige Elemente aus Liste auswählen: Schreibe ein Programm, das eine Liste von Namen erstellt und zufällig einen Namen aus der Liste auswählt.
- Neue Liste erzeugen: Schreibe ein Programm, das eine Liste von Zahlen erstellt und eine neue Liste erstellt, die nur die geraden Zahlen aus der ursprünglichen Liste enthält.
- Liste mit Namen: Schreibe ein Programm, das eine Liste mit den Namen deiner Freunde erstellt und die Namen auf dem Bildschirm ausgibt.
- Listen kombinieren: Schreibe ein Programm, das zwei Listen von Namen erstellt und eine neue Liste erstellt, die alle Namen aus beiden Listen enthält.
- Elemente in Liste sortieren: Schreibe ein Programm, das eine Liste von Zahlen erstellt und die Liste in aufsteigender Reihenfolge sortiert.
- Liste von Zahlen sortieren: Schreibe ein Programm, das eine Liste von Zahlen erstellt und die Liste in aufsteigender Reihenfolge sortiert.
- Liste von Wörtern rückwärts ausgeben: Schreibe ein Programm, das eine Liste von Wörtern erstellt und die Wörter rückwärts auf dem Bildschirm ausgibt.
- Werte in Liste ändern: Schreibe ein Programm, das eine Liste von Zahlen erstellt und die Werte in der Liste ändert, indem es eine For-Schleife verwendet.

- Liste von Werten filtern: Schreibe ein Programm, das eine Liste von Zahlen erstellt und nur die Werte in der Liste ausgibt, die größer als 10 sind.
- Zeichne ein einfaches Gesicht mit gturtle und Listen: Schreibe ein Programm, das eine Liste von Linien erstellt und eine For-Schleife verwendet, um das Gesicht auf dem Bildschirm zu zeichnen. Verwende die drawLine-Funktion, um die Linien zu zeichnen. Verwende die Farben "schwarz" und "gelb" für die Linien.
- Zeichne ein Haus mit gturtle und Listen: Schreibe ein Programm, das eine Liste von Linien erstellt und eine For-Schleife verwendet, um das Haus auf dem Bildschirm zu zeichnen. Verwende die drawLine-Funktion, um die Linien zu zeichnen. Verwende die Farben "schwarz" und "rot" für die Linien und fülle das Haus mit der Farbe "grün".
- Minimum und Maximum: Schreibe ein Programm, das eine Liste von Zahlen erstellt und das Minimum und Maximum der Liste berechnet.
- Erstelle eine Liste 1 bis 100 mit range
- Gegeben ist die folgende Liste von Farben: colors = ["red", "green", "blue", "yellow", "orange", "purple", "pink", "green"]
  Schreiben Sie ein Programm, das die Anzahl der Vorkommen von "green" in der Liste zählt und auf der Konsole ausgibt. Verwenden Sie dazu die count-Methode der Liste. Zeichnen Sie außerdem eine Linie für jede Farbe in der Liste mit der Turtle-Instanz von gturtle. Die Linien sollten jeweils 50 Pixel lang sein und die Farbe der entsprechenden Farbe in der Liste entsprechen.
  Ihr Programm sollte folgende Ausgabe erzeugen: Anzahl der grünen Farben: 2
- Gegeben sind die folgenden Listen von Punkten:
  square = [[-100, 100], [-100, -100], [100, -100], [100, 100], [-100, 100]]
  triangle = [[-50, -50], [50, -50], [0, 50], [-50, -50]]
  pentagon = [[-50, 50], [-100, 0], [-50, -50], [50, -50], [100, 0], [50, 50], [-50, 50]]
  shapes = [square, triangle, pentagon]
  Schreiben Sie ein Programm, das jede Form in der Liste von Formen zeichnet. Jede Form ist eine Liste von Punkten, die die Eckpunkte der Form darstellen. Jeder Punkt besteht aus zwei Zahlen, die die x- und y-Koordinaten des Punktes darstellen.

10.4.2.2 Schleifen (und Listen)

- Wiederhole eine einfache Form: Schreibe ein Programm, das eine einfache Form, wie zum Beispiel ein Quadrat oder ein Dreieck, mehrmals wiederholt. Verwende eine Schleife, um die Form automatisch mehrmals zu zeichnen, anstatt den Code mehrmals manuell zu wiederholen.
- Verwende eine for-Schleife: Schreibe ein Programm, das eine for-Schleife verwendet, um eine bestimmte Anzahl von Formen oder Schritten auszuführen. Verwende zum Beispiel eine for-Schleife, um ein Muster aus sich wiederholenden Formen zu erstellen.
- Verwende eine Schleife zum Animieren: Schreibe ein Programm, das eine Schleife verwendet, um eine Animation zu erstellen, indem du die Turtle in einer Schleife bewegst, um eine Bewegung zu simulieren, wie zum Beispiel das Bewegen der Turtle in einer Schleife, um eine sich drehende Spirale zu erstellen.
- Verwende eine verschachtelte Schleife: Schreibe ein Programm, das eine verschachtelte Schleife verwendet, um komplexe Formen oder Muster zu erstellen. Verwende zum

Beispiel eine verschachtelte Schleife, um ein Muster aus sich wiederholenden Formen in verschiedenen Farben zu erstellen.

10.4.3 Mehrdimensionale Listen

Übung 1: 2D-Array – Matrix-Summation

Aufgabe:

Schreibe ein Programm, das eine 3x3-Matrix erstellt und die Summe der Elemente in jeder Zeile, Spalte und der gesamten Matrix berechnet. Verwende ein **2D-Array** (mehrdimensionales Array), um die Matrix zu speichern.

Schritte:

9. Definiere ein 2D-Array mit festen Werten (eine 3x3-Matrix).
10. Berechne die Summe jeder Zeile und gib die Ergebnisse aus.
11. Berechne die Summe jeder Spalte und gib die Ergebnisse aus.
12. Berechne die Gesamtsumme aller Elemente der Matrix.

Übung 2: Zugriff auf Elemente eines 3D-Arrays

Aufgabe:

Schreibe ein Programm, das ein **3D-Array** darstellt. Jedes Element des Arrays soll eine Ebene von 2D-Matrizen darstellen. Das Programm soll den Zugriff auf bestimmte Elemente des 3D-Arrays demonstrieren und die Elemente einer ausgewählten Ebene (2D-Matrix) ausgeben.

Schritte:

13. Definiere ein 3D-Array, das aus drei 2D-Matrizen (2x2-Matrizen) besteht.
14. Zeige an, wie man auf ein bestimmtes Element innerhalb der 3D-Struktur zugreifen kann.
15. Gib alle Elemente einer bestimmten Ebene des 3D-Arrays aus (zum Beispiel die erste Ebene).

10.4.4 Dictionaries

Beispiel 1: Einfache Telefonbuch-Verwaltung

Aufgabe:

Erstelle ein einfaches Telefonbuch, in dem Namen als Schlüssel und Telefonnummern als Werte gespeichert sind. Das Programm soll es ermöglichen, nach Telefonnummern zu suchen und neue Einträge hinzuzufügen.

Beispiel 2: Bewertungssystem für ein Restaurant

Aufgabe:

Erstelle ein Bewertungssystem für ein Restaurant, in dem Gäste verschiedene Speisen bewerten können. Speisen werden als Schlüssel und die Bewertungen als Werte gespeichert. Das Programm soll die Durchschnittsbewertung für jede Speise berechnen.

Beispiel 3: Bestandsverwaltung für ein Geschäft

Aufgabe:

Erstelle ein Programm, das den Bestand eines kleinen Geschäfts verwaltet. Verwende ein Dictionary, bei dem Produkte die Schlüssel und deren Mengen die Werte sind. Ermögliche es, den Bestand eines Produkts zu aktualisieren oder ein neues Produkt hinzuzufügen.

10.5 Einheit 5

10.5.1 Wiederholung

10.5.1.1 Code analysieren

Beispiel 1:

```
x = 10
y = 15
z = 20
result = (x + y + z) / 3
```

Beispiel 2:

```
from gturtle import *

t = makeTurtle()

for i in range(5):
    x = i * 10
    t.goto(x, 0)
    t.write(str(i))

```

Beispiel 3:

```
from gturtle import *

t = makeTurtle()
x = 0

def draw_square():
    global x
    for i in range(4):
        turtle.forward(50)
        turtle.right(90)
    x += 1

draw_square()
turtle.up()
turtle.goto(0, -50)
turtle.down()
draw_square()

turtle.up()
turtle.goto(0, -100)
turtle.down()
draw_square()

turtle.up()
turtle.goto(0, -150)
turtle.down()
turtle.write("Squares drawn: " + str(x))
```

Beispiel 4:

```
from gturtle import *

t = makeTurtle()

for i in range(4):
    for j in range(4):
        t.forward(50)
        t.right(90)
    t.right(10)

t.hide()
```

Beispiel 5:

```
import turtle

for i in range(10):
    if i % 2 == 0 and i != 0:
        turtle.right(90)
    else:
        turtle.left(90)
    turtle.forward(50)
```

- Was tut der Code?
- Wie oft wird die Schleife durchlaufen?
- Was sind die Werte von i während der Schleife?
- Welchen Effekt hat die Bedingung i % 2 == 0 and i != 0?

Beispiel 6:

```
import turtle

while True:
    color = input("Gib eine Farbe ein: ")
    if color == "blau":
        turtle.color("blue")
        turtle.forward(100)
    elif color == "rot":
        turtle.color("red")
        turtle.forward(100)
    elif color == "grün":
        turtle.color("green")
        turtle.forward(100)
    elif color == "gelb":
        turtle.color("yellow")
        turtle.forward(100)
    else:
        print("Ungültige Farbe.")
```

- Was tut der Code?
- Was passiert, wenn der Benutzer eine ungültige Farbe eingibt?
- Wie oft wird die Schleife durchlaufen?

- Wie können wir die Schleife beenden?

10.5.1.2 Fehlersuche

Suche den Fehler

Beispiel 1:

```
1  from gturtle import *
2
3  t = makeTurtle()
4  for i in range(5):
5      for j in range(5):
6          t.forward(50)
7          t.right(90)
8      t.right(20)
9
10 t.hide()
```

Beispiel 2:

```
1  from gt import *
2
3  t = makeTurtle()
4
5  x = 0
6
7  def draw_circle():
8      for i in range(10):
9          t.forward(10)
10         t.right(36)
11     x += 1
12
13 draw_circle()
14 t.up()
15 t.goto(0, -50)
16 t.down()
17 draw_circle()
18
19 t.up()
20 t.goto(0, -100)
21 t.down()
22 draw_circle()
23
24 t.up()
25 t.goto(0, -150)
26 t.down()
27 print "Circles drawn: " + x
28
```

Beispiel 3:

```python
import turtle

t = turtle.Turtle()

for i in range(10):
    if i % 2 == 0 and i > 5:
        t.forward(50)
    else:
        t.right(90)
```

10.5.1.3 Ergänze oder verbessere den Code

Beispiel 1:

```python
from gt import *

t = maket()

# Schritt 1: Definiere eine t
t = gt.t()

# Schritt 2: Definiere eine Funktion, die ein Quadrat zeichnet
def draw_square():
    for i in range(4):
        t.forward(50)
        t.right(90)

# Schritt 3: Schleife, um das Quadrat 3 Mal zu zeichnen
for i in range(3):
    draw_square()
    t.right(30)

# Schritt 4: Verstecke die t
t.hide()
```

10.5.1.4 Wiederholendes Beispiel

Gegeben ist eine Liste mit Zahlen:

zahlen = [4, 7, 2, 9, 1, 5, 8, 3, 6]

- Schreibe eine Funktion, die die Summe aller Zahlen in der Liste berechnet und ausgibt.
- Schreibe eine Funktion, die das Minimum und Maximum der Zahlen in der Liste berechnet und ausgibt.
- Schreibe eine Funktion, die die Anzahl der geraden Zahlen in der Liste berechnet und ausgibt.
- Schreibe eine Funktion, die die Anzahl der Zahlen in der Liste berechnet, die größer als ihr Index in der Liste sind und ausgibt.
- Schreibe eine Funktion, die die Liste in umgekehrter Reihenfolge ausgibt.
- Schreibe eine Funktion, die die Liste in aufsteigender Reihenfolge sortiert und ausgibt.

10.5.2 Übungen

Übung 1: Berechnung von Flächen verschiedener geometrischer Formen (ohne Funktionen)

Schreibe ein Programm, das die Flächen eines Kreises, eines Rechtecks und eines Dreiecks berechnet. Dabei sollst du die Berechnungen für jede Form manuell durchführen. Beachte, dass der Code in dieser Übung absichtlich ohne Funktionen erstellt wird.

Schritte:

- Berechne die Fläche eines Kreises, nachdem der Benutzer den Radius eingegeben hat.
- Berechne die Fläche eines Rechtecks, nachdem der Benutzer Länge und Breite eingegeben hat.
- Berechne die Fläche eines Dreiecks, nachdem der Benutzer die Basislänge und die Höhe eingegeben hat.
- Gib die berechneten Flächen jeweils aus.

Reflexion:

Überlege nach der Übung, wie der Code durch den Einsatz von Funktionen vereinfacht werden könnte. Wie würde der Einsatz von Funktionen die Struktur des Programms verbessern?

Übung 2: Wiederholte Berechnung von Rabatten für verschiedene Produkte (ohne Funktionen)

Schreibe ein Programm, das den Endpreis von drei Produkten nach Abzug eines Rabatts berechnet. Der Benutzer gibt den ursprünglichen Preis und den Rabatt für jedes Produkt ein. Auch hier sollst du die Berechnung für jedes Produkt manuell wiederholen.

Schritte:

- Berechne den Endpreis für das erste Produkt basierend auf dem Preis und dem Rabatt in Prozent.
- Wiederhole die gleiche Berechnung für das zweite und dritte Produkt.
- Gib für jedes Produkt den Endpreis aus.

Reflexion:

Überlege nach der Übung, wie der Einsatz von Funktionen dir helfen könnte, den Code zu vereinfachen und die Berechnung für jedes Produkt effizienter zu gestalten. Wie könnte eine Funktion die Berechnung vereinfachen und Fehler vermeiden helfen?

Übung 1: Unterschied zwischen globalen und lokalen Variablen

Aufgabe:

Schreibe ein Programm, das den Unterschied zwischen einer globalen und einer lokalen Variable zeigt. Analysiere, wie sich der Gültigkeitsbereich der Variablen auf den Programmablauf auswirkt.

Schritte:

- Definiere eine Variable außerhalb einer Funktion (globale Variable).
- Definiere eine gleichnamige Variable innerhalb der Funktion (lokale Variable).
- Gib den Wert der Variablen sowohl innerhalb als auch außerhalb der Funktion aus und analysiere die Unterschiede.

Übung 2: Verwendung globaler Variablen innerhalb von Funktionen

Aufgabe:

Schreibe ein Programm, das eine globale Variable innerhalb einer Funktion verändert. Verwende das Schlüsselwort global, um sicherzustellen, dass die Änderung der Variablen auch außerhalb der Funktion sichtbar ist.

Schritte:

- Definiere eine globale Variable außerhalb der Funktion.
- Schreibe eine Funktion, die versucht, die globale Variable zu ändern.
- Teste, was passiert, wenn du die globale Variable ohne das Schlüsselwort global innerhalb der Funktion änderst.
- Verwende anschließend das Schlüsselwort global, um die Variable korrekt zu ändern.

Übung 3: Zugriff auf lokale und globale Variablen in einem Programm

Aufgabe:

Schreibe ein Programm, das sowohl lokale als auch globale Variablen verwendet. Das Programm soll eine mathematische Berechnung durchführen und dabei sowohl eine globale als auch eine lokale Variable berücksichtigen. Analysiere, wie die beiden Variablen den Programmablauf beeinflussen.

Schritte:

- Definiere eine globale Variable, die einen Startwert enthält.
- Schreibe eine Funktion, die eine lokale Variable verwendet, um den Wert zu verändern.
- Gib sowohl die globale als auch die lokale Variable nach den Berechnungen aus.

Übung 4: Verschachtelte Funktionen und Gültigkeitsbereich

Aufgabe:

Schreibe ein Programm, das eine Funktion enthält, die eine weitere Funktion innerhalb von sich aufruft. Analysiere, wie Variablen in den verschiedenen Ebenen der Funktionen verwendet werden und wie der Gültigkeitsbereich die Werte beeinflusst.

Schritte:

- Definiere eine globale Variable.
- Schreibe eine äußere Funktion, die eine lokale Variable verwendet.
- Definiere eine innere Funktion, die auf die globale Variable und die lokale Variable der äußeren Funktion zugreift.

- Analysiere die Ausgabe, nachdem die innere Funktion ausgeführt wurde.

10.5.3 Abbruch von Schleifen

- Schreibe ein Python-Programm, das eine Schleife verwendet, um eine Turtle auf der Leinwand zu bewegen und sie in jeder Iteration um einen festen Winkel zu drehen. Füge eine Bedingung hinzu, die die Schleife abbricht, wenn die Turtle in der Nähe des Ursprungs endet. Verwende anschließend die Turtle-Grafik, um einen Stern mit der Anzahl der Schleifendurchläufe zu zeichnen.
- Schreibe ein Python-Programm, das eine Schleife verwendet, um eine Turtle auf der Leinwand zu bewegen und sie in jeder Iteration eine zufällige Entfernung und einen zufälligen Winkel vorwärts bewegen lässt. Füge eine Bedingung hinzu, die die Schleife abbricht, wenn die Turtle den Bildschirmrand erreicht. Verwende anschließend die Turtle-Grafik, um ein Quadrat mit der Seitenlänge der Anzahl der Schleifendurchläufe zu zeichnen.
- Schreibe ein Python-Programm, das eine Schleife verwendet, um eine Turtle auf der Leinwand zu bewegen und sie in jeder Iteration eine feste Anzahl von Schritten vorwärts bewegen lässt. Füge eine Bedingung hinzu, die die Schleife abbricht, wenn die Turtle in der Nähe eines bestimmten Punktes auf der Leinwand landet. Verwende anschließend die Turtle-Grafik, um einen Kreis mit dem Radius der Anzahl der Schleifendurchläufe zu zeichnen.

10.6 Einheit 6

10.6.1 Wiederholende Beispiele

10.6.1.1 Code-Analyse Übungen

Übung 1: Funktionen und globale/lokale Variablen

Aufgabe:

Analysiere den folgenden Code und bestimme die Ausgabe des Programms. Achte darauf, wie sich der Wert von x innerhalb und außerhalb der Funktion unterscheidet. Erkläre, warum der Wert unterschiedlich ist.

```
x = 10

def meine_funktion():
  x = 5
  print("Wert innerhalb der Funktion:", x)

meine_funktion()
print("Wert außerhalb der Funktion:", x)
```

Übung 2: Schleifen und „break“

Aufgabe:

Der folgende Code verwendet eine Schleife und das „break“-Statement. Bestimme, wie viele Zahlen ausgegeben werden, bevor die Schleife abbricht. Erkläre den Einfluss von „break“ auf die Schleife.

```
def drucke_zahlen():
  for i in range(1, 10):
    if i == 5:
      break
    print(i)

drucke_zahlen()
```

Übung 3: Verschachtelte Funktionen

Aufgabe:

Hier wird eine Funktion innerhalb einer anderen Funktion aufgerufen. Analysiere den Code und erkläre, wie die Berechnung funktioniert. Was ist der Rückgabewert des Programms und wie wird er berechnet?

```
def multipliziere(a, b):
  def quadriere(x):
    return x * x
```

```
  return quadriere(a) * b

resultat = multipliziere(3, 2)
print("Resultat:", resultat)
```

Übung 4: Rekursion und Variablenbereich

Aufgabe:

Der folgende Code verwendet Rekursion, um die Fakultät einer Zahl zu berechnen. Analysiere den Code und erkläre, wie die Variable n in der Funktion „fakultaet“ funktioniert. Was passiert, wenn n gleich 0 ist?

```
def fakultaet(n):
  if n == 0:
    return 1
  else:
    return n * fakultaet(n - 1)

print(fakultaet(5))
```

Übung 5: „continue“ in Schleifen

Aufgabe:

Das „continue“-Statement überspringt bestimmte Schleifendurchläufe. Analysiere den folgenden Code und bestimme, welche Zahlen ausgegeben werden. Erkläre, was „continue“ bewirkt.

```
def drucke_ungerade():
  for i in range(10):
    if i % 2 == 0:
      continue
    print(i)

drucke_ungerade()
```

10.6.1.2 Fehlersuche Übungen

Übung 1: Fehler in der Variablenverwendung

Aufgabe:

Der folgende Code soll die Summe der Zahlen von 0 bis 4 berechnen, aber es tritt ein Fehler auf. Finde und behebe den Fehler. Achte dabei besonders auf den Gültigkeitsbereich der Variablen.

```
def addiere():
  summe = 0
  for i in range(5):
    summe += i
  print("Gesamtsumme:", summe)

print("Gesamtsumme nach Funktion:", summe)
```

Übung 2: Fehler in einer Funktion mit Parametern

Aufgabe:

Der Code soll den Durchschnitt von drei Zahlen berechnen, gibt jedoch nicht den richtigen Wert zurück. Finde den Fehler im Code und behebe ihn.

```
def berechne_durchschnitt(a, b, c):
  durchschnitt = a + b + c / 3
  return durchschnitt

print(berechne_durchschnitt(4, 5, 6))
```

Übung 3: Fehler bei der Verwendung von „break“

Aufgabe:

Der folgende Code soll die Schleife beenden, wenn der Wert 3 erreicht wird, aber das Programm funktioniert nicht wie erwartet. Finde und behebe den Fehler.

```
def suche_zahl():
  for i in range(1, 6):
    if i == 3:
      break
    print(i)

suche_zahl()
```

Übung 4: Funktion ohne Rückgabewert

Aufgabe:

Die Funktion „multipliziere“ soll zwei Zahlen multiplizieren und das Ergebnis zurückgeben. Der Code funktioniert jedoch nicht wie erwartet. Finde den Fehler und korrigiere ihn.

```
def multipliziere(a, b):
  resultat = a * b
```

```
print(multipliziere(3, 4))
```

Übung 5: Fehler in der Verwendung von „continue“

Aufgabe:

Der folgende Code soll alle Zahlen außer der Zahl 5 ausgeben. Stattdessen wird jedoch keine Ausgabe erzeugt. Finde den Fehler im Code und korrigiere ihn.

```
def filter_zahlen():
  for i in range(1, 10):
    if i == 5:
      continue
      print(i)

filter_zahlen()
```

10.6.1.3 Code-Vervollständigung Übungen

Übung 1: Quadrat einer Zahl berechnen

Aufgabe:

Vervollständige die Funktion „quadriere“, die das Quadrat einer gegebenen Zahl berechnet und zurückgibt. Ergänze die fehlenden Teile des Codes.

```
def quadriere(x):
  _____
  return _____

resultat = quadriere(5)
print("Quadrat von 5:", resultat)
```

Übung 2: Lokale und globale Variablen

Aufgabe:

Vervollständige den Code, um die Funktionsweise von lokalen und globalen Variablen zu demonstrieren. Die Funktion soll eine lokale Variable verwenden, die den globalen Wert nicht verändert.

```
x = 10

def meine_funktion():
  _____
  print("Wert innerhalb der Funktion:", x)
```

```
meine_funktion()
print("Wert außerhalb der Funktion:", x)
```

Übung 3: Funktionen mit Schleifen und „break“

Aufgabe:

Vervollständige die Funktion „finde_zahl“, die eine Zahl von 1 bis 10 sucht und die Schleife beendet, wenn die Zahl 7 erreicht wird.

```
def finde_zahl():
  for i in range(1, 11):
    if _____:
      break
    print(_____)

finde_zahl()
```

Übung 4: Rekursionsfunktion vervollständigen

Aufgabe:

Vervollständige die rekursive Funktion „summe_bis“, die die Summe aller Zahlen von 1 bis „n“ berechnet.

```
def summe_bis(n):
  if n == 1:
    return _____
  else:
    return _____ + summe_bis(_____)

print(summe_bis(5))
```

Übung 5: Funktion zur Primzahlprüfung vervollständigen

Aufgabe:

Vervollständige den Code, um eine Funktion zu schreiben, die prüft, ob eine Zahl eine Primzahl ist. Ergänze die fehlenden Teile.

```
def ist_primzahl(n):
  if n < 2:
    return _____
  for i in range(2, _____):
    if n % i == 0:
      return _____
  return True
```

```
print(ist_primzahl(7))
```

10.6.1.4 Übungen zum Debugging

Fehlerhafte Berechnung in einer verschachtelten Schleife

Aufgabe:

Das folgende Programm soll die Summe der Quadrate aller Zahlen von 1 bis 10 berechnen. Leider ist der Code fehlerhaft und liefert ein falsches Ergebnis. Finde und behebe den Fehler, sodass die richtige Summe der Quadrate ausgegeben wird.

```
summe = 0
for i in range(1, 11):
  for j in range(1, 11):
    summe += i * j
print("Summe der Quadrate:", summe)
```

Fehler in der Sortierung einer Liste

Aufgabe:

Das Programm soll eine Liste von Namen nach ihrer Länge sortieren, aber es gibt einen Fehler in der Sortierfunktion. Finde und behebe den Fehler, damit die Liste korrekt sortiert wird.

```
namen = ["Anna", "Sebastian", "Max", "Elena"]
sortierte_namen = sorted(namen, key=len())
print("Nach Länge sortierte Namen:", sortierte_namen)
```

Falsche Zuweisung in einer Schleife

Aufgabe:

Das Programm soll das Produkt der ungeraden Zahlen von 1 bis 10 berechnen. Es berechnet jedoch nicht das korrekte Ergebnis. Finde den Fehler in der Schleife und behebe ihn, damit das Produkt der ungeraden Zahlen korrekt ausgegeben wird.

```
produkt = 1
for i in range(1, 11):
  if i % 2 == 1:
    produkt = i
print("Produkt der ungeraden Zahlen:", produkt)
```

Fehler in der Listenverarbeitung

Aufgabe:

Das Programm soll alle positiven Zahlen aus der Liste "zahlen" auswählen. Es gibt jedoch auch negative Zahlen aus. Finde den Fehler und korrigiere den Code, sodass nur positive Zahlen ausgegeben werden.

```
zahlen = [1, -2, 3, -4, 5]
positive_zahlen = []
for zahl in zahlen:
  if zahl > 0:
    positive_zahlen.append(zahl)
  else:
    positive_zahlen.append(-zahl)
print("Positive Zahlen:", positive_zahlen)
```

Logikfehler in der Primzahlprüfung

Aufgabe:

Das folgende Programm soll überprüfen, ob eine Zahl eine Primzahl ist. Es gibt jedoch auch Nicht-Primzahlen als Primzahlen aus. Finde und behebe den Logikfehler im Code.

```
def ist_primzahl(n):
  if n < 2:
    return False
  for i in range(2, n):
    if n % i == 0:
      return True
  return False

zahl = int(input("Geben Sie eine Zahl ein: "))
if ist_primzahl(zahl):
  print(zahl, "ist eine Primzahl.")
else:
  print(zahl, "ist keine Primzahl.")
```

10.6.2 Übungen zur Code-Vervollständigung

Durchschnittswerte in Listen berechnen

Aufgabe:

Vervollständige den Code, um den Durchschnittswert jeder Unterliste in einer Liste von Listen zu berechnen und speichere diese Durchschnittswerte in einer neuen Liste.

```
def berechne_durchschnitt(liste_von_listen):
  durchschnittswerte = []
  for unterliste in liste_von_listen:
    summe = 0
    for zahl in unterliste:
```

```
        summe += _______
    durchschnittswerte.append(_______)
  return durchschnittswerte
```

Vokale aus einem Text entfernen

Aufgabe:

Vervollständige den Code, um alle Vokale aus einer Zeichenkette zu entfernen und die neue Zeichenkette zurückzugeben.

```
def entferne_vokale(text):
  vokale = "aeiouAEIOU"
  neuer_text = ""
  for buchstabe in text:
    if ________:
      neuer_text += ________
  return neuer_text
```

Multiplikationstabelle erstellen

Aufgabe:

Ergänze den Code, um eine Multiplikationstabelle von 1 bis 10 zu erstellen. Speichere das Ergebnis als verschachtelte Liste.

```
def multiplikationstabelle():
  tabelle = []
  for i in range(1, 11):
    reihe = []
    for j in range(1, 11):
      ________
    tabelle.append(reihe)
  return tabelle
```

Alle Elemente in einer Liste um 1 erhöhen

Aufgabe:

Vervollständige den Code, um jedes Element einer Liste von Zahlen um 1 zu erhöhen und die neue Liste zurückzugeben.

```
def erhoehe_alle(liste):
  neue_liste = []
  for zahl in liste:
    neue_liste.append(________)
```

```
  return neue_liste
```

Zeilen einer Datei lesen und speichern

Aufgabe:

Ergänze den Code, um jede Zeile einer Textdatei zu lesen und in einer Liste zu speichern.

```
def lese_datei(dateiname):
  zeilen = []
  with open(dateiname, "r") as datei:
    for zeile in datei:
      zeilen.append(_____)
  return zeilen
```

10.6.3 Übungen zur Strukturierung von Codesegmenten

1. Zahlen von 1 bis 100 drucken, durch 3 und 5 teilbare Zahlen ersetzen

Aufgabe:

Ordne die folgenden Codezeilen so, dass das Programm die Zahlen von 1 bis 100 ausgibt. Ersetze dabei Zahlen, die durch 3 teilbar sind, durch „Fizz“ und durch 5 teilbare Zahlen durch „Buzz“. Zahlen, die durch 3 und 5 teilbar sind, sollen durch „FizzBuzz“ ersetzt werden.

```
print("FizzBuzz")
if i % 3 == 0 and i % 5 == 0:
if i % 3 == 0:
print(i)
for i in range(1, 101):
elif i % 5 == 0:
print("Fizz")
print("Buzz")
```

2. Liste sortieren und Duplikate entfernen

Aufgabe:

Ordne die folgenden Codezeilen so, dass das Programm eine Liste von Zahlen sortiert und die Duplikate entfernt.

```
def entferne_duplikate(liste):
return list(einzigartig)
liste.sort()
einzigartig = set(liste)
```

3. Primzahlen in einem Bereich finden

Aufgabe:

Ordne die folgenden Codezeilen so, dass das Programm alle Primzahlen zwischen zwei eingegebenen Werten findet.

if zahl % i == 0:
for i in range(2, zahl):
def ist_primzahl(zahl):
return False
return True
for zahl in range(start, ende + 1):

4. Zahlen in einer Liste quadrieren

Aufgabe:

Ordne die folgenden Zeilen so, dass das Programm alle Zahlen in einer Liste quadriert und die neue Liste zurückgibt.

for zahl in liste:
return neue_liste
neue_liste.append(zahl ** 2)
neue_liste = []

5. Wörter nach Häufigkeit zählen

Aufgabe:

Ordne die Codezeilen so, dass die Häufigkeit der Wörter in einem Text gezählt und als Wörterbuch ausgegeben wird.

woerterbuch[w] += 1
if w in woerterbuch:
woerter = text.split()
def woerter_zaehlen(text):
for w in woerter:
else:
woerterbuch[w] = 1
woerterbuch = {}

10.7 Einheit 7

10.7.1 Code-Analyse:

Aufgabe:

Der folgende Code verwendet verschachtelte Schleifen, Bedingungen und eine Funktion. Analysiere den Code und beantworte die dazugehörigen Fragen.

```
def finde_primzahlen(limit):
  primzahlen = []
  for num in range(2, limit + 1):
    is_prime = True
    for i in range(2, num):
      if num % i == 0:
        is_prime = False
        break
    if is_prime:
      primzahlen.append(num)
  return primzahlen

grenze = 20
print("Primzahlen bis", grenze, "sind:", finde_primzahlen(grenze))
```

Fragen zur Code-Analyse:

1. Was macht die Funktion finde_primzahlen(limit) und was gibt sie zurück?
2. Wie arbeiten die beiden Schleifen zusammen, um Primzahlen zu finden? Erkläre den Ablauf der Schleifen.
3. Was passiert, wenn limit = 10 gesetzt wird? Welche Primzahlen werden gefunden?
4. Erkläre, wie die Bedingung if num % i == 0 dazu verwendet wird, um zu entscheiden, ob eine Zahl eine Primzahl ist.
5. Warum ist es notwendig, die Variable is_prime in jeder Iteration neu auf True zu setzen?

10.7.2 Code-Vervollständigung:

Aufgabe:

Vervollständige den folgenden Code, um die Funktion finde_durchschnitt zu implementieren. Die Funktion soll den Durchschnittswert einer Liste berechnen, die auch negative Zahlen enthalten kann. Die Funktion soll nur positive Zahlen berücksichtigen. Verwende Schleifen und Bedingungen, um die Lösung zu vervollständigen.

```
def finde_durchschnitt(liste):
  summe = 0
  zaehler = 0
  for zahl in liste:
```

```
    if _________:
      summe += _________
      zaehler += 1
  if zaehler > 0:
    return summe / zaehler
  else:
    return "Keine positiven Zahlen gefunden"

zahlen = [4, -1, 7, 3, -8, 2]
print("Durchschnitt der positiven Zahlen:", finde_durchschnitt(zahlen))
```

Fragen zur Code-Vervollständigung:

6. Vervollständige den Code an den markierten Stellen (_____).
7. Was passiert, wenn die Liste nur negative Zahlen enthält? Welche Ausgabe wird dann erzeugt?
8. Wie kann der Code modifiziert werden, damit die Funktion sowohl positive als auch negative Zahlen berücksichtigt, aber immer noch den Durchschnitt nur der positiven Zahlen zurückgibt?
9. Welche Änderungen würdest du vornehmen, wenn du auch Nullwerte aus der Liste ausschließen möchtest?

10.7.3 Fehlersuche:

Aufgabe:

Der folgende Code soll eine Liste von Zahlen prüfen und nur die geraden Zahlen in einer neuen Liste speichern. Es gibt jedoch Fehler, die dazu führen, dass das Programm nicht wie erwartet funktioniert. Finde und behebe die Fehler.

```
def finde_gerade_zahlen(liste):
  gerade_zahlen = []
  for i in range(1, len(liste)):
    if liste[i] % 2 == 1:
      gerade_zahlen.append(liste[i])
  return gerade_zahlen

zahlen = [5, 12, 9, 8, 13, 6]
gerade_zahlen = finde_gerade_zahlen(zahlen)
print("Gerade Zahlen:", gerade_zahlen)
```

Fragen zur Fehlersuche:

10. Welcher Fehler liegt in der Schleifenbedingung vor? Erkläre, warum der Code nicht korrekt alle Zahlen überprüft.
11. Der if-Bedingungscode prüft nach ungeraden Zahlen anstelle von geraden Zahlen. Wie sollte die Bedingung geändert werden, um das Problem zu beheben?

12. Was ist das Ergebnis, wenn die Liste zahlen leer ist? Sollte der Code angepasst werden, um dies zu berücksichtigen?
13. Modifiziere den Code so, dass der Index i korrekt von 0 beginnt, damit alle Zahlen in der Liste überprüft werden.

10.7.4 Übungen

1. Klassen und Objekte

Übung 1: Erstellen einer einfachen Klasse

Aufgabe:

Erstelle eine Klasse Auto, die ein Auto repräsentiert. Die Klasse soll Attribute wie marke, modell und baujahr haben. Erstelle dann ein Objekt dieser Klasse und weise die Attribute zu. Gib die Attribute des Objekts mit print() aus.

Übung 2: Mehrere Objekte einer Klasse

Aufgabe:

Erstelle eine Klasse Hund, die Attribute wie name, alter und rasse besitzt. Erstelle drei verschiedene Hunde-Objekte mit unterschiedlichen Attributwerten und gib deren Informationen aus.

Übung 3: Klasseneigenschaften für gemeinsame Werte

Aufgabe:

Erstelle eine Klasse Schule, die eine Klassenvariable anzahl_schueler besitzt. Diese Variable soll für alle Objekte der Klasse gelten. Jede Instanz der Klasse soll die schueler_namen als Attribut haben. Füge Schüler zu der Schule hinzu und aktualisiere die Klassenvariable.

Übung 4: Instanziierung von Objekten

Aufgabe:

Erstelle eine Klasse Rechteck mit den Attributen breite und hoehe. Füge eine Methode hinzu, die die Fläche des Rechtecks berechnet. Erstelle mehrere Objekte und berechne die Fläche für jedes Rechteck.

Übung 5: Vergleich von Objekten

Aufgabe:

Erstelle eine Klasse Buch mit den Attributen titel, autor und seitenzahl. Erstelle eine Methode, die prüft, ob zwei Bücher die gleiche Seitenzahl haben. Erstelle zwei Buch-Objekte und vergleiche sie anhand der Seitenzahl.

2. Attribute und Methoden

Übung 1: Methoden zur Berechnung

Aufgabe:

Erstelle eine Klasse Kreis mit dem Attribut radius. Füge eine Methode hinzu, die den Umfang des Kreises berechnet (Formel: 2 * pi * radius) und eine Methode, die die Fläche berechnet (Formel: pi * radius^2).

Übung 2: Attribute mit Standardwerten

Aufgabe:

Erstelle eine Klasse Person mit den Attributen name und geburtsjahr. Das Attribut geburtsjahr soll einen Standardwert von 2000 haben, falls kein Wert angegeben wird. Füge eine Methode hinzu, die das Alter der Person berechnet.

Übung 3: Methode zur Aktualisierung von Attributen

Aufgabe:

Erstelle eine Klasse Laptop mit den Attributen marke, ram (in GB) und speicher (in GB). Füge eine Methode hinzu, um den RAM des Laptops zu erhöhen, und eine Methode, um den Speicherplatz zu erweitern. Teste die Methode an einem Laptop-Objekt.

Übung 4: Getter- und Setter-Methoden

Aufgabe:

Erstelle eine Klasse Bankkonto mit den Attributen kontonummer und kontostand. Füge eine Methode hinzu, um Geld auf das Konto einzuzahlen, und eine Methode, um Geld abzuheben. Achte darauf, dass der Kontostand nicht negativ werden kann.

Übung 5: Methoden für spezielle Berechnungen

Aufgabe:

Erstelle eine Klasse Student, die die Attribute name und notenliste enthält. Füge eine Methode hinzu, die den Durchschnitt der Noten berechnet, und eine Methode, die die höchste Note zurückgibt.

3. Anwendung von OOP in realen Projekten

Übung 1: Verwaltung von Büchern in einer Bibliothek

Aufgabe:

Erstelle eine Klasse Bibliothek, die eine Liste von Buch-Objekten enthält. Füge Methoden hinzu, um Bücher zur Bibliothek hinzuzufügen, zu entfernen und alle vorhandenen Bücher anzuzeigen.

Übung 2: Auto-Mietservice

Aufgabe:

Erstelle eine Klasse AutoMietservice, die eine Liste von Auto-Objekten verwaltet. Jedes Auto hat die Attribute marke, modell, verfuegbar. Füge Methoden hinzu, um ein Auto zu mieten und zurückzugeben.

Übung 3: Mitarbeiterverwaltung

Aufgabe:

Erstelle eine Klasse Mitarbeiter, die die Attribute name, position und gehalt enthält. Erstelle eine Methode, die eine Gehaltserhöhung basierend auf der Leistung des Mitarbeiters gewährt. Erstelle eine Klasse Unternehmen, die eine Liste von Mitarbeitern verwaltet und eine Methode enthält, um alle Mitarbeiterinformationen anzuzeigen.

Übung 4: Einkaufswagen im Online-Shop

Aufgabe:

Erstelle eine Klasse Produkt mit den Attributen name, preis und menge. Erstelle eine Klasse Einkaufswagen, die eine Liste von Produkten enthält. Füge Methoden hinzu, um Produkte in den Warenkorb zu legen, die Gesamtsumme zu berechnen und den Einkaufswagen zu leeren.

Übung 5: Studentennotenverwaltung

Aufgabe:

Erstelle eine Klasse Kurs, die eine Liste von Student-Objekten enthält. Jeder Student hat einen Namen und eine Liste von Noten. Füge Methoden hinzu, um neue Noten hinzuzufügen und den Notendurchschnitt des gesamten Kurses zu berechnen.

10.8 Einheit 8

10.8.1 Code-Analyse:

Aufgabe:

Der folgende Code implementiert eine Klasse Konto, die ein einfaches Bankkonto simuliert. Analysiere den Code und beantworte die dazugehörigen Fragen.

```
class Konto:
  def __init__(self, kontonummer, inhaber, saldo=0):
    self.kontonummer = kontonummer
    self.inhaber = inhaber
    self.saldo = saldo

  def einzahlen(self, betrag):
    self.saldo += betrag
    print(f"{betrag} wurde eingezahlt. Neuer Kontostand: {self.saldo}.")

  def abheben(self, betrag):
    if betrag <= self.saldo:
      self.saldo -= betrag
      print(f"{betrag} wurde abgehoben. Neuer Kontostand: {self.saldo}.")
    else:
      print(f"Nicht genügend Guthaben. Abhebung fehlgeschlagen.")

  def kontostand_anzeigen(self):
    print(f"Kontostand für {self.inhaber}: {self.saldo}.")

# Hauptprogramm
konto1 = Konto("DE12345678", "Max Mustermann", 500)
konto1.einzahlen(150)
konto1.abheben(200)
konto1.kontostand_anzeigen()
```

Fragen zur Code-Analyse:

- Was passiert, wenn der Benutzer versucht, mehr Geld abzuheben, als auf dem Konto verfügbar ist? Erkläre anhand des Codes, wie dies verhindert wird.
- Was ist der Zweck der Methode __init__() in dieser Klasse, und was passiert, wenn kein Startsaldo angegeben wird?
- Wie greift das Programm auf die Attribute kontonummer, inhaber und saldo zu? Erkläre den Zugriff auf diese Attribute in den verschiedenen Methoden.
- Was passiert, wenn der Benutzer den Kontostand abfragt, ohne vorher Geld eingezahlt oder abgehoben zu haben?

- Wie könnte die Methode abheben verändert werden, um eine Gebühr für jede Abhebung hinzuzufügen?

10.8.2 Fehlersuche:

Aufgabe:

Der folgende Code enthält Fehler bei der Implementierung der Klasse Rechteck. Die Klasse soll den Umfang und die Fläche eines Rechtecks berechnen, funktioniert jedoch nicht wie erwartet. Finde und behebe die Fehler im Code.

```
class Rechteck:
  def __init__(self, breite, hoehe):
    self.breite = breite
    self.hoehe = hoehe

  def berechne_umfang(self):
    umfang = 2 * self.breite + 2 * self.hoehe
    return umfang

  def berechne_flaeche():
    flaeche = self.breite * self.hoehe
    return flaeche

# Hauptprogramm
r1 = Rechteck(5, 10)
print("Umfang des Rechtecks:", r1.berechne_umfang())
print("Fläche des Rechtecks:", r1.berechne_flaeche())
```

Fragen zur Fehlersuche:

- In der Methode berechne_flaeche() tritt ein Fehler auf. Was fehlt in dieser Methode, damit sie richtig funktioniert? Wie kannst du dies beheben?
- Was passiert, wenn du versuchst, die Methode berechne_flaeche() zu verwenden? Warum tritt ein Fehler auf?
- Überprüfe, ob die Berechnung des Umfangs korrekt ist. Was könnte verbessert werden, um den Code lesbarer zu machen?
- Füge eine Methode hinzu, um die Breite oder Höhe des Rechtecks nachträglich zu ändern. Was passiert, wenn der Benutzer versucht, eine negative Breite oder Höhe einzugeben? Passe den Code an, um dies zu verhindern.

10.8.3 Code-Vervollständigung:

Aufgabe:

Vervollständige den folgenden Code für eine Klasse Buch, die Informationen über ein Buch speichert und eine Methode zur Überprüfung hinzufügt, ob das Buch ein Bestseller ist. Ein Buch gilt als Bestseller, wenn es mehr als 100.000 Exemplare verkauft hat.

```
class Buch:
  def __init__(self, titel, autor, verkaufte_exemplare):
    self.titel = titel
    self.autor = autor
    self.verkaufte_exemplare = verkaufte_exemplare

  def __str__(self):
    return f"{self.titel} von {self.autor} hat {self.verkaufte_exemplare} Exemplare verkauft."

  def ist_bestseller(self):
    _____ # Füge die Bedingung hinzu, ob das Buch ein Bestseller ist.
    _____
    else:
      return False

# Hauptprogramm
buch1 = Buch("Das große Abenteuer", "Max Mustermann", 120000)
buch2 = Buch("Das kleine Abenteuer", "Erika Musterfrau", 80000)

print(buch1)
if buch1.ist_bestseller():
  print(f"{buch1.titel} ist ein Bestseller.")
else:
  print(f"{buch1.titel} ist kein Bestseller.")

print(buch2)
if buch2.ist_bestseller():
  print(f"{buch2.titel} ist ein Bestseller.")
else:
  print(f"{buch2.titel} ist kein Bestseller.")
```

Fragen zur Code-Vervollständigung:

- Vervollständige die Methode ist_bestseller(), um zu überprüfen, ob ein Buch mehr als 100.000 Exemplare verkauft hat.
- Wie würdest du den Code erweitern, um Bücher mit unterschiedlichen Verkaufszahlen zu vergleichen und das erfolgreichste Buch zu finden?
- Ändere die Methode __str__(), sodass die Ausgabe formatiert ist und auch das Erscheinungsjahr des Buches anzeigt.
- Implementiere eine Methode, um die Verkaufszahlen nachträglich zu aktualisieren, wenn neue Exemplare verkauft wurden.

10.8.4 Übungen

Übung 1: Erstellen eines einfachen Textabenteuers (Konsolidierung von Variablen, Schleifen und Bedingungen)

Aufgabe:

Du wirst ein einfaches Textabenteuer entwickeln, bei dem der Benutzer durch verschiedene Entscheidungen navigiert. Du musst mehrere **Variablen** verwenden, um den Zustand des Spiels zu speichern, sowie **Schleifen** und **Bedingungen**, um die verschiedenen Entscheidungen zu ermöglichen.

Anforderungen:

- Erstelle eine Geschichte mit mindestens drei Szenen, in denen der Spieler Entscheidungen trifft.
- Jede Entscheidung soll den Spieler zu einem anderen Ergebnis führen (z. B. „Du hast das Spiel gewonnen“ oder „Du bist gescheitert“).
- Verwende mindestens **zwei Schleifen**, um den Benutzer durch das Spiel zu führen und ihm die Möglichkeit zu geben, Entscheidungen zu wiederholen.
- Speichere den aktuellen Status des Spiels in **Variablen** und zeige diese am Ende des Spiels an.

Übung 2: Taschenrechner mit erweiterten Funktionen (Verwendung von Funktionen und Bedingungen)

Aufgabe:

Erstelle einen **Taschenrechner**, der dem Benutzer ermöglicht, verschiedene Rechenoperationen durchzuführen (Addition, Subtraktion, Multiplikation, Division). Du wirst Funktionen verwenden, um die Berechnungen zu modularisieren und den Code übersichtlich zu gestalten.

Anforderungen:

- Definiere mindestens **vier Funktionen** für die Grundrechenarten (Addition, Subtraktion, Multiplikation, Division).
- Implementiere eine Funktion, die den Benutzer nach einer Auswahl fragt, welche Berechnung er durchführen möchte.
- Verwende **Bedingungen**, um zu überprüfen, ob die Eingaben korrekt sind (z. B. darf nicht durch 0 geteilt werden).
- Implementiere eine **Schleife**, um den Benutzer so lange Berechnungen durchführen zu lassen, bis er das Programm beenden möchte.

Übung 3: Objektorientierter Ansatz – Klassenerstellung für ein Bankkonto (Konsolidierung von OOP)

Aufgabe:

Entwickle eine Klasse Bankkonto, die die **Grundprinzipien der objektorientierten Programmierung (OOP)** anwendet. Dein Programm soll mehrere Bankkonten verwalten können und dem Benutzer die Möglichkeit bieten, Geld einzuzahlen, abzuheben und den Kontostand zu überprüfen.

Anforderungen:

- Erstelle eine Klasse Bankkonto mit den Attributen kontonummer, inhaber und kontostand.
- Implementiere Methoden, um Geld einzuzahlen, abzuheben und den aktuellen Kontostand anzuzeigen.
- Verhindere, dass der Kontostand negativ wird.
- Erstelle mindestens **zwei Objekte** der Klasse und lasse den Benutzer mit diesen Objekten interagieren.

Übung 4: Simulation eines Einkaufswagens (Verwendung von OOP und Listen)

Aufgabe:

Erstelle eine Klasse Produkt und eine Klasse Einkaufswagen, die eine Liste von Produkten enthält. Dein Ziel ist es, ein einfaches Programm zu entwickeln, das dem Benutzer ermöglicht, Produkte in den Einkaufswagen zu legen, zu entfernen und den Gesamtpreis zu berechnen.

Anforderungen:

- Erstelle eine Klasse Produkt mit den Attributen name, preis und menge.
- Erstelle eine Klasse Einkaufswagen, die eine Liste von Produkten enthält.
- Implementiere Methoden, um Produkte in den Einkaufswagen hinzuzufügen und zu entfernen.
- Berechne den Gesamtpreis der Produkte im Einkaufswagen.
- Gib den aktuellen Inhalt des Einkaufswagens sowie den Gesamtpreis aus.

10.9 Closedbook Test

Grundlagen des Programmierens

1. Prüfung

Aufgabenbereich	Beispiel 1	Beispiel 2	Erreicht	Möglich
Programmieren				20
Suche den Fehler				20
Vervollständige den Code				20
Ordne den Code				20
Multiple Choice				20
Ergebnis				**100**

Sie haben 45 Minuten Zeit, um diese Fragen zu beantworten

Nachname, Vorname Ort und Datum

10.9.1 Programmieren

10.9.1.1 Beispiel 1

Eine Liste mit 5 Tiernamen soll sortiert nach dem Alphabet ausgegeben werden. Wenn der Name des Tiers "Tiger" ist, soll die Funktion nicht "Tiger" ausgeben, sondern die spezielle Nachricht "Der Tiger ist der König der Tiere!".

Input ist eine Liste mit: Elefant, Tiger, Giraffe, Affe, Löwe

Output ist: Affe, Elefant, Giraffe, Löwe, „Der Tiger ist der König der Tiere!“

10.9.1.2 Beispiel 2

Input ist der Name eines Objekts: "Schiff", "Haus" oder "Baum".

Eine Funktion namens zeichne_objekt, die einen Parameter namens objekt akzeptiert, soll prüfen, ob der Wert "Haus" ist. Wenn ja, zeichne mit der Turtle-Grafik in Tigerjython ein einfaches abstraktes Haus. Folgende Meldung soll ausgegeben werden: "Ein Haus wurde gezeichnet“. Ansonsten gib eine Meldung aus: „Ein [Name des Objekts] konnte nicht gezeichnet werden.“

Aufruf: zeichne_objekt(„Haus“)

Ausgabe: [Das gezeichnete Haus] Ein Haus wurde gezeichnet

Aufruf: zeichne_objekt(„Baum“)

Ausgabe: Ein Baum konnte nicht gezeichnet werden

10.9.2 Suche den Fehler

10.9.2.1 Beispiel 1

Unten findest du einen Code, der eine Liste von Zahlen durchläuft und jede Zahl, die größer als 10 ist, in eine neue Liste aufnimmt. Allerdings enthält der Code in fünf verschiedenen Zeilen Fehler.

Finde die Fehler im Code!

```
def filter numbers(numbers):
  new_list = [
  for number in numbers
    if numbers > 10:
      new_list_append(number)
    return new_list

numbers = [5 15 9 20 5 6]
filtered_numbers = filter_numbers numbers
Print(filtered_numbers)
```

10.9.2.2 Beispiel 2

Unten findest du einen Code, der eine Reihe von geometrischen Formen zeichnen soll – entweder Dreiecke oder Quadrate. Die spezifische Form und die Anzahl der zu zeichnenden Formen werden durch Benutzereingaben bestimmt.

Finde die Fehler im Code!

```
def draw_triangle():
  for i in range(4):
  forward(100)
  right(120)

def draw_square():
  for i in range(4):
    forward(100)
    right(80)

def draw_shapes(n, form):
  for i in range(n):
    if form == "viereck":
      draw_triangle()
    else:
      draw_square()
    penUp()
    forward(200)
    penDown()

makeTurtle()
anzahl_formen = int(input("Wie viele Formen möchten Sie zeichnen? "))
gewaehlte_form = input("Welche Form möchten Sie zeichnen? (Dreieck oder Viereck) ")
draw_shapes(anzahl_formen, gewaehlte_form.lower())
```

10.9.3 Vervollständige den Code

10.9.3.1 Beispiel 1

Folgender Code ist unvollständig. Vervollständige ihn!

```
from gturtle import *

def draw_square(size):
  for i in range(___):
    forward(___)
    right(___)

def draw_nested_squares():
  size = 100
  for i in range(4):
    draw_square(size)
    _______

makeTurtle()
draw_nested_squares()
```

10.9.3.2 Beispiel 2

Folgender Code ist unvollständig. Vervollständige ihn!

```
from gturtle import *

def draw_square():
  for i in ___:
    forward(100)
    right(___)

# Füge hier die Funktion draw_triangle() ein.
def draw_triangle():
  ________

def draw_shapes(numbers):
  for number in numbers:
    if number % ___ == 0:
      ___
    else:
      ___
    penUp()
    forward(120)
    penDown()

makeTurtle()
numbers = [7, 2, 9, 4, 1, 6, 3, 8, 5, 10]
draw_shapes(___)
```

10.9.4 Ordne den Code

10.9.4.1 Beispiel 1

Folgender Code ist ungeordnet. Ordne ihn!

```
makeTurtle()
forward(100)
def draw_square():
right(120)
number = int(input("Geben Sie eine Zahl ein: "))
   for i in range(3):
right(90)
from gturtle import *
else:
   for i in range(4):
if number <= 100:
      forward(100)
def draw_triangle():
draw_square()
      right(90)
draw_triangle()
```

10.9.4.2 Beispiel 2

Folgender Code ist ungeordnet. Ordne ihn!

```
number = int(input("Anzahl der Kreise: "))
makeTurtle()
def draw_circle():
forward(50)
   for i in range(number):
right(90)
if number > 0:
from gturtle import *
left(180 - angle)
      circle(50)
else:
      penUp()
angle = 360 / number
      penDown()
def circle(size):
   for i in range(number):
print("Die Anzahl muss größer als 0 sein.")
```

10.9.5 Multiple Choice

1. Welche Aussage über Variablen in Python trifft zu?

a) Eine Variable ist nur ein Alias für eine Speicheradresse.

b) Eine Variable kann immer nur einen einzigen Wert speichern.

c) Eine Variable in Python ist eine Referenz auf ein Objekt im Speicher.

d) Eine Variable kann keine Funktionen referenzieren.

2. Was ermöglicht die input() Funktion in Python?

a) Das Auslesen von Umgebungsvariablen.

b) Die Ausführung von Benutzerbefehlen.

c) Das Sammeln von Benutzereingaben über die Konsole.

d) Die Ausgabe von Informationen in die Konsole.

3. Welcher Codeausschnitt erstellt eine unendliche Schleife in Python?

a) while True:

b) for True:

c) while False:

d) for None:

4. Welches ist die korrekte Syntax, um in Python über eine Liste mit den Werten [1, 2, 3] zu iterieren und jeden Wert zu drucken?

a) for x in 1, 2, 3: print(x)

b) for x in [1, 2, 3]: print(x)

c) for x print in [1, 2, 3]:

d) print(x) for x in [1, 2, 3]:

5. Wie erstellt man korrekt eine Liste in Python mit den Elementen 1, 2 und 3?

a) list = (1, 2, 3)

b) list = [1, 2, 3]

c) list = {1, 2, 3}

d) list = '1, 2, 3'

6. Welche Syntax definiert korrekt eine Funktion in Python?

a) function my_function():

b) my_function[]:

c) my_function():

d) def my_function():

7. Welcher Codeausschnitt erstellt eine while-Schleife in Python, die mindestens einmal durchlaufen wird?

a) while x < 5:

b) x < 5 while:

c) while[x < 5]:

d) do while x < 5:

8. Welches ist die korrekte Syntax, um eine for-Schleife in Python zu erstellen, die von 0 bis 4 läuft?

a) for x in range(5):

b) x in range(5) for:

c) for[x in range(5)]:

d) x in range(5) for x:

9. Was bewirkt die folgende Python-Anweisung: numbers[1] = 0?

a) Sie setzt das zweite Element der Liste numbers auf den Wert zwei.

b) Sie löscht das zweite Element der Liste numbers.

c) Sie setzt das zweite Element der Liste numbers auf Null.

d) Sie fügt eine Null an der zweiten Position der Liste numbers ein.

10. Was ist der Unterschied zwischen einer lokalen und einer globalen Variablen in Python?

a) Lokale Variablen können in beliebigen Funktionen verwendet werden, globale nur in einer.

b) Globale Variablen speichern nur temporäre Werte, während lokale dauerhaft sind.

c) Lokale Variablen sind innerhalb einer Funktion definiert, während globale im gesamten Skript zugänglich sind.

d) Es gibt keinen Unterschied; in Python sind alle Variablen global.

10.10 Openbook Test

Grundlagen des Programmierens
2. Prüfung

Aufgabenbereich	Beispiel 1	Beispiel 2	Erreicht	Möglich
Programmieren				20
Suche den Fehler				20
Vervollständige den Code				20
Ordne den Code				20
Multiple Choice				20
Ergebnis				**100**

Sie haben 45 Minuten Zeit, um diese Fragen zu beantworten

Nachname, Vorname Ort und Datum

10.10.1 Programmieren

10.10.1.1 Beispiel 1

Erstellen Sie ein Programm, das eine einzige Liste verwendet, um die Noten von drei Schülern in drei verschiedenen Fächern zu speichern und darzustellen. Die Fächer sind Mathematik, Deutsch und Englisch. Die Schüler sind Anna, Ben und Carl. Die Noten sind in einer einzigen Liste in der folgenden Reihenfolge gespeichert:

noten = [[1, 2, 1], [2, 1, 2], [2, 2, 1]]

Diese Notenliste beinhaltet die Noten von Schülern: [Mathematik, Deutsch, Englisch]. Zuerst kommt Anna, dann Ben, dann Carl.

Das Programm soll für jeden Schüler den Namen und die Noten ausgeben. Wenn ein Schüler in Mathematik eine 1 hat, soll das Programm eine spezielle Nachricht ausgeben: "[Studentenname] hat in Mathematik eine 1!"

Die Ausgabe sollte wie folgt aussehen:

Anna: [1, 2, 1]

Anna hat in Mathematik eine 1!

Ben: [2, 1, 2]

Carl: [1, 2, 2]

Carl hat in Mathematik eine 1!

Überlege was wäre, wenn jemand nur 2 Fächer belegt hätte oder wenn eine Note eines Faches noch nicht feststeht? Wie könnte das umgesetzt werden?

10.10.1.2 Beispiel 2

Erstellen Sie ein Programm, das vier Quadrate in einander zeichnet. Jedes Quadrat sollte zentriert sein und größer als das vorherige. Die Seitenlängen der Quadrate werden als Liste angegeben, und das Programm soll die Quadrate zeichnen. Jedes Mal, wenn ein Quadrat gezeichnet wird, sollte die Ausgabe: „Quadrat mit Seitenlänge [hier die Seitenlänge] wird gezeichnet“

Seitenlängen: [15, 20, 24, 30]

Als Ergebnis wird neben der grafischen Ausgabe der Quadrate eine Funktion square(s) erwartet, die als Parameter s bekommt.

Überlege: was ist, wenn die Liste der Seitenlänge unsortiert ist [15, 30, 24, 20]?

10.10.2 Suche den Fehler

10.10.2.1 Beispiel 1

Im bereitgestellten Code sind 5 Fehler enthalten, die identifiziert und korrigiert werden müssen. Identifizieren Sie die fehlerhaften Zeilen und korrigieren Sie den Code.

```
1  from gturtle import *
2
3  def draw_square()
4     for i in range(5):
5      forward(100)
6        right(45)
7
8  def draw_triangle():
9     for i in range(3):
10       forward(100)
11       right(120)
12
13 makeTurtle()
14 number == int(input("Geben Sie eine Zahl ein: "))
15
16 if number % 4 == 0
17     for i in range(number):
18       draw_square()
19       penUp()
20       forward(10)
21       penDown()
22 else
23     for i in range(number):
24        draw_triangle()
25        penUp()
26        forward(10)
27        penDown()
```

10.10.2.2 Beispiel 2

Im bereitgestellten Code sind 5 Fehler enthalten, die identifiziert und korrigiert werden müssen. Identifizieren Sie die fehlerhaften Zeilen und korrigieren Sie den Code.

```
1  from gturtle import
2
3  def draw_circle():
4     for i in range(90):
5        forward(1)
6        right(1)
7
8  def draw_square():
9     for i in range(8)):
10       forward(100)
11       right(90)
12
13 makeTurtle()
14 number = input("Geben Sie eine Zahl ein: ")
15
16 if number < 50
17    for i in range(number):
18    draw_circle()
19       penUpward()
20       forward(200)
21       penDown()
22 else:
23    for i in range(number):
24       draw_square()
25       penDown()
26       forward(200)
27       penUp()
```

10.10.3 Vervollständige den Code

10.10.3.1 Beispiel 1

Vervollständigen Sie den folgenden Code, indem Sie die fehlenden Teile ergänzen:

```
def calculate_volume(length, width, height):
    ___

def print_volume(volume, threshold):
    ___

length = 10
width = 5
height = 2
threshold = 100

volume = calculate_volume(length, width, height)
print_volume(volume, threshold)
```

10.10.3.2 Beispiel 2

Vervollständigen Sie den folgenden Code, indem Sie die fehlenden Teile ergänzen:

```
def calculate_statistics(data):
    avg = sum(data) / len(data)
    min_val = _____(data)  # Der Schüler soll die Funktion zur Berechnung von min_val einsetzen
    max_val = _____(data)  # Der Schüler soll die Funktion zur Berechnung von max_val einsetzen
    return avg, min_val, max_val

def filter_data(data, min_val, max_val):
    # Der Schüler soll die Bedingung unter Verwendung der Spannweite vervollständigen
    return [x for x in data if x > min_val + 0.1 * (_____ - _____)]

def print_statistics(avg, min_val, max_val, filtered_data):
    print("Durchschnitt:", avg)
    print("Minimum:", min_val)
    print("Maximum:", max_val)
    print("Gefilterte Daten:", filtered_data)

data = [15, 20, 35, 40, 25, 30]
avg, min_val, max_val = calculate_statistics(data)
filtered_data = filter_data(data, _____, _____)  # Der Schüler soll die Argumente einsetzen
print_statistics(_____, _____, _____, filtered_data)  # Der Schüler soll die Argumente einsetzen
```

10.10.4 Ordne den Code

10.10.4.1 Beispiel 1

Der untenstehende Text enthält alle notwendigen Funktionen und Anweisungen, jedoch sind die Anweisungen in falscher Ordnung. Bringen Sie den Code in eine korrekte Reihenfolge.

print_average(average)

average = calculate_average(numbers)

def calculate_average(numbers):

def get_numbers(n):

numbers = get_numbers(5)

print("Der Durchschnitt der Zahlen ist:", average)

def print_average(average):

num = int(input("Geben Sie eine Zahl ein: "))

return sum(numbers) / len(numbers)

numbers = []

return numbers

for i in range(n):

numbers.append(num)

10.10.4.2 Beispiel 2

Der untenstehende Text enthält alle notwendigen Funktionen und Anweisungen, jedoch sind die Anweisungen in falscher Ordnung. Bringen Sie den Code in eine korrekte Reihenfolge.

```
data = get_data(5)

sum_val = calculate_sum(data)

def calculate_sum(data):

return sum(data)

print("Die Summe der Zahlen ist:", sum_val)

for i in range(n):

return data

data.append(num)

def get_data(n):

data = []

num = int(input("Geben Sie eine Zahl ein: "))

def print_sum(sum_val):

print_sum(sum_val)
```

10.10.5 Multiple Choice

1. Wie wird das Konstrukt [1, 10, 50, 65, 80] in Python bezeichnet?

a) Eine Liste

b) Ein Objekt

c) Ein Tuple

d) Ein Dictionary

2. Welches Verhalten zeigt der folgende Code?

```
x = 1
while (x == 1):
  print(x)
```

a) Der Code wird unendlich oft ausgeführt, druckt ständig die Zahl 1.

b) Der Code wird genau einmal ausgeführt und druckt die Zahl 1.

c) Der Code verursacht einen Syntaxfehler.

d) Der Code beendet sich sofort ohne etwas zu drucken.

3.Was definiert das folgende Python-Konstrukt?

```
def draw_square():
  for i in range(4):
    forward(100)
    right(90)
```

a) Ein Viereck wird gezeichnet.

b) Eine Funktion namens draw_square wird definiert, die ein Viereck zeichnet.

c) Eine unendliche Schleife.

d) Eine Klasse namens draw_square.

4. Was ist der Unterschied zwischen == und = in Python?

a) == ist ein Vergleichsoperator, = ist ein Zuweisungsoperator.

b) == wird in Schleifen verwendet, = wird in Funktionen verwendet.

c) Es gibt keinen Unterschied.

d) == wird für arithmetische Operationen verwendet, = für logische Operationen.

5. Was ist der Unterschied zwischen einer lokalen und einer globalen Variable in Python?

a) Lokale Variablen können nur innerhalb ihrer definierten Funktion verwendet werden, globale Variablen können im gesamten Code verwendet werden.

b) Lokale Variablen haben eine längere Lebensdauer als globale Variablen.

c) Es gibt keinen Unterschied.

d) Globale Variablen sind standardmäßig in allen Modulen sichtbar, lokale nicht.

6. Wie zeichnen Sie eine Linie, die 100 Pixel lang ist und nach rechts zeigt, mit der Turtle-Grafik?

a) right(90); forward(100)

b) forward(100); right(90)

c) left(90); forward(100)

d) forward(100); left(90)

7. Was macht der folgende Code in Python?

```
number = 50
if number < 50:
  print("HALLO")
elif number > 50:
  print("SERVUS")
else:
  print("GUTEN TAG")
```

a) Gibt "HALLO" aus.

b) Gibt "SERVUS" aus.

c) Gibt "GUTEN TAG" aus.

d) Gibt nichts aus.

8. Was beschreibt die Funktion range(1, 10, 2) in Python?

a) Erzeugt eine Liste mit Zahlen von 1 bis 10.

b) Erzeugt ein Iterator-Objekt mit Zahlen von 1 bis 10 in Zweierschritten.

c) Erzeugt eine Liste mit Zahlen von 1 bis 9 in Zweierschritten.

d) Erzeugt ein Iterator-Objekt mit Zahlen von 1 bis 9 in Zweierschritten.

9. Was ist der Wert von x nach Ausführung des folgenden Codes?

```
x = 5
x *= 3 + 2
```

a) 15

b) 25

c) 20

d) 10

10. Welches Verhalten zeigt der folgende Python-Code, wenn x = 5 ist?

```
if x < 10:
    print("Ein")
elif x < 20:
    print("Zwei")
else:
    print("Drei")
```

a) Druckt "Ein"

b) Druckt "Zwei"

c) Druckt "Drei"

d) Verursacht einen Syntaxfehler

10.11 Abschlussprojekte

Wählen Sie aus folgenden Projekten und lösen Sie die Aufgabe. Dokumentieren Sie deine Lösungsschritte mittels eines Lerntagebuchs. Deine Abgabe beinhaltet die Lösung und das Lerntagebuch.

Projekt 1: Erweiterter Rabattrechner mit mehreren Produkten und dynamischen Rabatten

Aufgabe:

Entwickeln Sie ein Programm, das mehrere Produkte gleichzeitig verwaltet und dynamische Rabatte auf Grundlage der Anzahl der gekauften Artikel sowie eines saisonalen Rabatts anwendet. Das Programm soll folgende Funktionalitäten umfassen:

Anforderungen:

- **Benutzereingabe für mehrere Produkte**: Der Benutzer soll den Namen, den Preis und die Anzahl der Produkte eingeben können.
- **Rabattregelungen**:
    - **Mengenrabatt**: Wenn der Benutzer mehr als 5 Einheiten eines Produkts kauft, soll ein zusätzlicher Rabatt von 10 % angewendet werden.
    - **Saisonaler Rabatt**: Zusätzlich soll ein saisonaler Rabatt erfasst werden, den der Benutzer eingibt (z. B. 15 % für den Sommer).
- **Endpreisberechnung**: Berechne den Endpreis jedes Produkts und den Gesamtpreis aller Produkte.
- **Zusammenfassung der Einkaufsliste**: Am Ende gibt das Programm eine Zusammenfassung aus, in der die Produkte, der ursprüngliche Preis, der Rabatt und der Endpreis für jedes Produkt aufgelistet werden.

Erweiterungen:

1. Fügen Sie eine Benutzeroption hinzu, um Produkte zu entfernen oder zu bearbeiten.
2. Lassen Sie den Benutzer mehrere Rabatte anwenden (z. B. einen speziellen Rabattcode).
3. Integrieren Sie eine Funktion, um einen Bericht über den gesamten Einkauf zu erstellen und in eine Datei zu speichern.

Projekt 2: Temperaturrechner mit automatischer Datenerfassung und erweiterter Analyse

Aufgabe:

Entwickeln Sie ein Programm, das eine Reihe von Temperaturwerten in Celsius erfasst, sie in Fahrenheit umrechnet und mehrere erweiterte Analysen durchführt. Das Programm soll folgende Funktionalitäten umfassen:

Anforderungen:

- **Erfassung von Temperaturen über eine Datei**: Der Benutzer lädt eine Datei hoch, die eine Liste von Temperaturwerten in Celsius enthält.

- **Umrechnung in Fahrenheit**: Jede Temperatur wird automatisch in Fahrenheit umgerechnet.
- **Erweiterte Analyse**:
    - Zeige an, wie viele Temperaturen unter dem Gefrierpunkt (0 °C) liegen.
    - Bestimme den Durchschnitt, die höchste und die niedrigste Temperatur (in Celsius und Fahrenheit).
- **Gefrierpunktanzeige**: Für jede Temperatur unter 0 °C soll „Gefrierpunkt erreicht“ ausgegeben werden.
- **Erstellen einer Übersicht**: Am Ende wird eine vollständige Übersicht der Temperaturwerte inklusive der Analyse angezeigt.

Erweiterungen:

1. Lassen Sie den Benutzer die Daten nicht nur aus einer Datei, sondern auch manuell eingeben.
2. Speichern Sie die Ergebnisse der Analyse in einer Datei ab.
3. Integrieren Sie einen Temperaturtrend (z. B. Steigung oder Senkung über die Zeit).

Projekt 3: Bibliothekssoftware zur Palindromerkennung und erweiterte Buchverwaltung

Aufgabe:

Erstellen Sie eine Bibliothekssoftware, die Buchtitel verwaltet und überprüft, ob ein Titel ein Palindrom ist. Die Software soll zudem eine erweiterte Buchverwaltung beinhalten, einschließlich der Möglichkeit, Bücher hinzuzufügen, zu entfernen und zu durchsuchen.

Anforderungen:

- **Eingabe von Buchtiteln**: Der Benutzer gibt eine Liste von Buchtiteln ein.
- **Palindromprüfung**: Jeder Buchtitel wird überprüft, ob er ein Palindrom ist (Wörter von vorne und hinten gelesen sind gleich, ohne Leerzeichen und Groß-/Kleinschreibung zu beachten).
- **Anzeige der Palindrome**: Zeige alle Palindrome und die Gesamtanzahl der Palindrome in der Liste an.
- **Buchverwaltung**: Fügen Sie Funktionen hinzu, um Bücher zur Bibliothek hinzuzufügen, zu entfernen und zu durchsuchen.
- **Erweiterte Suchfunktion**: Implementieren Sie eine erweiterte Suchfunktion, um Bücher anhand von Stichwörtern oder Teilen des Titels zu finden.

10.12 Einzelergebnisse – Closed Book

10.12.1 ChatGPT

Student	Programmieren Beispiel	Programmieren Beispiel	Suche den Fehler Beispiel	Suche den Fehler Beispiel	Vervollständige den Code Beispie	Vervollständige den Code Beispie	Ordne den Code Beispiel	Ordne den Code Beispiel	Multiple Choic	Gesamtpunktzah
1	0	0	0	0	0	0	10	0	14	24
2	2	10	10	0	10	10	10	10	20	82
3	10	10	0	10	10	10	10	10	20	90
4	0	6	10	10	10	0	0	0	20	56
5	10	10	10	10	10	0	10	10	14	84
6	6	10	10	10	8	0	6	0	20	70
7	4	4	2	0	10	10	8	8	20	66
8	6	8	0	10	0	10	10	10	20	74
9	6	0	10	10	0	0	0	0	20	46
	4,888888889	**6,444444444**	**5,777777778**	**6,666666667**	**6,444444444**	**4,444444444**	**7,111111111**	**5,333333333**	**18,66666667**	**65,77777778**

10.12.2 E-Book

Student	Programmieren Beispiel 1	Programmieren Beispiel 2	Suche den Fehler Beispiel 1	Suche den Fehler Beispiel 2	Vervollständige den Code Beispiel 1	Vervollständige den Code Beispiel 2	Ordne den Code Beispiel 1	Ordne den Code Beispiel 2	Multiple Choice	Gesamtpunktzahl
1	2	10	10	10	10	2	10	10	18	82
2	2	8	0	0	10	0	10	2	20	52
3	0	10	4	10	10	10	10	6	20	80
4	10	10	10	10	6	10	10	10	10	86
5	10	10	0	0	2	2	10	4	20	58
6	8	4	10	8	2	0	8	8	20	68
7	6	6	2	0	10	10	8	10	18	70
8	2	6	8	10	10	10	10	4	20	80
9	2	2	8	2	10	8	8	2	12	54
10	2	2	2	2	10	6	10	6	16	56
11	8	8	4	4	0	4	6	8	20	62
	4,727272727	6,909090909	5,272727273	5,090909091	7,272727273	5,636363636	9,090909091	6,363636364	17,63636364	68

10.13 Einzelergebnisse – Open Book

10.13.1 ChatGPT

Student	Programmieren Beispiel	Programmieren Beispiel	Suche den Fehler Beispiel	Suche den Fehler Beispiel	Vervollständige den Code Beispie	Vervollständige den Code Beispie	Ordne den Code Beispiel	Ordne den Code Beispiel	Multiple Choic	Gesamtpunktzah
1	10	8	10	10	10	10	10	10	0	78
2	10	6	10	10	10	0	10	10	20	86
3	10	8	10	10	0	10	6	8	18	80
4	10	6	2	0	10	0	8	8	20	64
5	10	8	10	10	0	10	2	2	20	72
6	8	8	10	10	10	0	4	6	17	73
7	10	8	10	10	10	10	10	10	11	89
8	10	6	0	10	0	10	8	8	20	72
9	10	6	4	8	10	10	10	10	15	83
10	10	10	4	10	6	8	10	10	20	88
11	10	8	8	0	0	0	10	10	17	63
12	8	8	6	8	8	2	10	10	20	80
13	8	4	10	10	6	10	8	6	20	82
14	10	6	4	8	10	10	10	10	19	87
15	10	8	4	10	10	10	10	10	17	89
	9,6	**7,2**	**6,8**	**8,266666667**	**6,666666667**	**6,666666667**	**8,4**	**8,533333333**	**16,93333333**	**79,06666667**

10.13.2 E-Book

Student	Programmieren Beispiel	Programmieren Beispiel	Suche den Fehler Beispiel	Suche den Fehler Beispiel	Vervollständige den Code Beispie	Vervollständige den Code Beispie	Ordne den Code Beispiel	Ordne den Code Beispiel	Multiple Choic	Gesamtpunktzah
1	10	10	10	8	10	10	10	10	20	98
2	10	10	10	10	4	10	10	10	20	94
3	10	10	4	0	10	10	8	8	20	80
4	10	10	10	10	6	10	10	10	19	95
5	10	10	10	6	0	10	10	10	20	86
6	10	10	10	10	10	6	10	10	18	94
7	10	10	4	2	2	4	8	8	16	64
8	10	10	4	2	2	4	8	6	17	63
9	10	10	8	0	10	0	10	8	18	74
10	10	8	8	10	10	0	10	8	20	84
11	8	10	4	0	10	10	10	10	20	82
12	10	6	4	0	0	6	10	10	20	66
13	10	8	10	10	10	10	6	8	20	92
14	10	8	6	2	4	4	10	10	20	74
	9,857142857	**9,285714286**	**7,285714286**	**5**	**6,285714286**	**6,714285714**	**9,285714286**	**9**	**19,14285714**	**81,85714286**