MASTERARBEIT | MASTER'S THESIS

Titel | Title

TEACHING GENERIC SKILLS AND PROGRAMMING SKILLS IN COMPUTER SCIENCE LESSONS BY USING PAIR PROGRAMMING AND GENERATIVE AI-TOOLS – A CASE STUDY

verfasst von | submitted by

Katrin Göltl BEd

angestrebter akademischer Grad | in partial fulfilment of the requirements for the degree of

Master of Education (MEd)

Wien | Vienna, 2025

Studienkennzahl lt. Studienblatt | Degree programme code as it appears on the student record sheet:	UA 199 510 514 02
Studienrichtung lt. Studienblatt | Degree programme as it appears on the student record sheet:	Masterstudium Lehramt Sek (AB) Unterrichtsfach Geographie und wirtschaftliche Bildung Unterrichtsfach Informatik
Betreut von | Supervisor:	Univ.-Prof. Dipl.-Ing. Dr. Renate Motschnig

1 Table of Contents

Abstract

This master's thesis investigates the potential of discovery-based learning in Computer Science education by designing, implementing, and evaluating a teaching sequence aimed at enhancing both programming skills and generic competencies such as creativity, collaboration, and communication. The teaching sequence included six double lessons (each lasting 100 minutes) and incorporated pair programming, with students alternating between human peers and generative AI tools as programming partners. A case study methodology was utilized to evaluate the effectiveness of this approach within a real-world classroom setting.

The findings indicate that discovery-based learning, though underrepresented in Computer Science education, shows potential when combined with contemporary programming didactics such as pair programming and AI-supported collaboration. Most students acquired at least some of the programming skills taught. However, students viewed working with AI as a constraint on creativity, pointing out significant limitations regarding problem-solving opportunities and original thinking. These findings suggest that while integrating AI into programming education can be advantageous, careful evaluation of its effects on cognitive processes and student engagement is needed. This study adds to the ongoing discussion on innovative teaching methods in Computer Science and lays the groundwork for further research on the effective integration of AI into programming education.

Zusammenfassung

Diese Masterarbeit untersucht das Potenzial des entdeckenden Lernens in der Informatikausbildung durch die Konzeption, Umsetzung und Evaluierung einer Unterrichtssequenz, die darauf abzielt, sowohl die Programmierfähigkeiten als auch allgemeine Kompetenzen wie Kreativität, Zusammenarbeit und Kommunikation zu verbessern. Die Unterrichtssequenz umfasste sechs Doppelstunden à 100 Minuten. Dabei bearbeiten die Schüler:innen Programmieraufgaben sowohl alleine, als auch im Pair-Programming-Modus, wobei sie einerseits mit einem Kollegen/einer Kollegin und andererseits mit generativen KI-Tools als Programmierpartner arbeiteten. Anhand einer Fallstudie wurde die Wirksamkeit dieses Ansatzes in einem realen Klassenzimmer untersucht.

Die Ergebnisse zeigen, dass entdeckendes Lernen, obwohl es in der Informatikbildung eher weniger verbreitet ist, Potenzial hat, wenn es mit modernen didaktischen Ansätzen wie Pair-Programming und KI kombiniert wird. Die meisten Schüler:innen erwarben zumindest einige der vermittelten Programmierkompetenzen. Allerdings sahen die Schüler:innen die Arbeit mit KI als Einschränkung ihrer Kreativität an und wiesen auf erhebliche Einschränkungen in Bezug auf Problemlösungsmöglichkeiten und originelles Denken hin. Diese Ergebnisse deuten darauf hin, dass die Integration von KI in die Vermittlung von Programmierkompetenzen zwar vorteilhaft sein kann, aber eine sorgfältige Bewertung ihrer Auswirkungen auf die kognitiven Prozesse und das Engagement der Schüler:innen erfordert. Diese Studie trägt zur laufenden Diskussion über innovative Lehrmethoden in der Informatik bei und legt den Grundstein für weitere Forschungen zur effektiven Integration von KI in die Programmierbildung.

1. Introduction

In today's world, the fields of education and business are under rapid development. This development comes mainly from the rapid growth of technology. The way we live and work is developing faster and faster and educational institutions are faced with the challenge of keeping pace with this development. One of the biggest challenges in modern teaching is to equip students with subject-specific knowledge and generic skills that enable them to operate in a complex, networked world successfully.

The current curriculum for 5th-grade upper secondary schools (AHS) in Austria includes both subject-specific goals and generic competencies intended to be taught in Computer Science classes. The computer science lessons aim to develop communication skills, collaboration skills, and creativity. A key component of the curriculum is also the teaching of programming skills. Combining programming skills with generic competencies such as teamwork, communication, problem-solving abilities, and creativity can provide students with various opportunities in their personal lives and professional careers.

To achieve the objective of concurrently teaching programming skills and generic competencies, the method of pair programming—wherein two learners collaborate at one computer, dividing the programming task and supporting each other (Xu & Correia, 2023)—appears particularly promising. This approach not only enhances a deeper understanding of programming concepts but also promotes the development of social and communication skills while increasing students' motivation and enjoyment of learning (Xu & Correia, 2023).

Pair programming has been used as a teaching method for programming for a long time, with extensive research conducted in this field. The introduction of artificial intelligence in education presents an opportunity to enhance and modernize this approach. The use of artificial intelligence has expanded significantly in recent years across various domains, including education. AI has the potential to be a valuable tool in teaching and learning by personalizing the educational experience and improving outcomes for each learner. Generative AI tools can explain complex programming problems, offer solutions, foster creative thinking, and assist students in developing innovative approaches to programming challenges (Spinellis, 2024).

The motivation for addressing this topic in this Master's thesis arises from the belief that the integration of pair programming and generative AI tools in computer science education represents a sustainable and future-oriented educational strategy. By combining these approaches, both subject-specific and generic competencies can be promoted in a way that meets the demands of a modern technology-driven society. This Master's thesis aims to analyze the potentials and challenges of these methods and to develop a practice-oriented recommendation for their implementation in an upper secondary school.

This Master's thesis should contribute to revolutionizing Computer Science education in schools. It should assist teachers in preparing their students as effectively as possible for the challenges of the digital future. Therefore, the underlying research question of this paper is: "To what extent does the integration of pair programming and generative AI tools in Computer Science lessons for the 5th grade of an upper secondary school (AHS) assist the development of selected generic competencies of the students as well as the development of their programming skills?"

A teaching sequence, based on discovery-based learning and pair programming, was developed and implemented to address this research question. A case study methodology accompanied this implementation. This will be presented in a two-part format. The first part of this Master's thesis covers the theoretical framework used for developing the teaching sequence, which includes principles of discovery-based learning, teaching programming, and using AI for educational purposes. The second part presents the teaching sequence developed based on these frameworks. Following this, the research design and results will be displayed. Finally, this practical part will be related to the literature, and final recommendations will be provided.

2 Theoretical Framework

This Master's thesis addresses the mediation of programming skills and generic skills in Computer Science classes through the use of a discovery-based learning approach, pair programming, and generative AI tools. To provide an overview of the current state of research and establish a foundation for the development, implementation, and analysis of a teaching sequence aimed at achieving this objective, an in-depth literature review has been conducted. The thesis begins with an introduction to the concept of discovery-based teaching and learning. Subsequently, it explores the foundations of teaching programming skills, followed by a detailed explanation of the pair programming teaching approach. An essential component of this thesis is the application of generative AI tools for educational purposes, which is discussed extensively in a dedicated sub-chapter of the theoretical framework.

2.1 Discovery-based Learning

Discovery-based learning is an instructional approach that diverges from the traditional method of imparting knowledge. In this approach, students are encouraged to uncover information previously unknown to them independently. Instead of receiving knowledge mediated by an expert on the subject, students acquire understanding through experimentation and exploration. Although students are expected to gain insights autonomously, educators must establish and maintain a conducive learning environment. This environment requires ongoing evaluation and adaptation to remain effective (Bönsch, 1999).

Discovery-based learning is not a novel concept within the educational realm. Jerome Bruner first introduced the idea of discovery-based learning in 1961. He defined it as an independent exploration of a knowledge domain, where the learner must engage their intellectual abilities to solve problems. This method enhances the learners' problem-solving skills. In this approach, the teacher plays a more supportive role, observing and assisting learners in acquiring new knowledge. Additionally, teachers should demonstrate problem-solving techniques. Discovery-based learning capitalizes on the inherent curiosity possessed by each individual, fostering intrinsic motivation to learn, particularly when learners are only partially familiar with the knowledge they are expected to acquire (Riedl & Schelten, 2012).

It is assumed that learners engaging with unfamiliar or new content attempt to associate this information with their pre-existing knowledge. They seek similarities within their existing knowledge structures and endeavor to elucidate the newly acquired information by identifying similarities and differences. The greater the disparity between the cases they encounter, the more refined their rules become. Based on this foundation, learners can deductively infer individual cases from the generalizations present in their minds (Riedl & Schelten, 2012).

Discovery-based learning involves allowing students to engage in individual cognitive processes through appropriate tasks. Effective implementation of this method requires suitable teaching and learning environments, which are closely related to problem-solving learning strategies. The objective of discovery-based learning is for students to independently acquire knowledge and identify patterns and relationships. For successful and pedagogically sound application of this method, it is essential to ensure that students possess sufficient and relevant prior knowledge to which they can connect newly acquired information. Consequently, this approach cannot be effectively employed with topics that are entirely unfamiliar to students and for which they have little or no prior knowledge (Riedl & Schelten, 2012).

Bönsch (1999) argues that certain prerequisites need to be ensured, for discovery-based learning to be successful:

- One cannot expect that discovering knowledge happens without any pre-knowledge. Students need to have certain basic knowledge in this field to develop questions, understand problems, and keep interest.
- Students need to be encouraged and motivated to have fun discovering new knowledge.
- Students need to try things out in the real world with real problems and not in theoretical scopes. Fourth
- The questions and problems the students need to solve need to be formulated as precise as possible
- Teachers must transform boring topics into vivid tasks
- Teachers need to encourage students to draft plans before they start solving the problem
- Teachers and students need to check whether the discovered solution and the knowledge that is built on this solution are adequate and accurate.

(Bönsch, 1999).

While the concept of discovery-based learning appears straightforward, there is no single precise definition for the term. The literature reveals various learning tasks that fall under the scope of discovery-based learning, ranging from implicit pattern detection to eliciting explanations, and from working through manuscripts or manuals to conducting simulations (Alfieri et al., 2011). Despite the differences in these definitions, they all share a common element: in a discovery-based learning environment, the teacher adopts a facilitative role, allowing students to uncover target information independently. For this Master's thesis, the following definition will be employed, which integrates various aspects discussed in the literature:

"Discovery-based learning means that the teacher provides the students with sufficient but minimal guidance and instruction in a desired field of knowledge, in such that they can acquire target information and skills on their own by being exposed to tasks and problems they need to solve themselves"

The subsequent chapters will delve into the primary advantages and considerations of this approach. Furthermore, specific implementation requirements and opportunities for discovery-based learning within Computer Science courses will be outlined.

2.1.1 Benefits, Pitfalls, and Considerations for Implementing Discovery Learning

Already in the 1950s research was concerned with the usefulness of discovery-based learning in comparison with other forms of instruction. There is vast agreement in literature, that discovery-based learning without the guidance and assistance of a teacher who ensures that learning takes place is quite useless as it lacks structure (Alfieri et al., 2011). Therefore, we will only consider discovery-based learning that follows the rules brought up by Bönsch and is as guided as necessary.

In early research, Ray (1961) argued that discovery-based learning has positive effects on the retention of information in contrast to traditional direct instruction. Moreover, being able to discover new information and knowledge yourself helps in developing strategies for lifelong learning (Dorier & Garcia, 2013).

Teaching something yourself, which is the base of discovery-based learning, means that you construct a conversation with yourself, and improve in your own scope of knowledge. This is directly linked to Vygotsky's Zone of Proximal Development (Vygotsky, 1978).

Baldwin (1995) suggests that discovery-based learning may be more effective than conventional instruction because it addresses the issue of "bulimic learning," where students learn material only for a test or exam and forget it afterward. In discovery-based learning, students engage in activities beyond simple memorization tasks; they are encouraged to identify and solve problems independently. Traditional instruction methods often involve students relying on the materials presented by the teacher without encouraging them to think outside the box.

Furthermore, discovery-based learning enables students to gain more than just lesson content or requisite skills. It equips them with the ability to teach themselves and understand the process of acquiring new knowledge. This meta-knowledge about learning can be beneficial in future educational contexts and throughout their lives (Baldwin, 1995).

However, there are some considerations when implementing a discovery-based learning approach in teaching. Initially, students may feel overwhelmed, and the cognitive workload can be substantial when this method is introduced. Therefore, it is advisable to familiarize students with this approach in advance. The more frequently the method is utilized, the more likely students are to develop an appreciation for learning. When students become accustomed to discovery learning, it encourages them to do more than actively participate in lessons. Rather they are constructing new information and extending their knowledge beyond the presented material. It is important to note that while this approach can lead to valuable insights into students' prior knowledge and facilitate its transformation, it does not guarantee that new information will be permanently retained (Alfieri et al., 2011).

Additionally, employing a discovery-based learning approach may not always align with the strict schedules of educational institutions. The timing of when discovery occurs is unpredictable, and it cannot be guaranteed that every student will acquire the intended knowledge and achieve the overall objectives. Therefore, the teacher needs to provide the necessary support, ensuring students receive as much assistance as needed while maintaining minimal interference (Baldwin, 1995).

Discovery-based learning requires a significant amount of time and ongoing monitoring. When a student becomes distracted by external factors, the teacher must ensure that continuous progress is maintained (Baldwin, 1995).

Finally, it needs to be noted that testing students in a discovery-based learning course can be challenging, as individual discovery is so diverse and different. This means that one should rather grade and assess the participation and the project than the process. Moreover, certain aspects of discovery-based learning, such as meta-learning are hard to measure and assess (Baldwin, 1995).

2.1.2 Discovery Learning in Computer Science Lessons

Although there is limited research on implementing discovery-based learning principles in Computer Science education (Arnold et al., 2005), certain general principles that require special consideration in Computer Science lessons must be observed.

Arnold et al. (2005) highlight that, particularly in Computer Science courses, it is crucial for the topic to possess a certain degree of openness. This implies that the knowledge domain should be sufficiently broad to allow students to experiment, test hypotheses, and compare their findings with those of their peers. The topic must be diverse enough to enable multiple problem-solving approaches. Additionally, educators must provide students with the necessary resources. These materials should be crafted to ensure that each student comprehends both the problem and the task at hand, minimizing the need for teacher assistance. Discovery-based learning emphasizes that students should generate new ideas and potential solutions for the given tasks. Finally, all assignments should be structured in such that they allow for various solutions, approaches, and perspectives, which should be acknowledged by the teacher. It is also vital to allocate adequate time for students to complete their tasks and develop their solutions (Arnold et al., 2005).

It is important to consider which topic to cover, as not every subject in a Computer Science classroom is suitable for discovery-based learning. Additionally, while the teacher does not have a leading role in this method of teaching, thorough and genuine preparation is necessary to ensure that specific competencies are conveyed effectively through discovery-based learning (Arnold et al, 2005).

There is limited evidence on best-practice examples for implementing a discovery-based learning course in Computer Science. Baldwin (1995), for instance, developed two university courses aimed at teaching computer graphics and programming skills in C and Unix through discovery-based learning. He concluded that discovery-based learning is effective, as students acquire more knowledge compared to a traditional course. Additionally, they found the learning process more enjoyable, the instructor was more relaxed, and students gained skills beyond the subject matter, such as independent learning.

Baldwin (1995) and Arnold (2005) advocate for the integration of discovery-based learning into Computer Science education, highlighting the potential advantages for both students and educators.

2.2 Learning to program – a definition

Computer programming is defined as "*the process of developing and implementing various sets of instructions to enable a computer to perform a certain task, solve problems and provide human interactivity. These instructions (source codes which are written in a programming language) are considered computer programs and help the computer to operate smoothly.*" (Balanskat & Engelhardt, 2015).

Acquiring programming skills is interconnected with various other essential day-to-day competencies. Prior to embarking on programming, it is imperative to undertake certain preparatory activities, which include:

- "the analysis, understanding and generically solving of a problem, resulting in an algorithm"
- "the verification of requirements of the algorithm including its correctness"
- "the implementation […] of the algorithm in a target programming language"

(Balanskat & Engelhardt, 2015).

Over the past decade, learning to program has become a significant objective in the educational field due to rapid digitalization, which presents new challenges and opportunities for society. Additionally, learning how to program supports the development of skills like problem-solving, logical thinking, and creativity (Scherer et al., 2020).

Numerous countries have acknowledged the significance of programming education and have incorporated, or are in the process of incorporating, this skill into their curricula (Scherer et al., 2020).

The same is true for the Austrian school context. When considering the current curricula for the lower and upper secondary schools (AHS) in Austria, one finds that the aspect of learning to program is already implemented there. The following teaching and learning goals are formulated and have been translated by the author :

- 1st grade: Students can understand, execute and independently formulate clear instructions (algorithms)
- 2nd grade: Students can create, test and debug (identify and fix errors) simple programs using a suitable development environment
- 3rd grade: Students can use examples to understand elements of computational thinking and use them to solve problems. They know how to implement solutions in a programming language
- 4th grade: Students can design and iteratively develop programs that combine control structures, including nested loops and compound conditionals and students can create simple

programs or web applications using appropriate tools to solve a specific problem or accomplish a specific task.

- 5th grade: Students can explain design, represent, and implement algorithms in a programming language and students can explain basic principles of automata, algorithms, data structures, and programs

(Bundesministerium für Bildung, Wissenschaft und Forschung, 2024)

As previously noted, instructing students in programming encompasses more than merely teaching them the structures, syntax, and semantics of a particular programming language. It also serves to enhance students' problem-solving capabilities and fosters the development of computational thinking skills. Computational thinking is associated with the concept of 21st-century skills.

The term 21st-century skills is an umbrella term that summarizes a set of skills relevant to successfully meeting the demands of the 21st-century workforce and education. The ten skills that are summarized under the term 21st-century skills are creativity and innovation, critical thinking, problem-solving, decision-making, learning to learn, metacognition, communication, collaboration, information literacy, ICT literacy, citizenship, life and career skills, and personal and social responsibility (van Laar et al., 2020).

A term pertinent to 21st-century skills, which will be referenced in this paper, is generic skills. Canning (2013) defines generic skills as "a set of discrete clusters of skills or competencies […] that are transferable across different work contexts". These skills are not confined to a static set of competencies; instead, they evolve over time in response to current workplace demands. Nevertheless, certain generic skills maintain their relevance across decades. These enduring skills include communication, collaboration, problem-solving, computational thinking, and information technology (Canning, 2013).

Returning to the concept of computational thinking, Wing (2010) defines it as "the thought process involved in formulating problems and their solutions in such a manner that the solutions can be effectively executed by an information processing agent."

Computational thinking is not a stand-alone ability. Rather it is a combination of the following five sub-skills: Abstraction, Decomposition, Algorithmic Thinking, Generalization and Evaluation (Selby & Woollard, 2013)

The importance of possessing programming skills as a European citizen has also been recognized by the European Commission. In their DigComp 2.2 – a digital competence framework for citizens, that has to be implemented in every member country. They devote a whole section to programming and another one to problem-solving skills. (Vuorikari et al., 2022).

While learning to program is closely associated with the development of computational thinking skills, it is important not to use these terms interchangeably. Instead, learning to program should be viewed as a method for teaching and cultivating computational thinking (Scherer et al., 2020).

There are various methods and approaches for teaching computer programming skills to students. A brief overview of these will be provided in the following chapter.

2.2.1 Methods of Teaching Programming Skills

Before considering various methodologies for teaching programming, particularly introductory programming, it is essential to recognize that the effectiveness of these approaches is significantly influenced by the chosen programming language (Koulori et al., 2014).

Generally, one can say that there are three approaches to teaching programming:

- Programming-first: using the traditional imperative paradigm
- Objects-first: emphasis lies on the use of objects
- Functional-first: introduction of algorithmic concepts with a simple functional syntax

(Koulori et al., 2014).

The question of which approach and programming language to use for introductory programming courses has been widely debated. However, there is no consensus yet, as studies have produced inconclusive results and each approach has its own advantages and disadvantages. (Koulori et al., 2014).

As of today, many programming tools are available and in use in the different classroom settings. Tikva and Tambouris (2021) have found that many educators utilize web-based simulation creation tools, media tools based on educational robots, graphical block-based programming tools and text code programming tools.

Different approaches and programming tools necessitate distinct requirements for lesson design. Consequently, the lesson design should align with the selected approach. In general terms, the lessons should:

- Instruct students in the process of program development
- Encourage problem-solving through the use of programming tools
- Facilitate a deeper understanding of programming concepts
- Promote the development of computational thinking skills

(Xu et al., 2023).

The programming tools employed in educational settings must possess certain characteristics to ensure effective teaching. Specifically, these tools should:

- Be readily accessible to both students and educators.
- Provide adequate support to assist in resolving potential issues.
- Feature an intuitive syntax that facilitates the efficient creation of applications and codes or the resolution of practical problems within a reasonable timeframe.

(Repenning et al., 2010).

Xu et al. (2022) conducted a meta-analysis of the effectiveness of different programming approaches in promoting computational thinking skills, and programming skills. They considered five different teaching methods and four kinds of programming tools.

The following programming approaches and tools were considered:

Programming Approaches:

- collaborative programming
- project-based programming
- game programming
- problem-based programming
- scaffolding programming

Programming tools:

- code text
- graphical block
- educational robot
- unplugged programming

Xu et al. (2022)

The study revealed that all teaching methods examined influenced the development of programming skills and computational thinking abilities, with the greatest effects observed in scaffolding programming and problem-based programming. Similarly, all programming tools evaluated had an impact, with educational robot programming and graphical block programming showing the most significant effect on cultivating programming skills and computational thinking abilities (Xu et al., 2022).

For the development of the teaching sequence being analyzed, an attempt was made to combine the approaches considered in Xu et al.'s work. Instead of traditional collaborative programming, where two or more programmers work simultaneously on a task, pair programming was chosen as a more guided and didactically innovative approach. Current research has found promising results by utilizing pair programming as a teaching method. These findings, along with an introduction to the concept of pair programming, will be described in detail in the following section.

2.3 Pair Programming

The concept of pair programming has gained significant recognition in educational settings, being regarded as an effective method for learning programming. Additionally, it aids students in acquiring various other essential 21st-century skills (Isaaee et al., 2022). The subsequent chapter will examine the fundamental principles of pair programming and its integration within the educational context.

2.3.1 Background and Definition of Pair Programming

Pair programming is a programming technique where two programmers work side-by-side at one computer on the same program, code, or design. Whereby, both programming partners have inherited certain roles – namely the driver and the navigator. The driver is the programmer in control of the mouse, and the keyboard, and is actively involved in the implementation of the program, the code, the algorithm, or whatever task they are engaged with. The navigator, on the other hand, is the one who observes the driver’s work. They have to continuously monitor the process to identify possible errors and mistakes that might occur. Those can range from syntactic and spelling mistakes to strategic ones. Whenever a challenging problem occurs in the course of programming, the programmers can brainstorm about how to solve this problem. After a pre-defined time, the two programmers switch roles. This leads to the fact that their contributions to the process and product are equal. (Williams, 2001).

Pair Programming is by no means a new concept. Rather it was and still is widely deployed in the industry. As early as the 1990s, this technique has been described in different works. It is said that with pair programming, programmers produce qualitative better code in a shorter amount of time. Moreover, it increases the performance and productivity of the programmers and aids their motivation and enjoyment (Williams, 2001).

One of the first attempts to transfer pair programming from the industrial sector into educational contexts was made in 1999. In a university course about Software Engineering in Utah, students could take part in an experiment that studied the costs and benefits of pair programming in educational contexts. Those who wanted to participate in the experiment were grouped with academically equivalent partners. The others had to solve the tasks on their own. It resulted in the fact that the code of the pairs was of higher quality in almost the same amount of time as the individuals needed. (Williams, 2001).

After the first reported usage at the University of Utah in 1999, many educators in the United States, the United Kingdom, New Zealand, India, and Thailand made their attempts to integrate pair programming into the classroom (Hanks et al., 2011).

In the following chapter, the integration of pair programming in the educational context and its advantages and pitfalls will be described.

2.3.2 Pair Programming in an Educational Context

Especially teachers of introductory programming courses make use of pair programming techniques. This is because, there is evidence that learning to program in this way, helps them to better understand and consequently to better learn to program. Although this use-case is widely used, there are also instances of higher education courses that use pair programming. (Hanks et al., 2011).

Different studies have concluded that students of introductory and intermediate programming courses who learned to program in pair programming mode were more likely to be successful. (Hanks et al., 2011). Umpathy and Ritzhaupt (2017) conducted a meta-analysis about the effectiveness of pair programming in computer programming courses. They reviewed 18 different articles that were concerned with this topic. They found out that pair programming if implemented properly, has a positive effect on the grades, the exam scores, and the persistence in computer programming courses compared to solo programming.

Pair programming not only enhances the retention and success rates of introductory courses but also offers significant pedagogical advantages. It creates a conducive environment for active and collaborative learning. Furthermore, pair programming alleviates students' frustration with challenging tasks by allowing them to depend on their partners. It also boosts students' self-confidence, interest, and enthusiasm in learning programming (Berenson et al., 2004).

The interaction with their peers and the continuous communication with each other create a more inclusive and supportive teaching and learning environment. This is not only beneficial to the enjoyment and motivation of the students, but it also aids in the development of generic skills such as collaboration, teamwork, and communication. (Williams et al, 2008).

Furthermore, pair programming is also beneficial for the teacher. Less support for the individual student is needed, as many low-level and comprehension problems can be solved in pair mode without the intervention of the teacher. In addition to that, students who would normally be hard to motivate might engage more in the tasks because of the pressure of their team partner and contribute more to the final team product (Williams et al., 2008).

While pair programming offers notable benefits in an educational context, certain considerations must be addressed to ensure its successful implementation. Firstly, it should be acknowledged that some students may prefer to work independently and thus require additional motivation or alternative assignments. Secondly, educators must allocate sufficient time for the preparation process to establish an effective environment. Lastly, continuous monitoring by the teacher is essential to ensure that the process operates smoothly (Williams et al., 2008).

As mentioned before, pair programming is only successful if implemented with care. Williams et al. (2008) proposed eleven guidelines that should be considered when implementing pair programming in the classroom:

1. Students need to become familiar with the setting and method and clear instruction of the tasks and the roles is necessary
2. The teacher needs to be aware of their new role that steps away from answering and teaching to monitoring and proactively guiding.
3. Regular attendance of both partners and suitable sanctions is mandatory to ensure that every partner participates in the process
4. Have students give feedback about their partner and the process
5. Evaluate on a balance of individual and collaborative work
6. Assign pairs that will most likely work well together

7. If pair programming is done over the whole course, ensure that students work with different partners
8. If any problem with the partner occurs, students must report this immediately to the teacher to ensure correction of the situation
9. Pairs should be able to comfortably sit next to each other and access to the mouse, monitor and keyboard needs to be as easy as possible
10. The pairs should work together towards a common goal
11. Encourage students to solve problems on their own rather than provide them with the answers right away.

(Williams et al., 2008)

To sum up, one can say that implementing a pair programming approach for an introductory programming course seems promising, as the advantages such as increased knowledge, higher test scores, improved skills, attitudes and self-efficacy, higher class participation and fewer dropout rates outweigh the efforts one has to make before implementing this practice (Papadakis, 2018).

The following chapter will examine two best-practice examples of how pair programming can be implemented in the classroom.

2.3.3 Best Practices of Teaching Programming Skills with Pair Programming

2.3.3.1 Block-based Programming with Pair Programming

In Serbia, an initiative was undertaken to incorporate pair programming into primary school education. This was achieved through the use of Scratch, a block-based programming software. Scratch is a complimentary graphical programming platform primarily designed for students aged 8 to 16, though it is not confined to this age range and is extensively utilized by beginners of all ages. Students can create programs by manipulating graphical blocks via a drag-and-drop interface. These blocks are color-coded and organized into categories, with each color representing a distinct functionality. This design facilitates visual differentiation and comprehension. The intuitive system aids users in following the logical flow of block sequences and reduces errors that would typically be detected by a compiler in a text-based programming language (Momcilovic-Iskrenovic, 2019).

For this study, the researcher divided the students into two groups. One group did the coding assignment independently and the other did the assignment in pair programming mode. As the main interest of this Master's thesis is on pair programming, only the second group will be described here. For the formation of the pairs, the researchers randomly assigned students or recognized students' wishes. Before the students were asked to complete the tasks, they received an introduction and a demonstration on how to handle Scratch. Then they had to solve specific tasks. After the students completed the tasks, they had to take an online knowledge test in which they were asked questions at different stages of knowledge based on Bloom's taxonomy. (Momcilovic-Iskrenovic, 2019).

The findings of this research indicate that students who participated in pair programming demonstrated greater motivation to acquire new knowledge, exhibited faster retention of new concepts, and effectively learned the programming language. The study concluded that students achieved more substantial and efficient results when working in pairs than those who programmed independently (Momcilovic-Iskrenovic, 2019).

2.3.3.2 Text-based Programming with Pair-Programming

In 2015, a university in the United States experimented to determine whether pair programming or solo programming is more beneficial for undergraduate students learning a text-based programming language. The experiment was carried out in three phases: an initial pilot study followed by two successive pair programming experiments (Dongo et al., 2016).

During the pilot phase, four randomly matched pairs and three solo programmers were tasked with completing a previously unknown assignment. All participants received a brief introduction to the concept of pair programming before starting their tasks. (Dongo et al., 2016).

In the second phase, additional students participated in the study. They were provided with an information sheet about pair programming before the experiment day. Participants were again randomly assigned to pairs. Before receiving their tasks, they watched a short video explaining the task, the pair programming concept, and data protection guidelines. Participants were also given a copy of "Do's and Don'ts" in pair programming along with the task instructions. They were allowed to choose their initial roles and were required to switch roles after 20 minutes. The third phase of the experiment followed the same procedure as the second phase (Dongo et al., 2016).

The outcome of this research was comparable to the study with Scratch. Students working in pairs engaged more in the process of creating a program than those who programmed alone. Moreover, the pairs collaborated effectively with each other. The students in pairs exchanged ideas with their partners and gained knowledge from one another. Additionally, the overall performance was higher in pair programming compared to the students who programmed individually. The code quality was superior and the students in pair mode demonstrated greater productivity (Dongo et al., 2016)

2.3.3.3 Differences to consider – Pair programming with visual programming vs text-based programming

Korber and Motschnig (2021) examined the differences to consider when employing pair programming in introductory programming courses with visual and text-based languages. They found that, regardless of whether students worked with a visual programming language or a text-based language, they performed better when working in pairs. Additionally, students were able to follow the tutorial more easily and found completing the tasks more enjoyable. This effect was even more pronounced with text-based programming languages. However, it is notable that students believed they learned more through solo programming than through pair programming (Korber & Motschnig, 2021).

Although pair programming with both language types is seen to be beneficial, there are some differences Korber & Motschnig (2021) have found. High-performance students saw little to no differences between text-based and visual programming languages, whereas low-performance students benefited more from pair programming with text-based languages than from pair programming with visual programming languages. The same holds for the motivational aspect. Students tend to profit more from the positive motivational benefits of pair programming with text-based languages. Again, this is especially dominant with students who struggle with learning to program.

Another notable difference between pair programming in visual and text-based environments is related to problem-solving approaches. Korber and Motschnig (2021) observed that certain students find it easier to systematically structure and organize their text-based code when collaborating in pairs. The inherent high level of structure provided by visual programming languages makes this positive effect more significant in the context of text-based languages (Korber & Motschnig, 2021).

2.4 Transfer from visual programming languages to text-based programming languages

In many educational environments, students begin their programming education by learning to use a visual programming language, commonly starting with Scratch. Subsequently, students often face the challenge of transferring the knowledge and skills obtained from Scratch to a text-based programming language. Frequently, the target language is Python (Skaarseth, 2023).

The objective of learning multiple programming languages is to provide students with a comprehensive and multi-faceted understanding of programming concepts. Additionally, in practical applications, programmers rarely depend on a single language. Instead, they must acquire proficiency in various languages and be able to transfer knowledge from one language to another. This ability to transfer skills is closely tied to self-directed learning and will be advantageous for their future professional careers (Skaarseth, 2023).

As previously mentioned, the typical approach for introducing programming to students involves starting with a visual programming language such as Scratch and subsequently transitioning to a text-based programming language such as Python. The rationale behind this progression is that Python is widely regarded as a "real" programming language, and students often perceive a need to learn a language that is utilized in practical applications, which differs from visual block-based languages. Furthermore, Python holds significant relevance in professional programming environments and is one of the most extensively used languages in practice. Additionally, Python is relatively user-friendly compared to other textual programming languages such as C++ or Java, due to its simplicity and readability (Skaarseth, 2023).

However, although, this is a common practice, students often struggle with transferring concepts from one language to another or do not see the connection between the two languages (Tshukudu & Jensen, 2020). Considering this problem, it is especially important to plan the transfer from one programming language to another, to avoid overwhelming the students and ensure a smooth transition. Tshukudu and Jensen (2020) proposed a model called MPLT that gives a framework for the implementation of this aim. First of all, it needs to be considered that there are three levels of programming language knowledge: the syntax level (how to write the code in the target programming language), the semantic level (how does the language work, how to give meaning to the syntax), and the concept level (knowledge about the concepts underlying programming languages). The teacher provides input on all the different knowledge levels, emphasizes similarities and differences between the two programming languages, and corrects the students' understanding when necessary (Tshukudu and Jensen, 2020).

If one utilizes the MPLT plan proposed by Tshukudu, it will be more likely that the transfer from one language to another will be successful and more effective as the likelihood of misunderstandings and false assumptions in the transfer process is minimized by the error correction the teacher does (Skaarseth, 2023).

As proposed by Skaarseth (2023), it is important to analyze the two programming languages based on their similarities, differences, and possible pitfalls students might encounter when transferring their knowledge from one language to another. The most obvious difference between the two languages Scratch and Python, is their visualization. While Scratch is purely block-based and event-driven, Python is text-based and sequential (Skaarseth, 2023).

With Scratch, the programmer has a pre-defined block repertoire and “writes” the code by simply connecting those pre-defined blocks by drag-and-drop with each other. While connecting the blocks, the user interface gives feedback, if the blocks are connected in a way that they function and make sense. This mechanism ensures that the programmer can hardly make any mistakes. Although there is the possibility to create customized blocks in Scratch, the majority of tasks can be solved with the pre-defined blocks. The programmer can create code fragments for the different objects. Those code snippets and the objects are independent of each other and do not need to be initialized from the very beginning. The visuals that are controlled by the code, the code itself and the execution are always visible next to each other (Skaarseth, 2023).

With Python, the programmer cannot choose from a pre-defined set of blocks. Rather, they have to create the code all by themselves, line by line, character by character. This code will be executed in sequential order by an interpreter. The programmer has to learn the syntax and semantics of the underlying concepts and how they are represented in the programming language Python. In comparison with Scratch, the programmer is not protected from making fatal syntax or semantic mistakes. Moreover, it does not have a visual feedback tool like Scratch and executes the code only after it has been written in a separate text-based interface (Skaarseth, 2023).

However, there are also some similarities between the two languages, namely that the representation of the underlying programming concepts such as operators, loops or conditionals is very similar, although they are presented in different environments. Moreover, words of the natural language are used in both cases (Skaarseth, 2023).

Skaarseth (2023), who analyzed how the transfer from Scratch to Python can work in practice, concluded that it is possible to utilize the MPLT, designed by Tshukudu and Jensen (2020), for the transfer from Scratch to Python, if certain prerequisites are met. Those are that, the students need to have sufficient experience and understanding of the first programming language – Scratch – in all levels of understanding, that the transfer process is communicated and made visible, and that the successful transfer to the new programming language needs to be measured (Skaarseth, 2023).

2.5 AI in Education

As Artificial Intelligence becomes more and more dominant in our day-to-day lives and is a topic of current public and scientific interest, it is no wonder that AI also yields potential for changing the way we learn and teach.

Statistik Austria researched the spread of AI use in Austrian companies in 2023 and found out that more and more companies rely on AI for their day-to-day work. In 2023 11% of all Austrian companies used AI in their work. This is an increase of 2% in only two years. Especially in companies that operate in the information and communication sector, AI usage is highly dominant. The most common use case for AI in companies is for text recognition and text production, followed by automatized data analysis, the automatization of procedures, and the creation of decision-making tools (Statistik Austria, 2023).

As educators, we are supposed to prepare the students for their future lives. The fact that more and more companies rely on artificial intelligence, provides a demand for integrating AI in the education of the future workforce as well, in such that they can use AI effectively, sustainably, reasonably, and critically. It is no secret, that AI has been widely used in teaching and learning. In a survey, conducted by Forbes Advisor, more than 50% of the participating teachers in the USA believed that using AI in the classroom has improved educational outcomes (Hamilton, 2024).

The Austrian Federal Ministry for Education, Science, and Research has also recognized the importance of integrating AI in the educational field. On the one hand, teachers are asked to integrate the topic of AI in their day-to-day teaching and on the other hand to use AI technology for creating a more inclusive and individualized learning experience for their students. (Bildungsministerium für Bildung, Wissenschaft und Forschung, 2023).

Some examples on the side of individualization and differentiation are:

- Drafting of personalized and individualized interactive tasks to meet the needs of the learners regarding difficulty levels, special needs, or interest fields.
- Creating graded texts in language classes to meet the specific demands and promote and encourage the individual learners.
- Using AI as a feedback tool for learners, such that they can receive feedback on their produced texts or solved tasks to become more proficient.
- Using AI for time and resource management and, thus, providing the possibility of establishing new feedback and assessment techniques in the classroom such as personal feedback, critical discourse, etc.
- Acquiring new competencies that go beyond traditional education such as the ability to formulate concise questions and instructions, or using provided information interdisciplinary.

(Bildungsministerium für Bildung, Wissenschaft und Forschung, 2023).

Besides the application examples for students and teachers in the day-to-day teaching, there are also possibilities for how to use AI for administrative and preparation work for the teachers. These include:

- Gaining insight into a topic or narrowing down a topic in such that it becomes suitable for the educational context.
- Assistance in the brainstorming process when drafting a teaching sequence.
- Assistance in the quality control of the creation of tasks, and tests, or in drafting individualized and differentiated versions of those.
- Creation of multimedia data such as pictures, videos, or audio that can be used for teaching.
- Facilitating the communication with parents, especially if there are language barriers.

(Bildungsministerium für Bildung, Wissenschaft und Forschung, 2023)

Considering this, one can assume that the way of teaching and learning will drastically change in the near future or has already changed in some areas.

However, it should be noted that AI and its use in educational contexts is not a new phenomenon, rather there have been attempts to use AI in education as early as the 1960s. The US Department of Defense developed systems that should assist human abilities (Chassignol et al., 2018). In the beginning, however, the spread and the tasks, such systems could do, were limited (Tahiru, 2021). Over the years, more and more research was done in this field and more and more different systems came into existence. Today, AI revolutionizes teaching and learning and facilitates those (Tahiru, 2021).

In the following chapter, it will be described how AI can be used in schools to facilitate teaching and learning and, how it can assist in acquiring programming skills.

2.5.1 Overview of the use of AI-Tools for Educational Contexts

As the short overview of the recommendations of the Austrian Federal Ministery of Education, Science and Research, presented in the previous chapter, has shown, there are multiple areas in the educational context, in which AI can prove helpful. In this chapter. A short overview of these application scenarios will be given.

Generally, it should be noted that the use of AI for educational purposes goes far beyond teaching itself. Tahiru (2021) divides the use cases into three broader types:

- Automation of Administrative Tasks
- Smart Content
- Intelligent Tutoring Systems

The first one is concerned with using AI for doing reoccurring and repetitive administrative tasks to save time. Those tasks include for example grading and assessing exams and homework, providing valuable feedback, or processing the admission of new students into a school (Johnson, 2019).

The second type of AI usage in educational contexts includes the use of AI for condensing textbook units into teaching and testing materials. For example, AI can be used to draft true or false statements of the unit's content to check, whether the students have understood the content they have learned. Moreover, it can be used for summarizing chapters or learning sequences, drafting flashcards, mock exams, or additional learning material. Finally, it can be used for drafting multimedia input to assist understanding (Johnson, 2019).

The last category is concerned with using AI for self-tutoring. Content that has not or hardly not been understood in the lessons, can be re-explained or re-formulated by the AI with the goal of creating a personalized tutoring system for students. (Tahiru, 2021). There are AI tools such as the "Mike" Software by Carniege Learning that are solely concerned with this goal (Singh et al., 2018). Another example is ReCo, a feedback tool used for Mathematics. The students can scan their solution and its solution process and the AI tool gives feedback if the solution and its solution process is correct or how it should be changed in order to be correct (Zehner, 2019).

Moreover, AI can be used for diagnosing and analyzing the behavior of students regarding a specific task or the overall behavior in schools. If the AI recognizes the undesired behavior of the students, the teacher can adapt their teaching or the teaching and learning environment. Therefore, it ensures the maximization of the student's well-being and improves the teaching and learning environment (Zehner, 2019).

However, AI is not only beneficial for teachers and learners. Various stakeholders can benefit from using AI for educational purposes. Duggan (2020) lists four stakeholders and their potential use of AI:

- **Students**: using AI for individual and personalized support, guidance, or practice regardless of their ages, achievement levels, or socio-economic backgrounds
- **Educators**: using AI for administration, individualization and differentiation of teaching materials, for effective and time-saving work, collaboration, professional development and self-reflection.
- **Parents**: using AI for involvement and engagement in their children's education and for reinforcing values.
- **School leaders**: using AI for administration, management, control, and communication and community purposes-
- **Local, Regional and National Administration**: using AI for processing information, collecting data, evaluating existing curricula, and resource planning

(Duggan, 2020).

Considering all those possible use cases of AI Tools, one should also consider that there are benefits as well as challenges when one decides to use AI Tools for educational contexts. The following chapter is concerned with these.

2.5.2 Benefits and Pitfalls of AI-Tools in Educational Contexts

Some of the benefits of using AI-Tools in Educational Contexts have already been mentioned in previous chapters. For overview reasons, they will be summed up here as well. As this paper is concerned with creating a teaching scenario that uses AI, this list will be limited to beneficial aspects that are directly involved in day-to-day teaching and learning:

- AI contributes to speeding up processes at the teacher's side such as administration, grading, creating learning materials, or drafting feedback.
- AI improves teaching and learning by providing individualized, differentiated, and personalized teaching and learning materials.
- AI can be used as an intelligent tutoring system by the students to recap and practice the content taught in classes.
- AI can be used to practice for examinations and self-assessment.
- AI can be used as a teaching and learning resource in the classroom.

(Johnson, 2019; Tahiru, 2021; Singh et al., 2018; Zehner, 2019 and Duggan, 2020).

However, although, the benefits are promising, there are also some challenges one should consider when implementing AI in the classroom.

Bailey (2023) lists six challenges that might come into play when using AI, especially generative AI, in the classroom:

1. The most obvious risk is that students might not only use AI for teaching and learning, but they might also use it for solving their homework or for cheating purposes. Especially in subjects, where texts are produced or with highly standardized tasks such as basic calculations, AI can easily be used for solving those tasks. This might lead to the fact that students do not understand the topics, and are then not learning and acquiring necessary skills. Therefore, it is important to overthink traditional tasks that require learning by heart or repeating things that have been learned in school. (Bailey, 2023).
2. Secondly, it should be considered that AI tools can have biases in their algorithms. These biases can range from ethnic, gender, or socioeconomic bias, to political biases or pedagogical philosophy biases. The responses or tasks generated by the AI systems might then generate unwanted output or output that is not in concordance with school or community values. Therefore, students need to have a basic understanding of the ethical or moral problems of AI and should be trained in critical usage. (Bailey, 2023).
3. Thirdly, with some AI tools privacy concerns play a role, as they might store data produced by the students or teachers to learn, posing the problem that confidential or private data is

provided to the public. Therefore, teachers must ensure that students do not share this information with AI systems. (Bailey, 2023).

4. Fourthly, the problem of decreased social contact might become even more of a problem, if students rely too much on AI for learning and neglect their fellow students or educators increasing mental health challenges and social isolation. Therefore, it is important to consciously include as much student-student interaction as well as teacher-student interaction in the classroom. (Bailey, 2023).
5. Fifthly, if AI is extensively used, it might lead to the fact that both teachers and students over-rely on AI technology without critically examining the output as well as the purpose of using it. Therefore, content produced by AI systems must receive a quality check before using it in the classroom. (Bailey, 2023).
6. Finally, there might also be the case that not all students have equal access to computer devices or AI tools, which leads to possible exclusion mechanisms based on socioeconomic differences. Therefore, the teacher needs to ensure that all students have the same possibility of participating in the classroom (Bailey, 2023).

Duggan (2020) also lists the problem of technology dependence. This means that without access to technology or a working technology, the use of AI is limited.

Moreover, students and teachers need to become aware of the fact that AI can produce wrong, imprecise, problematic, or plagiarized information and data (Mader, 2024).

All these challenges have in common, that they can be overcome if the use of AI is profoundly planned. AI must always be critically used and requires the teacher to professionally decide how and when to integrate it into their teaching. The role of the educator, therefore, will not be outweighed by AI. Rather it will be enhanced (Duggan, 2020).

To keep track of technological and pedagogical developments, it is not advisable to neglect AI altogether. Especially in subjects such as Computer Science, it is of great importance to learn about AI tools and to effectively use them for teaching and learning purposes. Therefore, the following chapter is concerned with the question of how to implement the topic of AI in Computer Science classes.

2.6 AI in Computer Science Classes

Considering the current curriculum for basic digital education and computer science of the lower and upper secondary school AHS, one finds several instances of AI. In the third grade of the lower secondary school, it is said that students should be able to describe how AI is used in software and other physical systems. In the fourth grade, students should be able to reflect on the possibilities and limitations of AI. And in upper secondary students should be able to describe areas in which AI systems are in use, explain and compare the difference between human and artificial intelligence and use AI systems. (Bundesministerium für Bildung, Wissenschaft und Forschung, 2024).

Therefore, the Computer Science classes prove especially designated to cover the correct handling of AI tools. Thereby, it is not only relevant to teach knowledge about how AI systems work, but also how to effectively use them. It is of even greater importance to equip the students with competencies such as critical thinking, question formulation, dialogic communication, and the ability to use knowledge created by AI. This springs from the fact that factual knowledge will lose its importance as it can be generated by AI more efficiently and in a shorter amount of time. Teachers of Computer Science classes should, thus, create a learning environment, in which students can develop those competencies, and in which they can critically reflect and grow (Fitzpatrick et al., 2023).

To ensure that students effectively acquire the competencies mentioned, the teacher is asked to gradually check the progress and to reflect on the learning process together with the students. It is of great importance to equip students with the ability of critical thinking, and to promote critical reflection of the content created by AI. Moreover, teachers should ensure that students apply their newly acquired knowledge and competencies in real-life situations (Fitzpatrick et al., 2023).

As the field of AI is rather broad (as seen in previous chapters), this Master's thesis focuses solely on generative AI Tools. In the following chapter some use case scenarios of generative AI Tools for Computer Science classes will be presented.

2.6.1 Generative AI-Tools in Computer Science Classes

In her Master's thesis, Michaela Mader has created three teaching scenarios, in which generative AI Tools are used in Computer Science classes. The first scenario is concerned with developing research competence and critically evaluate the output created by ChatGPT. The students should think about a topic of their interest and do research about it. First, they had to formulate a suitable research question. Then they should do a traditional internet search about this research question and note down the sources they have used. After that, they should use ChatGPT and ask it to answer their research question and compare it with their search results. Finally, they should reflect on their experience and on what they have learnt (Mader, 2024).

The second scenario is concerned with learning about AI. In the beginning, the students were asked to complete a quiz about AI to check their prior knowledge. Then the students were asked to teach themselves as much about AI as possible. One group received a guideline and was asked to answer some guiding questions about AI, the other group was free in their acquisition. Finally, they had to complete a quiz to check their knowledge gain (Mader, 2024).

The third teaching scenario was concerned with a programming task. Students were asked to program a task in Python and ChatGPT. After that they were asked to complete a quiz, to check, whether their programming knowledge has increased (Mader, 2024).

Lau & Guo (2023) developed the idea to integrate generative AI in the teaching method Think-Pair-Share. The students try to solve a task (e.g. a programming problem) on their own, and then prompt an AI to solve it. After that the students compare their human solution with the AI solution and share their experience in plenary format with the class.

As this Master's thesis is concerned with using AI for teaching and learning programming skills, use cases on how to integrate generative AI Tools in this specific field, will be described in the following section.

2.6.2 Learning to Program with generative AI-Tools

Generative AI Tools, such as ChatGPT, are especially useful for the acquisition of programming skills. This springs from the fact, that ChatGPT-3, the model which sparked the rapid spread of this tool, was trained with an enormous amount of code and human speech, in such that it should promote programming among many people. With ChatGPT-4 it is possible to generate code in Python, JavaScript, or Ruby. Therefore, ChatGPT can be used to teach basic programming skills in these languages to students without or with little prior knowledge in this field. Advanced students can use ChatGPT to speed up their programming process and create more efficient code. Moreover, ChatGPT can be used to translate task or problem descriptions from human language into code or to debug and optimize code written by the students. Finally, ChatGPT can assess code and give feedback. (Fitzpatrick et al., 2023).

Moreover, generative AI can be used to explain or teach a specific programming concept. It can either theoretically explain it or produce code that illustrates the usage of these concepts in code snippets. The explanation can, thus, happen in contexts that the students have directly specified in their prompts (Lau & Guo, 2023).

As it is common ground, that model tasks can assist in deeper understanding of programming concepts, generative AI can also be used to draft such model tasks for programming activities. They can even consider various versions of solutions and, therefore, understand that there are multiple ways to solve a programming task (Becker et al., 2023).

Therefore, one can assume that ChatGPT has a high potential for promoting programming skills in the Computer Science classroom.

2.6.3 Generative AI-Tools as Pair-Programming-Partners

In the current research, the effects of using artificial intelligence as a pair programming partner have become increasingly interesting for many researchers. In the meantime, even a specific term – pAIr-programming – has been designed to summarize research that is concerned with this topic. The term pAIr programming means that instead of two people working together on a programming task, it is one human and one generative AI Tool that is based on a large language model such as ChatGPT that work together on a programming task (Ma et al., 2023).

Spinellis (2024) summarizes the benefits, pitfalls, and hurdles and how to overcome the negative sides. He argues that generative AI receives its power from the rapid development and the huge amount of data with which the corpora are trained. Generative AI can, thus, generate code faster and in areas in which we lack knowledge. Moreover, generative AI is also a good tool for translating code from one programming language to another or for transforming data into executable code. Additionally, AI can help debug when provided with code and its related error message and improve the code. Finally, AI can be used as a programming language assistant or a tutor. (Spinellis, 2024).

However, there are also some downsides to using AI for programming purposes. First and foremost, AI can produce faulty code. Commonly, this springs from the so-called hallucinations when AI comes up with things that are wrong or non-existent. Moreover, it can produce erroneous code. This is not a big problem if the code does not run altogether, but it might become a problem if it does, and we rely on it, producing the wrong output. (Spinellis, 2024). Nevertheless, AI is a powerful tool and can be an assistive technology when learning to program.

Górecki (2023) researched how ChatGPT can be used in pair programming. His work focused on the benefits of having ChatGPT as the pair programming partner, who occupies the driver's role, while the human acts as the navigator. He discovered that this approach leads to enhanced productivity, improved code quality, knowledge sharing, better code documentation, and accessibility (Górecki, 2023).

The enhanced productivity aspect is based on the fact that ChatGPT can be used to generate repetitive yet time-consuming programming tasks. This helps the developer to free time and, thus, to the possibility of focusing on more demanding and creative work in the meantime. (Górecki, 2023).

The codes become better in such that ChatGPT can detect errors and bugs in the code before they manifest themselves into bigger problems. Moreover, ChatGPT can rewrite code and suggest improvements to existing code and design. (Górecki, 2023).

ChatGPT can also be used as a superior developer, as it can assist novice programmers in understanding more complex code. It can share explanations and act as a more experienced team member, providing suggestions and guidance (Górecki, 2023).

Moreover, the code will be better documented as ChatGPT can help to create more detailed and accurate code documentation by generating comments and annotations based on the code (Górecki, 2023).

Finally, ChatGPT allows people without a lot of programming knowledge to step into programming themselves and collaborate with developers and other stakeholders to contribute in projects that require programming in a more meaningful way. (Górecki, 2023).

Ma et al. (2023) researched whether there are differences between two humans working together on a programming task and the pAIr programming approach. They raised the questions of whether the two approaches are comparable, whether there are different application contexts, whether performance differs, and how the communication and interaction should vary in the two approaches. Although some research has been done in this field already, they argue that there is still too little data to give well-founded and satisfactory answers to the questions raised. However, some tendencies can be seen in current research. First, it needs to be considered, that AI-driven tools such as Copilot or ChatGPT can still produce code that does not work or that does contain errors and bugs. Moreover, the programming and communication dynamics seen between two people performing programming tasks cannot be seen by pAIr programming. (Ma et al., 2023).

However, in educational contexts, using artificial intelligence as a pair programming partner seems rather promising. Using AI as a programming partner has led to similar productivity, similar code quality, and similar self-efficacy results as with two humans performing a pair programming activity. They also found that knowledge transfer is successful when learning to program with an AI pair programming partner. (Kuttal et al., 2021)

Ma et al. (2023) have found out that using AI as a pair programming partner seems most effective for the educational context, as it does not bring much benefit to the professional developer. Nevertheless, some considerations need to be made, before using AI as a pair programming partner for students.

Bird et al. (2023) argue that there needs to be a shift of competencies students need to learn. Rather than being able to produce code on their own, students should be trained in such that they can assess and review code when using it. Teachers need to support students in developing debugging and testing skills as the code produced by the AI can still lack correctness and completeness (Ma et al., 2023). Moreover, teachers need to promote competencies such as self-reflection, question-forming, and explanation among their students (Ma et al., 2023).

In the research discussed above, AI is mostly used as the driver who writes the code. Phung et al. (2023) have researched how AI works as a navigator when a student provides it with some line of code. In their research, they prompted an incomplete program written by a student and asked the AI agent to complete the program while making as few modifications to the student's code as possible. They found out that ChatGPT in version 3.5 and version 4 could both solve the problem and provide feedback to the student, whereas ChatGPT 4 made more adaptions to the original code. However, this result should be taken with a grain of salt, as the model GPT 4 was just released when this research was done and adaptions and updates of the version are currently happening inflationary.

In the teaching scenario under consideration, we will focus on both use cases – AI as the navigator and AI as the driver.

3 Practical Part

3.1 Research Methodology (Theoretical Background)

The practical part of this Master's thesis consists of the creation, execution, and evaluation of a teaching sequence that mediates programming skills and generic skills by using pair programming and generative AI tools. This teaching sequence will also be based on the approach of discovery-based learning. The implementation of this teaching sequence is methodologically accompanied by a case study.

The research method of case studies has been used in various research fields, ranging from natural sciences to humanities to social sciences. Especially, in educational research domains, case studies have been used to research modules, programs, classrooms, schools, and universities (Gerring, 2016).

Although case studies have been widely used, there is no agreement in the definition of this research method. Rather, there are various definitions dependent on the research domain.

The definition that will be used in this paper stems from Yin (1994), who defined a case study as "an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident…[and] relies on multiple sources of evidence".
For a case study, the first necessary constituent to have is a case. Gerring (2016) defines a case as "a spatially and temporally delimited phenomenon of theoretical significance". Therefore, a case study is concerned with a special case. It can range from a single case to a small set of cases. The case under consideration should then shed light on a larger amount of cases (Gerring, 2016).

As a case study is concerned with one specific case, it is a highly focused research method, where the researcher invests much time in analyzing and presenting the chosen case to provide important evidence for the argument, the central point of the study, it tries to make (Gerring, 2016).

One major aspect of case studies is that the researcher does not intentionally adjust or manipulate the case rather his or her task is to observe the ongoing phenomena. This leads to the goals of this research design, namely to explain the case under investigation and shed light on other cases. Therefore, it must be possible to put the case into a larger context (Gerring, 2016).

The first step when considering implementing a case study research design is to select and define the case. There are some strategies and criteria in the selection process, that determine whether a case is suitable for a case study:

1. The case must be of importance for a specific group of readers.
2. The case should be independent of other cases.
3. The case needs to provide new evidence to subjects
4. The case needs to be easily accessible to the researcher
5. The case needs to be representative of a larger population

(Gerring, 2016).

In Table 1, the criteria will be used to assess the case underlying this research based on its suitability for a case study:

Criterion	Evidence in current case
Importance	As the case observes the suitability of a specific teaching scenario for the acquisition of specific teaching goals, it can be assumed that the importance goes beyond the classroom, in which it was taught. It is of importance for all teachers teaching Computer Science or programming classes in Austria and beyond. Moreover, it might also be of interest to other researchers in the educational field or to authorities and policy-makers.
Independence	This case was conducted in one class in one school. The students did not take part in any other research and their learning outcomes did

	not influence their success in this class. Moreover, they did not engage in a second round of treatment.
New evidence	A lot of research has been done for teaching programming skills with various methods including using pair programming as a teaching method. The same holds for the teaching of generic skills and the usage of AI in the classroom. However, no research has been found that connects teaching generic skills and programming skills by using generative AI and pair programming. Therefore, it can be assumed that new evidence can contribute to this knowledge domain.
Accessibility	As the students who took part in this research, were taught by me throughout the whole school year, the access was given. The only obstacle that needed to be overcome was getting consent from their parents in such that they could participate in this study
Representative of a larger population	As it can be assumed that many teachers in other schools are faced with similar problems (how to mediate generic skills and programming skills), the outcomes of this case study will most likely be representative for a larger population size.

Table 1: Assesment of case for this research based on its suitability for a case study

The second step, when doing a case study is to analyze the case. There are qualitative and quantitative methods one can use for analyzing a case. Sometimes, they are even combined. Based on the outcomes of this analysis, conclusions are drawn. (Gerring, 2016). This research uses a combination of both. The analysis will be described in greater detail in chapter 3.2.

3.2 Research Design

To answer the research question a multi-step research design has been chosen. First, a literature review was conducted to assess the status of the current research. The literature review formed the basis for developing a teaching concept, which has then been carried out in the context of the case study of this Master's thesis.

The literature review, which has been discussed in great detail in Chapter 2 of this thesis, was necessary to gain insight into the current research in this field and helped to increase the teaching concept's quality. The teaching concept will be presented in the next chapter. The core of the teaching concept has been developed in the seminar "Seminar Computer Science Didactics" in the summer term of 2024. This teaching concept was adapted as part of this Master's thesis and is now based on the insights gained from the literature review. The teaching sequence was implemented in the 5th grade of a secondary school in Burgenland, Austria. The whole teaching sequence was the base for this case study. The case study method was used to gain in-depth insights into the learning processes and developments of the students. The following data collection methods were used:

- Observations: Systematic observations of the students during the lessons Collection of the learning products
- Students' reflections and self-assessments

A detailed description of each data collection method, their analysis, as well as changes to the original teaching sequence are described in Chapter 3.2.2. to 3.2.4.

3.2.1 Research subject

14 students attended the class in which the teaching concept was implemented. Eleven of them were female and three of them were male. All were aged between 14 and 15 years, They had all in common that they attended the same class and had all attended the same lessons in lower secondary levels. As the subject "digital basic education" has only been introduced as mandatory in lower secondary levels in the school year 2022/23 (Bundesministerium für Bildung, Wissenschaft und Forschung), they did not attend these classes before entering the upper secondary level. Nevertheless, they all attended a school-intern course called ELSA (e-Learning in schools), so they still had some basic prior knowledge regarding the handling of the visual-block-based programming software "Scratch".

After the whole teaching sequence had been taught, the students were asked for consent to use their products and the observations for this Master's thesis research. The original form of the consent in German can be found in the Appendix. The students and their parents gave written consent to use their products and the observations in anonymized form. The reason, why students were only informed about the research at the very end of the teaching sequence was to achieve the most unbiased result possible. If the students had known that their work and products would be used for research purposes right from the beginning, two scenarios would have been possible, that would have falsified the results: First, they might have worked harder or differently than usual, or second, they might have worked less intensively or with less motivation.

3.2.2 Teaching sequence

The teaching sequence developed for this research lasts 6 double lessons (100 minutes each), whereby the double lessons take place weekly. All lessons were held using the discovery-based learning approach. The reasons for that were manifold. First, discovery-based learning in Computer Science classes of secondary education has hardly been researched. Thus, this study should contribute to research in this field. Secondly, discovery-based learning is an approach that fosters student-centered learning, which is a key aspect of self-regulated learning (Banson, 2022). With self-regulated learning, students have to set themselves goals, develop strategies to achieve these goals, and reflect on their performance. (Banson, 2022). These skills are also relevant for developing generic skills.

The teaching sequence is divided into three sections. Each section consists of 2 double lessons.

1. Introduction to visual block-based programming
2. Transition to textual programming
3. Introduction to generative AI tools as pair-programming partners in textual programming

For overview reasons, detailed lesson plans as well as the materials used for each lesson can be found in the Appendix. The lessons are now presented in chronological order:

3.2.2.1 Lessons of the first section (Introduction to visual block-based programming)

As the students were already familiar with the visual block-based programming tool Scratch, there was no introduction to the user interface or the handling of the tool necessary. If the students had not been familiar with this tool, one or two additional lessons would have been needed to familiarize them with this tool.

The first double lesson, therefore, was mostly used for activating their prior knowledge, gaining access to their previously registered accounts, and familiarizing themselves with the tool. The students were allowed to play around with the tool and to test the functionality of the different buttons and the user interface. At the end of the lesson, the students collected all the different categories of activities they had found, in the plenum. Their findings were then grouped to form the basis for introducing the basic programming concepts of loops, variables, and conditions. The concepts were explained based on their functionality and representation in Scratch and were revisited at the beginning of the second double lesson of the first section.

The second double lesson of the first section, which took place one week later, started with a short quiz about the introduced programming concepts. This was done to refresh the students' memory. After the quiz, the students were introduced to the concept of Pair-Programming. They were told about the two roles "Driver" and "Navigator" and received an information sheet about the roles as well as the Do's and Don'ts of pair programming.

After the introduction, the students were asked to form pairs on their own. They were asked to pay attention that both partners should approximately have the same knowledge level. This was feasible as the students knew each other very well for a long time and were good at self-assessing their skills. If this had not been the case, it would probably have been better for the teacher to assign the pairs. The students were told that the goal of this double lesson was to develop a simple game in Scratch. The reason for choosing games was to enhance students' motivation. As it can be assumed that games play an integral part in the students' free time, it can be assumed, that developing something that is positively connotated increases student motivation. This assumption has also been supported by research. Yuriniati et al. (2019) found a positive correlation between motivation and the quality of learning outcomes. This points to the fact that the more motivated students are the better they will perform in completing their tasks (Yuriniati et al. (2019).

The development of games within an educational context is closely related to constructivist approaches. It has been shown that this significantly enhances student motivation and improves learning outcomes. Constructivist learning environments emphasize active participation, collaboration, and the construction of knowledge through experience, which aligns well with game-based learning strategies. Research by Maheshwari and Thomas (2017) indicates that when students engage in constructivist teaching methodologies, they exhibit higher levels of motivation and achieve better

There were seven different games the students could choose from. As the students did not like to choose the games by themselves, the games were randomly assigned among the students. The game description was issued (see game description in the Appendix). There were some hints on the instruction sheet that could prove helpful to the students. No further instruction was given and the students were asked to develop the game on their own. This was done to adhere to the principle of discovery-based learning, where the students should develop own problem solving strategies and learn new things from that. In case, the task could not be solved at all, there was also a sample solution provided at the students' learning platform, but the students were asked not to copy that while programming.

After giving the instruction, the teacher stepped back and let the students work. The teacher observed the lesson and the time and only served as a moderator in this lesson by telling when to change roles. Other than that, the teacher did not intervene in creating the games. The students had to present their games and upload the link to their games to the learning platform in use.

3.2.2.2 Lessons of the second section (Transition to text-based programming)

The first lesson of the second section was dominated by the transition from visual-block-based programming to text-based programming. As the students had not yet finished creating the mini-games, they continued with their programming in the first 30 minutes of the lesson. After that they had time for debugging, presenting, and playing the games. The students were allowed to go around and play their colleagues' games, too. The second part of the first double lesson was then devoted to introducing text-based programming languages. First, the students had to research the purpose and distribution of programming languages. Then the teacher introduced the programming language Python by showing them a sample program written in Python. As no advanced functionalities were needed, the teacher decided to use the online integrated development environment (IDE) online-python.com, as it is very intuitive to use and no long installation and introduction is needed. The students then had to analyze this sample program by comparing the commands with those used in Scratch. The teacher then summarized the findings on the whiteboard, introduced further important commands, and grouped them based on their functionality.

In the second double lesson of the second section, the main focus was on learning how to program with Python. The objectives of this lesson were that the students should become familiar with the syntax and semantics of the programming language Python and that they should be able to write their own simple programs in Python. As the didactic principle of this teaching sequence is discovery-based learning, the teacher again refrained from lecturing, rather the students should discover the syntax and semantics by playing around with the online tool “Silent Teacher”. The teacher asked the students to open the following website: https://silentteacher.toxicode.fr/?theme=basic_python and work through the activities.

While working with this tool, the students learned about variables, variable types, Boolean operators, conditions, recursions, functions, and loops. When everyone had completed this tool, the teacher summarized what the students had discovered. For remembering purposes, the students were asked to fill out a distributed cheat sheet, where they should note the most important commands and structures, they had discovered while playing around.

After that, the students were asked to get together with the same partner they worked with while creating the games in Scratch. Then, they had to complete two out of three Python programming exercises in Pair Programming mode, whereby the first exercise was the easiest one and the difficulty of the exercises increased after each exercise. As with the Scratch tasks, the students were not given any precise instructions. They were told the aim of the program and given some small hints, that might be useful for completing the task. Otherwise, there were no instructions.

The students were asked to upload their programs to the learning platform. After discussing all three programs together in plenary format, they were given sample solutions to compare. As the third exercise was especially demanding and not a single group was able to solve this task, the program was then created together in plenary format with the input of the students.

3.2.2.3 Lessons of the third section (Introduction to generative AI tools as pair-programming partners in textual programming)

In these lessons, the focus was on introducing the students to the use of generative AI tools as pair programming partners. It should be noted that the students had been extensively introduced to the topic of AI, its functionality, advantages, disadvantages, and limitations, and to the handling of it, in previous lessons. Therefore, they had also created a ChatGPT account. If this had not been the case, one should invest in multiple lessons to introduce the students to the topic of AI before teaching the lessons in this section.

In the first double lesson of the third section, the students were asked to let ChatGPT program the programming tasks they had completed in the last lesson. After that, they should test if the program, provided by ChatGPT works. Then they should compare their program with the code written by ChatGPT, highlight differences and similarities, and rate the two codes based on predefined criteria. They had to fill in a sheet, where they took notes about the comparison and the rating and handed that in.

In the sixth and, thus, final lesson of this sequence, the class was divided into two groups. One half was assigned the role of the navigator and the other half the role of the driver. The half that was chosen as the navigator had the task of understanding the driver's code (ChatGPT), analyzing it, and, if necessary, adapting it, so that it is understandable for the navigator. The other half had the task of writing the code and having it analyzed and optimized by ChatGPT as the navigator. Afterward, both teams had to present their codes and answer reflection questions about the code and the process of working with ChatGPT as a programming partner (see reflection questions in the Appendix). At the end of this lesson, the students had to complete a quiz to determine whether they had gained programming skills or not.

3.2.3 Data Collection and Data Analysis Process

As this research should give insight into the suitability of this teaching approach for developing programming skills as well as generic skills, different data collection, and data analysis instruments were necessary.

To observe the progress in understanding and processing newly acquired information and skills, I decided to do a general observation of the students during the lessons. As the students were not informed about the observations during the lessons in the first place to ensure an unbiased behavior, I tried to keep the observations as discrete as possible. This means that I walked around the class, observed what the students were doing, walked back to the teacher's desk, took quick notes, and continued observing or teaching. After each of the lessons, I reflected on the impressions I got, evaluated the notes I took, reformulated or clarified those if necessary, and tried to interpret them. I continued this observation procedure until all lessons were completed. After the whole teaching sequence was over, I combined all notes into a document and read through the document multiple times. While reading, I realized that the notes I took over the lessons, were all rather similar. Thus, I realized that a thematic analysis of the notes would prove helpful to gain a greater understanding. While reading through the notes I tried to make sense of possible themes that would make sense for the analysis and came up with four broader categories. Those categories were:

1. questions asked by the students
2. student behavior
3. observable process progress
4. problems that occurred.

In the first round, I tried to assign the notes to those categories. I categorized per meaningful unit, meaning that I scanned the notes multiple times and broke them down into smaller units. Each unit holds only one piece of information at a time. If one sentence holds more information, it is broken down into multiple meaningful units. Splitting up the notes like this resulted in 74 meaningful units. The results of this analysis will be presented in Chapter 3.3.

To determine, if students had acquired programming skills, they were asked to hand in the final products of their programs, indicate specific programming concepts in their presentation, and answer a short quiz at the end of the teaching sequence. The programs, they handed in, were analyzed regarding the following aspects:

- correctness of syntax
- task fulfillment

If the respective category was met, the students received one point, if not, they received zero points.

In the presentation of the program, the students were also awarded points. If the students could indicate a loop, a variable, a condition, and a function in their program presentation, they also received one point for each correct indication.

The results of the final quiz, which can be found in the Appendix, also mounted to the final score of acquiring programming skills. In the final quiz, they had to indicate and explain programming concepts. For each correct answer, they received 1 point. The programming skills that were taught in the sequence were: concepts of programming – loops, conditions, recursions, functions, variables; Syntax of Scratch, Syntax of Python, Handling of an IDE, and Understanding of programming code.

All points from the program analysis, the presentation, and the final quiz were added up and added to an Excel sheet. At the end of the teaching sequence, each student had reached a final score. To give this score meaning, a range, based on the general grading system in use in this school, was developed. This assessment scale is described in Table 2:

Points	**Acquisition of Programming skills**
100-89%	Student acquired all programming skills taught
88-75%	Student acquired most of the programming skills taught
74-65%	Student acquired some of the programming skills taught
64-51%	Student acquired the most essential basics of the programming skills taught
Less than 50%	Student acquired too little or none of the programming skills taught

Table 2: Assessment scale used for assessing the acquisition of programming skills

The results of this Analysis will be presented in Chapter 3.3.

Finally, to determine, whether the students had acquired generic skills, the self-reflection sheets of the students about the pair programming process were evaluated. This was done as follows: All students received a reflection sheet (see Appendix), which specifically asked for their Collaboration, their Communication, and their Creativity. The responses were then analyzed looking at these themes specifically. For example: whenever a student talked positively about the collaboration or about some skill they acquired, it was noted with a plus. If a student talked negatively about a collaboration or didn't see any progress in this generic skill, a minus was noted. After each student's reflection was analyzed in this way, a big picture of the mood was created. This was done for the skills of collaboration, communication, and creativity separately but in the same manner. As it cannot be determined if a development or a progress has taken place by few reflection questions only, those results were combined with the analysis of the programming tasks and the observations made during the lessons. The results of this analysis will be discussed in Chapter 3.3

3.3 Results

First of all, the results of the observation will be discussed. In Table 3, the themes used for the thematic analysis of the observation notes will be described and the number of occurrences as well as example statements uttered by the students will be given:

Theme	Theme explanation	# of occurrences	Example Statements
questions asked by the students	In this theme, all questions the students posed regarding the understanding of the tasks, or the teaching method (pair programming or working with generative AI) are summarized	19	• "When do we have to swap roles [in pair programming] ?" • "Why can't we simply try to program together and not stick to our roles?" • "Can you help us in solving this task? We do not know what to do."
student behavior	In this theme behavior that deviates from everyday behavior of the student e.g. success-	32	• "I have absolutely no idea how to solve this" – Starts randomly clicking somewhere

	experience or frustration behavior is noted		• "[sigh!] this is way too hard" • "Does it really work?" The Student cannot believe that the program in Scratch does what it should
observable process progress	In this theme every instance of process progress is noted: e.g. students remind themselves to stick to the pair programming roles, students understand a programming concept	16	• "Hey, you are currently the navigator, give me the keyboard and the mouse. It's my turn" • "Why don't we use a loop for this instead of multiple if-statements" • "Have you seen, my ChatGPT gives clearer answers than yours, maybe you should rephrase your prompt"
problems that occurred.	In this theme every instance, whenever a problem occurred is noted. Problems range from technical problems over problems with the teaching approach to misunderstandings	7	• "I can't remember my Scratch password" • "My partner is sick today, what should I do now? Should I continue on my own?" • "I did not save my Python program, what should I use for the ChatGPT vs Me task?"

Table 3: Themes of thematic analysis

This table indicates that the most occurring theme was notes about student behavior followed by questions asked by the students. This might spring from the fact that I utilized a teaching approach – discovery-based learning and pair programming, the students were not familiar with. Although, I used a learner-centered and self-regulated teaching approach throughout the semester, discovery-

based learning with mostly no guidance was new to them. Moreover, they have never worked in pair programming mode and, thus, needed some time to get acquainted with it.

Both categories mostly occurred in the notes of the first and second lessons and can, thus be interpreted in such that they got acquainted with this teaching approach in the course of the teaching sequence. This also leads to the conclusion that, especially with students who are not yet familiar with the two teaching approaches, sufficient time – even more than originally planned – needs to be dedicated to explaining the concepts and developing rules together with the students.

Considering the problems that occurred, they also prove fruitful for further improvements in the teaching concepts. I did not pay attention to scenarios like these, and therefore, the teaching scenario needs to include some plan B options. Problems with forgotten registration data are a common problem and are easily solvable by resetting the password, given that the student has access to his/her e-mail account. The example presented, where one student was absent, is a bit more tricky. It does not really make sense that the student continues working on their own, as it would counteract the principle of the pair programming approach. So, in that case, I would advise letting the student either work as an "expert" who helps the other teams or let him/her join another team, and they have three roles now: one driver and two navigators. I decided on the latter. The same issue might occur when there is an odd number of students in the classroom. The examples provided in the themes "problems that occurred" give an insight into which scenarios should be considered when designing the teaching scenario and which plan Bs should be thought through.

Secondly, the results of the acquisition of programming skills will be analyzed. In Table 4, the assessment used to award points to the students is described in detail. A maximum of 26 points was possible:

Criterion	Max. Points
Scratch	
Task fulfillment: The Scratch program has all features that are required in the task description and is playable	1
Correct use of Syntax: The students used the building blocks provided in Scratch in a meaningful and correct way	1
Python	

Task fulfillment: The Python program has all the features that are required in the task description and the code is executable	3 (1 for each program)
Correct use of Syntax: The students used the Python syntax correctly and in a semantically correct way	3 (1 for each program)
Presentation	
Indication of Programming concepts (a loop, a variable, a condition, and a function)	8 (4 points for Scratch, 4 points for Python)
Final Quiz	
Students' performance on the final quiz. 10 questions. 1 point each	10

Table 4: detailed assesment scale for assessing acquisition of programming skills

As some students did not hand in a Python program at all or did not hand in a program in Python format and, thus preventing the analysis of their programs, they received 0 points in the Python category. Figure 1 indicates the distribution of the share of programming skills acquired. Those categories were already described in Table 1 and Table 4:

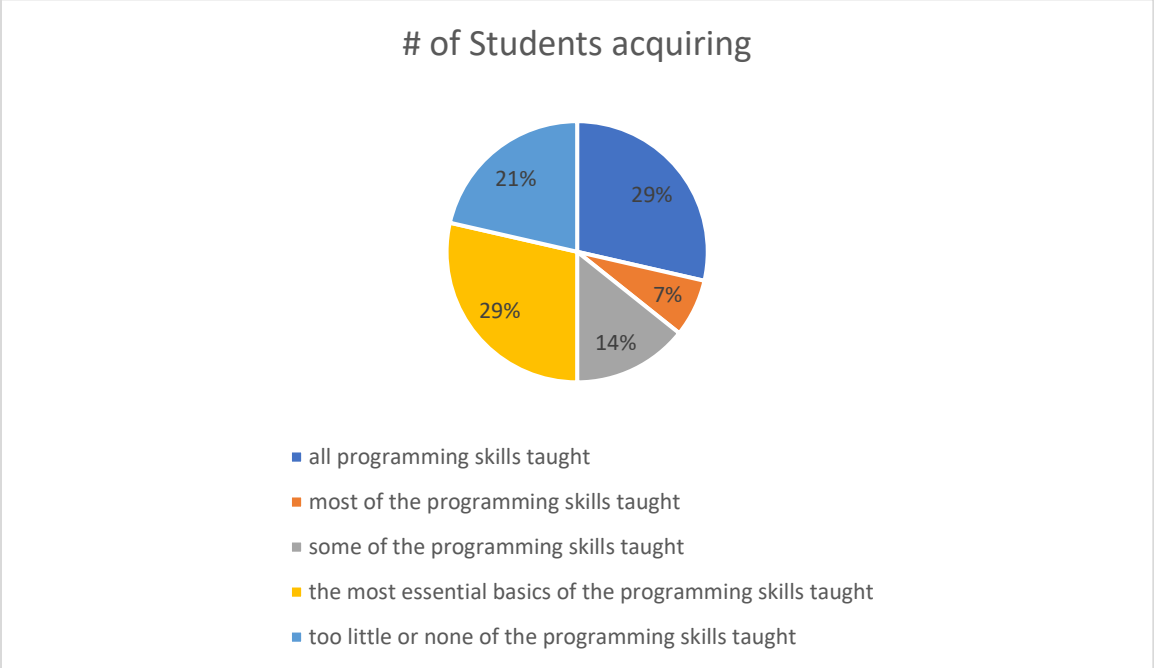


Figure 1: Distribution of share of programming skills acquired

Figure 1 indicates that more than 50% of the students acquired at least some of the programming skills taught. There was only one student who had less than 50% on the final programming quiz. The others were above 50%. All of the 21% of the students who were assigned to the category “acquiring

too little or none of the programming skills taught" did not hand in a Python program or handed in a file that could not be opened. However, only two students in this category reached only 5 points in the final programming quiz. They also had only 2 and 4 points in the presentation.

Therefore, we can conclude that out of the 21%, only two people fall into this category, as the others outperformed the other subcategories of this category.

Finally, the students were asked to reflect on their work with ChatGPT and with their classmates while working on the programs. Figure 2 shows the students' perceptions of their development of generic skills:

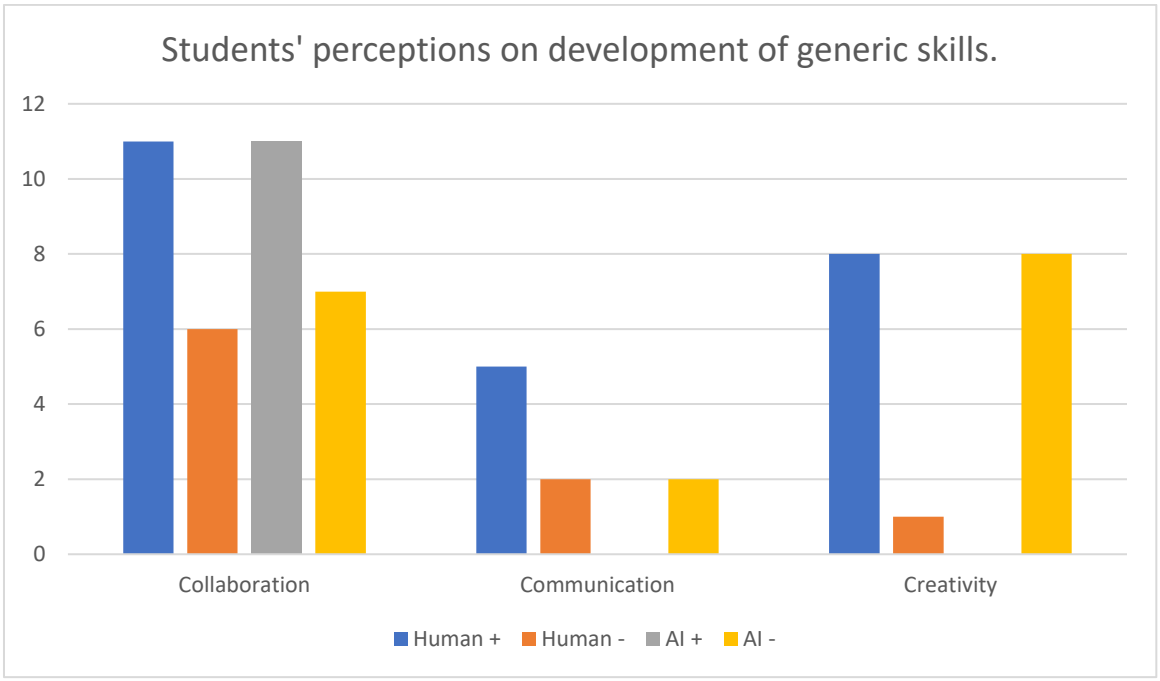


Figure 2: Students perceptions on development of generic skills

Considering Figure 2, it can be concluded that students perceived an increase in collaboration competence. They feel that this teaching sequence has positively contributed to their collaboration competence in both – their collaboration with other human beings and with artificial intelligence. Both aspects play a crucial role in the development of generic skills and are of high importance for the future lives of the students. With creativity, students see only a contribution to the development of this skill while working with another human being. On the contrary, students argue that their creativity was not promoted at all while working with AI. Finally, the majority of students see a development in their communication skills, especially while working with a classmate, as they had to resolve conflicts and misunderstandings and find a common understanding. Two students argued that they were having small fights with each other while programming and, thus, enjoyed working with the AI more than with the other partner, as they felt misunderstood.

This self-reflection correlates with the observations made by the teacher. While observing, it was obvious that the students did communicate very well with each other, they talked through misunderstandings, rephrased their thoughts if the programming partner does not understand what they want to tell and asked for clarification. There was hardly any sign of communication breakdowns or fights. As indicated by the students, communication with the AI did not really contribute to a development of this skill as non-verbal communication skills cannot be met by an AI. However, it was observed that the students did learn something in connection with communication. If one considers the prompts the students used for completing their AI tasks, an improved prompting could be observed. They prompted more clearly, and with greater detail. The students worked really well as a team. They stuck to their roles and helped each other, there were hardly any major issues. They also collaborated well with the AI. This was seen in the working process, but also in the presentation phase. Students did not see an increase in their creativity while working with AI, as they perceived that AI did all the work and they did not have to be creative to solve the task. This might be true on the one hand, however, I perceive that some creativity is also necessary to find the correct way to prompt or to come up with creative solutions for the presentation.

In general, it can be said that measuring generic skills such as creativity, collaboration, and communication presents significant challenges for educators and researchers. These skills, are essential for success in the 21st century but are inherently complex and multifaceted, making their assessment difficult. One of the primary challenges in assessing these skills is their subjective nature. As noted by Tamela et al. (2024), while lesson plans may incorporate indicators for creativity, collaboration, and communication, the actual measurement of these skills often lacks clarity and consistency. Purjiyanta et al. (2020) highlight that while experimental activities can foster these skills, the lack of standardized assessment tools makes it difficult to evaluate their development accurately.

Considering these obstacles, it can be noted that it is hard to tell whether, students actually did improve their generic skills under consideration. The self-reflection and the observations made by the teacher as well as the lesson design that allows room for the development of those skills, do at least prove promising in promoting the generic skills of communication, creativity, and collaboration and indicate some growth.

In response to the research question “To what extent does the integration of pair programming and generative AI tools in Computer Science lessons for the 5th grade of an upper secondary school (AHS) assist the development of selected generic competencies of the students as well as the development of their programming skills?” the findings suggest that the majority of students successfully acquired at least some of the programming skills taught, with only a small percentage struggling significantly. The combination of pair programming and AI-assisted learning supported the development of

programming competencies for most students. Regarding generic competencies, while improvements in communication and collaboration were noted, students perceived AI tools as a hindrance to creativity. This suggests that while AI can enhance coding efficiency and facilitate learning, it may also limit opportunities for independent problem-solving and original thinking. Furthermore, the subjective nature of assessing generic skills remains a challenge, highlighting the need for more refined evaluation methods. Overall, the integration of pair programming and AI tools appears to be beneficial for programming skill development, though careful consideration is needed to balance AI support with fostering creativity in students.

3.4 Discussion

The results of this research indicate that utilizing a discovery-based learning approach in combination with pair programming and generative AI tools, can be a promising approach for mediating programming skills and the generic skills communication, collaboration, and creativity. Each of these pedagogical strategies contributes uniquely to the learning experience, fostering an environment conducive to skill acquisition.

3.4.1 Discovery-based learning

Although, discovery-based learning has hardly been in used in Computer Science education (Baldwin, 1995 & Arnold, 2005) and little research has been done in this field, it proved very suitable for this specific teaching and learning scenario. Discovery-based learning is a learning approach that counts to active learning methodologies, where the students are in the center of teaching and learning and not the teacher. Students are asked to acquire knowledge through interaction, experimentation, and reflection (Córdova-Esparza, et al. 2024). This approach also fosters the development of generic competencies such as collaboration, critical-thinking, and problem-solving skills (Córdova-Esparza, et al. 2024 & Khabibah et al., 2017). This active engagement not only deepens understanding but also cultivates a sense of ownership over the learning process, which is crucial for developing skills that extend beyond the classroom (Saptura et al., 2022). Moreover, discovery-based learning aligns with the constructivist theory of learning (Chuang & Lee, 2021). The constructivist theory of learning is said to be a teaching theory, that positively contributes to student motivation and a higher student performance (Maheshwari & Thomas 2017). Those effects were also visible in the current study. Students felt more motivated and were keen on finding solutions to their proven problems (especially, when they had to develop games with Scratch).

It has been seen, however, that especially in the beginning, when the students are not familiar with the teaching approach yet, frustration and confusion might be present. Liu et al. (2013) found out that, when students experience these motions for a short term, it can lead to an increased motivation for overcoming these emotions. However, if students experience these emotions over a lengthier time span, they lose motivation and encouragement. Therefore, it is especially important for the teacher to keep the motivation high, and encourage students to keep on with the work. Therefore, the teacher needs to step into the role of the coach and the motivator and step back from the role of the instructor. Although, this research addresses this approach a positive outcome, it should be noted that only one aspect of Computer Science education – namely the teaching of programming skills – has been observed. Moreover, it needs to be considered that this teaching and learning design, focused mostly on the collaboration aspect as the students mainly worked in pairs or small teams. Further research would be necessary to determine whether discovery-based learning, also has positive outcomes if students try to acquire programming skills in solo-programming mode, and if it is also promising for mediating other concepts in the Computer Science Education.

3.4.2 Pair-Programming

The findings also indicate that pair programming positively contributed to the learning outcome. Besides that the students noted a higher motivation and enjoyment while performing pair programming activities. The same has been found in a Iskrenovic-Momcilovic study (2019), which found out that pair programming provides higher motivation for acquiring new knowledge, faster memory of new concepts, and a more effective learning of programming languages. A study, conducted by Hughes et al. (2020) gives reasons why pair programming has a better performance outcome than solo programming. In their study, they found out that the continuous and immediate feedback-circle with the peers, and the collaborative programming solving leads to a better output (Hughes et al., 2020). Another study, by Hannay et al. (2009), who performed a meta-analysis of 18 studies that focused on the outcome differences between solo and pair-programming came to a similar conclusion. They found that students were able to complete tasks in less time when doing pair programming and came to better solutions of complex task (Hannay et al., 2009). As in the current study, the students did not perform solo-programming tasks, the second point can neither be verified nor falsified, however, what has been seen is that students were able to solve even more complex tasks in pair programming.

Moreover, the findings of the current study, also indicate that pair programming positively contributed to the development of the generic skills communication, collaboration, and creativity. This has been seen in the teacher observation and the final student products, but also in the self-reflection of the students. Kanika et al. (2020) also highlights that students profit from pair programming not only in such that they produce code of higher quality, but they also acquire essential generic skills such as communication and collaboration. The reason for that is that the students need to articulate their thought processes and negotiate solutions with their partner (Kanika et al., 2020). Finally, research suggests that pair programming also contributes to a better learning environment, as students are encouraged to share their knowledge, learn from each other, and, thus increases the feeling of belonging, and reduces isolation, which is most likely the case with other programming tasks (Xinogalos et al., 2017).

3.4.3 Generative AI tools

Although students did not really see the benefits of using AI for developing generic skills, they stated that the enjoyed working with AI as a programming partner. They highlighted that they enjoyed the benefit of quick reply, and that they learned how to prompt effectively, in order to receive desired output. It should be noted, however, that students definitely need some prior knowledge before they can use AI for their own learning process. First, they need to be familiar with prompting. Second, they need to be familiar with the programming concepts to verify the output they receive from ChatGPT or other AI tools. Third, they need to have strategies at hand to assess the output and to make decisions. Finally, it needs to be ensured that the students are able to solve the task they outsource to ChatGPT or any other AI on their own, to ensure a deep understanding of the concepts. Literature suggests that AI can be a valuable partner when acquiring programming skills, as AI can facilitate personalized feedback and support, which allows students to progress at their own pace while receiving guidance on complex programming concepts. Moreover, generative AI can assist in automating repetitive tasks, freeing up cognitive resources for more complex problem-solving and creative thinking (Hughes et al., 2020). The current research does support these findings in general but also gives some limitations, as students did not have the experience of becoming more creative. Rather they saw a hindrance in developing this specific skill. The reason for that might spring from the programming task design. Students felt more creative while working with a human pair programming partner. Further research in this area would be necessary to determine if pair programming with AI hinders the development of creativity, and if so, how this could be overcome.

The combination of these three approaches creates a synergistic effect that amplifies their individual benefits. For instance, the inquiry-driven nature of discovery learning complements the collaborative aspects of pair programming, as students can explore programming concepts together while applying critical thinking skills. This collaborative exploration is further enhanced by the use of generative AI tools, which can provide real-time insights and suggestions, thereby enriching the learning experience (Hughes et al., 2020). The interplay between these elements fosters an environment where students can develop not only their technical skills but also the generic skills necessary for success in the modern workforce.

3.5 Implications, Limitations, and Further work

The findings of this research are of high relevance for computer science teachers teaching at secondary schools, as it provides a framework for mediating programming skills and generic skills at once. It fosters a higher motivation and a higher learning outcome by the students and combines previous literature-based findings (e.g. pair programming as a teaching method) with new developments in the field of Computer Science Education (e.g. AI). However, some limitations should be considered. First, the teaching scenario that was developed was only taught in one specific school with one specific class, whereby the researcher was the teacher of this class. Considering this, some subjectivity cannot be neglected with this case study despite the research trying to be as objective as possible. To verify these findings, further research will be necessary, meaning that the teaching scenario needs to be carried out multiple times in multiple different (cultural) settings to ensure validity. In the future, this teaching scenario will be held with additional classes of the same school. Possible diverging outcomes will be reported in future studies. A second limitation of this study is the constellation of this class. 11 out of the 14 students in the class were female, resulting in a gender imbalance. In future studies, possible differences in gender results might be interesting to observe.
Moreover, the study took place in a more rural area, leaving place for speculation if results would be different in urban areas.

In addition, it should be noted that currently, students who proceed to the upper secondary level have (ideally) some prior experience in programming, as they had 4 year of the subject general basic education, resulting in the question of how the results would vary.

Finally, in this study, students assigned themselves to pair programming teams. It would be interesting to find out if results, would differ if students were assigned randomly or according to their knowledge levels.

To verify, if these results hold for other computer science education topics, future research is necessary.

3.6 Personal Reflection

Conducting this research has been a valuable experience, both in terms of my professional development and my perspective on teaching Computer Science. One of the key takeaways from this study is that teaching programming skills and demonstrating how students can use AI for their own improvement means equipping them with skills for the future. As AI continues to shape various fields, enabling students to understand and work with it early on ensures they are prepared for a rapidly evolving technological landscape.

Beyond technical skills, this research reaffirmed the importance of integrating generic competencies into Computer Science lessons. Critical thinking, collaboration, communication, creativity, and problem-solving are just as essential as coding itself. Observing my students throughout the study has reinforced my commitment to embedding these skills into my teaching practice more deliberately. Encouraging teamwork and fostering creativity in my classroom can significantly benefit students beyond their academic careers.

Moreover, analyzing my own lessons and reflecting on their design has been an important step in my professional growth. It allowed me to see what worked well and what could be improved. Even though the lessons ran smoothly overall, student questions often took discussions in unexpected directions, highlighting the need for flexibility and a well-prepared Plan B. This adaptability is crucial in a dynamic classroom setting, ensuring that students remain engaged and that learning objectives are met effectively.

Additionally, this research has reinforced my belief that Computer Science education does not have to be boring. By incorporating engaging methodologies such as pair programming and AI tools, lessons can be interactive, stimulating, and relevant to students' interests. Seeing students enjoy learning while developing valuable skills has been one of the most rewarding aspects of this project.

Finally, conducting research has opened my eyes to further opportunities beyond classroom teaching. Investigating effective teaching methods, evaluating their impact, and exploring innovative pedagogical approaches have been incredibly enriching experiences. This has motivated me to continue exploring ways to enhance Computer Science education, whether through further research, curriculum development, or professional collaboration.

Overall, this journey has not only deepened my understanding of effective teaching strategies but has also inspired me to continually evolve as an educator. I look forward to applying these insights in my future teaching and exploring new ways to create meaningful learning experiences for my students.

4 Conclusio

Summing up, this study has shown that using an approach that has not received a lot of attention in the field of Computer Science Education – namely discovery-based learning, and combining it with current developments in programming didactis (pair programming) and education in general (AI) – proves promising for mediating programming skills and generic skills at once resulting in lessons full of motivation, enjoyment, and a good learning outcome. This study has shown the possibilities of stepping away from a boring and monotonous teacher-centered teaching to an active, engaging, modern and student-centered teaching. This findings should encourage other educators to try out this teaching and learning design and contribute to a paradigm change in computer science education.

5 References

Arnold, Ruedi; Hartmann, Werner & Reichert, Raimond (2005). *Entdeckendes Lernen im Informatik-Unterricht.* Bonn: Gesellschaft für Informatik e.V. S. 197-205.

Balanskat, Anja & Engelhardt, Katja (2015). *Computing our future. Computer programming and coding. Priorities, school curricula and initiatives across Europe*. European Schoolnet. https://www.researchgate.net/publication/284139559_Computing_our_future_Computer_programming_and_coding_-_Priorities_school_curricula_and_initiatives_across_Europe (Accessed October, 21st, 2024)

Baldwin, Doug (1996). *Discovery learning in computer science*. SIGCSE Bulletin. 28(1). S. 222-226. https://doi.org/10.1145/236462.236544.

Bailey, John. (2023). *AI in Education: The leap into a new era of machine intelligence carries risks and challenges, but also plenty of promise*. Education Next. 23(4).

Becker, Brett. A.; Denny, Paul; Finnie-Ansley, James; Luxton-Reilly, Andrew; Prather, James, & Santos, Eddie. Antonio. (2023). *Programming Is Hard - Or at Least It Used to Be.* Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1, 500–506. https://doi.org/10.1145/3545945.3569759

Berenson, Sarah. B., Slaten, Kelli. M., Williams, Laurie., & Ho, Chih.-Wei. (2004). *Voices of Women in a Software Engineering Course: Reflections on Collaboration*. Journal on Educational Resources in Computing, 4(1), 3. https://doi.org/10.1145/1060071.1060074

Bird, Christian; Ford, Denae; Zimmermann, Thomas; Forsgren, Nicole; Kalliamvakou, Eirini; Lowdermilk, Travis & Gazit, Idan. (2023). *Taking Flight with Copilot: Early insights and opportunities of AI-powered pair-programming tools*. Queue 20(6), 35-57 https://doi.org/10.1145/3582083

Banson, Justice. (2022). *Co-regulated learning and online learning: A systematic review*. Social Sciences & Humanities Open 6(1). https://doi.org/10.1016/j.ssaho.2022.100376

Bundesministerium für Bildung, Wissenschaft und Forschung. (2023*) Auseinandersetzung mit Künstlicher Intelligenz im Bildungssystem*. https://www.bmbwf.gv.at/dam/jcr:b77eacd7-3926-460e-955a-0754e419e577/ki_bildungssystem.pdf (Accessed on November, 16th, 2024).

Bundesministerium für Bildung, Wissenschaft und Forschung. (2024). *Bundesrecht konsolidiert: Gesamte Rechtsvorschrift für Lehrpläne – allgemeinbildende höhere Schulen, Fassung vom 21.10.2024.* https://www.ris.bka.gv.at/GeltendeFassung.wxe?Abfrage=Bundesnormen&Gesetzesnummer=10008568 (Accessed on October, 21st, 2024)

Canning Roy (2013). *Rethinking generic skills*. In: European journal for Research on the Education and Learning of Adults 4 (2013) 2, p. 129-138. https://doi.org/10.25656/01:8293

Chassignol, Maud; Khoroshavin, Aleksandr; Klimova, Alexandra; Bilyatdinova, Anna. (2018). *Artificial Intelligence trends in education: a narrative overview*. Procedia Computer Science. 136. 16-24. https://doi.org/10.1016/j.procs.2018.08.233

Córdova-Esparza, Diana-Margarita, Romero-González Julio-Alejandro, Córdova-Esparza, Karen-Edith, Terven Juan & López-Martinez Rocio-Edith. (2024). *Active Learning Strategies in Computer Science Education: A Systematic Review.* Multimodal Technologies and Interaction, 8(6), 50. https://doi.org/10.3390/mti8060050

Dongo, Tendai, Reed, April H, O'Hara Margaret (2016). *Exploring Pair Programming Benefits for MIS Majors.* Journal of Information Technology Education: Innovations in Practice. 15. 223-239, https://doi.org/10.28945/3625

Dorier, Jean-Luc, & García, Francisco Javier (2013). *Challenges and opportunities for the implementation of inquiry-based learning in day-to-day teaching*. ZDM – Mathematics Education, 45(6), 837-849. https://doi.org/10.1007/s11858-013-0512-8

Duggan, Steven (2020). *AI in education: change at the speed of learning*. Moskau: UNESCO Institute for Information Technologies in Education.

Fitzpatrick, Daniel; Fox, Amanda & Weinstein, Brad. (2023). *The AI classroom: The ultimate guide to artificial intelligence in education*. TeacherGoals Publishing, LLC.

Gerring, John. (2016). *Case Study Research: Principles and Practices* (2nd ed.). Cambridge: Cambridge University Press.

Górecki, Jan (2023). *Pair Programming with ChatGPT for sampling and estimation of copulas*. Computational Statistics. https://doi.org/10.1007/s00180-023-01437-2

Hamilton, Ilana. (2024). *Artificial intelligence in education: Teachers' opinions on AI in the classroom*. https://www.forbes.com/advisor/education/it-and-tech/artificial-intelligence-in-school/ (Accessed on, 16th November, 2024).

Hanks, Brian, Fitzgerald, Sue, McCauley, Renée, Murphy, Laurie, & Zander, Carol (2011). *Pair programming in education: a literature review.* Computer Science Education, 21(2), 135–173, https://doi.org/10.1080/08993408.2011.579808

Hannay, Jo E.,Dybå, Tore, Arisholm, Erik, Sjøberg, Dag I.K. (2009). *The effectiveness of pair programming: A meta-analysis*, Information and Software Technology, 51(7). 1110-1122, https://doi.org/10.1016/j.infsof.2009.02.001.

Hughes, Janet, Walshe, Ann, Law, Bobby & Murphy, Brendan. (2020). *Remote Pair Programming*. Proceedings of the 12th International Conference on Computer Supported Education - Volume 2: CSEDU; 476-483, https://doi.org/10.5220/0009582904760483

Issaee, Arash; Motschnig, Renate & Göltl, Katrin. (2022). *Learning to program in secondary classrooms: Students' and Teachers' perceptions of the pair-programming setting*. 2022 IEEE Frontiers in Education Conference (FIE), Uppsala, Sweden, 1-9, https://doi.org/10.1109/FIE56618.2022.9962635

Johnson, Alyssa (2019). *5 ways AI is changing the Education Industry*. https://elearningindustry.com/ai-is-changingthe-education-industry-5-ways (Accessed on November, 16t, 2024).

Khabibah, Elok Norma, Masykuri, Mohammad, & Maridi, Maridi. (2017). *The effectiveness of module based on discovery learning to increase generic science skills.* Journal of Education and Learning (Edulearn), 11(2), 146-153. https://doi.org/10.11591/edulearn.v11i2.6076

Korber, Patrick, Motschnig, Renate. (2021). *The effects of pair-programming in introductory programming courses with visual and text-based languages.* 2021 IEEE Frontiers in Education Conference (FIE), *Lincoln, NE, USA, 2021. 1-9,* https://doi.org/10.1109/FIE49875.2021.9637186

Koulouri, Theodora;Lauria, Stansilao & Macredie, Robert D. (2014). *Teaching Introductory Programming: A Quantitative Evaluation of Different Approaches*. ACM Transactions on Computing Education. 14(4). http://dx.doi.org/10.1145/2662412

Kozakiewicz, Nicolai; Bosset, Isabelle; Buchner, Josef & Reitinger, Johannes. (2024). E*ntdeckend-forschendes Lernen im Kontext einer Bildung für nachhaltige Entwicklung*. Haushalt in Bildung & Forschung, 13(1), 68–84. https://doi.org/10.25656/01:29220

Kuttal, Sandeep Kaur; Ong, Bali; Kwasny, Kate & Robe, Peter. (2021). *Trade-offs for Substituting a Human with an Agent in a Pair Programming Context: The Good, the Bad, and the Ugly.* Proceedings of the 2021 CHI Conference on Human Factors in Computig Systems (CHI '21). Association for Computing Machinery, New York, NY, USA, Article 243, 1-20. https://doi.org/10.1145/3411764.3445659

Lau, Sam & Guo, Philip (2023). *From "Ban It Till We Understand It" to "Resistance is Futile": How University Programming Instructors Plan to Adapt as More Students Use AI Code Generation and Explanation Tools such as ChatGPT and GitHub Copilot.* Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 1, 106–121. https://doi.org/10.1145/3568813.3600138

Ma, Qianou; Wu, Tongshuang & Koedinger, Kenneth (2023*). Is AI the better programming partner? Human-Human Pair Programming vs. Human-AI Pair Programming*. arXiv. https://doi.org/10.4850/arXvis.2306.05153

Mader, Michaela (2024). *ChatGPT in der Schule – Fluch oder Segen*. Masterarbeit: Universität Wien.

Maheshwari, Greeni, & Thomas, Susan. (2017). *An Analysis of the Effectiveness of the Constructivist Approach in Teaching Business Statistics.* Informing Science: The International Journal of an Emerging Transdiscipline 20, 89-97. https://doi.org/10.28945/3748

Momcilovic-Iskrenovic, Olivera (2019). *Pair programming with Scratch*. Education and Information Technologies. 2019 (24). 2943 –2952. https://doi.org/10.1007/s10639-019-09905-3

Liu, Zhongxiu, Pataranutaporn, Visit, Ocumpaugh, Jaclyn & Baker, Ryan S.J.d (2013). *Sequences of Frustration and Confusion, and Learning.* Proceedings of the 6th International Conference on Educational Data Mining (EDM 2013). 114-120

Papadakis, Stamatios (2018*). Is Pair Programming More Effective than Solo Programming for Secondary Education Novice Programmers? A Case Study*. International Journal of Web-Based Learning and Teaching Technologies. 13 (1). http://dx-doi-org.uaccess.univie.ac.at/10.4018/IJWLTT.2018010101

Phung, Tung; Pădurean, Victor-Alexandru; Cambronero, José; Gulwani, Sumit; Kohn, Tobias; Majumdar, Rupak; Singla, Adish & Soares, Gustavo. (2023*). Generative AI for Programming Education: Benchmarking ChatGPT, GPT-4, and Human Tutors*. https://doi.org/10.48550/arXiv.2306.17156

Purjiyanta, Eka, Wiyanto, Wiyanto, Sarwi, Sarawi, & Nugroho, Sunyoto Eko. (2020). *A hypothetical design of the contextual science book to develop generic skills.* Proceedings of the Proceedings of the 5th International Conference on Science, Education and Technology, ISET 2019, 29th June 2. https://doi.org/10.4108/eai.29-6-2019.2290415

Ray, Willis. E. (1961). *Pupil Discovery vs. Direct Instruction*. The Journal of Experimental Education, *29*(3), 271–280. https://doi.org/10.1080/00220973.1961.11010692

Repenning, Alexander, Webb, David C., & Ioannidou, Andri (2010). *Scalable game design and the development of a checklist for getting computational thinking into public schools*. Proceedings of the 41st ACM Technical Symposium on Computer Science Education (SIGCSE'10), 265–269. ACM Press. https://doi.org/10.1145/1734263.1734357

Riedl, Alfred & Schelten, Andreas (2012). *Entdeckendes Lernen*. Grundbegriffe der Pädagogik und Didaktik beruflicher Bildung. Deutschland: Franz Steiner Verlag.

Saputra, Dicky, Rosilawati, Ila & Utami, Gamilla Nuri. (2022). *The Effectiveness of Blended Learning with Discovery Learning Model on Buffer Solution Materials to Improve Science Process Skills*. Jurnal Pendidikan Dan Pembelajaran Kimia, 11(3). https://doi.org/10.23960/jppk.v11.i3.2022.02

Scherer, Ronny; Siddiq, Fazilat ; Sánchez Viveros Bárbara. (2020). *A meta-analysis of teaching and learning computer programming: Effective instructional approaches and conditions.* Computers in Human Behavior 109. https://doi.org/10.1016/j.chb.2020.106349

Singh, Ninni; Ahuja, Neelu Jyothi & Kumar, Amit. (2018). *A Novel Architecture for Learner-Centric Curriculum Sequencing in Adaptive Intelligent Tutoring System.* Journal of Cases on Information Technology (JCIT), 20(3), 1-20. https://doi.org/10.4018/JCIT.2018070101

Skaarseth, Lars Kristian (2023). *Investigating the transfer from Scratch to Python in Norwegian secondary school.* Master thesis –University of Oslo.

Spinellis, Diomidis (2024). *Pair Programming with Generative AI*. IEEE Software 41(3). 16-18. https://doi.org/10.1109/MS.2024.3363848

Statistik Austria (2023). *11 % der österreichischen Unternehmen nutzen künstliche Intelligenz*. https://www.statistik.at/fileadmin/announcement/2023/10/20231017IKTU2023.pdf (Accessed on February 17th, 2025)

Tahiru, Fathi Ho (2021*) AI in Education: A systematic Literature Review*. Journal of Cases on Information Technology. 23(1). https://doi.org/10.4018/JCIT.2021010101

Tamela, Ellis, Palupi, Tara Mustikaning, & Pujiati, Hanip (2024). *How english teachers' lesson plans incorporate learning and innovation skills.* E-Link Journal, 11(1), 1-16. https://doi.org/10.30736/ej.v11i1.1016

Tehlan, Kanika, Chakraverty, Shampa, Chakraborty, Pinaki & KKhapra, Shradha. (2020). *A genetic algorithm-based approach for making pairs and assigning exercises in a programming course.* Computer Applications in Engineering Education 28(6). 1708-1721. https://doi.org/10.1002/cae.22349

Tikva, Christian, & Tambouris, Efthimios (2021). *Mapping computational thinking through programming in k-12 education: a conceptual model based on a systematic literature review*. Computers & Education, 162(1), 104083. https://doi.org/10.1016/j.compedu.2020.104083.

Tshukudu, Ethel & Jensen, Siri Annethe Moe. (2020).*The Role of Explicit Instruction on Students Learning their Second Programming Language*. In: UKICER '20: United Kingdom and Ireland Computing Education Research conference (2020), 10–16. doi: https://doi.org/10.1145/3416465.3416475.

Umpathy, Karthikeyan & Ritzhaupt, Albert D. (2017). *A Meta-Analysis of Pair-Programming in Computer Programming Courses: Implications for Educational Practice*. ACM Transactions on Computing Eduction. 17(4). 1-13. https://doi-org.uaccess.univie.ac.at/10.1145/2996201

van Laar, Ester, van Deursen, Alexander. J. A. M., van Dijk, Jan. A. G. M., & de Haan, Jos. (2020). *Determinants of 21st-Century Skills and 21st-Century Digital Skills for Workers: A Systematic Literature Review.* SAGE Open, 10(1). https://doi.org/10.1177/2158244019900176

Vuorikari, Riina; Kluzer, Stefano & Punie, Yves. (2022).*DigComp 2.2: The Digital Competence Framework for Citizens -With new examples of knowledge, skills and attitudes*. Luxemburg: Publications Office of the European Union. https://doi.org/10.2760/115376

Vygotsky, Lew Semjonowitsch. (1978). *Mind in Society: The Development of Higher Psychological Processes*; Harvard University Press: Cambridge

Williams, Laurie. (2001) *Integrating pair programming into a software development process, Proceedings 14th Conference on Software Engineering Education and Training*. Search of a software engineering profession Charlotte, NC, USA, 27-36, https://doi.org/10.1109/CSEE.2001.913816

Wing, Jeannette M. (2010). *Computational Thinking : What and Why?* The link magazine. http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf. (Accessed on October, 21st 2024).

Xinogalos, Stelios, Satratzemi, Maya, Chatzigeorgiou, Alexander, & Tsompanoudi, Despina. (2019). Factors Affecting Students' Performance in Distributed Pair Programming. Journal of Educational Computing Research, 57(2), 513-544. https://doi.org/10.1177/0735633117749432

Xu, Enwei; Wang, Wei & Wang, Qiangxia (2022*). A meta-analysis oftheeffectiveness ofprogramming teaching inpromoting K-12 students'computational thinking*. Education and Information Technologies. 2023(28). 6619-6644. https://doi.org/10.1007/s10639-022-11445-2

Xu, Fan & Correia, Ana-Paula (2023). *Adopting distributed pair programming as an effective team learning activity: a systematic review*. Journal of Computing in Higher Education. https://doi.org/10.1007/s12528-023-09356-3

Yin, Robert K. (1994). *Discovering the Future of the Case Study. Method in Evaluation Research.* Evaluation Practice, 15(3), 283-290. https://doi.org/10.1177/109821409401500309

Yurniati, Deri, Syofyan, Efrizal, Usman,, Marwan. (2019). *The Effect of Teacher's Role, Learning Motivation and Students' Creativity toward Learning Outcome on Workshop and Entrepreneurship's Subject of XI Grade Students in Management Business Vocational School*. Proceedings of the 2nd Padang

International Conference on Education, Economics, Business and Accounting (PICEEBA-2 2018). https://doi.org/10.2991/piceeba2-18.2019.103

Zehner, Fabian (2019). *Künstliche Intelligenz. Ihr Potenzial und der Mythos des Lehrkraft*. Schulmanagement-Handbuch 38. 169. 6-30. https://doi.org/10.25656/01:17561

AI-Disclaimer

In the preparation of this master's thesis, artificial intelligence (AI) tools were utilized to support specific aspects of the writing process. Grammarly was used for grammar and spelling checks, Microsoft Co-Pilot for rephrasing, ChatGPT for structuring thoughts, and DeepL for translation. These tools were employed solely to enhance linguistic clarity and coherence. The conceptualization, research, analysis, and original arguments presented in this thesis were developed independently, without AI-generated content beyond the aforementioned supportive functions.

No AI tools were used for data analysis, literature review synthesis, or the generation of research findings. All intellectual contributions, interpretations, and conclusions are my own.

6 List of Tables

7 List of Figures

8 Appendix

8.1 Teaching Materials

8.1.1 Lesson plans

1. Lesson – Revision of Scratch

Goals:

- Students can access their Scratch accounts
- Students become familiar with the Scratch developing environment
- Students become familiar with the functionality of Scratch
- Students can group similar building blocks based on their functionality

Time frame: 100 minutes

Time	Activity	Material/Media
5 min	Introduction to today's lesson	-
25 min	Getting access to previously registered Scratch accounts and familiarizing with developing environment	PCs
35 min	Discovery of Scratch functionality and different building blocks	PCs
20 min	Collection of found building blocks and grouping together based on their functionality	Whiteboard
15 min	Introduction of basic programming terms loops, functions, variables, conditions	Whiteboard

Table 5: Lesson plan 1st double lesson

2. Lesson – Programming with Scratch

Goals:

- Students can develop a game using Scratch in pair programming mode
- Students can use most common programming concepts
- Students can explain the most common programming concepts.

Time frame: 100 minutes

Time	Activity	Material/Media
5 min	Introduction to today's lesson	-
10 min	Quiz on basic concepts to re-fresh memory from previous lesson	PCs
15 min	Introduction to pair program-ming (roles, do's and don'ts)	Sheet – "Pair programming Rollenverteilung"
10 min	Pair formation	Cards for picking pairs, if pairs are not formed alone
10 min	Task description, assigning of games to develop	Sheet – "Aufgabenblatt Scratch"
50 min	Game development in Scratch	PCs

Table 6: Lesson plan 2nd double lesson

3. Lesson – Programming in Scratch, Transformation to text-based programming with Python

Goals:

- Students can develop a game using Scratch in pair programming mode
- Students can present their developed games and explain their code by giving justifications
- Students can transfer programming knowledge from Scratch to the text-based programming language Python

Time frame: 100 minutes

Time	Activity	Material/Media
5 min	Introduction to today's lesson	-
25 min	Game development (continuation of lesson 2)	PCs
20 min	Game presentation, playing the games of the other groups, giving feedback to colleagues	Projector, PCs
15 min	Discussion about the purpose of programming languages, key features of programming languages – Internet search	PCs, Whiteboard
15 min	introduction to sample program in Python by the teacher, comparison of commands with those used in scratch	Sample Program written in Python (not too complex), teacher's PC
20 min	summarizing of the findings on the whiteboard, introduction of further important commands. Students note that down, too	PCs, Whiteboard

Table 7: Lesson plan 3rd double lesson

4. Lesson – Programming in Pythong

Goals:

- Students know the most relevant demands in Python (loops, conditions, variables, variable types, functions, Boolean operators,…)
- Students can develop simple programs in the programming language Python in pair programming mode

Time frame: 100 minutes

Time	Activity	Material/Media
5 min	Introduction to today's lesson	-
30 min	Students complete the tool Silent teacher that introduces them to most relevant demands.	PCs, Silent teacher
15 min	Students fill out a distributed word document to note down the demands they just discovered using the tool silent teacher to draft a class-cheat sheet	Shared word document, PCs
5 min	Forming Pairs (same pairs as with Scratch)	-
5 min	Distribution of task description and explanation of tasks.	Sheet – "Assignments Python"
35 min	Developing of programs described by the task description in pair programming mode	PCs, task description
15 min	Discussion of solved programs in plenary format, reflection about experience	-

Table 8: Lesson plan 4th double lesson

5. Lesson – Programming with AI

Goals:

- Students can develop a game using AI assistance (ChatGPT)
- Students can reflect on the code provided by AI and compare it to their own development
- Students can evaluate the code provided by AI.

Time frame: 100 minutes

Time	Activity	Material/Media
5 min	Introduction to today's lesson	-
15 min	Revisiting AI (do's and don'ts, how to prompt, etc.) to activate prior knowledge	Whiteboard (to create Mind-Map of Brainstorming activity)
30 min	Programming with ChatGPT: letting ChatGPT program programming assignments of previous lesson and trying to understand the code provided by AI	PCs, programming assignments for Python
40 min	analyzing ChatGPTs code, comparing it to their own code, highlight differences and similarities	PCs, reflection sheet "ChatGPT vs me"

Table 9: Lesson plan 5th double lesson

6. Lesson – AI as a pair programming partner

Goals:

- Students can develop a program using AI assistance (ChatGPT) as either the driver or the navigator
- Students can reflect on the code provided by AI and give refinement suggestions/students can use the suggestions provided by AI
- Students can reflect on their personal development in their programming competence and their generic competences collaboration, creativity, and communication.
- Students understand block-based and text-based programming languages' most relevant programming concepts (loops, variables, conditions, etc.).

Time frame: 100 minutes

Time	Activity	Material/Media
5 min	Introduction to today's lesson	-
5 min	dividing class into two groups (one group pair programming role driver, one group pair programming role navigator)	-
10 min	explaining task – driver: write code, let ChatGPT analyze and optimize it; navigator: let ChatGPT write code, analyze and optimize it	-
30 min	Code development, code presentation and justification	PCs, projector
30 min	Reflection sheet on the work with AI as a pair programming partner	Reflection sheet "my experience with pair programming"
20 min	Final quiz	Final quiz (forms quiz(

Table 10: Lesson plan 6th double lesson

8.1.2 Teaching Materials for the Lessons

8.1.2.1 Scratch Game Description

Gruppe 1) Spiel – Früchte Sammeln

Programmiert mit Scratch ein Spiel. In dem Spiel sollt ihr mit einem Korb Früchte, die vom Himmel fallen, einsammeln. Für jeden Apfel, den du einsammelst, bekommst du 2 Punkte. Für jede Banane, die du einsammelst, bekommst du 4 Punkte. Fängst du die Früchte nicht auf, werden die Punkte abgezogen. Versuche möglichst viele Punkte zu sammeln, bis der Timer von 3000 Sekunden abgelaufen ist. Vermerke die Punkte in einem Punktekonto.

Tipps für die Programmierung:

- Du benötigst mehrere Figuren, die unabhängig voneinander zu steuern sind
- Du benötigst ein Feld (Variable), dass die Punkte zählt und einen Timer, der abläuft
- Wenn die Frucht den Korb berührt, sollen die Punkte hinzugezählt werden.
- Überlege dir, wie du Minuspunkte rechnest (Tipp: Einen Boden definieren und, wenn dieser berührt wird, abziehen)
- Die Früchtefiguren gehen immer in eine Zufallsposition bei y=180 zurück.

Gruppe 2) Apfelernte

Programmiere ein Spiel bei der eine Katze Äpfel von einem Baum erntet. Die Katze soll über die Pfeiltasten gesteuert werden. Für jeden Apfel, den die Katze sammelt, bekommst du einen Punkt und die Katze wächst. Die Äpfel sollen zu zufälligen Zeitpunkten vom Baum fallen. Wenn ein Apfel die Katze berührt, bekommst du einen Punkt und soll danach gelöscht werden. Fällt der Apfel zu Boden, wird er ebenso gelöscht. Das Spiel soll 50 Sekunden dauern. Die Katze soll am Ende des Spiels sagen, wie viele Punkte gesammelt wurden.

Tipps zur Programmierung:

- Du benötigst 3 Figuren (Katze, Baum, Apfel)
- Da der Baum auch ein Objekt ist und die Katze nicht verdeckt werden soll, brauchen wir «komm nach vorn» (findest du bei Aussehen).
- Erstelle nur einen Apfel und klonen diese. Erst, wenn der Code für einen Apfel programmiert wurde, wird er geklont.
- Du benötigst zwei Variablen (Punkte und Stoppuhr)

Gruppe 3) Ballonjagd

Programmiere ein Spiel, bei dem Ballons abgeschossen werden sollen. Die Ballons sollen zufällig aufsteigen. Nun soll der Spieler mit der Maus den Ballon abschießen (draufklicken). Erwischt er ihn, bekommt man einen Punkt. Berührt der Ballon den Bildschirmrand ist das Spiel vorbei. Das Spiel endet, wenn ein Ballon den Bildschirmrand berührt oder, wenn die Stoppuhr abgelaufen ist. Gib am Ende aus, wie viele Punkte erreicht wurden

Tipps zur Programmierung:

- Erstelle nur einen Ballon und klonen diese. Erst, wenn der Code für einen Apfel programmiert wurde, wird er geklont.
- Du benötigst zwei Variablen (Punkte und Stoppuhr)

Gruppe 4) Weltraumrennen

Programmiere ein Spiel, bei dem ein Raumschiff einen Stern fangen soll. Das Raumschiff soll mit den Pfeiltasten gesteuert werden. Der Stern soll zufällig auftauchen. Wenn der Stern vom Raumschiff berührt wird, bekommst du einen Punkt und der Stern verschwindet und taucht an einer anderen zufälligen Stelle wieder auf. Immer, wenn 2 Sekunden vergangen sind und der Stern nicht berührt wurde, soll er verschwinden und woanders im Bild wieder auftauchen. (Tipp: Benutze hierfür eine Stoppuhr. Du findest diese in der Kategorie: Fühlen. Achte darauf, dass die Stoppuhr auch wieder zurückgestellt wird). Jedes Spiel soll nur eine Minute dauern. Wenn die Zeit abgelaufen ist, soll das Spiel gestoppt werden (Tipp: Erstelle dafür eine Variable)

Gruppe 5: Jump'N'Run- Spiel

Programmiere ein Spiel, bei der die Katze entlang einer Straße läuft. Die Katze soll mittels Pfeiltasten gesteuert werden. Wird, die Leertaste gedrückt, springt die Katze hoch und landet wieder an derselben Stelle. Auf die Katze kommen immer wieder Autos zu. Die Katze muss über diese Autos drüber springen. Berührt die Katze das Auto, ist das Spiel vorbei.

Tipps zur Programmierung:

- Erstelle nur ein Auto und klont dieses. Erst, wenn der Code für ein Auto programmiert wurde, wird er geklont.
- Die Autos sollen verschwinden, wenn sie die Wand berühren

Gruppe 6: Ping-Pong

Programmiere ein Spiel, bei dem du mit einem Balken einen Ball steuerst. Der Ball fällt vom Himmel und kann mittels Balken aufgefangen werden. Berührt der Ball den Balken, soll er nach oben hin wegprallen. Berührt der Ball den Spielfeldrand, prallt er ab. Berührt der Ball den Boden, hat man verloren und das Spiel ist vorbei. Der Balken soll dabei mittels Mausbewegung gesteuert werden. Definiere dazu den Y-Wert des Balkens zu Beginn.

8.1.2.2 Role description pair programming

ROLLENVERTEILUNG DRIVER – NAVIGATOR

Driver:

- Person programmiert aktiv und konzentriert sich darauf kleine, unmittelbare Ziele zu erreichen und ignoriert zunächst größere Problem
- Kommuniziert mit Navigator die einzelnen Arbeitsschritte (was mache ich gerade)

Navigator:

- Beobachte die Arbeit des Drivers
- Überprüfe fortlaufend den Code
- Gib Verbesserungsvorschläge und teile deine Gedanken
- Behalte das große Ganze im Blick und arbeite darauf hin.
- Mache dir Notizen, bei gröberen Probleme

Wechselt nach 20 Minuten die Rollen

8.1.2.3 Task descriptions Python

Aufgaben Programmierung mit Python)

1) Schreibe ein Programm, welches Celsius in Fahrenheit umwandelt bzw. Fahrenheit in Celsius:
    a. Das Programm soll zunächst danach fragen, ob Celsius in Fahrenheit (1) oder Fahrenheit in Celsius (2) umgerechnet werden soll.
    b. Danach soll der User die entsprechende Zahl eingeben.
    c. Auf Basis dessen soll die Berechnung im Hintergrund durchgeführt werden.
    d. Das Programm soll ausgeben. X Grad Celsius entsprechen Y Grad Fahrenheit bzw. X Grad Fahrenheit entsprechen Y Grad Celsius

    Tipp:

    - Celsius = 5/9 * (Fahrenheit - 32).
    - Fahrenheit = (9/5 × Celsius) + 32.
    - Benutze zur Fallunterscheidung die if-else Verzweigung.

2) Schreibe ein Programm, welches die Zahlen von 1 bis zu einem beliebigen Wert summiert.
    a. Das Programm soll zunächst danach fragen, bis zu welcher Zahl summiert werden soll.
    b. Das Programm soll danach alle Zahlen inklusive dem genannten Wert erhöhen.
    c. Das Programm soll ausgeben: „Die Summe der Zahlen von 1 bis X beträgt Y“.

    Tipp:

    - Die Verwendung einer while Schleife kann hier sehr hilfreich sein!

3) Schreibe ein Programm, das eingibt, wie oft ein bestimmter Buchstabe in den Vornamen eurer Klassenkolleg:innen vorkommt.
    a. Das Programm soll zunächst nach den Vornamen der Klasse fragen.
        i. Speichere die Vorname in einem sogenannten Array. Diesen kannst du erzeugen, indem du zB folgendes schreibst:
        ```
        namen = ["Susi", "Max", "Peter"]
        ```
    b. Das Programm soll dann danach fragen, welcher Buchstabe gezählt werden soll.
    c. Nachdem das Programm dann durch den Array durchgelaufen ist und gezählt hat, wie oft der gewählte Buchstabe vorkommt, soll es folgende Meldung ausgeben: „Der Buchstabe „X“ kommt Y-mal in unserer Klasse vor“

    Tipp:

    - Die Einträge im Array kannst du mit einer Schleife durchlaufen. Beachte, dass immer mit 0 zu zählen begonnen wird (erster Eintrag = 0-ter Eintrag).
    - Um innerhalb der Einträge die Buchstaben durchzulaufen, brauchst du eine weitere Schleife
    - Erstelle eine Variable, die zählt, wie oft ein Buchstabe vorkommt und immer dann erhöht wird, wenn der Buchstabe im Wort vorkommt.
    - Damit auch Großbuchstaben erkannt werden, gibt es den Befehl lower(), also zum Beispiel:
      name = „Susi“ → name.lower() = „susi“

8.1.3 Final quiz

1. Du möchtest einen Befehl in deinem Code öfters durchführen, welches Programmierkonzept, dass wir in den letzten Stunden besprochen haben, eignet sich, um dies zu realisieren? „Schleife“
2. Du möchtest eine Variable in Python erstellen, welche der folgenden Optionen ist korrekt?
    a. variable x = 10
    b. x := 10
    c. x = 10
    d. def variable 10
3. Mit welchem Befehl in Python erstellst du eine Bedingung (wenn x, dann y)? Gib bitte nur den Befehl an und nicht den gesamten Code! If
4. Der folgende Befehl in Python überprüft, ob die Variable x den Wert 5 enthält: x = 5
    a. Richtig
    b. Falsch
5. Ordne die Scratch Blöcke der jeweiligen Funktionsweise zu:

1	If ____- then _____- els ….	2	Wiederholt einen Block solange, bis eine bestimmte Bedingung eintrifft
2	Repeat until _____	4	Stellt eine Nachricht am Bildschirm da
3	Set x to _____	3	Speichert eine Variable x mit einem bestimmten Wert
4	Say „Hello“	1	Führt einen Block unter einer bestimmten Bedingung aus

6. Was gibt der folgende Python code aus? A

```
x = 3
if x > 2:
  print („A“)
else:
  print („B“)
```

7. Welchen der folgenden Variblentypen gibt es in Scratch NICHT?
    a. Zahl
    b. Liste
    c. Boolean
    d. Funktion

8. Warum funktioniert der folgende Python code nicht? Es fehlt der Doppelpunkt nach der Klammer

   ```
   def add_number (a, b)
     return a+b
   ```

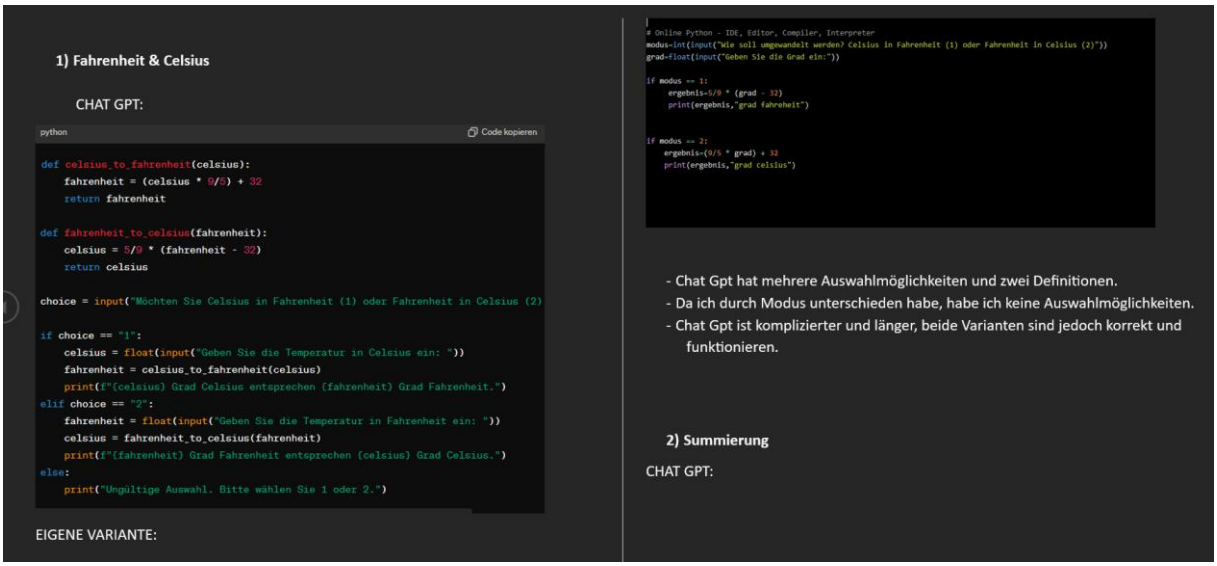
9. Was ist der Hauptzweck einer Funktion beim Programmieren?
   a. Um einen einzelnen Wert zu speichern
   b. Um mehrere Befehle mehrfach auszuführen, ohne diese im Code wiederholen zu müssen
   c. Um eine Schleife zu erzeugen
   d. Um eine neue Variable zu erzeugen
10. Was ist der Unterschied zwischen einer Bedingung (IF) und einer Schleife (while)?
    a. Eine Bedingung führt den Code mehrfach aus, während eine Schleife den Code nur einmal ausführt
    b. Eine Schleife überprüft eine Bedingung und läuft nur, wenn die Bedingung wahr ist, während eine Bedingung (If-Anweisung) endlos läuft
    c. Eine Schleife wiederholt einen Code mehrfach, während eine Bedingung einen Code-teil nur dann ausführt, wenn die Bedingung auch erfüllt wird
    d. Eine Schleife und eine Bedingung sind dasselbe und werden nur anders geschrieben.

8.2 Research Materials

8.2.1 Self-reflection-sheets

In the following chapter, examples of reflection sheets used for this research are displayed. If there is a request for all reflection sheets, I am ready to provide those by getting in touch with me.

8.2.1.1 Self-reflection-sheets ChatGPT vs me (Examples)



python

```python
# Vornamen der Klasse als Array
namen = ["Susi", "Max", "Peter", "Lisa", "Anna", "Tom"]

# Funktion zur Zählung der Buchstaben
def count_letter_in_names(letter, names):
    count = 0
    for name in names:
        for char in name.lower():  # Um Groß- und Kleinbuchstaben zu berücksichtigen
            if char == letter.lower():
                count += 1
    return count

# Benutzereingabe
letter = input("Für welchen Buchstaben möchtest du die Häufigkeit prüfen? ")

# Zählfunktion aufrufen
letter_count = count_letter_in_names(letter, namen)

# Ausgabe
print(f"Der Buchstabe \"{letter}\" kommt {letter_count}-mal in unserer Klasse vor.")
```

EIGENE

VARIANTE:

```
namensliste = [];
name = input ("Geben Sie einen Namen oder geben Sie - ein falls Sie keinen Namenmehr eingeben wollen")
namensliste.append(name);
suchbuchstabe = input("Geben Sie den Buchstaben ein nachdem gesucht werden soll");
count=0;

while name != "-":
    name =input("Geben Sie einen Namen ein oder geben Sie - ein falls Sie keinen Namen mehr eingeben wollen")
    namensliste.append(name);


for name in namensliste:
   for buchstabe in name:
      if buchstabe == suchbuchstabe.lower():
         count= count+1;
         |
print("Der Buchstabe" , suchbuchstabe, "kommt in eure Klasse" , count, "mal vor.");
```

- Im Programm von ChatGPT sind die Namen schon vorgegeben, somit kann man später keine eigenen Namen auswählen.
- Beide Programme funktionieren jedoch und sind auch verständlich.

Ende des Dokuments

ChatGPT:

Temperatur:

Eigenes:

```
print("Wollen Sie Fahrenheit in Celsius oder Celsius in Fahrenheit umwandeln? Wenn Sie die erste Möglichkeit haben wollen, geben Sie 1 an, wenn Sie die zweite Möglichkeit hab

modus=int(input("Modus"))
grad=int(input("Geben Sie die Grad ein"))

if modus==1:
    ergebnis=5/9*   (grad-32)
    print(grad," Fahrenheit entsprechen ",ergebnis," Celsius ")

else:
    ergebnis=(9/5*Celsius)+32
    print(grad," Celsius entsprechen ",ergebnis," Fahrenheit ");
```

ChatGPT

```
def celsius_to_fahrenheit(celsius):
    return (celsius * 9/5) + 32

def fahrenheit_to_celsius(fahrenheit):
    return (fahrenheit - 32) * 5/9

choice = int(input("Möchten Sie Celsius in Fahrenheit umwandeln (1) oder Fahrenheit in Celsius (2)? "))

if choice == 1:
    celsius = float(input("Geben Sie die Temperatur in Celsius ein: "))
    fahrenheit = celsius_to_fahrenheit(celsius)
    print(f"{celsius} Grad Celsius entsprechen {fahrenheit} Grad Fahrenheit.")
elif choice == 2:
    fahrenheit = float(input("Geben Sie die Temperatur in Fahrenheit ein: "))
    celsius = fahrenheit_to_celsius(fahrenheit)
    print(f"{fahrenheit} Grad Fahrenheit entsprechen {celsius} Grad Celsius.")
else:
    print("Ungültige Eingabe. Bitte wählen Sie 1 oder 2.")
```

ChatGPT macht die Programmierung schwieriger und mehr. Das eigene Programm ist sowohl übersichtlicher als auch einfacher gekennzeichnet. Im Pogramm von ChatGPT sind die If und Else Clauses überflüssig. - Würde meine Version nehmen.

Zahlen:

Eigenes:

```
zahl= int(input(" Bis zu welcher welcher beliebigen Zahl wollen Sie summieren? "))
erhhung= int(input(" Um wie viel sollen die Zahlen erhöht werden? "))
counter = 1
summe = 0

while counter <= zahl:
    summe += counter
    counter += 1

print("Die Summe der Zahlen von 1 bis", zahl, "inklusive der Erhöhung um", "erhöhung", "beträgt", summe)
```

Auch hier schreibt ChatGPT ein komplizierteres Programm. Außerdem funktioniert dieser auch nicht richtig, da jede Zahl, die man eingibt mit 1 addiert wird, obwohl dies nicht die Aufgabenstellung war. Also würde ich diesmal wieder mein Programm wählen.

ChatGPT:

```
def sum_numbers_up_to(target):
    total = 0
    current_number = 1
    while current_number <= target:
        total += current_number
        current_number += 1
    return total

target = int(input("Bis zu welcher Zahl sollen die Zahlen summiert werden? "
target += 1  # Erhöhe die Zielzahl um 1, um sie einzuschließen
total_sum = sum_numbers_up_to(target)
print(f"Die Summe der Zahlen von 1 bis {target - 1} beträgt {total_sum}.")
```

Eigenes:

```
import array
namensliste= [];
name= input("Nenne mir die Vornamen deiner Klassenkollegi/innen oder gib - ein, falls du keinen mehr Namen mehr eingeben willst")
namensliste.append(name);
suchbuchstabe = input("Gib den Buchstaben ein, nachd dem gesucht werden soll")
count = 0;
while name != "-":
        name = input("Nenne mir die Vornamen deiner Klassenkollegi/innen oder gib - ein, falls du keinen mehr Namen mehr eingeben willst")
        namensliste.append(name);
for name in namensliste:
    for buchstabe in name:
        if buchstabe.lower() == suchbuchstabe.lower():
            count = count+1;
print("Der Buchstabe", suchbuchstabe, "kommt in eurer Klasse", count, "mal vor");
```

Beide Programme sind gleich “viel”, jedoch funktioniert das von ChatGPT gar nicht. Also würde ich meines nehmen.

ChatGPT:

```
names = []  # Array für die Vornamen

# Eingabe der Vornamen
num_names = int(input("Wie viele Namen gibt es in der Klasse? "))
for i in range(num_names):
    name = input(f"Geben Sie den {i+1}. Namen ein: ")
    names.append(name)

# Eingabe des Buchstabens, der gezählt werden soll
letter = input("Welcher Buchstabe soll gezählt werden? ").lower()

# Zähler für das Vorkommen des Buchstabens initialisieren
letter_count = 0

# Durchlaufen der Namen und Zählen des Vorkommens des Buchstabens
for name in names:
    for char in name.lower():  # Um auch Großbuchstaben zu erkennen, wird alles in Kleinbuchstaben umgewandelt
        if char == letter:
            letter_count += 1

# Ausgabe des Ergebnisses
print(f"Der Buchstabe '{letter}' kommt {letter_count}-mal in unserer Klasse vor.")
```

Ende des Dokuments

CHAT GPT VS ME

Aufgabe 1 hat mit Chat Gpt funktioniert. Er hat den Code kürzer und einfacher zusammengefasst ohne viele Variablen oder Input Abfragen zu benutzen. Er hat mehrere Input abfragen von mir in eine zusammengefasst.

Chat GPT

```
def celsius_to_fahrenheit(celsius):
    fahrenheit = (celsius * 9/5) + 32
    return fahrenheit

# Eingabe der Temperatur in Celsius vom Benutzer
celsius = float(input("Geben Sie die Temperatur in Celsius ein: "))

# Umwandlung von Celsius in Fahrenheit
fahrenheit = celsius_to_fahrenheit(celsius)

# Ausgabe des Ergebnisses
print(f"{celsius} Grad Celsius entsprechen {fahrenheit} Grad Fahrenheit.")
```

Mein Code

```
def celsius_to_fahrenheit(celsius):
    return (celsius * 9/5) + 32

def fahrenheit_to_celsius(fahrenheit):
    return (fahrenheit - 32) * 5/9

def main():
    choice = int(input("Möchten Sie Celsius in Fahrenheit (1) oder Fahrenheit in Celsius (2) umwandeln? Geben Sie 1 oder 2 ein: "))
    if choice == 1:
        celsius = float(input("Geben Sie die Temperatur in Celsius ein: "))
        fahrenheit = celsius_to_fahrenheit(celsius)
        print(f"{celsius} Grad Celsius entsprechen {fahrenheit} Grad Fahrenheit.")
    elif choice == 2:
        fahrenheit = float(input("Geben Sie die Temperatur in Fahrenheit ein: "))
        celsius = fahrenheit_to_celsius(fahrenheit)
        print(f"{fahrenheit} Grad Fahrenheit entsprechen {celsius} Grad Celsius.")
    else:
        print("Ungültige Eingabe. Bitte geben Sie entweder 1 oder 2 ein.")
if __name__ == "__main__":
    main()
```

Bei Aufgabe 2 hat er eine Funktion mehr gemacht als ich. Er hat mehr Variablen benutzt und einen Input weniger benutzt. Er hat eine Return Variable mehr benutzt als ich.

Chat Gpt

```
def sum_numbers_up_to(target):
    total = 0
    current_number = 1
    while current_number <= target:
        total += current_number
        current_number += 1
    return total

def main():
    target = int(input("Bis zu welcher Zahl sollen die Zahlen summiert werden? "))
    target = target + 1  # Erhöhe die Zielzahl um 1
    total = sum_numbers_up_to(target)
    print(f"Die Summe der Zahlen von 1 bis {target - 1} beträgt {total}.")

if __name__ == "__main__":
    main()
```

Mein Code

```
def sum_numbers(up_to):
    total = 0
    current_number = 1
    while current_number <= up_to:
        total += current_number
        current_number += 1
    return total

def main():
    up_to = int(input("Bis zu welcher Zahl möchten Sie die Zahlen summmieren? Geben Sie eine positive ganze Zahl ein: "))

    if up_to <= 0:
        print("Bitte geben Sie eine positive ganze Zahl ein.")
        return

    total_sum = sum_numbers(up_to)
    print(f"Die Summe der Zahlen von 1 bis {up_to} beträgt {total_sum}.")

if __name__ == "__main__":
    main()
```

Bei Aufgabe 3 hat er zuerst einen Input gemacht in dem man die Anzahl der Klassenkollegen eingeben muss. Erst danach fragt er nach den Namen.

Mein Code

```
import array # Array Funktionalität integrieren

namensliste = [];
name = input("Geben Sie einen Namen ein oder geben Sie - ein, falls Sie keinen Namen mehr eingeben wollen")
namensliste.append(name);

count = 0;

while name != "-":
    name = input("Geben Sie einen Namen oder geben Sie - ein, falls Sie keinen Namen eingeben wollen")
    namensliste.append(name);

suchbuchstabe = input("Geben Sie den Buchstaben ein nachdem gesucht werden soll: ")
```

```
for name in namensliste:
    for buchstabe in name:
        if buchstabe.lower() == suchbuchstabe.lower():
            count = count+1
print("Der Buchstabe", suchbuchstabe, "kommt in eurer Klasse", count, "mal vor")
```

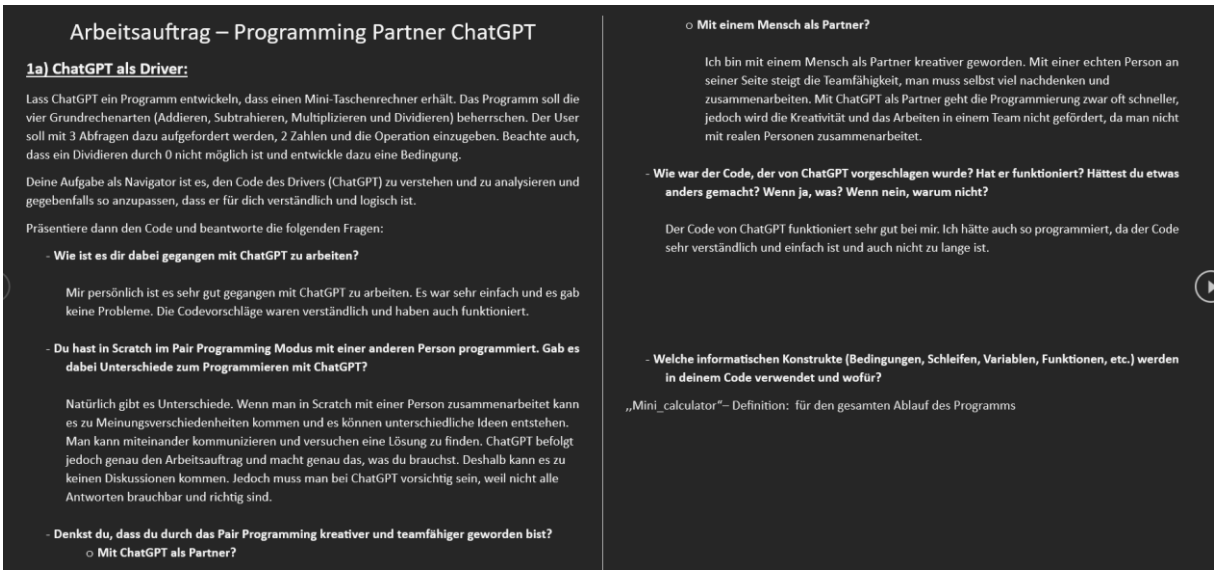
Chat Gpt

```
def count_letters(names, letter):
    count = 0
    for name in names:
        for char in name:
            if char.lower() == letter.lower():
                count += 1
    return count

namensliste = []

while True:
    name = input("Geben Sie einen Namen ein oder geben Sie '-' ein, um die Eingabe zu beenden: ")
    if name == "-":
        break
    elif name.strip():  # Überprüfen, ob der Name nicht leer ist
        namensliste.append(name)

if not namensliste:
    print("Keine Namen eingegeben.")
else:
    suchbuchstabe = input("Geben Sie den Buchstaben ein, nach dem gesucht werden soll: ")

    if len(suchbuchstabe) != 1 or not suchbuchstabe.isalpha():
        print("Ungültige Eingabe für den Buchstaben.")
```

```
    count = count_letters(namensliste, suchbuchstabe)
    print("Der Buchstabe", suchbuchstabe, "kommt in eurer Klasse", count, "mal vor.")
```

Ende des Dokuments

8.2.1.2 Self-reflection on pair programming (examples)



Arbeitsauftrag – Programming Partner ChatGPT

1a) ChatGPT als Driver:

Lass ChatGPT ein Programm entwickeln, dass einen Mini-Taschenrechner erhält. Das Programm soll die vier Grundrechenarten (Addieren, Subtrahieren, Multiplizieren und Dividieren) beherrschen. Der User soll mit 3 Abfragen dazu aufgefordert werden, 2 Zahlen und die Operation einzugeben. Beachte auch, dass ein Dividieren durch 0 nicht möglich ist und entwickle dazu eine Bedingung.

Deine Aufgabe als Navigator ist es, den Code des Drivers (ChatGPT) zu verstehen und zu analysieren und gegebenfalls so anzupassen, dass er für dich verständlich und logisch ist.

Präsentiere dann den Code und beantworte die folgenden Fragen:

- **Wie ist es dir dabei gegangen mit ChatGPT zu arbeiten?**

  Mir persönlich ist es sehr gut gegangen mit ChatGPT zu arbeiten. Es war sehr einfach und es gab keine Probleme. Die Codevorschläge waren verständlich und haben auch funktioniert.

- **Du hast in Scratch im Pair Programming Modus mit einer anderen Person programmiert. Gab es dabei Unterschiede zum Programmieren mit ChatGPT?**

  Natürlich gibt es Unterschiede. Wenn man in Scratch mit einer Person zusammenarbeitet kann es zu Meinungsverschiedenheiten kommen und es können unterschiedliche Ideen entstehen. Man kann miteinander kommunizieren und versuchen eine Lösung zu finden. ChatGPT befolgt jedoch genau den Arbeitsauftrag und macht genau das, was du brauchst. Deshalb kann es zu keinen Diskussionen kommen. Jedoch muss man bei ChatGPT vorsichtig sein, weil nicht alle Antworten brauchbar und richtig sind.

- **Denkst du, dass du durch das Pair Programming kreativer und teamfähiger geworden bist?**
  - **Mit ChatGPT als Partner?**
  - **Mit einem Mensch als Partner?**

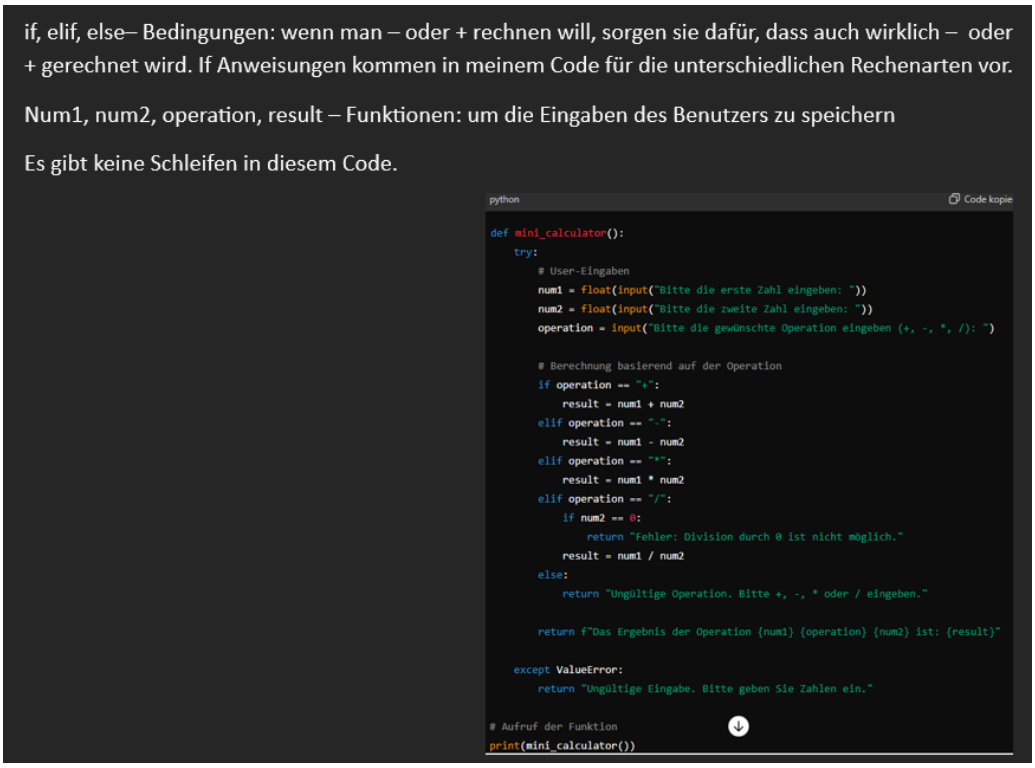
    Ich bin mit einem Mensch als Partner kreativer geworden. Mit einer echten Person an seiner Seite steigt die Teamfähigkeit, man muss selbst viel nachdenken und zusammenarbeiten. Mit ChatGPT als Partner geht die Programmierung zwar oft schneller, jedoch wird die Kreativität und das Arbeiten in einem Team nicht gefördert, da man nicht mit realen Personen zusammenarbeitet.

- **Wie war der Code, der von ChatGPT vorgeschlagen wurde? Hat er funktioniert? Hättest du etwas anders gemacht? Wenn ja, was? Wenn nein, warum nicht?**

  Der Code von ChatGPT funktioniert sehr gut bei mir. Ich hätte auch so programmiert, da der Code sehr verständlich und einfach ist und auch nicht zu lange ist.

- **Welche informatischen Konstrukte (Bedingungen, Schleifen, Variablen, Funktionen, etc.) werden in deinem Code verwendet und wofür?**

„Mini_calculator"– Definition: für den gesamten Ablauf des Programms



if, elif, else– Bedingungen: wenn man – oder + rechnen will, sorgen sie dafür, dass auch wirklich – oder + gerechnet wird. If Anweisungen kommen in meinem Code für die unterschiedlichen Rechenarten vor.

Num1, num2, operation, result – Funktionen: um die Eingaben des Benutzers zu speichern

Es gibt keine Schleifen in diesem Code.

python — Code kopie

```python
def mini_calculator():
    try:
        # User-Eingaben
        num1 = float(input("Bitte die erste Zahl eingeben: "))
        num2 = float(input("Bitte die zweite Zahl eingeben: "))
        operation = input("Bitte die gewünschte Operation eingeben (+, -, *, /): ")

        # Berechnung basierend auf der Operation
        if operation == "+":
            result = num1 + num2
        elif operation == "-":
            result = num1 - num2
        elif operation == "*":
            result = num1 * num2
        elif operation == "/":
            if num2 == 0:
                return "Fehler: Division durch 0 ist nicht möglich."
            result = num1 / num2
        else:
            return "Ungültige Operation. Bitte +, -, * oder / eingeben."

        return f"Das Ergebnis der Operation {num1} {operation} {num2} ist: {result}"

    except ValueError:
        return "Ungültige Eingabe. Bitte geben Sie Zahlen ein."

# Aufruf der Funktion
print(mini_calculator())
```

Arbeitsauftrag – Programming Partner ChatGPT

1a) ChatGPT als Driver:

Lass ChatGPT ein Programm entwickeln, dass einen Mini-Taschenrechner erhält. Das Programm soll die vier Grundrechenarten (Addieren, Subtrahieren, Multiplizieren und Dividieren) beherrschen. Der User soll mit 3 Abfragen dazu aufgefordert werden, 2 Zahlen und die Operation einzugeben. Beachte auch, dass ein Dividieren durch 0 nicht möglich ist und entwickle dazu eine Bedingung.

Deine Aufgabe als Navigator ist es, den Code des Drivers (ChatGPT) zu verstehen und zu analysieren und gegebenfalls so anzupassen, dass er für dich verständlich und logisch ist.

Präsentiere dann den Code und beantworte die folgenden Fragen:

- Wie ist es dir dabei gegangen mit ChatGPT zu arbeiten?
  Es war super, da ChatGPT durch die Fragestellung das Erwartete direkt übernommen und ein Programm entwickelt hat, das einwandfrei funktioniert hat.

- Du hast in Scratch im Pair Programming Modus mit einer anderen Person programmiert. Gab es dabei Unterschiede zum Programmieren mit ChatGPT?
  Es lässt sich sagen, dass das Programmieren mit ChatGPT einfacher, aber dafür auch viel monotoner ist. Man gibt die Aufgabenstellung ein und bekommt direkt ein Ergebnis. Gleichzeitig muss man ohne KI wirklich nachdenken und sich anstrengen. Man muss gemeinsam auf Lösungen kommen und zusammenarbeiten, da es sonst nicht funktioniert.
  Beide Programme sind in ihrer eigenen Art anspruchsvoll, jedoch empfand ich Scratch aufgrund der vorgegebenen Module übersichtlicher und hat mir besser gefallen.

- Denkst du, dass du durch das Pair Programming kreativer und teamfähiger geworden bist?
  - Mit ChatGPT als Partner?
    Nicht wirklich, man musste kaum überlegen und einfach die Aufgabenstellung eintippen. Falls Fehler aufliegen muss man diese dann noch verbessern oder fragt einfach weiterhin nach Lösungsvorschläge. Also keine Teamarbeit und kaum denken.
  - Mit einem Mensch als Partner?
    Es ist definitiv kreativer mit einer anderen Person, weil man sich Lösungen überlegen muss, falls etwas nicht klappt und nicht einfach von ChatGPT kopiert. Teamgeist ist dafür auch notwendig, um neue Lösungsvorschläge und einen Ansatz für die Programmierung zu finden.

- Wie war der Code, der von ChatGPT vorgeschlagen wurde? Hat er funktioniert? Hättest du etwas anders gemacht? Wenn ja, was? Wenn nein, warum nicht?
  Der erste angezeigte Code von ChatGPT hat direkt ohne Probleme funktioniert. Die Angabe wurde wirklich so umgesetzt und auch so Kleinigkeiten wie, dass man nicht durch 0 dividieren kann wurden berücksichtigt. An sich wirkt der Code verständlich und ich persönlich kann den völlig nachvollziehen. Ich hätte diesen jedoch etwas kompakter und verkürzt gemacht, da sehr viele einzelne Operatoren benutzt wurden.

- Welche informatischen Konstrukte (Bedingungen, Schleifen, Variablen, Funktionen, etc.) werden in deinem Code verwendet und wofür?
  Es wurden Operatoren eingesetz, um die einzelenen Komponenten wie *, +,- , / auszudrücken. Außerdem wurde, falls 0 zum dividieren verwendet wird (If), return verwendet um eine ungültige Operation auszudrücken. Durch die Bneutzereingaben (num1, num2) wir dem Benutzer die Frage gestellt, was er eingeben will und dazu das Resultat „print" (Ergebnis).

```
        if num2 == 0:
            return "Fehler: Division durch 0 ist nicht möglich!"
        else:
            return num1 / num2
    else:
        return "Ungültige Operation!"

# Hauptprogramm
def main():
    try:
        # Benutzereingaben
        num1 = float(input("Geben Sie die erste Zahl ein: "))
        num2 = float(input("Geben Sie die zweite Zahl ein: "))
        operation = input("Geben Sie die gewünschte Operation ein (+, -, *, /): ")

        # Berechnung durchführen und Ergebnis ausgeben
        result = taschenrechner(num1, num2, operation)
        print("Das Ergebnis ist:", result)
    except ValueError:
        print("Fehler: Bitte geben Sie gültige Zahlen ein.")

if __name__ == "__main__":
    main()
```

```
def taschenrechner(num1, num2, operation):
    if operation == '+':
        return num1 + num2
    elif operation == '-':
        return num1 - num2
    elif operation == '*':
        return num1 * num2
    elif operation == '/':
        if num2 == 0:
            return "Fehler: Division durch 0 ist nicht möglich!"
        else:
            return num1 / num2
    else:
        return "Ungültige Operation!"

# Hauptprogramm
def main():
    try:
        # Benutzereingaben
        num1 = float(input("Geben Sie die erste Zahl ein: "))
        num2 = float(input("Geben Sie die zweite Zahl ein: "))
        operation = input("Geben Sie die gewünschte Operation ein (+, -, *, /): ")
```

Arbeitsauftrag – Programming Partner ChatGPT

1b) ChatGPT als Navigator:

Präsentiere dann den Code und beantworte die folgenden Fragen:

- Wie ist es dir dabei gegangen mit ChatGPT zu arbeiten?
- Sehr gut!
- Auf Wunsch hat er Sachen hinzugefügt und verbessert, jedoch manchmal zu kompliziert und nicht gleich das erfüllt was ich von ihm haben wollte.
-
- Du hast in Scratch im Pair Programming Modus mit einer anderen Person programmiert. Gab es dabei Unterschiede zum Programmieren mit ChatGPT?
- Ja!
- Mit anderen realen Personen kann man sich schneller austauschen und seine Meinungen sagen, jedoch hat ChatGpt immer Verbesserungsideen und kann es immer wieder verbessern und in wenigen Sekunden erneuern.
-
- Denkst du, dass du durch das Pair Programming kreativer und teamfähiger geworden bist?
    - Mit ChatGPT als Partner?
      Nein, da entweder ChatGpt oder ich programmiert haben und nicht gleichzeitig unsere Ideen austauschen konnten.
    - Mit einem Mensch als Partner?
      Ja, da wir viel miteinander geredet und diskutiert haben. Uns gegenseitig geholfen und ausgebessert haben. Dadurch sind wir besser beim Programmieren geworden und gleichzeitig auch teamfähiger.
- Welche Änderungen und Optimierungsvorschläge gab es von ChatGPT? Hast du die Vorschläge angenommen? Warum? Warum nicht? Haben die Anpassungen funktioniert?
- Er hat den Code funktionierend gemacht und Fehler behoben, manche Konstruktionen hat er vereinfacht und hinzugefügt. Ich habe die meisten Verbesserungsvorschläge angenommen haben, da sie hilfreich und optimierend waren.
-
- Welche informatischen Konstrukte (Bedingungen, Schleifen, Variablen, Funktionen, etc.) werden in deinem Code verwendet und wofür?
- Variable-Speichern die Benutzereingaben
- Funktionen-Führen die spezifischen Rechenoperationen aus und machen den Code übersichtlicher.
- Bedinungen-Wählen die richtige Rechenoperation basierend auf die Benutzereingabe.
- Eingabe und Ausgabe- input- erfasst die Benutzereingaben
- Print- zeigt das Ergebn9s der Berechnung an

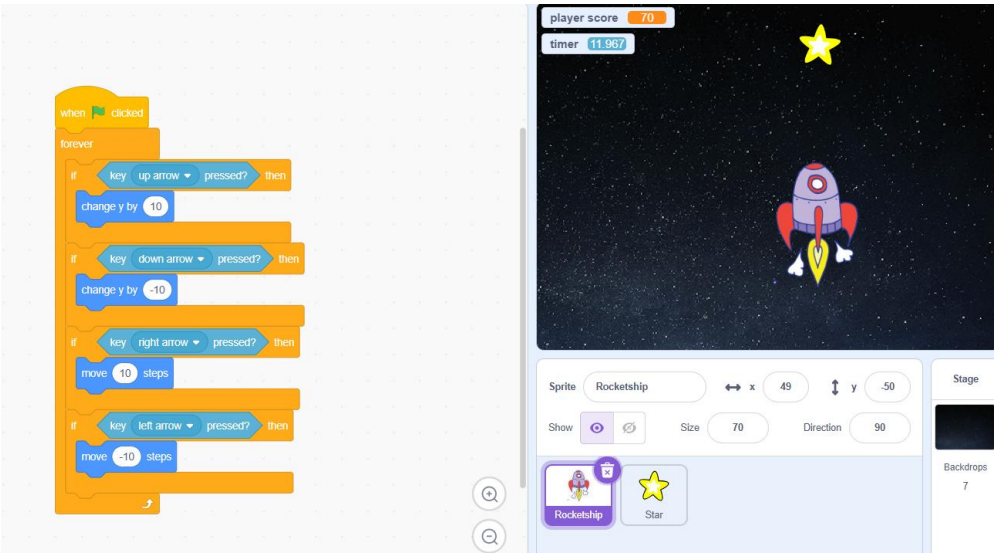
Ende des Dokuments

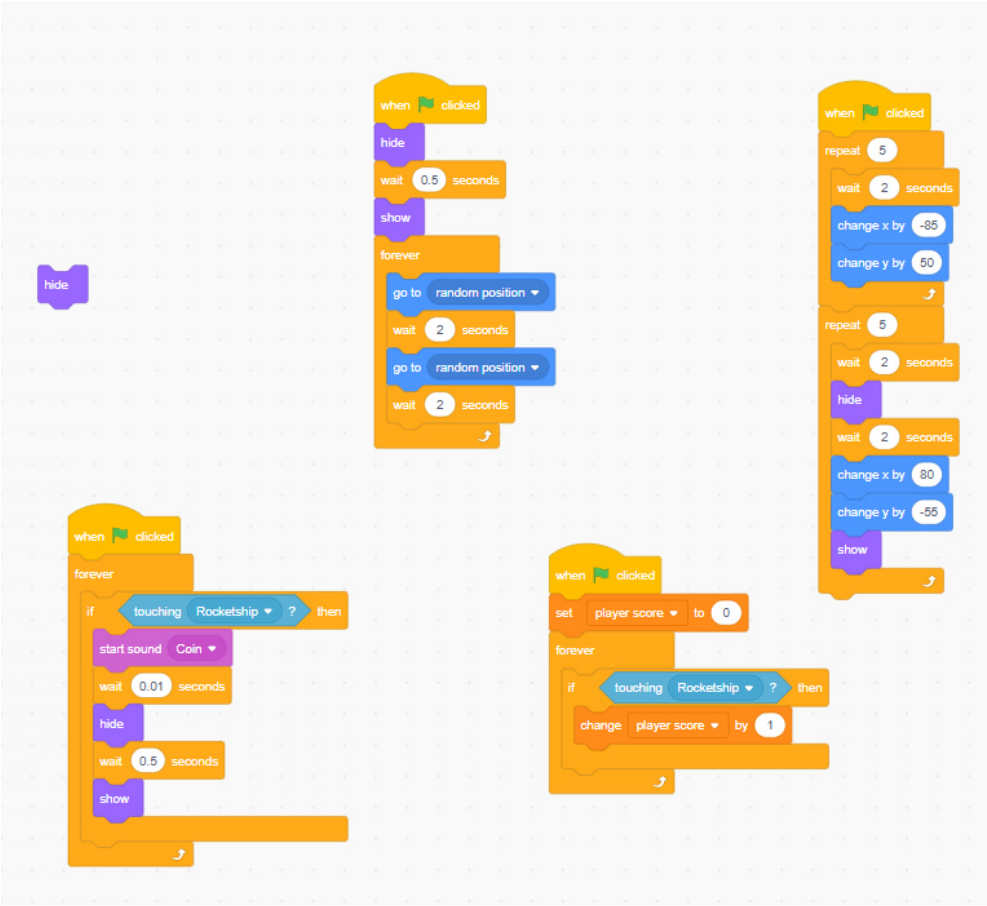
8.2.2 Programming assignments of students

In the following chapter, examples of the programs developed by the students and used for this research are displayed. If there is a request for all programs, I am ready to provide those by getting in touch with me.

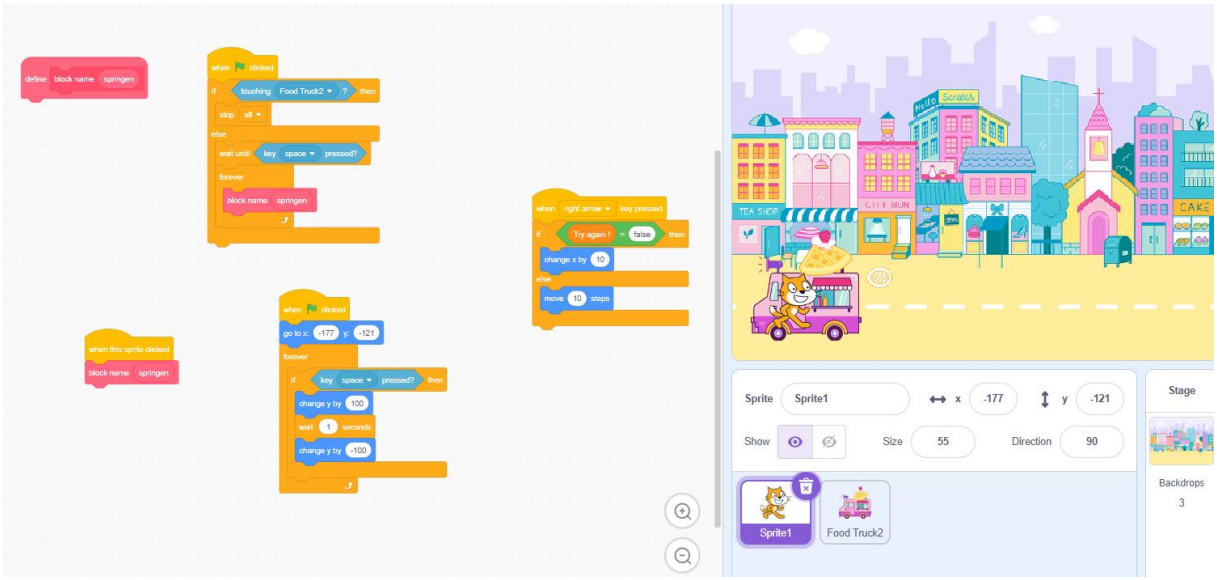
8.2.2.1 Scratch

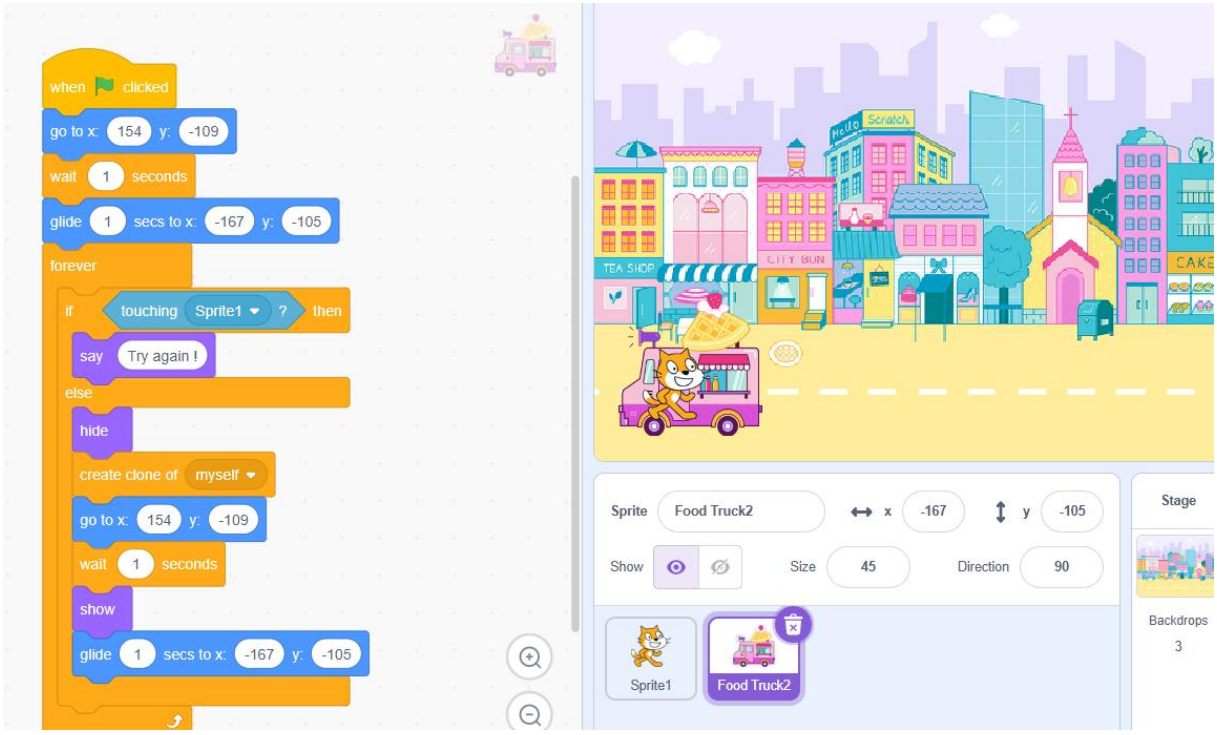
8.2.2.1.1 Game1



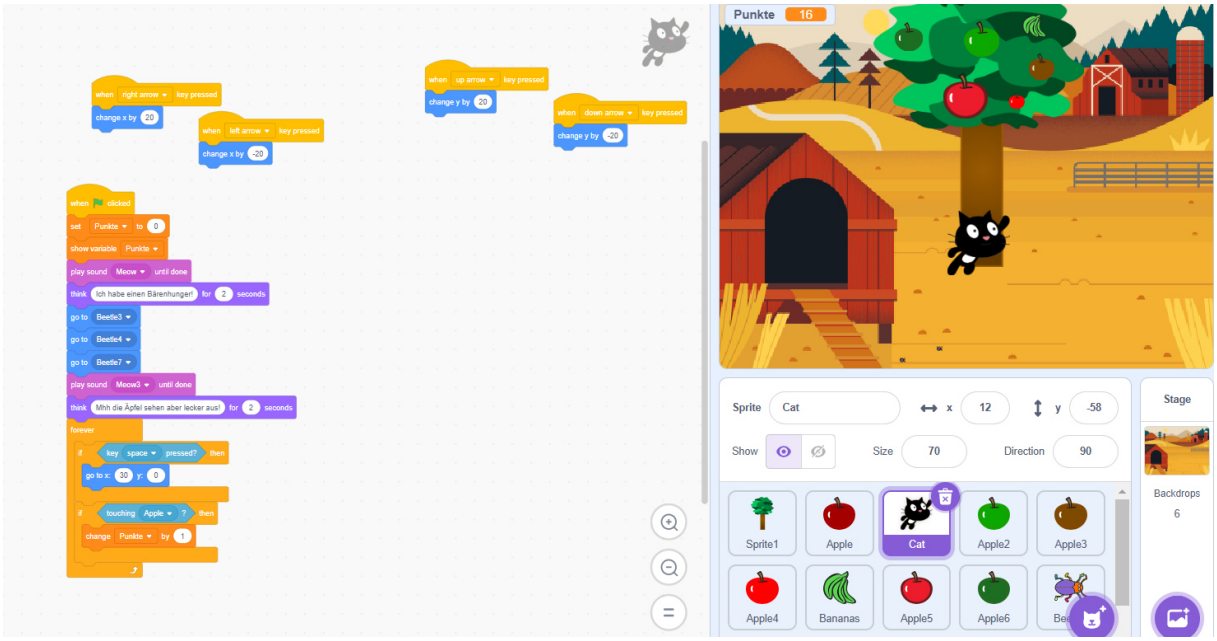


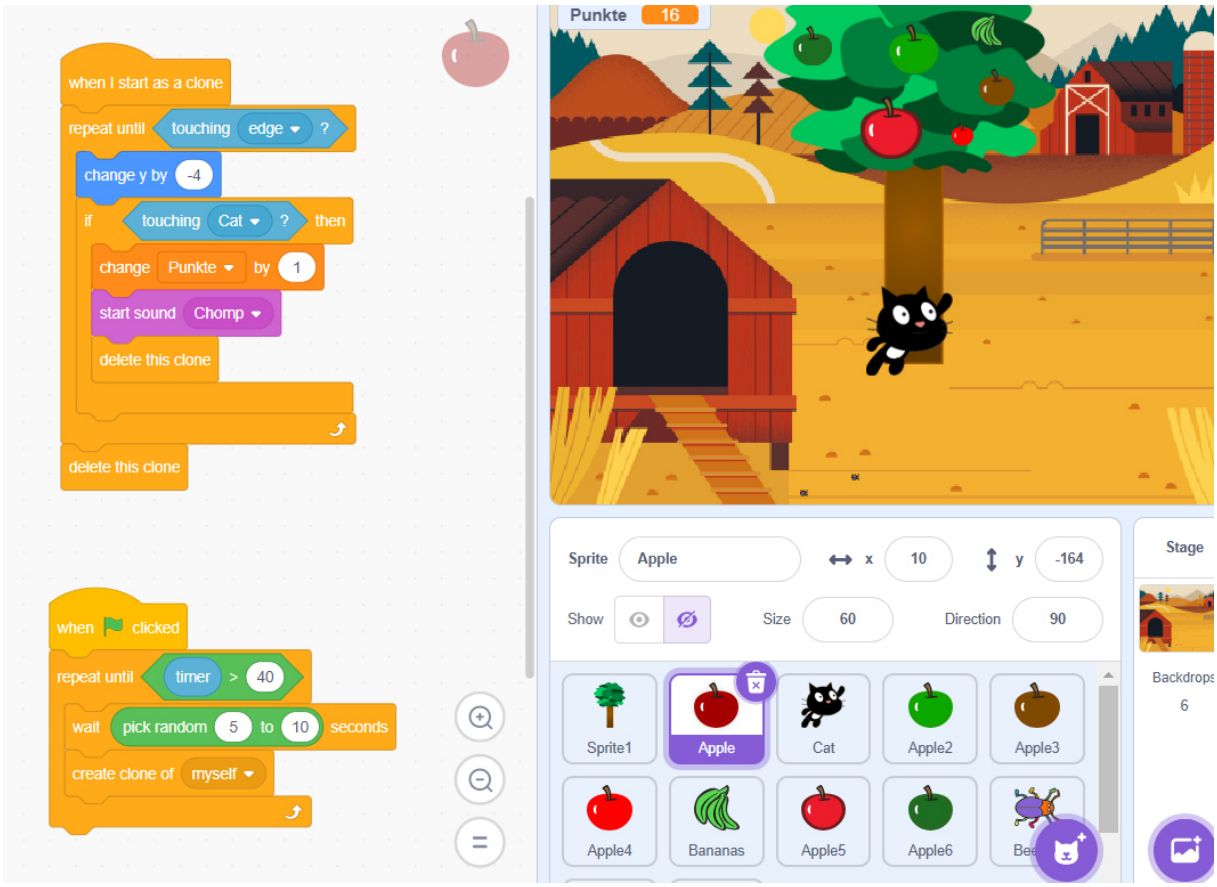
8.2.2.1.2 Game2

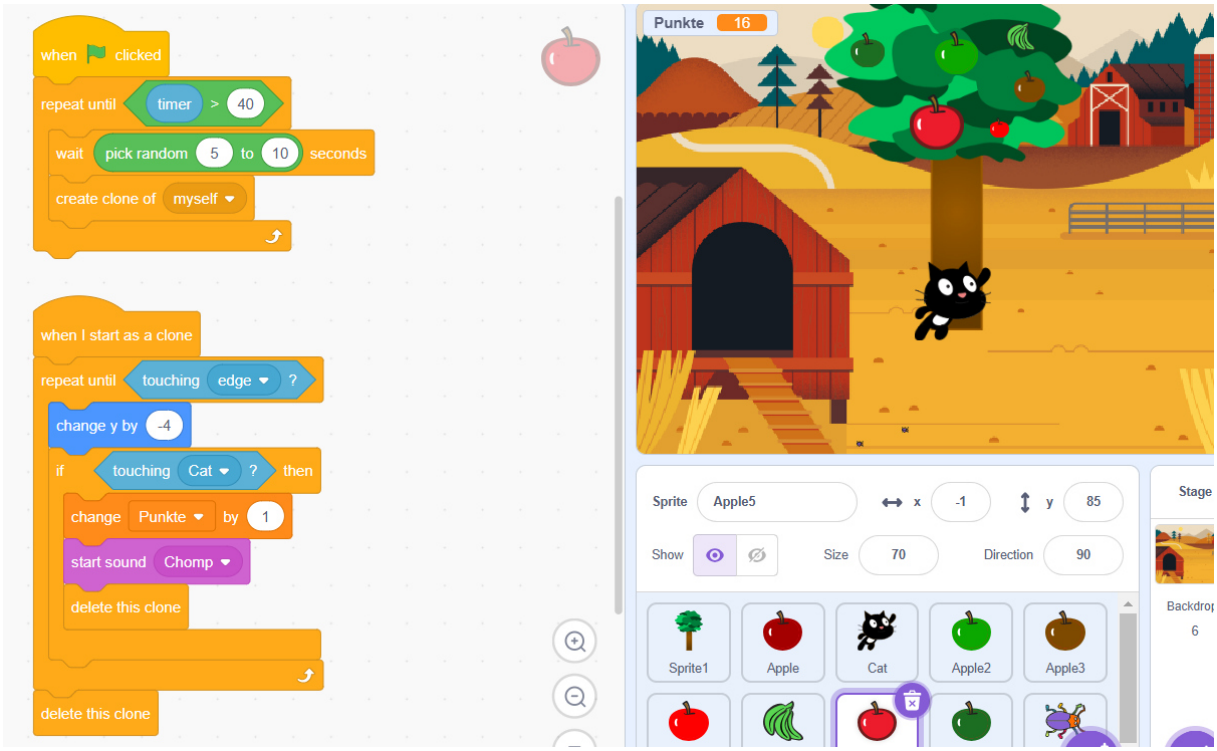

define block name springen
when clicked
if touching Food Truck2 ? then
stop all
else
wait until key space pressed?
forever
block name springen
when right arrow key pressed
if Try again ! = false then
change x by 10
else
move 10 steps
when this sprite clicked
block name springen
when clicked
go to x: -177 y: -121
forever
if key space pressed? then
change y by 100
wait 1 seconds
change y by -100
Sprite Sprite1 x -177 y -121
Show Size 55 Direction 90
Sprite1
Food Truck2
Stage
Backdrops 3


when clicked
go to x: 154 y: -109
wait 1 seconds
glide 1 secs to x: -167 y: -105
forever
if touching Sprite1 ? then
say Try again !
else
hide
create clone of myself
go to x: 154 y: -109
wait 1 seconds
show
glide 1 secs to x: -167 y: -105
Sprite Food Truck2 x -167 y -105
Show Size 45 Direction 90
Sprite1
Food Truck2
Stage
Backdrops 3

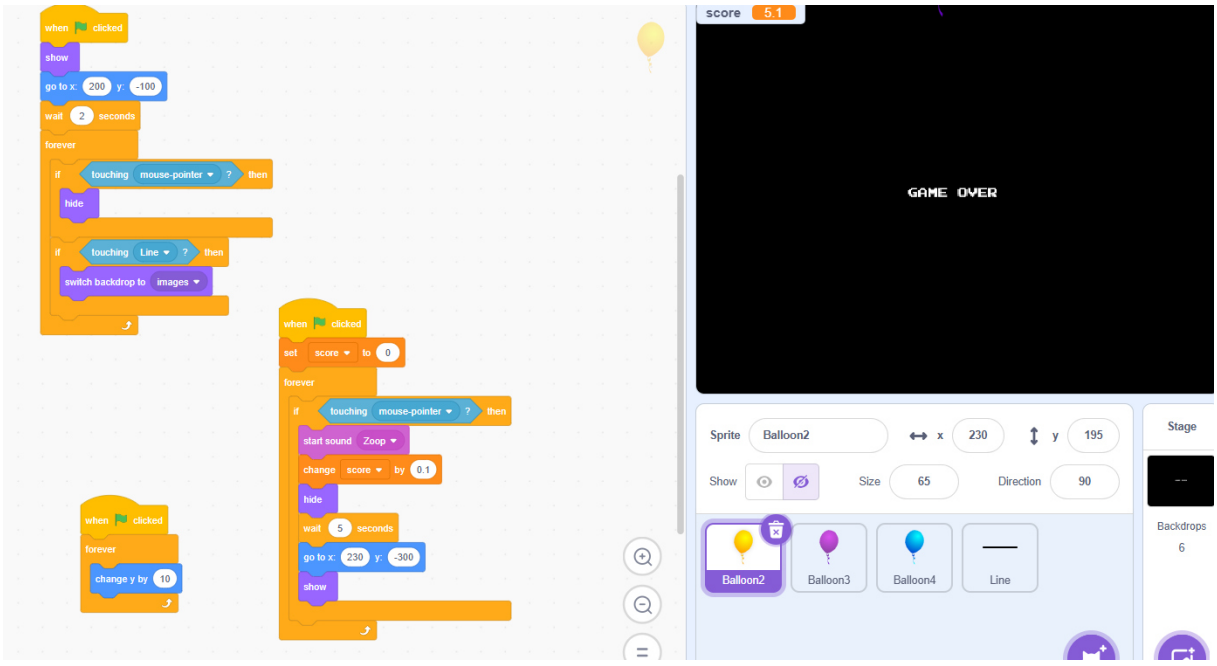
8.2.2.1.3 Game3

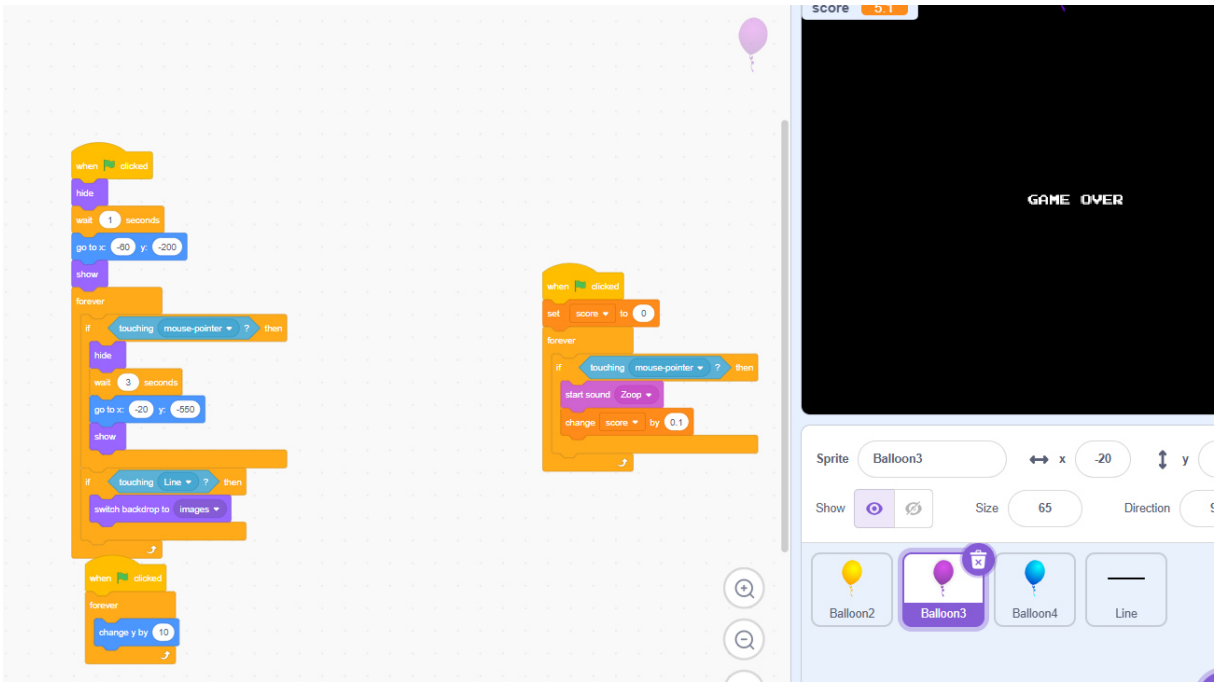

when right arrow key pressed
change x by 20
when left arrow key pressed
change x by -20
when up arrow key pressed
change y by 20
when down arrow key pressed
change y by -20
when clicked
set Punkte to 0
show variable Punkte
play sound Meow until done
think Ich habe einen Bärenhunger! for 2 seconds
go to Beetle3
go to Beetle4
go to Beetle7
play sound Meow3 until done
think Mhh die Äpfel sehen aber lecker aus! for 2 seconds
forever
if key space pressed? then
go to x: 30 y: 0
if touching Apple ? then
change Punkte by 1
Punkte 16
Sprite Cat
x 12
y -58
Show
Size 70
Direction 90
Stage
Backdrops 6
Sprite1
Apple
Cat
Apple2
Apple3
Apple4
Bananas
Apple5
Apple6

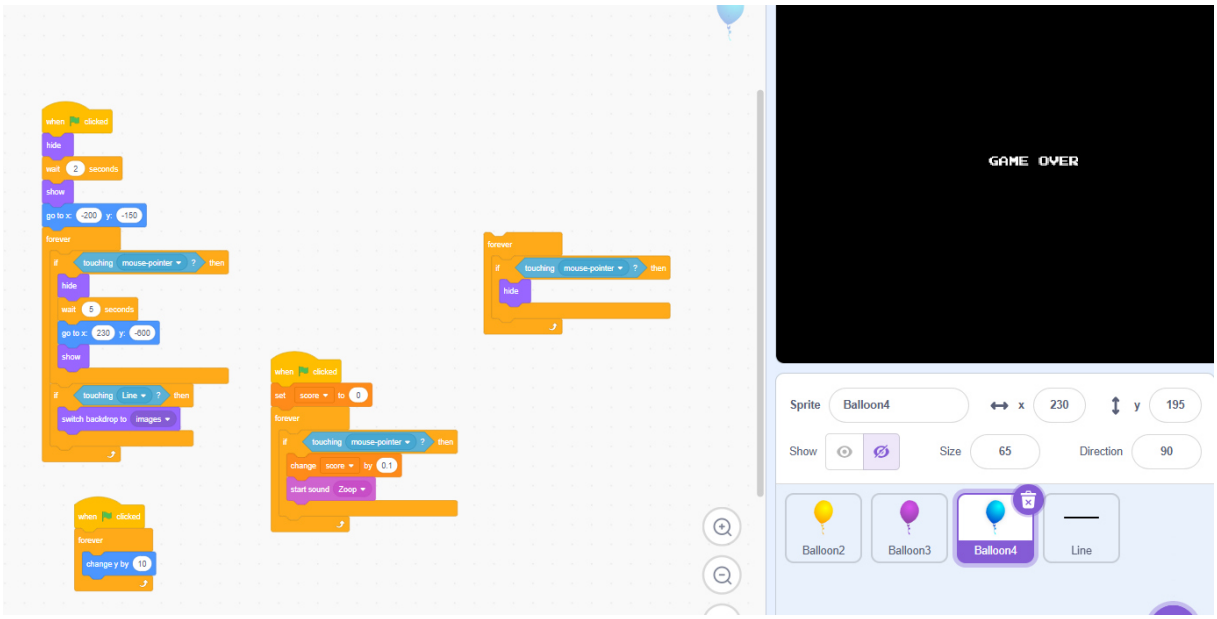

when I start as a clone
repeat until touching edge ?
change y by -4
if touching Cat ? then
change Punkte by 1
start sound Chomp
delete this clone
delete this clone
when clicked
repeat until timer > 40
wait pick random 5 to 10 seconds
create clone of myself
Punkte 16
Sprite Apple
x 10
y -164
Show
Size 60
Direction 90
Stage
Backdrops 6
Sprite1
Apple
Cat
Apple2
Apple3
Apple4
Bananas
Apple5
Apple6

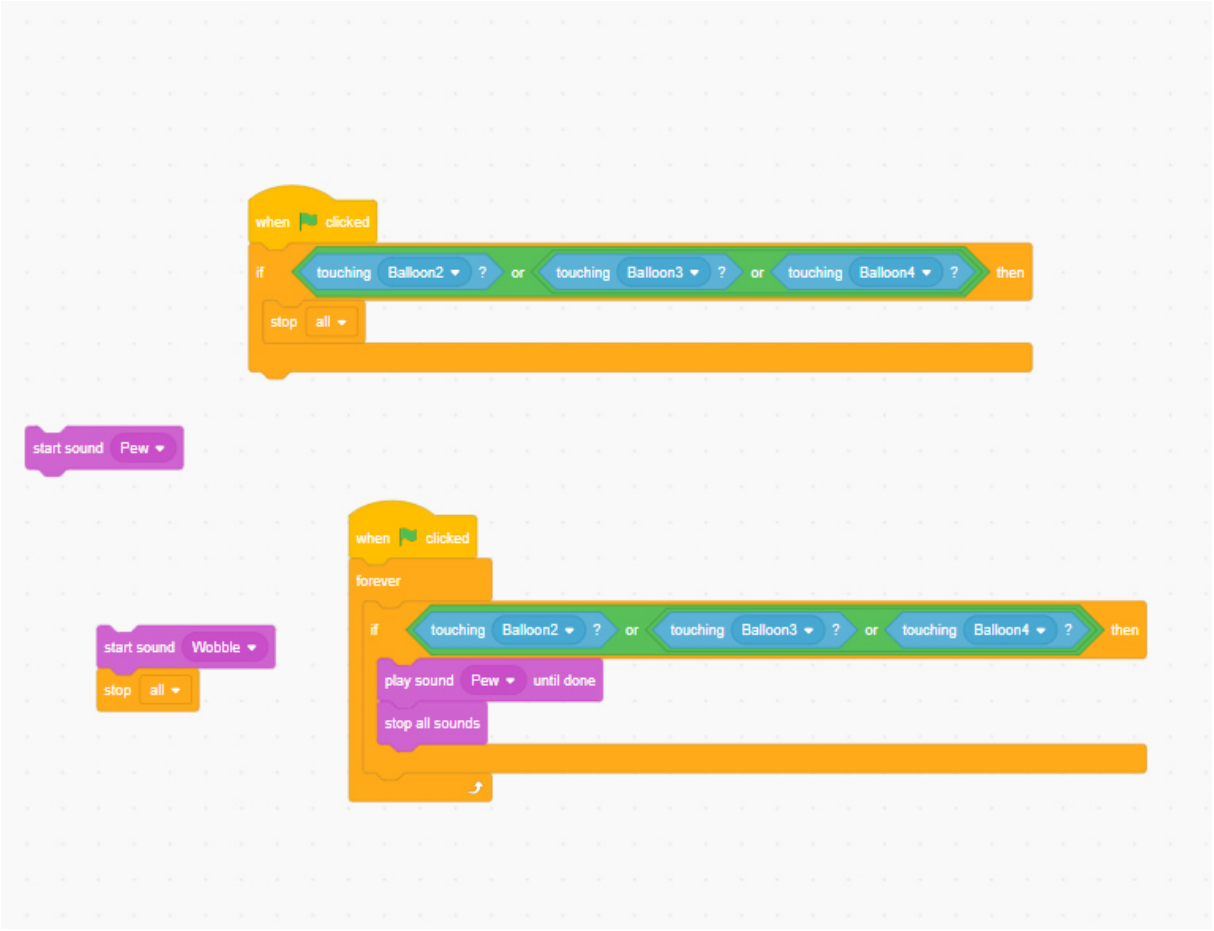

when clicked
repeat until timer > 40
wait pick random 5 to 10 seconds
create clone of myself
when I start as a clone
repeat until touching edge ?
change y by -4
if touching Cat ? then
change Punkte by 1
start sound Chomp
delete this clone
delete this clone
Punkte 16
Sprite Apple5
x -1
y 85
Show
Size 70
Direction 90
Stage
Sprite1
Apple
Cat
Apple2
Apple3

8.2.2.1.4 Game4

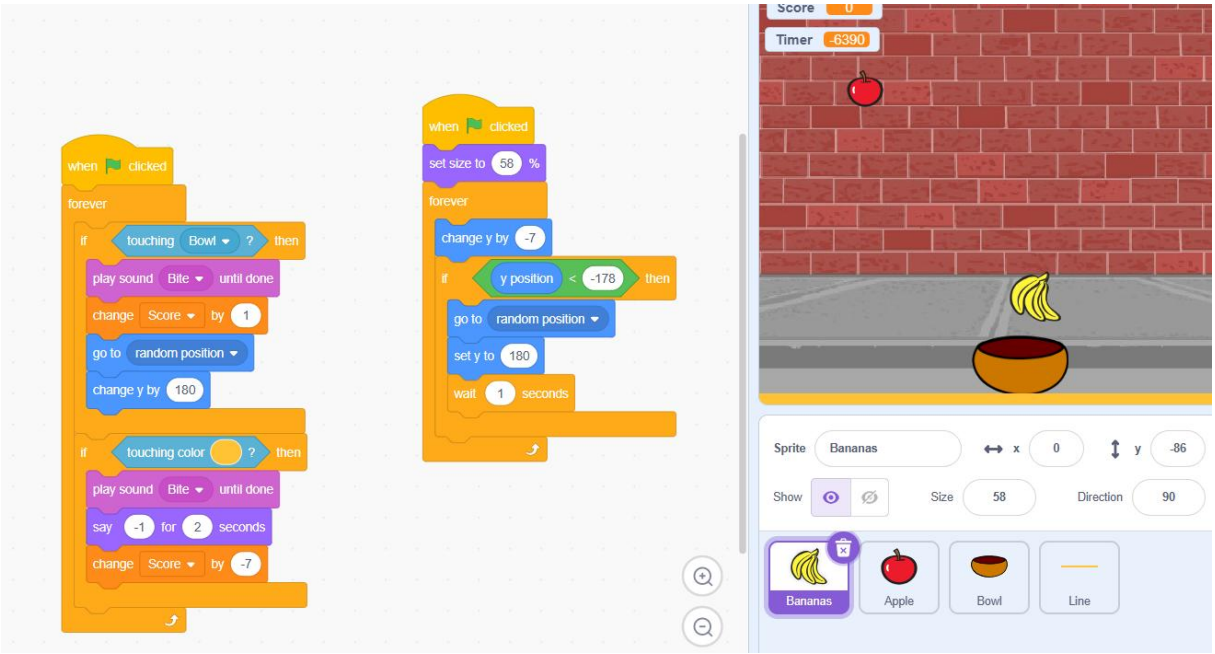

when clicked
show
go to x: 200 y: -100
wait 2 seconds
forever
if touching mouse-pointer ? then
hide
if touching Line ? then
switch backdrop to images
when clicked
forever
change y by 10
when clicked
set score to 0
forever
if touching mouse-pointer ? then
start sound Zoop
change score by 0.1
hide
wait 5 seconds
go to x: 230 y: -300
show
score 5.1
GAME OVER
Sprite Balloon2
x 230
y 195
Show
Size 65
Direction 90
Stage
Balloon2
Balloon3
Balloon4
Line
Backdrops 6

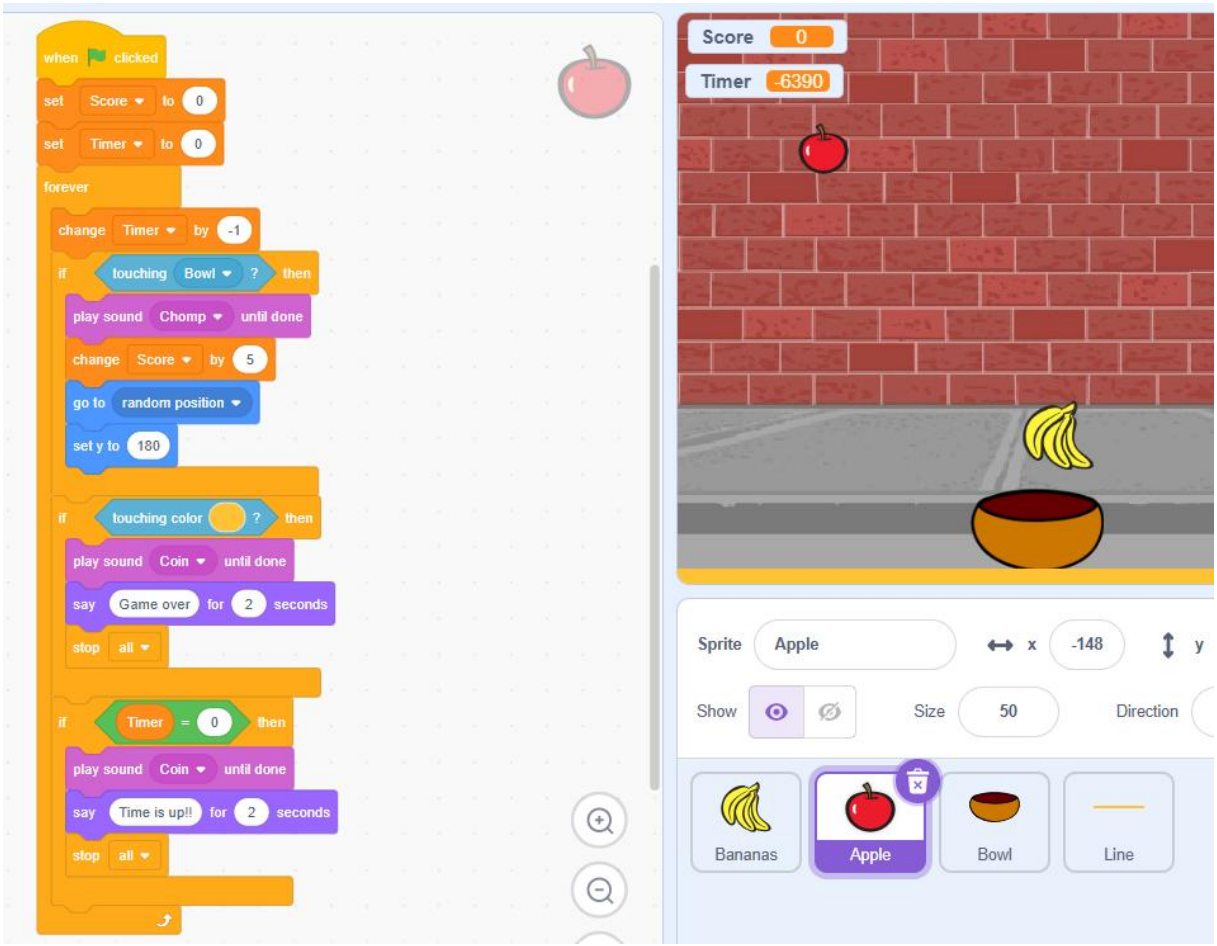

when clicked
hide
wait 1 seconds
go to x: -60 y: -200
show
forever
if touching mouse-pointer ? then
hide
wait 3 seconds
go to x: -20 y: -550
show
if touching Line ? then
switch backdrop to images
when clicked
forever
change y by 10
when clicked
set score to 0
forever
if touching mouse-pointer ? then
start sound Zoop
change score by 0.1
score
GAME OVER
Sprite Balloon3
x -20
y
Show
Size 65
Direction
Balloon2
Balloon3
Balloon4
Line

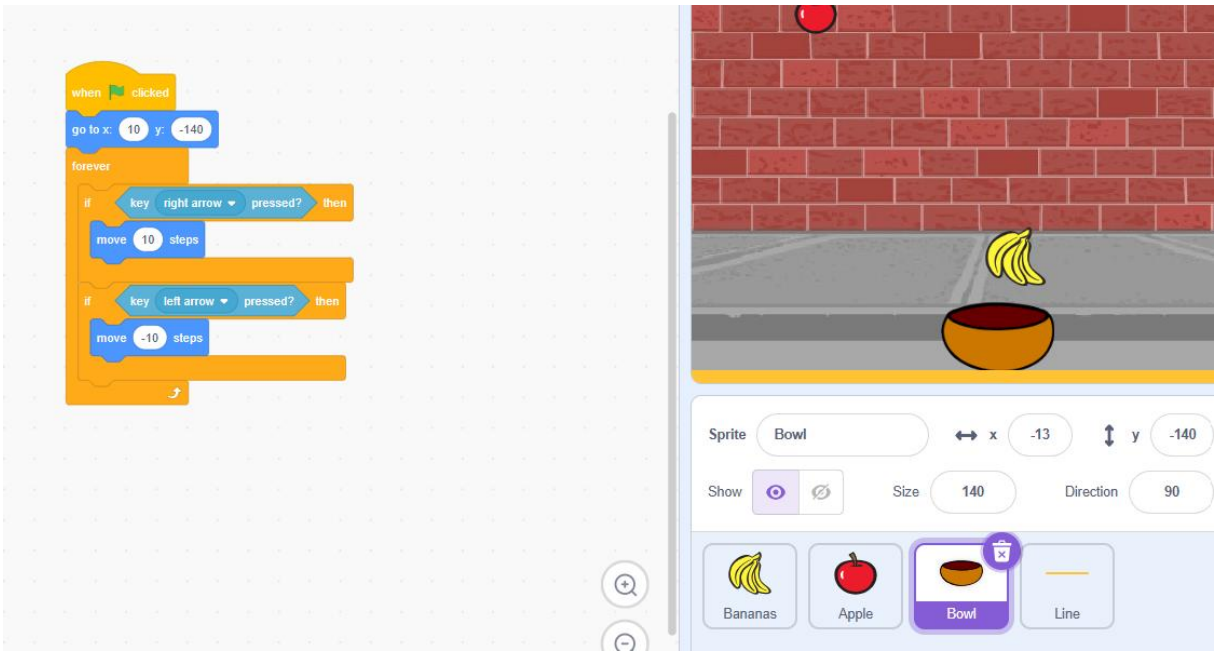

when clicked
hide
wait 2 seconds
show
go to x: -200 y: -150
forever
if touching mouse-pointer ? then
hide
wait 5 seconds
go to x: 230 y: -600
show
if touching Line ? then
switch backdrop to images
when clicked
forever
change y by 10
when clicked
set score to 0
forever
if touching mouse-pointer ? then
change score by 0.1
start sound Zoop
forever
if touching mouse-pointer ? then
hide
GAME OVER
Sprite Balloon4
x 230
y 195
Show
Size 65
Direction 90
Balloon2
Balloon3
Balloon4
Line


when clicked
if touching Balloon2 ? or touching Balloon3 ? or touching Balloon4 ? then
stop all
start sound Pew
start sound Wobble
stop all
when clicked
forever
if touching Balloon2 ? or touching Balloon3 ? or touching Balloon4 ? then
play sound Pew until done
stop all sounds

8.2.2.1.5 Game5


when clicked
forever
if touching Bowl ? then
play sound Bite until done
change Score by 1
go to random position
change y by 180
if touching color ? then
play sound Bite until done
say -1 for 2 seconds
change Score by -7
when clicked
set size to 58 %
forever
change y by -7
if y position < -178 then
go to random position
set y to 180
wait 1 seconds
Score 0
Timer -6390
Sprite Bananas
x 0
y -86
Show
Size 58
Direction 90
Bananas
Apple
Bowl
Line


when clicked
set Score to 0
set Timer to 0
forever
change Timer by -1
if touching Bowl ? then
play sound Chomp until done
change Score by 5
go to random position
set y to 180
if touching color ? then
play sound Coin until done
say Game over for 2 seconds
stop all
if Timer = 0 then
play sound Coin until done
say Time is up!! for 2 seconds
stop all
Score 0
Timer -6390
Sprite Apple
x -148
y
Show
Size 50
Direction
Bananas
Apple
Bowl
Line


when clicked
go to x: 10 y: -140
forever
if key right arrow pressed? then
move 10 steps
if key left arrow pressed? then
move -10 steps
Sprite Bowl
x -13
y -140
Show
Size 140
Direction 90
Bananas
Apple
Bowl
Line

8.2.2.2 Python Programs (Examples) – Pair Programming with Human Partner

```
modus = int(input("Celsius in Fahrenheit (1) oder Fahrenheit in Celsius (2)"))
grad = int(input("Geben Sie an wie viel Grad es sind!"))

if modus == 1:
    ergebnis=(9/5 * grad) + 32
    print(grad," grad celsius entsprechen ",ergebnis, " grad fahrenheit ")

if modus == 2:
    ergebnis=5/9 * (fahrenheit - 32)
    print(grad," grad fahrenheit entsprechen ", ergebnis, " grad celsius ")
```

```
# Online Python - IDE, Editor, Compiler, Interpreter
modus=int(input("Wie soll umgewandelt werden? Celsius in Fahrenheit (1) oder Fahrenheit in Celsius (2)"))
grad=float(input("Geben Sie die Grad ein:"))

if modus == 1:
    ergebnis=5/9 * (grad - 32)
    print(ergebnis,"grad fahreheit")


if modus == 2:
   ergebnis=(9/5 * grad) + 32
   print(ergebnis,"grad celsius")
```

```
def celsius_to_fahrenheit(celsius):
    return (celsius * 9/5) + 32

def fahrenheit_to_celsius(fahrenheit):
    return 5/9 * (fahrenheit - 32)

def main():
    choice = int(input("Möchten Sie Celsius in Fahrenheit (1) oder Fahrenheit in Celsius (2) umwandeln? "))

    if choice == 1:
        celsius = float(input("Geben Sie die Temperatur in Celsius ein: "))
        fahrenheit = celsius_to_fahrenheit(celsius)
        print(f"{celsius} Grad Celsius entsprechen {fahrenheit} Grad Fahrenheit.")
    elif choice == 2:
        fahrenheit = float(input("Geben Sie die Temperatur in Fahrenheit ein: "))
        celsius = fahrenheit_to_celsius(fahrenheit)
        print(f"{fahrenheit} Grad Fahrenheit entsprechen {celsius} Grad Celsius.")
    else:
        print("Ungültige Eingabe. Bitte wählen Sie 1 oder 2.")

if __name__ == "__main__":
    main()
```

```python
zielwert = int(input("Bis zu welcher Zahl soll summiert werden? "))

summe = 0
i = 1

while i <= zielwert:
    summe += i
    i += 1

zielwert += 1

print("Die Summe der Zahlen von 1 bis", zielwert, "beträgt", summe)
```

```python
zahl=int(input("Bis zu welcher Zahl soll summiert werden?"))
summe = 0

zähler = 1
while zähler <= zahl:
    summe= summe+zähler
    zähler=zähler+1

print ("Die Summe der Zähler von 1 bis " ,zahl,"beträgt" , summe)
```

```python
zahl=int(input("Bis zu welcher Zahl soll summiert werden:"))
summe = 0
zähler = 1
while zähler <= zahl:
    summe = summe + zähler
    zähler= zähler + 1

print( "Die Summe der Zahlen von 1 bis ",zahl," beträgt",summe)
```

```python
def sum_numbers(up_to):
    total = 0
    current_number = 1
    while current_number <= up_to:
        total += current_number
        current_number += 1
    return total

def main():
    up_to = int(input("Bis zu welcher Zahl möchten Sie die Zahlen summmieren? Geben Sie eine positive ganze Zahl ein: "))

    if up_to <= 0:
        print("Bitte geben Sie eine positive ganze Zahl ein.")
        return

    total_sum = sum_numbers(up_to)
    print(f"Die Summe der Zahlen von 1 bis {up_to} beträgt {total_sum}.")

if __name__ == "__main__":
    main()
```

```
# Online Python - IDE, Editor, Compiler, Interpreter
import array


namensliste = [];
name = input ("Geben Sie einen Namen oder geben Sie - ein falls Sie keinen Namenmehr eingeben wollen")
namensliste.append(name);
suchbuchstabe = input("Geben Sie den Buchstaben ein nachdem gesucht werden soll");
count=0;

while name != "-":
      name =input("Geben Sie einen Namen ein oder geben Sie - ein falls Sie keinen Namen mehr eingeben wollen")
      namensliste.append(name);


for name in namensliste:
    for buchstabe in name:
        if buchstabe == suchbuchstabe.lower():
            count= count+1;

print("Der Buchstabe" , suchbuchstabe, "kommt in eure Klasse" , count, "mal vor.");
```