Erzincan Üniversitesi Eğitim Fakültesi Dergisi
Erzincan University Journal of Education Faculty
2024 Cilt 26 Sayı 4 (554-564) https://doi.org/10.17556/erziefd.1485760
Araştırma Makalesi / Research Article

Coding Analogy
Analojiyi Kodlamak

Aykan Koç [1] Elif Taşlıbeyaz[2]

[1] Asst. Prof. Dr. Erzincan Binali Yıldırım University, Faculty of Education, Erzincan, Türkiye
[2] Assoc. Prof. Dr. Erzincan Binali Yıldırım University, Faculty of Education, Erzincan, Türkiye

Makale Bilgileri

Geliş Tarihi (Received Date)

17.05.2024

Kabul Tarihi (Accepted Date)

30.09.2024

Sorumlu Yazar

Ayhan Koç

Erzincan Binali Yıldırım University, Faculty of Education, Erzincan, Türkiye

ayhankoc@Erzincan.Edu.Tr

Abstract: This study aimed to examine the pre-service teachers' experiences in preparing and using analogies within a programming course to better understand and evaluate the concepts. The research design of this study was identified as a case study. The 21 participants of the study took an elective introductory programming (Python) course at a state university over 14 weeks. They prepared analogies during the course, and 11 of them attended a focus group interview at the end of the course. The data collection tools used in the study included a questionnaire in which learners indicated the topics they found challenging within the programming course, the analogies they created based on programming education, and interview questions that explored their views at the end of the course. These data were analyzed descriptively. The results indicated that although learners experienced difficulties in understanding algorithms, programming logic, and learning loops, the use of analogies in programming education offered positive contributions.

Keywords: Analogy, programming, programming education, Python, pre-service teachers

Öz: Bu çalışmanın amacı, programlama dersinde kavramları daha iyi anlamak ve değerlendirmek için analoji hazırlama ve kullanma konusunda öğretmen adaylarının deneyimlerini incelemektir. Bu çalışmanın araştırma tasarımı bir durum çalışması olarak belirlenmiştir. Çalışmanın 21 katılımcısı, 14 hafta boyunca bir devlet üniversitesinde seçmeli bir Programlamaya Giriş (Python) dersi almıştır. Ders sırasında analojiler hazırlamışlar ve 11'i dersin sonunda odak grup görüşmesine katılmıştır. Araştırmada kullanılan veri toplama araçları arasında; öğrencilerin programlama dersi içerisinde zorlandıklarını belirttikleri bir anket, programlama öğretimine dayalı olarak oluşturdukları analojiler ve ders sonunda onların görüşlerini ortaya çıkaran görüşme soruları yer almaktadır. Bu veriler betimsel olarak analiz edilmiştir. Sonuçlar, öğrencilerin algoritmaları, programlama mantığını ve öğrenme döngülerini anlamada zorluklar yaşamalarına rağmen, programlama eğitiminde analoji kullanımının olumlu katkılar sağladığını göstermiştir.

Anahtar Kelimeler: Analoji, programlama, programlama öğretimi, Python, öğretmen adayları

Koç, A & Taşlıbeyaz, E. (2024). Coding analogy, *26*(4), 542-564. *Erzincan University Journal of Education Faculty* https://doi.org/10.17556/erziefd.1485760

Introduction

21st century skills encompass a wide array of abilities aimed at preparing individuals for social and occupational demands. Key skills include critical thinking, problem-solving, technological literacy, effective communication, collaboration, programming, and computational thinking (Altbach et al., 2009; Ambrosio, 2014; Dede, 2013; Türel et al., 2023). These skills are crucial for keeping pace with contemporary developments and contributing to society. One of the most significant areas focused on this contribution is undoubtedly education. Therefore, educational institutions are increasingly emphasizing the development of these skills to adapt to the rapidly evolving and changing conditions (Longjun, 2023).

Programming skill is frequently emphasized in educational environments (Raman, 2020). It is a fundamental skill for securing success and employment opportunities in the future society (Yang et al., 2018). Programming is also recognized as a valuable 21st century skill that prepares individuals for the digital era and the future job market (Yang et al., 2018). Particularly, the development of problem-solving and computational thinking skills, often highlighted as essential for the 21st century, is associated with programming-related activities. It is even suggested that programming education starting at an early age can enhance these skills (Laato et al., 2020). In Turkey, programming education is increasingly incorporated into the curricula at primary and secondary education levels (Atabaş, 2018; Betchie, 2019; Deniz & Eryılmaz, 2019). This integration of programming skills development into educational settings emphasizes the importance of preparing learners for the demands of the contemporary world.

Numerous academic studies have been published in the literature with an increasing interest in programming education. Some of these studies have explored the relationship between programming education and different skills. The findings indicate that programming education positively affects students' computational thinking and problem-solving skills, as well as their ability to think algorithmically (Boom, 2022; Hromkovic et al., 2017; Kiss & Arki, 2017; Kong & Wang, 2020; Rim, 2017; Selby, 2015). The problem-solving skills present in learners positively contribute to the ability to learn programming (Yıldız Durak, 2020). On the other hand, there has been a positive correlation between mathematical skills and programming (Pörn et al., 2021).

Some of the research in programming education also focuses on the methods and techniques used to enhance its effectiveness and success. For example, problem-based learning approaches are identified as effective in programming education (Bawamohiddin & Razali, 2017; Chung et al., 2020; Goletti et al., 2021; Peng, 2010). To overcome the challenges in teaching and learning programming languages, a problem-based e-learning model that integrates traditional problem-based learning with e-learning environments has also been proposed (Bashir & Hoque, 2016). The positive impact of online instructional design in programming education has been emphasized, highlighting the importance of peer-assessment and the design of online learning environments (Sabarinath & Quek, 2020). Another method utilized in programming education is blended learning. Studies suggest that blended learning models in programming courses can improve educational outcomes (Shi & Zheng, 2019). Specifically, programming education conducted using the Flipped Classroom Model has been found to be more efficient than traditional methods, with positive effects on student learning, motivation, and engagement in the courses (Alper & Öztürk, 2019; Herala et al., 2015; Tolano-Gutiérrez et al., 2022). Lastly, it has been emphasized that game-based learning methods are increasingly popular techniques to enhance learning, interest, and comprehension in programming, particularly among younger students (Kanika & Chakraborty, 2020).

One of the techniques employed in programming education involves the use of analogies. Analogies are cognitive mechanisms that facilitate the application of knowledge from one situation to another by identifying similarities and differences (Gentner & Hoyos, 2017). They are extensively used in science and mathematics education to assist students in understanding complex concepts and reasoning (Jonāne, 2015; Heywood, 2002). Analogies, which support engagement in the learning process (Heywood, 2002), can simplify the teaching of challenging scientific concepts, making them more accessible and comprehensible to learners. This approach can be particularly effective in programming education, where abstract concepts and logical structures often parallel patterns in mathematics and science, allowing analogies to bridge the gap between familiar knowledge and new programming skills.

The focus is on the use of analogies, metaphors, and various scenarios in programming education studies in literature. Although analogies and metaphors have different properties and roles, they are similar elements used interchangeably (Yıldırım & Gürsu, 2018). In fact, metaphors can shape thought, especially in relation to complex concepts. Sometimes metaphors are used in a situation where words are insufficient or in a situation where the expression needs to be strengthened. On the other hand, analogy helps to explain an unknown, unfamiliar phenomenon with similarities and differences (Gentner & Hoyos, 2017) and directly compares two fields (Nakiboğlu & Yıldırım, 2019). One of the studies on this subject Gökoğlu (2017) investigated computer programming students' perceptions of the concept of algorithms through metaphor analysis, aiming to categorize the emergent metaphors under conceptual categories. Kandin's (2019) thesis examined the use of metaphors and scenarios in early programming education while other studies have also focused on the impact of analogy techniques used in teaching concepts in Mathematics, Information Technologies, and Science courses (Kaya, 2011; Ketenci, 2019; Özcan, 2013). The results of these studies generally highlight the positive effect of metaphors and analogies used in classrooms on learning the subjects. These research contributions are significant in understanding the effects of various methods and techniques used in education and improving educational processes. While metaphors are generally used in programming teaching, which is the subject of our research, the number of studies using analogies is quite limited. However, in teaching some courses such as programming, explaining abstract and difficult concepts by comparing them with familiar concepts can facilitate learning. Therefore, the current study focused specifically on analogies.

Moreover, the present research selected pre-service teachers taking a programming course, and analogies related to programming education were prepared by these students. Similarly, Harper et al., (2023) formed groups within a programming course, asking students to prepare analogies about key concepts. These analogies were then presented, as it was believed that analogies prepared by learners could contribute more to learning and encourage deeper reflection than those prepared by teachers (Fincher et al., 2020; Harper et al., 2023). Additionally, as stated in the constructivist learning approach, better learning outcomes emerge when learners are active and construct their own learning (Mascolo & Fischer, 2005). In our study, students individually prepared analogies and these were presented in a classroom setting to gather peer and instructor feedback.

In conclusion, our study will provide guidance on the use of analogies in programming, as the participants are pre-service teachers, and will be able to offer a different perspective to overcome the challenges in programming teaching. In addition, the study is notable in terms of focusing on the use of analogies in programming teaching and the preparation of these analogies by pre-service teachers. This process can enable them to prepare analogies to help their own learning and to use them in their future teaching to facilitate the learning of their students. Therefore, this study aims to examine the pre-service teachers' experiences in preparing and using analogies within a programming course to better understand and evaluate the concepts. The research questions of the study are as follows:

1. How have the analogies prepared in the programming course affected the pre-service teachers' understandibility of the subjects?
2. What are the experiences of pre-service teachers in preparing and using analogies in the context of programming education?
3. What are the views of pre-service teachers on preparing analogies in other subjects?

Method

Research Design

This study is qualitative research. A case study was used in the study. The case study included an in-depth examination of the research question (Yıldırım & Şimşek, 2013). Prior to the analogy generation process, a pre-questionnaire and post-questionnaire focusing on programming difficulties were administered. Following this, the experiences of pre-service teachers regarding the analogy generation process and views about preparing analogies were deeply examined. Finally, a focus group interview was conducted with students to examine their views on the use of analogy in programming education in depth.

Participants

The participants of the study were 21 pre-service teachers who took an elective course on introductory programming (Python) at a state university for 14 weeks. These participants were students in the 2nd, 3rd, and 4th years of the Mathematics Department at the Faculty of Education. These participants were students in the 2nd, 3rd, and 4th years of the Mathematics Department at the Faculty of Education. They enrolled in the introductory programming (Python) course. There were 21 people, 13 girls and eight boys. Since they were selected from among the students taking the programming course, a purposive sampling method was used. Given the subject matter focused on teaching programming, this course's students were included in the study. In our study, before and after the programming course, volunteers responded to a questionnaire titled "Topics that participant found most challenging" (Appendix-1), with 18 participating in the pre-questionnaire and 16 in the post-questionnaire. Following the course, a focus group discussion was conducted with 11 participants who were selected from among the participants voluntarily.

Data Collection Instruments

The data collection instruments used in the study include a questionnaire where participants indicated the topics, they struggled with during the programming course, analogies prepared by them based on programming education, and the interview questions that examined learners' views on learning and their views on analogies at the end of the course.

The questionnaire asked learners to mark the topics covered in the course that they found challenging. It was created and administered using Google Forms. This questionnaire was reapplied after the analogies were prepared and reviewed in class. This questionnaire was prepared by the course instructor. The questionnaire included topics covered in the course. Students were asked to choose one of the topics in this questionnaire that they had difficulty with.

The interview questions (Appendix-2) were designed to explore learners' attitudes and learning experiences within the scope of the research questions. These questions were prepared by the researchers of the study in light of the research questions. These were then reviewed by field experts and finalized. These questions were posed to students during a focus group interview. This format was chosen to facilitate rich data collection, allowing participants to remind each other as necessary. In order to ensure consistency in the research, expert opinions were sought during the preparation of data collection tools and data analysis stages. Both quantitative and qualitative data were presented to support the research findings and for credibility. In addition, while presenting qualitative data for confirmability, sample answers to the questions were presented in the findings as direct quotes.

Analogy Preparation Process

The analogies were created by the participants on topics mentioned in the course selected during the course and were reviewed in class with peer and instructor evaluations. The analogies were updated by them in the following week. While creating the analogies, participants followed the stages below (Harper et al., 2023):

1. Identifying the target concept and its essential characteristics: This involves understanding the new or complex concept that the analogy aims to explain. Detailed information and example applications are provided to students during the class on the topics they would use to create analogies.
2. Brainstorming on potential source domains that share similarities with the target concept: This encourages divergent thinking and helps students explore various familiar concepts that can be used in the analogy. After selecting the topics for their analogies, students are shown various examples of analogies and given the opportunity to develop ideas.
3. Selecting the most suitable source domain: This step emphasizes the need to carefully choose the source domain that best fits the target concept. Students are asked to find an example analogy related to the topic they chose.
4. Mapping the similarities between the source and target domains: This step focuses on creating a clear and accurate correlation between the two domains. Students are asked to list the similarities between the topic they chose and the analogy.
5. Identifying and addressing differences or limitations in the analogy: This step fosters critical thinking by helping students recognize and address the limitations of the analogy. In this part, students are asked to list the differences between the topic they chose and the analogy. The analogies in this study aim at aiding subsequent learners and facilitating the acquisition of programming skills, are presented in the appendix (see Appendix-3).

The Role of Researchers

One of the researchers in the study was the instructor of the course. This researcher asked the learners to fill out a questionnaire regarding the topics they struggled with during the course. In this questionnaire, students individually marked the topics they found difficult to learn. Subsequently, learners were assigned to prepare analogies related to the topics covered. Each student selected a topic within the course and created an analogy related to it. These analogies were presented in class a week later and subjected to peer and instructor evaluations. After the evaluations, the analogies were finalized in the following week. At the end of the course, the researcher reapplied the questionnaire and asked the learners to reflect on whether the analogies helped them understand the topics they struggled with.

The researchers of this study reviewed and edited the analogies submitted by the students. They also developed interview questions as part of the study. In the final week of the course, they conducted a focus group interview with 11 voluntarily participating students, recorded the interview with the students' permission, and later analyzed the responses to the interview questions and other data.

Data Analysis

The data obtained from the questionnaire were descriptively analyzed, and frequency values were derived. The results were presented in tables and charts. The data from the focus group interview were subjected to descriptive content analyses. In the analysis, the participants' perspectives on the topics they found challenging in programming education were coded according to the subjects and objectives of the course. Additionally, the responses regarding the contributions and suggestions of the process were subjected to content analysis. In content analysis, data collected from participants are analyzed, similar data are grouped under a common theme, and interpreted (Yıldırım & Şimşek, 2013). During the analysis, the qualitative findings were coded by the researchers and these codes were subjected to expert opinion. In order to ensure the transferability of the research findings, the participants were described in detail and the codes obtained were presented with their coding numbers. In addition, the participant names were kept confidential and coded as Participants 1-11 in the findings section, and direct quotes from some participants were included in the findings section.

Findings

The study aims to examine the pre-service teachers' experiences in preparing and using analogies within a programming course to better understand and evaluate the concepts. The findings of these studies are presented below in light of the research questions.

Table 1. Topics that participant found most challenging

Topics	Pre-questionnaire		Post-questionnaire	
	f	%	f	%
Loop Structures (for, while)	13	68,4	13	76,5
Nested Control Structures (if, else, elif)	12	63,2	4	23,5
Control Structures (if, else)	11	57,9	3	17,6
String Operations (len, etc.)	10	52,6	4	23,5
Data Type Conversions	5	26,3	3	17,6
Logical Operators (and, or, not)	5	26,3	2	11,8
Array Definition and Usage (list definition)	4	21,1	6	35,3
Comparison Operators (<, >)	3	15,8	0	0,0
Problem Solving Process	2	10,5	1	5,9
Algorithms	2	10,5	0	0,0
Variables and Data Types	2	10,5	1	5,9
Assignment Operators (=, +, -)	2	10,5	1	5,9
Computer and Programming & What is Programming?	1	5,3	0	0,0
Total (Participants)	18		16	

Participants' Understanding of the Subjects

In this section, the results of the "Topics that participant found most challenging" questionnaire, administered as pre- and post-questionnaire, before and after the analogy preparation process, were evaluated and presented in Table 1.

It was evident from the pre-questionnaire responses in the programming education session that participants predominantly struggled with loop structures (f=13), nested control structures (f=12), control structures (f=11), and string operations (f=10) in Table 1. Moderate difficulties were observed in other topics.

Following the analogy creation process, the post-questionnaire responses revealed a significant decrease in difficulties previously identified in the pre-questionnaire, particularly in nested control structures (f=4), control structures (f=3), and string operations (f=4). Additionally, there were notable decreases in other areas such as data type conversions (f=3), logical operators (f=2), comparison operators (f=0), problem-solving processes (f=1), algorithms (f=0), variables and data types (f=1), assignment operators (f=1), and an introduction to computers and programming (f=0). According to Table 1, loop structures (for, while) remained a challenging topic for the students, with no observed improvement post-analogy process (f=13). Conversely, an increase in difficulty was noted in the area of array definition and usage (list definition) in the post-questionnaire (f=6).

Participants' Experiences on Analogy Preparation and Usage Process

The process of preparing analogies was examined after which, based on the guiding research questions, interview questions were developed. These questions were then administered through a focus group discussion. The outcomes were analyzed using content analysis, and the results were presented in Table 2. organized by categories, codes, frequencies, and sample quotes.

It was observed that, in accordance with the feedback from the participants, the most challenging topics in programming education are syntax rules and loops in Table 2. Participants expressed that they struggle with writing code due to spelling errors (such as the need to close a parenthesis, not using Turkish characters when defining variables, etc.), thus encountering difficulties during the coding process. Moreover, both types of loops, whether with an undefined repetition structure or a defined one, were found to be the subjects participants struggled with the most in terms of comprehension. This finding was further supported by Table 1.

Table 2. Perspectives on the most challenging topics in programming education

Category	Code	f	Sample Quotes
Loops	Code 1: While Loop	6	Code1 “While loops…” P3
	Code 2: For Loop	6	Code 2 “For loops…” P2
Spelling Rules	Code 3: Syntax Rules	3	Code 3 “...Spelling rules and parentheses are a bit of a hassle.” P4

Table 3. Topics of analogy prepared by participants

Code (Node)	f	Sample Quotes
Code 1: Comparison Operators	3	"Mine was one of the comparison operators." P10
Code 2: Loops	2	"Mine was the While loop..." P2
Code 3: Arrays (Lists)	1	"I had done an array, it was a list." P9
Code 4: Nested Controls	1	"It was nested controls..." P8
Code 5: Logical Operators	1	"... logical operators." P1
Code 6: String Operations	1	"Mine was also string operations." P6
Code 7: Problem-Solving Process	1	"Mine was the problem-solving process." P5
Code 8: Programming Logic	1	"Computer programming and what programming is about, that was the topic." P11

Table 4. The views on difficulties encountered when creating analogies

Category	Code	f	Sample Quotes
Thinking Process	Code 1: Defining and Selecting Characteristics	2	"... I struggled a bit there, wondering which features to add..." P7
	Code 2: Setting Boundaries	2	"... when you determine a very large area, you can't just make it up, you have to find boundaries. It's a bit difficult to set those boundaries." P3
	Code 3: Identifying Differences	1	"... at first, we focus on their similarities and start writing. Then, when it comes to their differences, well, now you have to think a little about what those differences are." P10

Table 5. The views of analogy creation process

Category	Code	f	Sample Quotes
Peer Support	Code 1: Idea Exchange	3	"When preparing with P5, we looked at each other's work. We had a for loop in P5 too. We looked together." (P 4)
	Code 2: Comparison	2	"So, I did mine, then they did theirs. Finally, we compared each other's work. We supported each other to make it look a bit more professional, for example." (P3)
	Code 3: Peer Review	1	"I write and send it. I wonder if it's okay? P7 sends it back, asking where I can fix it." (P6)

It was noted that, according to the feedback from the participants, the topics most frequently analogized were comparison operators and loops in Table 3. Additionally, analogies were formed in topics such as arrays, nested controls, logical operators, string operations, problem-solving processes, and programming logic. This finding suggested that the analogies created by participants in these topics facilitated the understanding when interpreted alongside the pre-questionnaire and post-questionnaire data from Table 1. Interestingly, the lack of sufficient analogies in topics such as loops and array declaration might indicate that difficulties persist in these areas.

Participants primarily struggled with the process of thinking about how to create analogies. They mentioned experiencing difficulty in translating abstract concepts in programming education into analogies. In the analogies they created, participants found it most challenging to identify the characteristics, set boundaries or in other words, determine similarities and differences in programming concepts when making comparisons.

Table 5 presents the participants' views on the process of preparing analogies. Although assignments were given individually, participants were found to collaborate during the process of selecting a topic from programming subjects and creating analogies related to that topic. These collaborating individuals expressed engaging in idea exchange, comparing, and verifying the analogies they prepared during the analogy creation process. This finding, while not imposing any limitations on our research, actually encouraged collaboration, demonstrating that mutual exchanges of ideas not only led to the formation of better examples but also contributed more to the participants' learning process through mutual idea exchanges.

According to Table 6, participants not only expressed a positive view (f=5) regarding the contribution of the analogy preparation process to their understanding of programming topics but also noted its beneficial aspects in making programming topics more meaningful (f=2), facilitating the understanding of both the chosen topic in the analogy and the programming subjects (f=1), and reinforcing the subjects (f=1). Additionally, participants expressed the opinion that the experience gained from the process was enjoyable (f=1).

Table 6. The views in contributions of analogy creation process

Category	Code	f	Sample Quotes
Learning	Code 1:Contribution	5	"... I think it contributes." (P7)
	Code 2: Making Information Meaningful	2	"At first, these loops, operators, and so on seem meaningless. Then, when you relate them, they become a bit more meaningful." (P6)
	Code 3: Mutual Understandability	1	"At first, to think of an analogy, you need to understand its definitions first. When creating an analogy, we also understand its definition at the same time, it happens simultaneously. So, both make it easier for each other to understand." (P3)
	Code 4: Reinforcement	1	"... helped with reinforcement." (P8)
Experience	Code 5: Experience	1	"Moreover, it is a good experience, in my opinion." (P6)

Table 7. The views about usage of analogies in different courses

Category		Code	f	Sample Quotes
Positive	Mathematics	Code 1: Analytical Geometry	2	"... in analytical geometry, most of the class, even the teacher sometimes gets stuck. If they were connected to an analogy, and then discussed in a narrative process, there would likely be easier learning, especially with newly added topics." (P3)
		Code 2: Analysis	1	"... can be used in subjects like analysis, which are a bit more concrete, manual, and suitable for calculation." (P4)
		Code 3: Algebra	1	"... example topics can be very abstract in algebra for children. What are these x's and y's? How am I going to find this? It's like going from basic arithmetic to algebra. Analogies can be very useful in these topics..." (P7)
		Code 4: Probability	1	"Probability also comes to mind. In probability, for example, an event occurs, then when another example is given, it presents another event. ... probabilities can be built upon a single analogy." (P3)
	Computer Science	Code 5: Algorithm and Programming	1	"... used in classes involving programming or algorithms. Especially Python, Java..." (P4)
Partially		Code 6: Algebra	1	"... can be used to some extent in algebra, I think..." (P3)
Negative		Code 7: Abstract contents and subjects	2	"So, the very abstract ones are very difficult. Because they inherently have very abstract expressions." (P4)

The Views of Participants on Preparing Analogies in Other Subjects

According to Table 7, participants expressed their views regarding the use of analogies, particularly in mathematics classes. It was noteworthy that these pre-service teachers, who are students of mathematics education, emphasized the use of analogies, especially in teaching abstract concepts in mathematics (f=5). There was also emphasis on the importance of using analogies in computer science classes (f=1). Additionally, participants highlighted the challenge of structuring highly abstract subjects (f=2) using analogies as a negative comment.

Discussion

The remarkable responses given to the interview questions prepared under the research questions were discussed in light of previous studies in this section.

Students mentioned that they had difficulty in creating analogies due to the abstract structure of programming. This first major finding has also been emphasized in previous studies and the difficulty in teaching has been expressed as the difficulty of teaching programming (Gomes & Mendes, 2007). The topic of algorithms, the problem-solving process, and establishing programming logic were also the difficulties encountered in programming, mentioned by the participants (Günbaş & İlgün, 2023; Özmen & Altun, 2014; Saygıner & Tüzün, 2017). Other challenging topics in programming were syntax rules, loops, and decision structures. Participants expressed struggling with spelling errors (such as the need to close parentheses, not using Turkish characters when defining variables, etc.) while coding, leading to difficulties in the coding process. The finding was interpreted as consistent with previous research by Baltalı (2016) and Jancheski (2017) indicating that students faced difficulties in syntax in programming education. Furthermore, participants’ struggles with topics such as loops, decision structures, and operators align with other challenges encountered in text-based programming languages (Kadin, 2019).

Another finding was related to the participants' processes of creating analogies. The process of creating analogies contributed to making topics more meaningful and reinforced the subjects. This finding aligned with the findings of studies conducted by Dinçer (2005), Erümit et al. (2019), Harper et al. (2024) and Kaya & Durmuş (2011) in the field of Computer Science. Similarly, there were studies suggesting that creating analogies in different subjects supports learning (Bayazit, 2011; Bozkurt, 2019; Şahin et al., 2001; Yılmaz et al., 2002).

Participants encountered difficulty in determining the characteristics and differences when selecting source and target concepts during the analogy preparation process. This finding was supported by studies suggesting that students may face challenges in establishing connections between source and target concepts during the analogy preparation process (Harper et al., 2023; Uçar, 2021). According to Kobal et al. (2014), it was also believed that this difficulty experienced during the process may stem from students' insufficient prior knowledge about the source concept. Based on the research findings, collaborative work with peers during the analogy preparation process, involving mutual idea exchanges and comparisons, proved effective in overcoming this difficulty at various stages of the process. In addition to peer support during the analogy preparation process, instructor was contributed to completing participants' prior knowledge about the source and target concepts in the analogy and reinforcing the understanding of unfamiliar topics.

Finally, the participants had positive views mentioned in Table 7 regarding the use of analogies in their field (mathematics education) for teaching abstract concepts. This finding was supported by research on the use of analogies in mathematics education (Bayazit, 2011; Özcan, 2013; Saygılı, 2008). Analogies were commonly used in science and mathematics education to assist learners in understanding complex concepts, reasoning, and forming correct interpretations (Jonāne, 2015; Heywood, 2002). Additionally, analogies facilitated the learning of abstract concepts and helped overcome misconceptions (Zorluoğlu & Sözbilir, 2016).

Conclusion and Suggestions

The research findings demonstrated that the use of analogies in programming courses yielded positive contributions to programming education. It was observed that the process of preparing analogies by pre-service teachers contributed to making programming topics more understandable for learning. The utilization of analogies in programming courses emerged as a facilitator for students' comprehension of abstract concepts and supported the learning process. These findings underscore the significance of employing analogies in programming education and indicate their potential to contribute to students' learning more effectively.

In the research, participants were involved in creating analogies. Although they encountered certain difficulties during the process of analogy formation, it enhanced their ability to establish connections between source and target concepts. Collaborative work among them and teacher support during the analogy preparation process played a significant role in helping them overcome challenging topics. Additionally, the fact that the target audience of this study was pre-service teachers may contribute to both assisting their own learning through analogy preparation and facilitating their future professional practices.

Based on the research findings, specialized training programs can be provided to teacher candidates to enhance their skills in preparing analogies for programming courses. These programs could assist them in understanding the analogy formation process and effectively implementing it. Additionally, receiving regular feedback is crucial for evaluating the impact of using analogies in programming courses. To achieve this, new research could be planned by employing various analogy techniques in different groups to assess their effectiveness.

Author Contributions

All authors took an equal part in all processes of the article. All authors have read and approved the final version of the study.

Ethical Declaration

This study was conducted with the approval of the Erzincan Binali Yıldırım University Applied Research Ethics Center, Human Research Ethics Committee (Protocol No. 06/17), obtained at the meeting held on 29.03.2024.

Conflict of Interest

The authors declare that there is no conflict of interest with any institution or individual related to this study.

References

Alhazbi, S. (2016, December). Using flipped classroom approach to teach computer programming. In *2016 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)* (pp. 441-444). IEEE.

Alper, A., & Öztürk, S. (2019). *Programlama Öğretimindeki Ters-Yüz Öğretim Yönteminin Öğrencilerin Başarılarına, Bilgisayara Yönelik Tutumuna ve Kendi Kendine Öğrenme Düzeylerine Etkisi. Bilim Eğitim Sanat Ve Teknoloji Dergisi, 3(1),* 13-26.

Altbach, P. G., Reisberg, L. & Rumbley, L. E. (2009). *Trends in global higher education: Tracking an academic revolution.* The United Nations Educational, Scientific and Cultural Organization. http://atepie.cep.edu.rs/public/Altbach,_Reisberg,_Rumbley_Tracking_an_Academic_Revolution,_UNESCO_2009.pdf

Ambrosio, A. P., Almeida, L. S., Macedo, J., & Franco, A. H. R. (2014). *Exploring core cognitive skills of computational thinking.* Psychology of Programming Interest Group Annual Conference 2014, Brighton. https://hdl.handle.net/1822/30076

Anne, Jantos., Lisa-Marie, Langesee. (2023). 21st century skills in higher education - an empirical analysis of current challenges and potentials at a university of excellence. INTED proceedings, https://doi.org/10.21125/inted.2023.0438

Atabaş, S. (2018). *Programlama başarısını etkileyen bazı faktörlerin incelenmesi*. [Investigation of some factors affecting programming success] [Unpublished Master's thesis]. Ondokuz Mayıs Üniversitesi Eğitim Bilimleri Enstitüsü, Samsun.

Baltalı, S. (2016). *Programlama öğretiminde kullanılabilecek yazılımlara ilişkin öğretmen görüşleri.* [Teachers' views on software that can be used in teaching programming][Unpublished Master's Thesis]. Uludağ Üniversitesi Eğitim Bilimleri Enstitüsü, Bursa.

Bashir, G.M., & Hoque, A.S. (2016). An effective learning and teaching model for programming languages. Journal of Computers in Education, 3, 413 - 437. https://doi.org/10.1007/s40692-016-0073-2

Bawamohiddin, A. B., & Razali, R. (2017). Problem-based learning for programming education. *İnternational Journal on Advanced Science Engineering Information Technology, 7 (6).*2035-2050 https://www.doi.org/10.18517/ijaseit.7.6.2232

Bayazit, İ. (2011). Öğretmen adaylarının matematik öğretiminde analoji kullanımları konusundaki görüş ve yeterlilikleri. *Selçuk Üniversitesi Ahmet Keleşoğlu Eğitim Fakültesi Dergisi, 31,* 139-158.

Betchie, E., Aguinaldo. (2019). 21st Century Learning Skills Predictive Model Using PART Algorithm. https://doi.org/10.1145/3310986.3310992

Boom, K. D., Bower, M., Siemon, J., & Arguel, A. (2022). Relationships between computational thinking and the quality of computer programs. *Education and Information Technologies, 27*(6), 8289-8310.

Bozkurt, Ü. (2019). *Öğretmenlerin analojiye yönelik görüşlerinin değerlendirilmesi.* [Evaluation of teachers' views on analogy] [Unpublished Master's thesis]. Erzincan Binali Yıldırım Üniversitesi Fen Bilimleri Enstitüsü, Erzincan.

Chang, C. S., Chung, C. H., & Chang, J. A. (2020). Influence of problem-based learning games on effective computer programming learning in higher education. *Educational technology research and development, 68,* 2615-2634.

Chen, H. R., & Hsu, W. C. (2022). Do flipped learning and adaptive instruction improve student learning outcome? a case study of a computer programming course in Taiwan. *Frontiers in Psychology, 12,* 768183. https://doi.org/10.1007/s11423-020-09784-3

Dede, C., Mishra, P., & Voogt, J. (2013, October). Working group 6: Advancing computational thinking in 21st century learning. In *EDUsummIT 2013, International summit on ict in education.* http://www.edusummit.nl/fileadmin/contentelementen/kennisnet/EDUSummIT/Documenten/2013/Advancing_computational_thinking_in_21st_century_learning.pdf

Deniz, G., & Eryılmaz, S. (2019). Türkiye'de Programlama Eğitimi ile İlgili Yapılan Çalışmaların İncelenmesi: Bir Betimsel Analiz Çalışması. Eğitimde Kuram Ve Uygulama, 15(4), 319-338. https://doi.org/10.17244/eku.645387

Dinçer, S. (2005). Bilgisayar ve teknolojileri öğreniminde analoji (benzetme) yönteminin yararları ve yöntemleri. *Akademik Bilişim Konferansı*, Gaziantep.

Erümit, K. A., Karal, H., Şahin, G., Aksoy, D. A., Gencan, A. A., & Benzer, A. İ. (2019). A model suggested for programming teaching: Programming in seven steps. *Egitim ve Bilim*, *44*(197), 155–183. https://doi.org/10.15390/EB.2018.7678

Fincher, S., Jeuring, J., Miller, C. S., Donaldson, P., Du Boulay, B., Hauswirth, M., ... & Petersen, A. (2020). Notional machines in computing education: The education of attention. In *Proceedings of the working group reports on innovation and technology in computer science education* (pp. 21-50). https://doi.org/10.1145/3437800.3439202

Gentner, D., & Hoyos, C. (2017). Analogy and abstraction. *Topics in cognitive science*, *9*(3), 672-693.https://doi.org/10.1111/tops.12278

Gökoğlu, S. (2017). Programlama eğitiminde algoritma algısı: Bir metafor analizi. *Cumhuriyet International Journal of Education*, *6*(1), 1-14. https://doi.org/10.30703/cije.321430

Goletti, O., Mens, K., & Hermans, F. (2021, June). Tutors' Experiences in Using Explicit Strategies in a Problem-Based Learning Introductory Programming Course. In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1* (pp. 157-163). https://doi.org/10.1145/3430665.3456348

Gomes, A., & Mendes, A. J. (2007). Learning to program - difficulties and solutions | Academic Conference Paper. *In International Conference on Engineering Education– ICEE*, *7*(May), 3–7. https://www.researchgate.net/publication/228328491_Learning_to_program_-_difficulties_and_solutions

Günbaş, N., & İlgün, Ş. (2023). Algoritma ve Programlama Dersinin Matematik Öğretmen Adayları Perspektifinden Değerlendirilmesi. In *Ondokuz Mayis University Journal of Education Faculty* (Vol. 42, Issue December). https://doi.org/10.7822/omuefd.1298139

Gutierrez, H. T., Valdez, L. A., Peñuñuri, L. T. P., & Brindis, J. C. V. (2022). Methodology for teaching programming: Integrating best practices in the teaching and learning process with undergraduate students. *Revista de Docencia e Investigación Educativa: Journal of Teaching and Educational Research*, *8*(22), 1-7. https://www.doi.org/10.35429/JTER.2022.22.8.1.7

Harper, C., Bockmon, R., & Cooper, S. (2023). Investigating Themes of Student-Generated Analogies. CompEd 2023 - Proceedings of the ACM Conference on Global Computing Education, 1, 64–70. https://doi.org/10.1145/3576882.3617914

Harper, C., Rance, J., Owens, P., & Cooper, S. (2024). Tool-Driven Scaffolding of Student-Generated Analogies in CS1. ACM International Conference Proceeding Series, 5–8. https://doi.org/10.1145/3633053.3633061

Herala, A., Vanhala, E., Knutas, A., & Ikonen, J. (2015, November). Teaching programming with flipped classroom method: a study from two programming courses. In *Proceedings of the 15th Koli Calling Conference on Computing Education Research* (pp. 165-166). https://doi.org/10.1145/2828959.2828983

Heywood, D. (2002). The place of analogies in science education. *Cambridge Journal of Education*, *32*(2), 233-247. https://doi.org/10.1080/03057640220147577

Hromkovič, J., Kohn, T., Komm, D., & Serafini, G. (2016). Examples of algorithmic thinking in programming education. *Olympiads in Informatics*, *10*(1-2), 111-124. https://doi.org/10.15388/ioi.2016.08

Jancheski, M. (2017). Improving teaching and learning computer programming in schools through educational software. *Olympiads in Informatics*, *11*, 55–75. https://doi.org/10.15388/ioi.2017.05

Jonāne, L. (2015). Analogies in science education. *Pedagogika/Pedagogy*, *119*(3), 116-125. https://doi.org/10.15823/p.2015.027

Kandin, E. (2019). *5. sınıf öğrencilerine programlama öğretiminde hedefe dayalı senaryo kullanımının etkisi ve öğrenci görüşleri.* [The effect of goal-based scenario used for programming education of 5th graders and students' opinions] [Unpublished Master's thesis], Ondokuz Mayıs Üniversitesi Eğitim Bilimleri Enstitüsü, Samsun.

Kanika, Chakraverty, S., & Chakraborty, P. (2020). Tools and techniques for teaching computer programming: A review. *Journal of Educational Technology Systems*, *49*(2), 170-198. https://doi.org/10.1177/0047239520926971

Kaya, S., & Durmuş, A. (2011). Bilişim Teknolojileri Öğretimi İçin Geliştirilen Örnek Analojilerin İncelenmesi. *Ahi Evran Üniversitesi Eğitim Fakültesi Dergisi*, *12*(2), 235–254.

Ketenci, Ö. (2019). *Madde ve Isı Konusunda Uygulanan Analoji (Benzeşim) Üzerine Bir Araştırma.* [A research on analogy applied in matter and heat] [Unpublished Master's Thesis] Necmettin Erbakan Üniversitesi Eğitim Bilimleri Enstitüsü, Konya.

Kiss, G., & Arki, Z. (2017). The influence of game-based programming education on the algorithmic thinking. *Procedia-Social and Behavioral Sciences*, *237*, 613-617. https://www.doi.org/10.1016/j.sbspro.2017.02.020

Kobal, S., Şahin, A., & Kara, İ. (2014). Fen ve teknoloji dersinde analojilere dayalı öğretimin öğrencilerin başarıları ve hatırda tutma düzeyi üzerindeki etkisi. *Pamukkale Üniversitesi Eğitim Fakültesi Dergisi, 36(36)*, 151-162.

Kong, S. C., & Wang, Y. Q. (2020). Formation of computational identity through computational thinking perspectives development in programming learning: A mediation analysis among primary school students. *Computers in Human Behavior*, *106*, 106230. https://doi.org/10.1016/j.chb.2019.106230

Laato, S. Rauti and E. Sutinen, (2020). The Role of Music in 21st Century Education-Comparing Programming and Music Composing. http://dx.doi.org/10.1109/ICALT49669.2020.00088

Mascolo, M. F., & Fischer, K. W. (2005). Constructivist theories. Cambridge Encyclopedia of Child Development (pp. 49-63). Cambridge, England: Cambridge University Press.

Mithun, S., & Evans, N. (2018, June). Impact of the flipped classroom on students' learning and retention in teaching programming. In *2018 ASEE Annual Conference & Exposition*. http://dx.doi.org/10.18260/1-2--30608

Mozelius, P., Tomos, F., Shabalina, O., Miller, C., Malliarakis, C., C Balan, O., & Chickerur, S. (2016). Game-based technologies in teaching programming in higher education: Theory and practices. *Recent Patents on Computer Science*, *9*(2), 105-113. http://dx.doi.org/10.2174/2213275908666151030212745

Nakiboğlu, C., & Yıldırım, Ş. (2019). 10. Sınıf Öğrencilerinin

Kimyasal Bağ ile ilgili Algıları, Kimyasal Bağı Tanımlamada Kullandıkları Metaforları ve Yaptıkları Benzeşimler. *Turkiye Kimya Dernegi Dergisi Kısım C: Kimya Egitimi*, *4(*2), 61-80.

Özcan, F. Z. (2013). *Analoji tekniğinin öğrencilerin akademik başarılarına etkisinin incelenmesi ve bu sürece ilişkin öğrenci görüşlerinin belirlenmesi: 5.sınıf matematik dersi örneği*. [The analysis of effects of analogy method on students academic success and the determination of students opinions about the process: A sample of 5th grade maths class] [Unpublished Master's Thesis]. Gazi Üniversitesi Eğitim Bilimleri Enstitüsü, Ankara.

Özmen, B., & Altun, A. (2014). Undergraduate Students' Experiences in Programming: Difficulties and Obstacles Üniversite Öğrencilerinin Programlama Deneyimleri: Güçlükler ve Engeller. *Turkish Online Journal of Qualitative Inquiry*, *5*(3), 9–27. https://doi.org/10.17569/tojqi.20328

Pawelczak, D. (2017, June). Comparison of traditional lecture and flipped classroom for teaching programming. In *Proceedings of the 3rd International Conference on Higher Education Advances* (pp. 391-398). Editorial Universitat Politècnica de València. http://dx.doi.org/10.4995/HEAd17.2017.5226

Peng, W. (2010, September). Practice and experience in the application of problem-based learning in computer programming course. In *2010 International Conference on Educational and Information Technology* (Vol. 1, pp. V1-170). IEEE. https://doi.org/10.1109/ICEIT.2010.5607778

Pörn, R., Hemmi, K., & Kallio-Kujala, P. (2021). Inspiring or Confusing--A Study of Finnish 1-6 Teachers' Relation to Teaching Programming. *LUMAT: International Journal on Math, Science and Technology Education*, *9*(1), 366-396. http://dx.doi.org/10.31129/LUMAT.9.1.1355

Raman, Nambiar. (2020). Coding as an Essential Skill in the Twenty-First Century. https://www.doi.org/10.1007/978-981-15-7018-6_29

Rim, H. (2017). A Study on Teaching using Website'Code. org'in Programming Education based on Computational Thinking. *Journal of Korea Multimedia Society*, *20*(2), 382-395.http://dx.doi.org/10.9717/kmms.2017.20.2.382

Sabarinath, R., & Quek, C. L. G. (2020). A case study investigating programming students' peer review of codes and their perceptions of the online learning environment. *Education and Information* Technologies, 25(5), 3553-3575. https://doi.org/10.1007/s10639-020-10111-9

Şahin, F., Mertoğlu, H., & Çömek, A. (2001). Öğrencilerin oluşturdukları analojilerin öğrenmeye etkisi. *Yeni Bin Yılın Başında Türkiye'de Fen Bilimleri Eğitimi Sempozyumu, İstanbul.*

Saygılı, S. (2008). *Analoji ile öğretim yönteminin 9. sınıf öğrencilerinin matematik başarılarına ve yaratıcı düşünmelerine etkisi.* [The effect of analogy-enhanced teaching on mathematical success and creative thinking ability of 9th high school students] [Unpublished Master's thesis]. Çanakkale Onsekiz Mart Üniversitesi Eğitim Bilimleri Enstitüsü, Çanakkale.

Saygıner, Ş., & Tüzün, H. (2017). Programlama Eğitiminde Yaşanan Zorluklar ve Çözüm Önerileri. *11th International Computer Education and Instructional Technologies Symposium*.

Selby, C. C. (2015, November). Relationships: computational thinking, pedagogy of programming, and Bloom's Taxonomy. In *Proceedings of the workshop in primary and secondary computing education* (pp. 80-87). https://doi.org/10.1145/2818314.2818315

Shi, Y., Huang, S., & Zheng, C. (2019, June). Research on programming courses teaching based on blended learning. In *Proceedings of the 5th International Conference on Frontiers of Educational Technologies* (pp. 30-34). http://dx.doi.org/10.1145/3338188.3338198

Taşpolat, A., Özdamli, F., & Soykan, E. (2021). Programming language training with the flipped classroom model. *Sage Open*, *11*(2), https://doi.org/10.1177/21582440211021403

Tolano-Gutierrez, K., Amavizca-Valdez, O., Tadeo Portelapeñuñuri, L., & Vazquez-Brindis, C. (2022). Methodology for teaching programming: Integrating best practices in the teaching and learning process with undergraduate students. *Journal of Teaching & Educational Research/Revista de Docencia & Investigación Educativa*, *8*(22). http://dx.doi.org/10.35429/JTER.2022.22.8.1.7

Türel, Y. K., Şimşek, A., Şengül Vautier, C. G., Şimşek, E., & Kızıltepe, F. (2023). 21. Yüzyıl Becerileri ve Değerlere Yönelik Araştırma Raporu. https://ttkb.meb.gov.tr/meb_iys_dosyalar/2023_05/11153521_21.yy_becerileri_ve_degerlere_yonelik_arastirma_raporu.pdf

Uçar, E. Ü. (2021). *Fen bilgisi öğretmen adaylarının bireysel analoji oluşturmalarına ve uygulamalarına yönelik bir araştırma*. [A study on the making and applications of individual analogy of science teacher candidates] [Unpublished Master's Thesis]. Kastamonu Üniversitesi Fen Bilimleri Enstitüsü, Kastamonu.

Yang, J., Wong, G.K.W., Dawes, C. (2018). An Exploratory Study on Learning Attitude in Computer Programming for the Twenty-First Century. In: Deng, L., Ma, W., Fong, C. (eds) New Media for Educational Change. Educational Communications and Technology Yearbook. Springer, Singapore. https://doi.org/10.1007/978-981-10-8896-4_5.

Yıldız Durak, H. (2020). The effects of using different tools in programming teaching of secondary school students on engagement, computational thinking and reflective thinking skills for problem solving. *Technology, Knowledge and Learning*, *25*(1), 179-195. https://doi.org/10.1007/s10758-018-9391-y

Yıldırım, A., & Şimşek, H. (2013). Sosyal bilimlerde nitel araştırma yöntemleri. Ankara: Seçkin Yayınevi

Yılmaz, S., Eryılmaz, A., & Geban, Ö. (2002). Birleştirici benzetme yönteminin lise öğrencilerinin mekanik konularındaki kavram yanılgıları üzerindeki etkisi. *V. Ulusal Fen Bilimleri ve Matematik Eğitimi Kongresi Tam Metin Kitabı, Ankara.*

Zhao, D., Muntean, C. H., Chis, A. E., Rozinaj, G., & Muntean, G. M. (2022). Game-based learning: enhancing student experience, knowledge gain, and usability in higher education programming courses. *IEEE Transactions on Education*, *65*(4), 502-513. http://dx.doi.org/10.1109/TE.2021.3136914

Zhou, L. (2023). How to Develop 21st Century Skills in Students: The Role of LEGO Education. *Science insights education frontiers,15*(2), 2281–2283. https://www.doi.org/10.15354/sief.23.co066

Zorluoğlu, S. L., & Sözbilir, M. (2016). İyonik ve kovalent bağlar konusunda uygulanan analoji tekniğinin öğrenci başarısına etkisi. *Bayburt Eğitim Fakültesi Dergisi, 11(1)*, 84-99.

Appendix-1

Topics that participants found most challenging
Loop Structures (for, while)
Nested Control Structures (if, else, elif)
Control Structures (if, else)
String Operations (len, etc.)
Data Type Conversions
Logical Operators (and, or, not)
Array Definition and Usage (list definition)
Comparison Operators (<, >)
Problem Solving Process
Algorithms
Variables and Data Types
Assignment Operators (=, +, -)
Computer and Programming & What is Programming?

Appendix-2

Interview Questions

- What are the most challenging topics in programming education?
- Which topic did you create an analogy on?
- Can you tell us about your experience creating an analogy?
- What were the difficulties you encountered during the process?
- Were there any parts you liked during the process?
- Do you think the analogy creation process contributed to your learning?
- What do you think about its use in other courses?

Appendix-3

Analoji-1: Canlı yaşamı ve Su ilişkisi

1-) Koşul İfadeleri (if-else):
Analoji: Koşul ifadeleri, gezegenlerdeki canlı yaşamıyla benzetilebilir. Örnek; "Eğer gezegende su varsa, canlı yaşamı vardır. Aksi halde canlı yaşamı yoktur." şeklinde bir düşünce, koşul ifadeleriyle benzerlik taşır.
2-) Koşul İfadeleri (if):
Analoji: "Gezegende su varsa" ifadesi, Gezegenlerdeki canlı yaşamında bir koşul ifadesini temsil eder. Bu durumda, belirli bir şart gerçekleşirse (Gezegende su varsa) belirli bir duruma varılır (canlı yaşamı vardır).
3-) Else (aksi halde):
Analoji: "Aksi halde, canlı yaşamı yoktur" ifadesi, koşul ifadesinde sağlanan şartın karşılanmaması durumunda yapılacak çıkarımı belirtir. Yani, eğer su yoksa canlı yaşamı yoktur.
Bu analoji, Canlı yaşamı ve su ilişkisi üzerinden koşul ifadelerini açıklar. Her iki durumda da belirli şartlar altında farklı durumlar gerçekleştirilmesi gerekliliği, koşul ifadelerinin temel mantığıyla benzerlik gösterir.

Benzerlikler

- Canlı yaşamı ve su ilişkisiyle kontrol yapılarının benzer olmasının sebebi işlem aşamasında gerçekleştirilen eylemlerin bir koşula bağlı olacak şekilde yapılmasıdır.
- Sıralama yönünden benzerdirler. Mesela canlı yaşamını su ile ilişkilendirdiğimiz zaman suyun olmadığı bir durumda (bu bir koşul ifadesidir) belirli bir sonuca varılabilir. Aynı şey kontrol yapıları içinde geçerlidir. Çözüm belli bir koşulu baz alarak yapıldığı için bu kısımda aşamalar benzerlik gösterir.

Farklılıklar

- Doğal yaşamda, suyun akışı ve etkileşimi doğal olarak gelişirken, programlama kontrol yapıları insanlar tarafından bilinçli bir şekilde tasarlanır ve uygulanır.
- Doğal yaşamdaki kontrol yapıları genellikle doğal seçilim ve evrimsel süreçler tarafından şekillenirken, programlama kontrol yapıları insanlar tarafından bilinçli olarak uygulanır.

Analoji-2: Sayıların karşılaştırılması

Karşılaştırma operatörlerini, iki sayı arasındaki ilişkiyi karşılaştırmak için kullanılan birer terazi olarak düşünebiliriz. Terazinin bir kefesine bir sayıyı, diğer kefesine ise diğer sayıyı koyarsak, terazinin hangi kefenin ağır bastığını görebiliriz. Bu bize iki sayının birbirine göre büyüklük, küçüklük veya eşitlik ilişkisini verir. Örneğin, 5 ve 3 sayılarını karşılaştırmak için 5'i bir kefeye, 3'ü ise diğer kefeye koyarsak, terazinin 5'in bulunduğu kefeye doğru ağır bastığını görebiliriz. Bu bize 5'in 3'ten büyük olduğunu gösterir.

Benzerlikler

- Her ikisi de iki değeri karşılaştırır.
- Her ikisinin de sonucu bir değerdir.
- Her ikisinin de sonucu, karşılaştırılan değerler arasındaki ilişkiyi gösterir.

Farklılıklar

- Karşılaştırma operatörleri, sayılar, karakterler, dizeler, listeler ve diğer veri türleri gibi farklı değerleri karşılaştırabilir. Terazi ise sadece ağırlıklar karşılaştırabilir.
- Karşılaştırma operatörlerinin sonuçları, sayısal değerler olabilir. Terazinin sonuçları ise ağırlık değerleridir.

Analoji 3- Buluşma Planı

Döngüleri, öğrencilerin her haftasonunda bir kafede buluşmalarına benzetebiliriz. Her hafta belirli bir koşul, yani hafta sonu, sağlandığında buluşma tekrarlanır. Ancak, bir hafta sonu herkesin uygun olmadığı durumda veya başka bir etkinlik planlandığında, bu döngü dışındaki bir durumu temsil eder. Yani, belirli bir düzeni olan ancak esneklik sağlayan bir yapı söz konusudur.

Benzerlikler

- Her hafta belirli bir düzene göre tekrarlanır.
- Belirli bir koşul, hafta sonu, sağlandığında tekrarlanır.
- Belirli bir işlemi tekrarlamak için kullanılır.
- Döngü içindeki işlemler belirli bir düzene göre yapılır.

Farklılıklar

- For döngüleri, belirli bir iterable (liste, demet vb.) üzerinde dolaşır.
- Sona erme koşulu, otomatik olarak iterable'ın sona ermesidir. Buluşma düzeni, hafta sonu veya uygunluk durumu gibi kullanıcı tarafından belirlenen bir şarta bağlıdır.

- For Döngüsü; sayaç veya iterable'ın içindeki elemanları kontrol eder. Diğer taraftan kişiler belirlenen bir şartın sağlanıp sağlanmadığını kontrol eder.
- For Döngüsü; sabit bir düzeni tekrarlar, değişkenlik sağlamaz. Buluşma planı, belirli bir düzen içinde olmasına rağmen, esneklik ve adaptasyon sağlar.