

Camtree Digital Library



Investigating the Integration of Parson's Programming Puzzles into a Lesson Study Framework in Computer Science Education

Author	Iskaziyeva, Agis;Bejerano, John Paul
Title	Investigating the Integration of Parson's Programming Puzzles into a Lesson Study Framework in Computer Science Education
Publication date	2025
Download date	2026-06-16 12:01:50
Link to Item	https://hdl.handle.net/20.500.14069/941

RESEARCH LESSON STUDY REPORT

Investigating the Integration of Parson's Programming Puzzles into a Lesson Study Framework in Computer Science Education

Agis Iskaziyeva, John Paul Bejerano

Nazarbayev Intellectual School, Atyrau, Kazakhstan

Abstract

Background and purpose: Research indicates that students often struggle with coding logic, experiencing frustration and disengagement. This study explores the integration of Parson's Programming Puzzles into computer science education as a pedagogical tool, through lesson study. By placing scrambled lines of code into the correct sequence, Parson's Puzzles aim to reduce frustration and highlight the core logic of programming in a more game-like format. The interactive and hands-on nature of these puzzles aims to make programming more accessible, engaging, and effective for diverse learners.

Aims: The primary aim of this research is to evaluate the impact of integrating Parson's Programming Puzzles on student engagement in computer science lessons, perseverance in solving coding problems, and understanding of coding logic and programming skills. A secondary aim is to investigate how collaborative lesson planning and reflection (the lesson study approach) can further optimize these teaching strategies.

Study design or methodology: Participants in the study included 15 secondary school students (aged 15–17) enrolled in computer science classes at Nazarbayev Intellectual School, Atyrau, Kazakhstan. Three research lessons were observed over a period of three weeks. Lessons were designed collaboratively by a team of educators using Parson's Puzzles to address specific learning objectives. Data collection methods included classroom observations to evaluate student engagement and interaction, pre- and post-assessments to measure improvement in problem-solving and coding skills, and student surveys and interviews to gather qualitative feedback on their learning experiences. The collected data were analysed using both quantitative (assessment scores) and qualitative (thematic analysis of interviews and surveys) methods.

Findings: Group A (High Performers) demonstrated significant improvement in problem-solving strategies and actively engaged with complex puzzles. Group B (Intermediate Learners) developed a better understanding of coding logic and structure, finding the puzzles a balance between challenge and support. Group C (Lower Performers) reported reduced frustration with coding and increased persistence in completing tasks.

Conclusions, originality, value and implications: Integrating Parson's Programming Puzzles within a lesson study framework fosters engagement, persistence, and improved coding outcomes in diverse learner groups. This approach provides educators with an innovative, hands-on method to address common challenges in programming education. The findings contribute to instructional practices, highlighting the value of collaborative lesson planning and innovative tools in enhancing computer science education. Future research could explore the long-term impact of this method and the integration of puzzles of varying complexity.

Keywords: Parson's programming puzzles; lesson study framework; computer science education; student engagement; coding logic.

Context

The study was conducted with secondary school students aged 15–17 at Nazarbayev Intellectual School in Atyrau, Kazakhstan. The school provides a strong emphasis on STEM education and innovative teaching approaches, making it an ideal environment for implementing collaborative lesson study frameworks.

Co-researchers

The study also involved Nurgul Zhanaliev (Computer Science Teacher).

Overall aims of lesson study

The primary goal was to enhance student engagement, perseverance, and problem-solving skills in computer science education by integrating Parson's Programming Puzzles (see examples in appendices) into lessons. The lesson study aimed to foster collaborative teaching practices, improve student outcomes, and address challenges in learning programming concepts.

Motivation/need for this lesson study & review of existing approaches

To better situate this work within current research, this section outlines key studies that inform the rationale behind using 'Parson's Puzzles' in a lesson study context. The rationale for this lesson study was the observation that students often face frustration and disengagement when learning coding due to its abstract nature and complexity. Challenges included ensuring differentiated support for students with varying skill levels while maintaining engagement in the classroom. Previous methods of teaching programming lacked interactive, hands-on activities that address diverse student needs. Parson's Programming Puzzles, first introduced by Parsons and Haden (2006), are rearrangement-based tasks that require students to put jumbled lines of code into the correct logical order, promoting a focus on structure and problem-solving. These puzzles were selected as an innovative tool to enhance engagement, perseverance, and logical reasoning. They help reduce cognitive load by removing the need to focus on syntax, allowing students to concentrate on control flow and logic.

Research by Ericson et al. (2017) supports this approach, showing that Parson's Puzzles are particularly effective in supporting novice learners' understanding of programming structures. Other structured learning tools, such as block-based programming (e.g. Scratch) and visual debugging tools (e.g. Python Tutor), have also shown positive results in supporting logic comprehension among beginners (Price & Barnes, 2015).

However, there is limited research on embedding Parson's Puzzles within a collaborative professional development model such as lesson study. The lesson study framework, widely used to improve instructional strategies through co-planning, observation, and reflection, has demonstrated

impact in various subjects (Lewis, Perry, & Hurd, 2009). By integrating these puzzles into a lesson study cycle, this research explores a novel combination of tool and methodology to improve student engagement and problem-solving in programming education. This study contributes to the literature by being one of the first to examine the integration of Parson's Programming Puzzles into a collaborative lesson study framework in a secondary school computer science context.

- The expected outcome was improved student persistence, reduced frustration, and better comprehension of coding logic through collaborative lesson planning and reflection.

Focus and research questions

The primary research question was how the integration of Parson's Programming Puzzles into a lesson study framework affects student engagement, perseverance, and achievement in computer science. Specifically, the research addressed the following questions:

- (1) How does the integration of Parson's programming puzzles into the lesson study framework affect student achievement in computer science education?
- (2) What impact does the inclusion of Parson's programming puzzles in lesson study have on student perseverance and engagement in computer science classes?
- (3) How does the use of Parson's programming puzzles influence students' perceptions of learning within the computer science curriculum?

We were particularly interested in comparing these outcomes to traditional lesson formats, where students write code from scratch without puzzle-based scaffolding.

Lesson study plan and activities

Teachers collaborated to design lessons integrating Parson's Puzzles (see appendices for example puzzles). In addition to providing puzzles, teachers offered scaffolding such as hints, peer support opportunities, and live demonstrations of puzzle logic to facilitate student learning and engagement. The lessons were implemented in classrooms with observations to monitor student engagement and interaction. Post-lesson reflection sessions were conducted to refine teaching strategies. Data was collected through surveys, interviews, and assessments to evaluate learning outcomes.

Choice of and rationale case students

The three case students were selected to reflect diverse learning profiles—high achiever, mid-level learner, and struggling student—based on previous assessment data and teacher recommendations. This purposeful sampling enabled an in-depth exploration of how Parson's Puzzles affected students with varying levels of coding experience and confidence.

- **Case Student 1:** This student is an A* learner who excels in problem-solving and logical reasoning. They consistently achieve top grades and demonstrate advanced understanding of programming concepts. They were chosen to evaluate how Parson's Programming Puzzles can

challenge high-performing students and further enhance their problem-solving strategies and coding skills.

- **Case Student 2:** This student is a B- learner who demonstrates a moderate understanding of coding logic but occasionally struggles with consistency and applying concepts. Their academic performance indicates a solid foundation with room for improvement. They were selected to determine how Parson's Programming Puzzles can bridge gaps in their understanding and improve their ability to structure and debug code effectively.
- **Case Student 3:** This student is a D learner who often faces significant challenges, including frustration and disengagement during programming lessons. Their academic results reflect struggles in grasping coding concepts. They were chosen to assess whether Parson's Programming Puzzles can reduce coding anxiety, foster persistence, and provide a more accessible way to learn and apply basic programming skills.

Assessment of how each student would have progressed if we had taught the usual/previous curriculum for these lessons

If the usual curriculum were applied, the expected outcomes for the students would have been as follows:

- Case Student 1 (A* learner) would likely achieve the overall goals but may not have been adequately challenged to further enhance their advanced problem-solving skills.
- Case Student 2 (B- learner) would likely meet the basic goals but may continue to struggle with consistency in applying coding logic and structuring programs.
- Case Student 3 (D learner) would likely remain disengaged, struggling to meet the goals, and require additional support to grasp the fundamental coding concepts.

In the class of 15 students:

- 6 students were expected to achieve the overall goals.
- 3 students were predicted to exceed the overall goals.
- 6 students were expected to continue struggling and require further teaching or support.

Research Lesson 1 (RL 1)

Aims of RL1

To introduce Parson's Programming Puzzles and evaluate their initial impact on student engagement and problem-solving skills.

Predictions for each case student at two/three points mid lesson and end of lesson

Case student 1:

- Mid-lesson: Expected to excel with minimal difficulty, engaging with more complex puzzles.
- End of lesson: Likely to demonstrate strong problem-solving strategies and higher engagement levels.

Case student 2:

- Mid-lesson: Expected to engage but require guidance to navigate the puzzle logic effectively.
- End of lesson: Likely to show moderate improvement in understanding coding structure.

Case student 3:

- Mid-lesson: Expected to struggle initially with frustration but gradually engage with support.
- End of lesson: Likely to demonstrate improved persistence and reduced coding anxiety.

Observations of case students (including comparison with predictions, and what this suggested for RL2.

- Student 1: Successfully engaged with challenging puzzles, showing advanced problem-solving skills.
- Student 2: Required occasional support but demonstrated steady progress in logical thinking.
- Student 3: Showed visible persistence and reduced frustration, completing simpler puzzles with encouragement.

Student Interview Feedback on RL1

- Student 1: Enjoyed the complexity and felt it enhanced their skills.
- Student 2: Appreciated the structure and found the puzzles moderately challenging.
- Student 3: Reported reduced frustration and found the puzzles easier to approach than writing code from scratch.

Research Lesson 2 (RL 2)

Aims of RL2

To build on the engagement from RL1 by introducing intermediate puzzles focused on debugging and logical structuring.

Predictions for each case student at two/three points mid lesson and end of lesson

Case student 1:

- Mid-lesson: Expected to confidently tackle intermediate-level puzzles, exploring creative and alternative solutions. Likely to work independently with minimal guidance.
- End of lesson: Predicted to complete all puzzles, demonstrating refined problem-solving strategies. May take the initiative to assist peers or suggest further improvements to the puzzles.

Case student 2:

- Mid-lesson: Likely to engage steadily, showing improved logical reasoning with debugging tasks. May require occasional prompts or peer support to maintain momentum.
- End of lesson: Expected to complete most puzzles, demonstrating a noticeable improvement in structuring and debugging skills. Confidence in coding abilities is likely to increase.

Case student 3:

- Mid-lesson: Anticipated to struggle initially but show persistence with guided support. Likely to complete simpler sections of the puzzles and begin making connections between concepts.
- End of lesson: Predicted to complete several puzzles with reduced frustration, demonstrating an improved understanding of basic coding logic and a willingness to continue tackling challenges.

Observations of case students (including comparison with predictions, and what this suggested for RL3.

- Student 1: Excelled and began experimenting with alternate approaches to puzzles.
- Student 2: Improved logical thinking, though occasionally required support to debug.
- Student 3: Demonstrated increased confidence and persistence, completing tasks with minimal frustration. Specifically, the student independently completed a debugging puzzle with three logic branches, which they had previously struggled with. They asked fewer questions and worked steadily throughout the lesson.

Student Interview Feedback on RL2

- Student 1: Found the puzzles increasingly engaging and enjoyable.
- Student 2: Felt the challenges were balanced and appreciated the structured feedback.
- Student 3: Reported growing confidence in solving tasks independently.

Research Lesson 3(RL 3)

Aims of RL3

To evaluate the cumulative impact of Parson's Programming Puzzles by introducing advanced problems requiring multi-step solutions.

Predictions for each case student at two/three points mid lesson and end of lesson

Case student 1:

- Mid-lesson: Expected to excel with advanced, multi-step puzzles, showcasing strong analytical and creative thinking. Likely to be highly engaged and motivated by the challenge.
- End of lesson: Predicted to achieve full mastery of the puzzles, demonstrating innovative problem-solving approaches. May provide insights or feedback for further refinement of the puzzles.

Case student 2:

- Mid-lesson: Likely to engage actively with the puzzles, applying previous learning to solve complex problems. May occasionally need guidance for particularly challenging tasks.

- End of lesson: Expected to complete most puzzles with a stronger understanding of multi-step problem-solving and debugging. Likely to exhibit a notable increase in confidence and coding proficiency.

Case student 3:

- Mid-lesson: Anticipated to approach puzzles with more confidence, showing persistence in tackling increasingly complex problems. Likely to require support but will demonstrate gradual progress.
- End of lesson: Predicted to complete simpler sections of advanced puzzles, showing significant improvement in coding persistence and reduced anxiety. Likely to express pride in their accomplishments and an increased interest in programming.

Observations of case students (including comparison with predictions and what this means beyond the study).

- Student 1: Excelled with advanced puzzles, showcasing innovative problem-solving skills.
- Student 2: Demonstrated notable improvement in debugging and logical structuring.
- Student 3: Persisted through tasks and showed measurable improvement in confidence and problem-solving. This was evidenced by the student volunteering answers during class discussion, initiating peer support, and solving three puzzles in sequence without teacher assistance.

Student Interview Feedback on RL3

- Student 1: Felt fully challenged and satisfied with the learning experience.
- Student 2: Expressed confidence in applying newly acquired skills to future coding tasks.
- Student 3: Reported a significant reduction in coding anxiety and an increased interest in programming.

Findings

The lesson study revealed significant insights into the learning progress of the case students and the effectiveness of integrating Parson’s Programming Puzzles into the curriculum. Case Student 1, a high-achieving learner, consistently excelled across all lessons, demonstrating advanced problem-solving abilities and strong engagement with the puzzles. They explored creative solutions and provided valuable feedback during class discussions, such as suggesting alternative code sequences for the puzzles and identifying errors in peer solutions, which facilitated class-wide reflection. Case Student 2, a mid-level performer, showed steady improvement in logical reasoning and debugging skills. While initially requiring occasional guidance, they gained confidence and competence, particularly in solving intermediate puzzles, bridging gaps in their understanding of coding logic. Case Student 3, who initially struggled with frustration and disengagement, exhibited notable progress in persistence and confidence. By the end of the lesson sequence, they completed simpler puzzles independently, demonstrating reduced coding anxiety and a growing interest in programming.

These results confirm that Parson’s Puzzles, when embedded in a lesson study framework, can benefit learners across various proficiency levels.

The findings highlight the versatility and impact of Parson’s Programming Puzzles in fostering engagement and skill development among diverse learners. This approach not only supported advanced students in achieving mastery but also provided accessible pathways for struggling learners to build foundational coding skills.

The success of this approach underscores the need to incorporate interactive and differentiated tools like Parson’s Programming Puzzles into teaching practices. In future lessons for this class, puzzles and similar hands-on activities will continue to play a key role in balancing the needs of high achievers and struggling learners while fostering a collaborative learning environment. For other classes, this curricular sequence will be adapted to include puzzles aligned with specific learning objectives, ensuring accessibility for students of varying skill levels. Teacher collaboration in designing and reflecting on these lessons will be promoted to enhance both teaching strategies and student outcomes across the board.

Post RLS student assessments and comparative table

	Total students achieving overall learning goals of the RLS	Total students exceeding the overall goals of the RLS	Total students still struggling and requiring further support/teaching
a. Likely outcome predicted if we had taught as we always have done	6	3	6
b. Outcome at end of lesson study	12	1	2
b minus a =	6	-2	-4

These results demonstrate the effectiveness of the Parson’s Programming Puzzles in fostering engagement and improving student outcomes across various skill levels. The significant reduction in the number of struggling students highlights the accessibility of this approach, which successfully supported learners in building confidence and achieving learning goals.

By explicitly comparing the predicted vs. actual results, we provide clear evidence that Parson’s Programming Puzzles, combined with reflective lesson study, directly contributed to the gains observed.

Any resource or approach that you created or trialled

The primary resource used in this lesson study was Parson’s Programming Puzzles, which are interactive tools designed to help students learn coding logic by arranging scrambled lines of code into the correct order. The puzzles were tailored to align with the specific curriculum objectives for computer science and were differentiated by complexity to cater to varying skill levels. The purpose of this resource was to foster engagement, reduce coding anxiety, and build problem-solving skills among students. An illustrative example involves a loop and conditional statements for printing prime numbers, which students must reorder correctly to achieve the intended output—reinforcing logical sequence and syntax awareness without the overwhelming nature of writing every line from scratch. This approach could be used by other educators to promote active learning and logical reasoning in programming lessons, especially in diverse classrooms with varying skill levels.

Significant quotations

Students and teachers provided valuable feedback during and after the lesson study, revealing the impact of the approach:

- *“I liked that the puzzles felt like solving a game instead of writing boring code.” (Student 3)*
- *“It helped me understand the structure of a program much better. I feel more confident debugging code now.” (Student 2)*
- *“Watching students help each other and discuss solutions made me rethink how collaborative activities can improve learning outcomes.” (Colleague)*

These statements highlight the positive reception of Parson’s Programming Puzzles and their effectiveness in improving both student engagement and teacher practices.

Changes to practice

As a result of observations and findings, I incorporated more collaborative and interactive activities into programming lessons, such as peer discussions during puzzle-solving. Additional data, such as pre- and post-assessment scores, were collected to measure the impact on student learning outcomes. The biggest impact was seen in how students approached coding tasks with greater confidence and persistence. Notably, teachers held reflection sessions after each lesson, analysing puzzle complexity and adjusting hints or instructions to better support struggling learners. The most significant change to my teaching was the adoption of differentiated tools to better support diverse learners, ensuring that both advanced and struggling students could benefit from the lessons.

Reflective evaluation

The parts of this RLS that worked well included the collaborative lesson planning and the use of differentiated puzzles. These approaches effectively engaged students and supported their learning at various levels. A surprising outcome was the level of peer interaction and collaboration that emerged naturally during puzzle-solving activities. Support from colleagues and shared reflections during post-

lesson discussions were instrumental in the success of the lesson study. External challenges included time constraints for planning and adapting puzzles, but these were managed effectively through collaboration. Going forward, we plan to refine the puzzle designs further, possibly offering optional “bonus challenges” for advanced students. Developing my practice through this RLS highlighted the importance of designing engaging and accessible learning tools. In hindsight, I would have allocated more time for refining the puzzles based on student feedback. My advice to others conducting Research Lesson Study is to prioritize collaboration and be open to iterative improvements based on observations and feedback.

Overall, I learned that incorporating innovative resources like Parson’s Programming Puzzles can significantly enhance student engagement and outcomes. Collaborative lesson study frameworks provide valuable insights for refining teaching practices and improving learning experiences.

Ethical considerations and relationships

Several ethical issues were considered during the study. Student participation and feedback were collected with their informed consent, ensuring their privacy was safeguarded. Compliance with school policies on data protection and e-safety was strictly adhered to. The professional relationship with colleagues and students was maintained by fostering a respectful and collaborative environment. Communication with stakeholders, including school administrators and parents, ensured transparency and support for the project. Safeguarding measures, such as avoiding the use of identifiable student data and following the school’s child protection policies, were implemented to protect students and maintain ethical standards throughout the study.

Making the lesson study findings public

The findings of this lesson study will be communicated and shared through various methods to maximize their impact and reach. Within my institution, I plan to present the results during a staff meeting to highlight the effectiveness of Parson’s Programming Puzzles in improving student engagement and coding skills. Additionally, an open research lesson will be organized after school, allowing colleagues to observe the puzzles in action and discuss their application in other subjects. A visual poster summarizing key findings will also be displayed in the staff room to encourage further interest and discussion.

Locally, the findings will be shared during a professional development meet-up for educators, with a focus on innovative teaching tools for STEM subjects. I also plan to conduct an open lesson for local educators, demonstrating the implementation of Parson’s Programming Puzzles and inviting feedback for potential refinement.

For wider dissemination, the Camtree publication will be promoted through social media platforms and professional networks to reach a broader audience of educators and researchers. Additionally, I will explore opportunities to present the findings at regional and international conferences on education and STEM teaching.

For individuals or schools attempting this approach in a lesson study, I recommend starting with a collaborative planning process to tailor puzzles to the specific needs of their students. Regular reflection and adjustment based on student feedback are crucial for ensuring the effectiveness of the approach. Sharing resources and outcomes with colleagues can also help foster a community of practice.

Next steps

The next step involves further refining the use of Parson's Programming Puzzles by experimenting with different levels of complexity and integrating them into other programming topics. I plan to focus on exploring how the puzzles can be adapted for group activities to encourage collaborative learning. Another direction is to evaluate the long-term impact of this approach on students' coding proficiency and interest in computer science.

We will also document the changes made lesson by lesson, so that subsequent reviews can clearly see how feedback has shaped each revision.

Supplementary materials/resources

The following supplementary materials were created during this lesson study:

- A series of Parson's Programming Puzzles designed to teach coding logic and debugging skills. These puzzles were differentiated into beginner, intermediate, and advanced levels to cater to diverse learners.
- A lesson plan template for incorporating Parson's Programming Puzzles, including objectives, expected outcomes, and strategies for classroom implementation.
- A spreadsheet summarizing student assessment outcomes before and after the intervention, providing insights into the effectiveness of the puzzles.

These materials can be used by other educators to replicate the approach or adapt it for their specific classroom needs. For access to these materials, refer to the appendices.

References

- Ericson, B. J., Margulieux, L. E., & Guzdial, M. (2017). Investigating the impact of cognitive load in Parsons problems. In *Proceedings of the 2017 ACM Conference on International Computing Education Research* (pp. 60–68). ACM.
- Lewis, C., Perry, R., & Hurd, J. (2009). Improving mathematics instruction through lesson study: A theoretical model and North American case. *Journal of Mathematics Teacher Education*, 12(4), 285–304.
- Parsons, D., & Haden, P. (2006, January). Parson's programming puzzles: A fun and effective learning tool for first programming courses. In *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52* (pp. 157-163). Australian Computer Society, Inc.

Price, T. W., & Barnes, T. (2015). Comparing textual and block interfaces in a novice programming environment. *In Proceedings of the eleventh annual International Conference on International Computing Education Research* (pp. 91–99). ACM.

About Camtree

Camtree: the Cambridge Teacher Research Exchange is a global platform for close-to-practice research in education. Based at Hughes Hall, University of Cambridge, Camtree draws on high-quality research from around the world to support educators to reflect on their practice and carry out inquiries to improve learning in their own classrooms and organisations. The outcomes of these inquiries, once peer reviewed, can be published within the Camtree digital library under a Creative Commons Licence (CC-BY 4.0). You can find out more about Camtree and its digital library at www.camtree.org.

Appendix 1: A Series of Parson's Programming Puzzles

Lesson 1: Represent Student Data

Rearrange the code blocks to correctly create and access a nested list of students.

Shuffled Code Block	Order
<code>]</code>	
<code>["Bob", 17, "B"], # Student 2</code>	
<code>for student in students:</code>	
<code>["Charlie", 16, "A"] # Student 3</code>	
<code>["Alice", 16, "A"], # Student 1</code>	
<code>students = [</code>	
<code>print(student[0])</code>	

Lesson 2: Access Elements in Nested Lists

Rearrange the code blocks to calculate the average grade and print grades for a specific student.

Shuffled Code Block	Order
<code>avg = sum(grades[0]) / len(grades[0])</code>	
<code>[75, 80, 82], # Student 2</code>	
<code>grades = [</code>	
<code>print("Student 1 Average:", avg)</code>	
<code>print(grade)</code>	
<code>for grade in grades[1]:</code>	
<code>[88, 92, 94] # Student 3</code>	
<code>]</code>	
<code>[85, 90, 78], # Student 1</code>	

Lesson 3: Dynamically Create a Nested List

Rearrange the code to correctly input student data and display the nested list.

Shuffled Code Block	Order
<code>grade = input("Enter student grade: ")</code>	
<code>for i in range(3):</code>	
<code>students = []</code>	
<code>age = int(input("Enter student age: "))</code>	
<code>print("Student Data:", students)</code>	
<code>students.append([name, age, grade])</code>	
<code>name = input("Enter student name: ")</code>	

Lesson 4: Enter and Analyse Scores

Rearrange and debug the code to input scores for students and calculate averages.

Shuffled Code Block	Order
<code>avg = sum(student) / len(student)</code>	
<code>student_scores = list(map(int, input().split()))</code>	
<code>for i in range(3):</code>	
<code>scores = []</code>	
<code>for i, student in enumerate(scores):</code>	
<code>print("Top Student is Student", top_student + 1, "with an average score of", highest_avg)</code>	
<code>top_student = -1</code>	
<code>print("Enter scores for Student", i + 1)</code>	
<code>scores.append(student_scores)</code>	
<code>if avg > highest_avg:</code>	
<code>highest_avg = 0</code>	
<code>top_student = i</code>	
<code>highest_avg = avg</code>	

Lesson 5: Weekly Schedule

Rearrange the code blocks to modify and print a weekly schedule.

Shuffled Code Block	Order
<code>["Monday", ["Math", "Physics"]],</code>	
<code>schedule[0][1].append("Art")</code>	
<code>schedule = [</code>	
<code>for day in schedule:</code>	
<code>schedule[1][1].remove("Biology")</code>	
<code>["Wednesday", ["Chemistry", "History"]]</code>	
<code>]</code>	
<code>print(day[0], ":", " ", ".join(day[1]))</code>	
<code>["Tuesday", ["English", "Biology"]],</code>	

Lesson 6: Analyze and Rank Student Performance

Rearrange and debug the code to calculate total grades and rank students.

Shuffled Code Block	Order
<code>for student in students:</code>	
<code>["Alice", [85, 90, 78]],</code>	
<code>total = sum(student[1])</code>	
<code>totals.sort(key=lambda x: x[1], reverse=True)</code>	
<code>print(f"{rank}. {student[0]}: {student[1]}")</code>	
<code>["Bob", [75, 80, 82]],</code>	
<code>totals.append([student[0], total])</code>	
<code>]</code>	
<code>students = [</code>	
<code>for rank, student in enumerate(totals, 1):</code>	
<code>["Charlie", [88, 92, 94]]</code>	
<code>totals = []</code>	

Appendix 2: Lesson Plan

Long-term plan unit: Unit 11.3A: Nested Lists and Problem Solving in Python		School: Nazarbayev Intellectual school Atyrau	
Date: 6.02-9.02.24		Teacher name: Agis Iskaziyeva	
Grade: 11 F		Number present: 15	absent:
Theme of the lesson		Introduction to Nested Lists	
Learning objectives that are achieved at this lesson(Subject Programmed reference)		<ul style="list-style-type: none"> • Create nested lists to store and access data. • Enter elements into nested lists from the keyboard. • Solve applied problems using nested lists in real-life scenarios. 	
Lesson objectives		<ul style="list-style-type: none"> • Understand the concept of nested lists and how to create them. • Dynamically input data into nested lists using Python. • Apply problem-solving skills to analyze and solve real-world problems using nested lists. 	
Success criteria		Success criteria Knowledge: Define and create nested lists. Understanding: Access and manipulate elements in nested lists. Applying: Solve tasks related to nested lists in Python Analysis: Analyse problem to design Evaluation: Compare and analyze solutions for nested list-based problems.	
Language objectives		Learners can: <ul style="list-style-type: none"> • Explain the structure and use of nested lists in Python. • Use key terms such as "nested lists", "elements", and "iteration" accurately during discussions. 	
Values instilled at the lesson		Mutual support and mutual respect of group work, academic honesty	
Cross-curricular links		Mathematics: Use lists to calculate averages and solve data-related problems.	
Previous learning		Unit 11.2A and 11.2B must be completed before attempting this unit. Knowledge of one-dimensional arrays will be essential.	
Planned stages of the lesson	Planned activities at the lesson		Resources
	Teacher's activities	Student's activities	

<p>Beginning 10 min</p>	<p>Organizing time. The teacher greets the students. Takes the register.</p> <p>Actualization of knowledge Recap the Analysis stage. Checks report of students.</p> <p>Questions. Evaluation: to recap previous lesson</p>	<p>Greets the teacher. Concentrates their attention.</p> <p>Answer. Discuss.</p>	<p>Presentation</p>
<p>Middle 60 min</p>	<p>1. Introduction to Nested Lists (10 minutes) Explain the concept of nested lists using examples. Show a simple nested list on the board.</p> <hr/> <p>2. Task 1: Represent Student Data (15 minutes) Activity: Use Parson's Programming Puzzles to arrange code blocks for creating and accessing nested lists. Evaluation: Students successfully rearrange the blocks and explain the final structure.</p> <hr/> <p>3. Task 2: Dynamically Input Data (15 minutes) Activity: Use Parson's Puzzles to dynamically input student names, ages, and grades into a nested list. Evaluation: Students complete the task and share results with the class.</p> <hr/> <p>4. Task 3: Solve an Applied Problem (20 minutes) Activity: Work in pairs to solve a problem: Calculate and print the average grades for all students from a nested list. Use shuffled blocks to solve the problem. Evaluation: Students compare solutions and explain their logic.</p>	<p>Take notes and ask questions for clarification.</p> <p>Answers.</p> <p>Feedback.</p> <p>Creates code</p> <p>Debug and run the program.</p> <p>Discuss.</p>	<p>Presentation</p> <p>Presentation Task from "Lesson 1: Represent Student Data" (Shuffled Table in Docx).</p> <p>Python interpreter or IDE for running code. Presentation A4</p> <p>Python interpreter or IDE for running code. A4</p>

End 10 min	Consolidation of the studied topic. Discuss what was learned and how it applies to future lessons. Teacher summarizes key points and reinforces the importance of problem-solving skills.	Students reflect on their understanding and complete a quick checklist: Did they understand nested lists? Could they solve the tasks successfully?	Presentation
Home task	Write full report for Design stage.		Presentation
Differentiation – how do you plan to give more support? How do you plan to challenge the more able learners?	Assessment – how are you planning to check students’ learning? Use this section to record the methods you will use to assess what students have learned during the lesson.	Health and safety regulations Ensure ergonomic seating and screen positioning in the computer lab.	
Differentiation can be by task, by outcome, by individual support, by selection of teaching materials and resources taking into account individual abilities of learners (Theory of Multiple Intelligences by Gardner). Differentiation can be used at any stage of the lesson keeping time management in mind.	Monitor individual and group work during tasks. Provide feedback on code accuracy and logic. Provide additional examples for students who need more support. Offer advanced tasks for students who complete early.	During the lesson students should keep ergonomics and safety rules in CS lab room	

Note: Reflection and Evaluation Sections in main report

Appendix 3: Pre- and Post-Test Scores

Student ID	Pre-Assessment Score	Post-Assessment Score	Improvement
1	65.23	65.42	0.19
2	54.89	67.08	12.19
3	81.25	84.89	3.64
4	66.67	85.23	18.56
5	85.57	85.42	-0.15
6	78.75	85.57	6.82
7	75.42	86.48	11.06
8	84.58	86.48	1.90
9	84.55	86.67	2.12
10	88.75	89.20	0.45
11	85.42	89.55	4.13
12	87.08	90.80	3.72