Bachelor's Thesis Computing Science

Educational Escape Rooms in Secondary Computer Science Education

A content analysis, design of an escape room, and evaluation of learning outcomes

Stefan Vlastuin
s1044388

June 10, 2022

First supervisor/assessor:
Dr. Mara Saeli

Second assessor:
Prof. Erik Barendsen

Radboud University

Abstract

Recently, teachers have started bringing escape rooms into their classrooms. Due to its novelty, research on this topic is limited. Specifically, not much research has been done on the question of how to use this teaching method in secondary computer science education. We answered this question in two phases. First, we developed content criteria for using an escape room as a teaching method, by studying characteristics of the content of existing computer science escape rooms. Second, we created an escape room and evaluated its learning outcomes, in two ways: using learner reports, in which students could report about a variety of possible learning outcomes, and using exercises, that objectively measured achievement of the learning objectives. We concluded that the learning objectives were achieved. Students also reported learning about content themselves. Moreover, we found that escape rooms can be used as a self-reflection tool for students, as students reflect on their own knowledge and skills. The results inform teachers on when and how to use escape rooms in secondary computer science education.

Acknowledgements

I would like to thank my supervisor Mara Saeli for always guiding and supporting me in the process of writing my thesis. Several other staff members of the Science Education Research group helped me as well, which was highly appreciated. I also owe many thanks to Renske Weeda for allowing me to test the escape room in her classroom. Finally, I want to thank Martin Bruggink, for his extremely useful advice and feedback on the design of the escape room.

Contents

1 Introduction **6**
- 1.1 Escape Rooms & Education . . . 6
- 1.2 Research Questions . . . 7
- 1.3 Structure of Thesis . . . 8

2 Related Work **9**
- 2.1 Educational Escape Rooms . . . 9
  - 2.1.1 Goals of Educational Escape Rooms . . . 9
  - 2.1.2 Debriefing . . . 11
  - 2.1.3 Design of Educational Escape Rooms . . . 11
  - 2.1.4 Escapp Web Platform . . . 13
  - 2.1.5 Examples in Computer Science . . . 14
  - 2.1.6 Evaluation Methods . . . 15
- 2.2 Computer Science Education . . . 16

3 Content Analysis **18**
- 3.1 Methodology . . . 18
- 3.2 Results . . . 19
  - 3.2.1 Computer Science ERs . . . 19
  - 3.2.2 Groups . . . 19
  - 3.2.3 Technical vs. Social Topics . . . 22
  - 3.2.4 Content Criteria . . . 22

4 Design of an ER **24**
- 4.1 Choice of Topic . . . 24
- 4.2 Methodology . . . 24
- 4.3 Results . . . 26
  - 4.3.1 Participants . . . 26
  - 4.3.2 Objectives . . . 27
  - 4.3.3 Theme . . . 29
  - 4.3.4 Puzzles . . . 30
  - 4.3.5 Equipment . . . 33

4.3.6 Evaluation 34

5 Evaluation of Learning Outcomes 35
5.1 Methodology 35
5.1.1 Data Collection 35
5.1.2 Data Analysis 37
5.2 Results 37
5.2.1 Exercises 38
5.2.2 Learner Reports 39

6 Discussion 42
6.1 Interpretation of Results 42
6.2 Implications 43
6.3 Limitations 44
6.4 Recommendations 45

7 Conclusions 46
7.1 Content Criteria 46
7.2 Achieved Learning Outcomes 47
7.3 Integration of ERs into Education 48

A Exam Program 53

B Computer Science ERs 55

C Material of the ER 58

D Evaluation Form 71

Chapter 1

Introduction

1.1 Escape Rooms & Education

In recent years, the popularity of escape rooms as a leisure activity has rapidly increased. An escape room, abbreviated as ER, is a 'live-action team-based game where players discover clues, solve puzzles, and accomplish tasks in one or more rooms in order to accomplish a specific goal in a limited amount of time' (Nicholson, 2015). Each clue or solution to a puzzle leads to the next step. Teamwork and communication are essential to complete all steps and escape within the time limit.

Nowadays, ERs are used in more ways than just as a leisure activity. They have found their way into the classroom, in several subjects, including computer science. For example, ERs have been used to teach about cryptography concepts (Borrego et al., 2017; Deeb and Hickey, 2019; Ho, 2018; Seebauer et al., 2020; Streiff et al., 2019). In several university courses related to software engineering, ERs were used as a method to reinforce the course material (Gordillo et al., 2020; López-Pernas et al., 2019a,b). Other ERs let students practice with problem-solving in computer science (Hacke, 2019) or general ICT competences (Musil et al., 2019).

ERs present several advantages. For example, students can both learn curriculum content and skills, and develop soft skills, such as teamwork and communication skills. Teachers also use ERs as an active learning environment, or as a means to increase the motivation and engagement of students (Taraldsen et al., 2020; Veldkamp et al., 2020). Interestingly, none of these ER benefits is unique on its own. It is their combination that is unique and appealing to teachers (Veldkamp et al., 2020), which explains why ERs have started appearing in classrooms.

1.2 Research Questions

Most existing studies on educational ERs are case studies that create and evaluate an ER. Informally, we can say that they try to answer the question: 'Does this work?' (Taraldsen et al., 2020). Unfortunately, the number of such studies for secondary computer science education is limited. Only three examples were found (Hacke, 2019; Michaeli and Romeike, 2020; Streiff et al., 2019), each with their own limitations. In order to gain more knowledge on the use of ERs in this educational setting, more well-designed and evaluated ERs are needed.

There is another issue with the current state of the research area. Although case studies provide useful insights, other, more general, research is needed as well. In order to bring the research area into a new phase, research on how, why and when to use ERs as a didactic tool is necessary, according to Taraldsen et al. (2020). Such research leads to more general results, and helps to improve the usage of ERs in the classroom.

In order to address both aforementioned issues, this thesis answers the following research question and sub-questions:

How can we integrate educational escape rooms into secondary computer science education?

- *What are content criteria for using an educational escape room in secondary computer science education?*
- *What are the achieved learning outcomes of an educational escape room for secondary computer science education?*

The answer to the first sub-question will give information on how and when ERs can be used in secondary computer science education. This is an example of the previously mentioned general research requested by Taraldsen et al. (2020), on the use of ERs as a didactic tool. In this way, this thesis goes a step further than the case studies.

We also need more example ERs, due to the limited number of ERs in secondary computer science education. Therefore, by answering the second sub-question, we create and evaluate an example ER. This ER differs from the three existing examples for secondary computer science education, because it is used as a teaching method on a single topic, contrary to the existing ERs (the differences are explained in more detail in section 2.1.5). Besides creating a new ER, we also evaluate the learning outcomes, in order to get more general insights into what can be taught using ERs. Finally, the main question, on how to integrate educational escape rooms into secondary

computer science education, is answered by combining the answers to both sub-questions.

Since this thesis contains an analysis, design and evaluation of a learning activity, it can be classified as educational design research (van den Akker et al., 2013). The advantage of this research approach is that we both create an activity usable in educational practice, and advance our knowledge in order to inform future work, for instance on how to design and use these activities.

1.3 Structure of Thesis

In Chapter 2, we discuss existing work on educational ERs and consider secondary computer science education in the Netherlands. The research of this study consists of three phases, and is described in Chapters 3 to 5. Each phase has their own chapter, describing the research methods and results. In Chapter 3, we investigate the content of existing computer science ERs, in all educational settings, in order to extract content criteria. Then, we design an actual ER for secondary computer science education. The design process and results are described in Chapter 4. The evaluation on learning outcomes of this ER is outlined in Chapter 5. Finally, the results are discussed in Chapter 6, and the conclusions are described in Chapter 7.

Chapter 2

Related Work

2.1 Educational Escape Rooms

Although educational ERs are relatively new, some research has already been done on this topic. In this section, we explore several aspects of this research. We also encounter some existing examples of ERs in computer science.

2.1.1 Goals of Educational Escape Rooms

The use of an ER in education can serve several goals. According to a literature review on ERs in education by Veldkamp et al. (2020), educators used an ER '1) to explore an active learning environment which is said 2) to increase student's motivation and/or engagement, 3) to foster learning, while 4) practising or developing teamwork and communication skills' in 36 out of 39 studies. The review also described different types of learning goals, which differ from the above-mentioned reasons to implement an ER. For example, increasing motivation is not a learning goal in itself, but can be an intention to use an ER in the classroom. The following types of learning goals were identified (Veldkamp et al., 2020):

- Specific content knowledge and content related skills
  Most ERs (32) with content-related learning goals were used to foster, demonstrate or assess students' knowledge and skills. Fewer (10) were used to introduce, extend or integrate knowledge and skills.

- General skills
  Common examples are teamwork and communication skills, problem-solving, critical thinking and analytic thinking/reasoning.

- Affective goals
  This type of goal occurs most often in medical ERs. It relates to career situations, such as performing under pressure, which is highly relevant in medical education.

Another literature review on educational ERs, by Taraldsen et al. (2020), identified four fields of research attention:

- Scenario
  Students are exposed to real-life scenarios. Skills as working together, trusting each other and handling time constraints are important. This is relevant in a profession like health care.

- Curriculum
  The focus is on knowledge and skills. The ER provides an opportunity to apply the curriculum content.

- 21st-century skills
  Mentioned examples are critical thinking, creativity, group dynamics, initiative and problem-solving.

- Motivation
  An ER creates a fun experience, generating enthusiasm and curiosity.

The aspects identified in the two literature reviews overlap. Both reviews mention the integration of curriculum content (both knowledge and skills) in ERs. Moreover, general skills, not specific to a subject, are identified as a goal of ERs in both reviews, as well as increasing motivation. Finally, the *affective goals* of Veldkamp et al. (2020) can be combined with the *scenario* aspect of Taraldsen et al. (2020), since both are about students experiencing a real-life situation that is relevant for their profession. Therefore, the goals of using an educational ER can be summarized into four aspects:

- Knowledge and skills in the curriculum

- General skills

- Motivation

- Real-life scenario

2.1.2 Debriefing

In recreational ERs, there is usually a debriefing at the end (Nicholson, 2015). In the review by Veldkamp et al. (2020), 25 out of 39 educational ERs used some form of debriefing. Students indicate that debriefing is important in the learning process. In the recommendations for implementing an ER, the implementation of a debrief is mentioned as a crucial element. It is not only a source of feedback, but also supports students in actively linking and decontextualizing knowledge (Sanchez and Plumettaz-Sieber, 2018).

This raises the question: what happens during the debriefing? First, there is often a cooling down period after the intense activity. The participants can also exchange their experiences and ask questions. It is possible to discuss and explain the puzzles themselves, but the focus can also be on learning aspects: which knowledge and skills were needed to solve the puzzles? In this regard, relation to the learning goals is important. Moreover, the encountered concepts can be related to each other, or to other contexts. There might also be room for feedback on students' performances or reflection on the learning process. (Sanchez and Plumettaz-Sieber, 2018; Veldkamp et al., 2020)

There is no unique approach for debriefing. It can be implemented in several forms. A structured debriefing could have the form of a PowerPoint presentation. However, some teachers apply more unstructured methods, in which the students' questions and comments are leading, while the teacher acts as a guide rephrasing students' ideas or introducing scientific vocabulary. Teachers also mention the use of concept maps, infographics, or brainstorming sessions with post-its during the debriefing. (Sanchez and Plumettaz-Sieber, 2018)

2.1.3 Design of Educational Escape Rooms

Researchers have started to create frameworks for designing educational ERs (Taraldsen et al., 2020). An example is the EscapED framework (Clarke et al., 2017), shown in Figure 2.1. This model takes a step-by-step approach, resulting in clarity during the design process. However, such a linear approach has disadvantages as well, since some aspects, such as goals, puzzle path and teacher support, occur in more complex, non-linear patterns (Veldkamp et al., 2020).

Figure 2.1: EscapED Framework (Clarke et al., 2017)

A different approach is used in the SEGAM model (Guigon et al., 2018), depicted in Figure 2.2. An ER is split up into 'levels', which contain riddles and clues. Therefore, contrary to EscapeED, this model puts more emphasis on the structure of the ER. Guigon et al. also describe other aspects, such as the learning objectives and theme, but these are not part of the model itself. Debriefing is mentioned as an important step, whereas this is not part of EscapED.

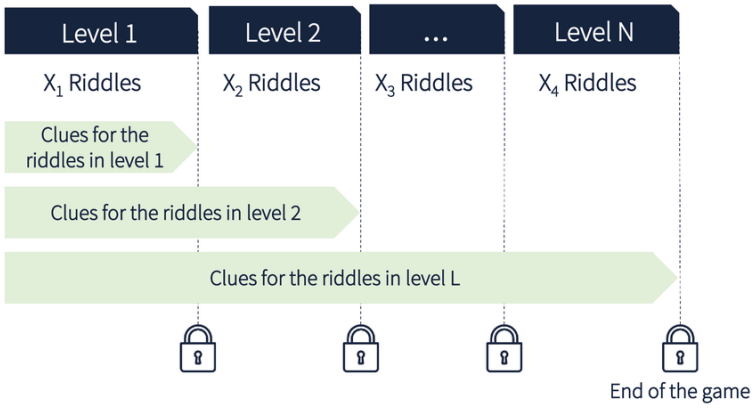


Figure 2.2: SEGAM Model (Guigon et al., 2018)

Another framework is based on the theory of intrinsic integration, meaning that educational learning and the game mechanics should be integrated (van der Linden et al., 2018). In order to achieve this, there must be alignment between the game goal, learning goal, pedagogical approach and game mechanics, as shown in Figure 2.3.

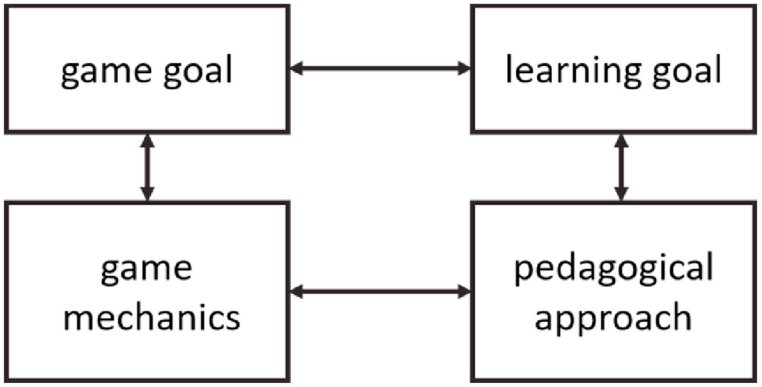


Figure 2.3: Intrinsic Integration Framework (van der Linden et al., 2018)

This framework was created for educational games in general, but can be applied to ERs. Educational ERs have the game goal of escaping, but also have learning goals. Students should only escape if the learning goals have been achieved. Game mechanics, such as the team size and puzzle structure, play a role as well. For example, it was shown that the team size (game mechanic) impacted the achievement of collaborative learning (pedagogical approach) (Veldkamp et al., 2020).

2.1.4 Escapp Web Platform

Besides frameworks for designing educational ERs, a tool for conducting them has been developed as well. Escapp is a web platform, which assists teachers in delivering content, checking solutions, giving hints, monitoring progress and evaluating the activity (López-Pernas et al., 2021). The platform can both be used for remote ERs and face-to-face ERs, with physical and digital puzzles being combined in the latter. A limitation of Escapp is the requirement that the puzzle structure must be linear, meaning that one puzzle must be solved before the team can continue to the next. Other puzzle structures suggested by Nicholson (2015), such as open or path-based structures, in which multiple puzzles can be solved at the same time, are therefore not possible using this tool. The Escapp tool has been evaluated during three case studies, with 413 participants in total. Students' responses indicate that the platform provides an enjoyable and useful experience (López-Pernas et al., 2021).

2.1.5 Examples in Computer Science

The existing research on the use of ERs in computer science education consists of case studies, in which an ER is used to teach about one or more computer science topics. There are few examples for secondary education, since most ERs were made for higher education. For example, an ER was used in a university course on software engineering, in order to reinforce the most important concepts of the course (Gordillo et al., 2020). The learning objectives of the ER focused on understanding, interpreting and creating several diagrams relevant to software engineering (UML use case, sequence, activity, class and state diagrams). This ER used the previously described Escapp platform (López-Pernas et al., 2021).

Multiple ERs relate to security concepts (Borrego et al., 2017; Seebauer et al., 2020; Streiff et al., 2019). For example, an ER was used to raise awareness about potential vulnerabilities (Beguin et al., 2019). As a result, the participants, not necessarily with a computer science background, learnt ways to protect themselves from attacks. The teachers of a university course about cryptography used an ER to give students practice in applying several ciphers that were taught during the course (Ho, 2018). Applying a cipher was also part of the ER created by Deeb and Hickey (2019).

ERs were also used to teach about front-end development (López-Pernas et al., 2019b), back-end web development (López-Pernas et al., 2019a), debugging (Michaeli and Romeike, 2020), networks (Borrego et al., 2017), logic (Otemaier et al., 2020), or a combination of diverse computer science topics (Combéfis and De Moffarts, 2019; Dimova et al., 2020; Hacke, 2019; Kahila et al., 2020; Musil et al., 2019). The existing computer science ERs are discussed more elaborately in Chapter 3.

The most common educational setting for computer science ERs is higher education (Borrego et al., 2017; Deeb and Hickey, 2019; Gordillo et al., 2020; Hacke, 2019; Ho, 2018; López-Pernas et al., 2019a,b; Otemaier et al., 2020; Seebauer et al., 2020), which is consistent with educational ERs in general (Taraldsen et al., 2020; Veldkamp et al., 2020). A reason for this might be that teachers expect ERs to be more suitable for older students, since participants of recreational ERs are more often adults (over 21) than not (Nicholson, 2015). However, since ERs train general skills that are important in secondary education, as will be discussed in section 2.2, there is a lot of potential in the use of ERs in secondary education. Unfortunately, the number of computer science ERs designed for secondary education level is currently severely limited. Only three examples were found (Hacke, 2019; Michaeli and Romeike, 2020; Streiff et al., 2019). In order to investigate the use of this new teaching method in secondary computer science education,

more examples are necessary, to which this thesis contributes. The ER created in this thesis will be used as a teaching method on a single topic. Therefore, it differs from the existing three examples in the following ways:

- Hacke (2019) focuses on general problem-solving. The ER does not teach about a specific topic.
- Michaeli and Romeike (2020) used their ER as a research method rather than a teaching method.
- Streiff et al. (2019) describe some stand-alone activities that could be used in an ER, but they are not actually incorporated into one ER.

Another new perspective of this thesis deals with the choice of topic. In the case studies, it is not explained why the chosen topics or learning objectives are suitable to be taught using an ER, with one exception (Beguin et al., 2019). Therefore, this thesis also provides a novel contribution by determining content criteria for the use of ERs in computer science education.

2.1.6 Evaluation Methods

An educational ER can be evaluated in several ways. The literature reviews by Taraldsen et al. (2020) and Veldkamp et al. (2020) indeed show that a variety of methods is used, of which the most occurring are:

- Informal observations: The researchers, or other observers, observe the participants during gameplay. This can be used to get some information on student motivation or engagement, or the quality of teamwork and communication skills, but the research value is limited.
- Feedback: Participants give feedback, in the form of a feedback session, group discussion (possibly during the debriefing), or a post-activity survey. Students can indicate, for example, their general opinion on the ER, their thoughts on motivation and teamwork, and whether the ER helped them to achieve the learning goals. Here, focus is on student self-perception.
- Pre- and post-test: Participants make a test before and after the ER. Improvement in content knowledge can be measured in this way. It can also be used to measure an increase in motivation. This method gives more valuable information than the opinions of participants, since there is a discrepancy between perceived and actual learning in ERs (Veldkamp et al., 2020).

The previously described Escapp platform enables an additional evaluation

method (López-Pernas et al., 2021). The platform automatically records data that gives insight into student performance. For example, it keeps track of how many teams solved the puzzles, how much time the puzzles took and how many hints were needed. This information has been used as part of evaluations, in addition to a pre- and post-knowledge test and a survey on students' opinions (Gordillo et al., 2020; López-Pernas et al., 2019a). Escapp only gives information about the performance of students on the puzzles; it does not provide insight into obtained knowledge outside the context of the ER, motivation, teamwork, or other aspects. Therefore, this evaluation method is indeed suitable to be used in combination with other methods, but not as the only method.

2.2 Computer Science Education

Research on the didactics of computer science is limited, since computer science is a relatively new scientific field (Barendsen and Tolboom, 2016). This thesis contributes to research on computer science education by investigating the use of a novel didactic approach, namely educational ERs. In order to get insight into the use of such a new teaching method, a curriculum is a useful resource, because it summarizes the most important content knowledge and skills of the subject. However, the curriculum contents differ per country. In this thesis, we will focus on the Dutch computer science curriculum. The research is conducted in the Netherlands, and the developed ER will be implemented in the Netherlands. Therefore, connection to the Dutch exam program is desirable. Exploring the connection to other curricula, that might contain different skills or content, is outside the scope of this thesis.

The authors of the Dutch computer science curriculum identify three crucial skills for computer science (Barendsen and Tolboom, 2016):

- Design and development (which the authors relate to problem-solving)
- See things in a computer science perspective (which the authors relate to analytical skills)
- Collaboration and interdisciplinarity

The fact that these skills are part of the curriculum indicate that ERs can be useful in computer science education, because there is a partial overlap with the goals of educational ERs mentioned in section 2.1.1. For example, ERs are used to improve problem-solving and collaboration skills (Taraldsen et al., 2020; Veldkamp et al., 2020). Besides the development of general skills, ERs can improve students' content knowledge as well. In the Dutch com-

puter science curriculum, the content is divided into 17 domains (Barendsen and Tolboom, 2016), which can be found in Appendix A. In the development of content criteria for the use of ERs in computer science education, the division of content into these domains will be used.

Chapter 3

Content Analysis

Research on 'the "if, how, why, and when" regarding the use of escape rooms as a didactic tool' is needed to bring the research area on educational ERs into a new phase (Taraldsen et al., 2020). In this chapter, the focus will be on the content for which an ER is a suitable teaching approach. According to Wiemker et al. (2015), an ER can be themed with almost any topic. However, since all students need to experience a learning process during the ER, content-related constraints do exist (Taraldsen et al., 2020). The appropriateness of an ER might depend on the topic. Therefore, this chapter investigates content criteria for the use of an ER in secondary computer science education.

3.1 Methodology

For the content analysis, examples of computer science ERs were used. A list of these ERs was created in a literature review by Lathwesen and Belova (2021). Since their list is relatively recent, we assume that it is reasonably complete. On top of the list of Lathwesen and Belova, some new examples were added. Most descriptions of learning objectives and/or subtopics were taken or adapted from the list of Lathwesen and Belova (2021).

The existing computer science ERs differ with regard to several aspects. For example, the goals of these ERs are different: some were used to increase motivation, whereas others mainly focused on content knowledge or skills. ERs were used to introduce new concepts, but also to apply or review content. There are also differences in educational setting and game aspects. However, in this chapter, we do not consider these differences; we purely focus on the content of the ERs.

After having assembled a list of computer science ERs, we grouped them by topic. For each group, we described the content that was included in the ERs. In this way, we identified characteristics of the content. By finding intersecting characteristics, we developed content criteria. Furthermore, we mapped the content of each group to the Dutch exam program, to get insight into the domains for which ERs are and are not used. Identifying patterns allowed us to extract more content criteria.

3.2 Results

3.2.1 Computer Science ERs

The full list of computer science ERs can be found in Appendix B. There is some inconsistency in the descriptions of the ERs, since the several authors explained the content of their ERs in different ways: some articles included concrete learning objectives, whereas other articles only mentioned subtopics, and a few articles only gave some examples of puzzles. One ER that was labelled as a computer science ER by Lathwesen and Belova (2021), was left out, since this ER focused on research methods rather than content specific to computer science. Also note that Borrego et al. (2017) are present in the table twice, because the created ER included content on two different topics.

3.2.2 Groups

In Appendix B, the ERs are grouped by general topic. The groups are indicated by colours. The largest group, consisting of six ERs, contains the ERs about security-related concepts. There is a group of four ERs that deal with software development. Moreover, there is one ER about networks, and one about maths for software engineering. Finally, five ERs do not focus on one topic, but include various computer science topics. We will go over each group.

Security

The ERs in this group relate to Domain N (Security) of the exam program. However, the theme of security is present in the exam program twice more, namely in subdomains E2 and F4. In E2, the focus lies on the technical side, whereas F4 pays attention to the social and human aspects of security. This distinction between the technical and human side is visible in the ERs as well: there are four ERs about the technical side (Borrego et al., 2017; Ho, 2018; Seebauer et al., 2020; Streiff et al., 2019), one about the human side (Beguin et al., 2019), and one that includes aspects of both (Deeb and Hickey, 2019).

An interesting observation is that in five of the six ERs, manipulation of data plays a role. The manipulation happens in different ways: using ciphers, compression algorithms, hash algorithms or other cryptographic methods. An ER seems to be a suited environment for students to apply these concepts. ERs consists of puzzles; using a cipher, algorithm or other method to modify data is a puzzle in itself. The fact that there is often only one correct answer when applying such a method is also helpful, because answers must be easy to verify in ERs. For example, decrypting a message can directly lead to a unique code. For these reasons, using an ER is suitable for content related to the manipulation of data.

Software Development

This group relates to Domain D (Programming) of the exam program. However, the actual programming that students needed to do in the ERs is limited (Gordillo et al., 2020; López-Pernas et al., 2019a,b; Michaeli and Romeike, 2020). Students did not have to design or program large parts of an application themselves. Such a task is not suitable for an ER, because of the time constraints, and because it is hard to verify within an ER whether such a task has been completed satisfactorily. Verification is necessary, because the students' solutions must lead to a code or another puzzle in some way.

Therefore, the ERs included different kinds of tasks, that are easier to verify. These were smaller programming tasks, such as invoking functions, debugging something, modifying a function, or writing a couple lines of code (López-Pernas et al., 2019a,b; Michaeli and Romeike, 2020). Besides making students practice with small programming tasks, the ERs were used to teach about concepts related to programming, such as software modelling diagrams (Gordillo et al., 2020) or the basics of a programming language (López-Pernas et al., 2019a,b). From the ERs in this group, we can conclude that ERs are not suitable for design tasks or larger programming tasks, but they can be used for practising smaller pre-defined tasks or improving the understanding of concepts related to programming.

Networks

The single ER in this group relates to Domain L (Networks) of the exam program (Borrego et al., 2017). In the ER, students need to apply the skill of sniffing a network. Students also need to understand the components of the packets. Moreover, students need to understand the TCP protocol in order to be able to rearrange the instructions. This shows that ERs can be used to train skills or improve understanding of concepts related to networks.

Maths

The single ER in this group focused on logic (Otemaier et al., 2020), which is related to Subdomain G3 (Logic). The students had to solve puzzles that required them to understand propositions and predicates. This indicates that the use of ERs is a possible teaching approach for logic.

Various Topics

Finally, some ERs do not focus on a specific topic (Combéfis and De Moffarts, 2019; Dimova et al., 2020; Hacke, 2019; Kahila et al., 2020; Musil et al., 2019). The reason for this is that their goal is either to teach about several basic computer science concepts (Combéfis and De Moffarts, 2019; Dimova et al., 2020), or to train more general skills, such as problem-solving and computational thinking (Hacke, 2019; Kahila et al., 2020; Musil et al., 2019). As a result, these ERs contain several subtopics, related to different domains of the exam program. Some of these subtopics, such as encryption/decryption, logic and programming, belong to the previously described groups. However, some other subtopics are included as well:

- Algorithmic concepts. This includes the basic notion of an algorithm and the working of stepwise, deterministic program execution. Examples include the bubble sort algorithm and the shortest-route problem. These topics relate to Subdomain B1 (Algorithms).

- Finite automata, which relates to Subdomain B3 (Automata).

- Binary logic and bit flips. Since this is about binary representations, this relates to Subdomain C4 (Standard representations).

- Digital skills. These are general skills, such as using software programs and connecting hardware. Although such basic skills are mentioned as skills in the exam program, they are not part of the content students need to learn. Therefore, such skills are not relevant for the content analysis, and will not be considered in the rest of this chapter.

Similar to the ERs about software development, the students do not have to create something themselves in these ERs. For example, students do not need to design an algorithm or create an automaton. Instead, these concepts must be understood and used to solve the puzzles, by applying an existing algorithm or tracing an automaton.

3.2.3 Technical vs. Social Topics

In the previous section, ERs were linked to domains of the exam program. Several domains were not matched with an existing ER. However, this does not imply that an ER is not a suitable teaching method for these domains; the research field of educational ERs is relatively new and the number of computer science ERs is severely limited. Therefore, a single domain not being matched to an ER does not say much. Nevertheless, some conclusions can be drawn by looking at the complete list of domains.

Previously, it was mentioned that the theme of security could be split up into a technical and a social side. This distinction can also be made for the complete exam program. Some domains are about technical topics, such as domains B (Foundations), D (Programming) and E (Architecture), whereas other domains focus on the social side, such as domains F (Interaction), O (Usability), P (User Experience) and Q (Societal and individual influence of computer science). Appendix B contains sixteen distinct ERs. Fourteen of them focus on the technical side, whereas only two, both about security, focus on the human side. These two ERs show that it is possible to teach about some aspects of the social side using an ER (Beguin et al., 2019; Deeb and Hickey, 2019). However, considering the fact that the large majority of ERs is about the technical side, this side is likely to contain more suitable content for ERs.

3.2.4 Content Criteria

Based on the content of existing computer science ERs, we can develop criteria for content to be used in an educational ER. However, there is no strict separation between appropriate and non-appropriate content. ERs can be used in novel and unexpected ways. Therefore, the developed criteria are not strict rules; they should merely be seen as an indication of suitable content.

One characteristic of existing ERs is about the kind of tasks that are present. Tasks that involve the reproduction of facts do not occur often, just like design or creation tasks (e.g. designing an algorithm, developing a larger program). Instead, most puzzles are either about *understanding concepts* (e.g. interpreting diagrams, understanding the basics of a programming language) or *applying concepts* (e.g. manipulating data using cryptographic methods, using an algorithm).

Another general pattern is that the ERs contain puzzles of which the *solutions are easily verifiable*. This is important in an ER, because a correct solution must automatically lead to a new hint or puzzle. Verification is more difficult for tasks in which students have much freedom, such as de-

sign, development or creation tasks. This explains why such tasks do not appear in ERs often, as mentioned before. In the existing ERs, there are many examples of puzzles of which the solution can automatically be verified. For example, applying a cipher, using an algorithm, or solving a logic puzzle results in a unique answer (Ho, 2018; Combéfis and De Moffarts, 2019; Otemaier et al., 2020). It is also possible to include small, pre-defined programming tasks that can easily be verified, as some ERs did (López-Pernas et al., 2019a,b). For example, if the correct functions are called or a bug is fixed, a new hint or puzzle is automatically loaded.

As discussed in section 3.2.3, ERs are not often used for the domains focusing on the social side of computer science. ERs are mostly about understanding and applying concepts. The technical side of computer science contains a lot of suitable concepts for this, whereas the social domains put emphasis on other aspects, such as analysing the influence of computer science and the use of computers by humans, which are less suited to be part of an ER. Therefore, ERs are more suitable for the *technical side of computer science*.

In summary, based on the existing computer science ERs, we identified the following content criteria:

- The content must contain concepts that students can apply, or concepts of which students' understanding can be tested using a puzzle. Other content, for example about factual information or creation tasks, is less suitable.

- It must be possible to easily verify correct application or understanding of the content. This means that rather than allowing for much freedom, the puzzles should be small, pre-defined tasks.

- Technical concepts are more appropriate for an ER than domains focusing on the social side of computer science.

Chapter 4

Design of an ER

After having established the content criteria, we designed an educational ER for secondary computer science education. In this chapter, the design process and resulting ER are described.

4.1 Choice of Topic

First, a teacher was contacted who was willing to test the ER in her classroom. In consultation with the teacher, we decided to match the topic of the ER with the topic that was planned in the period in which the ER would be tested. In this way, the ER actually connects to the students' learning process. At the moment the ER would be tested, the students would be learning the basics of programming in Python. Therefore, this was chosen as the topic of the ER. A more specific description of the topic follows in section 4.3.

The chosen topic satisfies the content criteria that were developed in Chapter 3. Although the ER cannot contain large programming tasks, it can be used to improve or test students' understanding of Python concepts. Moreover, it is possible to create small tasks about such concepts that are easily verifiable. Finally, the topic is technical rather than social.

4.2 Methodology

As discussed in section 2.1.3, several frameworks for the design of an educational ER exist. We used the EscapED framework (Clarke et al., 2017), because it describes, contrary to the SEGAM model (Guigon et al., 2018) and the model based on the intrinsic integration theory (van der Linden

et al., 2018), all steps that need to be taken in the design process. Its linear nature has disadvantages, since the process of designing an ER might be complex and non-linear at times. However, it is possible to slightly deviate from the provided steps, as was shown in an account describing a use case of the framework (Clarke et al., 2017).

The other described frameworks, the SEGAM model (Guigon et al., 2018), and the model based on the intrinsic integration theory (van der Linden et al., 2018), were not used directly. However, we adopted some ideas of these frameworks. For example, debriefing is an essential part of an ER, as described in section 2.1.2, but it is not part of the EscapED framework, whereas it is mentioned in the explanation of the SEGAM model. Therefore, debriefing was explicitly added to the EscapED framework. The model based on the intrinsic integration theory was taken into consideration in the choice of certain game mechanics, such as the group size.

The EscapED framework is illustrated in Figure 2.1. It contains the following steps (Clarke et al., 2017):

1. **Participants**
   The target audience needs to be analysed. For example, the user type, which includes demographic information and educational needs, must be determined. Since the required level of difficulty depends on the audience, it is part of this step as well, just like other game mechanics, such as the time and scale (number of participants). Finally, a choice of 'mode' needs to be made. This can, for example, be the cooperation-based mode or the competitive-based mode.

2. **Objectives**
   By designing the learning objectives early, they can be integrated into the theme and puzzles. In this way, the ER is designed purposefully. It also needs to be decided in this step whether one discipline or multiple disciplines will be present. Moreover, choices about the role of soft skills and problem-solving must be made.

3. **Theme**
   A theme needs to be chosen in order to create a compelling game experience. It can be 'Escape Mode' (escaping a locked room) or 'Mystery Mode' (solving a mystery). A narrative must be designed to keep participants interested. It also needs to be decided whether the ER is a stand-alone experience or part of a larger experience, consisting of multiple games or ERs.

4. **Puzzles**
   This step involves the creation of the actual puzzles. They must match the learning objectives, so it is important to regularly check whether this is still the case. Moreover, creating clear instructions and rules is part of this step. Finally, hints and a method to deliver them must be developed.

   As mentioned before, we add a debriefing session to the EscapeED framework. We develop the debriefing as part of this step.

5. **Equipment**
   This step involves the choice of equipment. This includes the location in which the ER takes place. Moreover, both physical and technical props can be used. Real-life actors can also be used to make the experience more believable, and possibly to indicate the time or to give hints.

6. **Evaluation**
   The last step is the evaluation of the ER. It needs to be tested before the actual target audience plays it. Afterwards, a reflection with the participants on their views and experiences can be useful. This step also includes a formal evaluation of the learning objectives. Based on the evaluation, adjustments can be made to the ER. Finally, the 're-set' involves checking whether all puzzles are in the correct state before the ER is played again.

4.3 Results

The results for each step of the EscapED framework are described in this section.

4.3.1 Participants

The target audience of the ER consists of students in the fourth grade of *havo* or *vwo*[1]. In collaboration with a teacher, we decided to create an ER to become part of a module about programming in Python. In order to match the ER with the students needs, the students' foreknowledge was considered. Although the students have used a block-based programming environment, they have not had previous lessons on actual imperative programming.

In the weeks before students will play the ER, the first two chapters of the

[1] *Havo* and *vwo* indicate levels of the Dutch education system. They can be translated as 'senior general secondary education' and 'pre-university education', respectively. In the fourth grade of *havo* or *vwo*, students are generally 15 or 16 years old.

module about Python are covered in the classroom. The students learn to write and run a simple Python program and use a library, called Turtle, to draw figures in the first chapter. Afterwards, in the second chapter, they learn about data types, printing information, user input and performing calculations. The students also practice with the material of the first two chapters. It was decided to start the lesson in which the students play the ER with a short explanation about the third chapter, which covers variables. This means that, before playing the ER, the students hear an explanation about this topic, but they have not yet practised with it.

Besides an analysis of the participants, this step also involved the choice of certain game mechanics. The playtime of existing educational ERs varies from 20 to 120 minutes, with most ERs taking 60 minutes (Veldkamp et al., 2020). However, specific pedagogical reasons for a playtime are barely ever mentioned; some studies simply refer to classroom time slots (Veldkamp et al., 2020). Since the duration of our ER is bounded by the duration of the lesson, the choice of play time is primarily based on this. The lesson takes 60 minutes, but time is needed for the explanation of the third chapter of the module, the explanation of the ER, the debriefing and the evaluation. A play time of 30 minutes was chosen, because this fits in the planning and gives the students enough time to actually practice with the material in the ER.

Another game mechanic is the team size. In most existing educational ERs, the team size ranged from three to six students (Veldkamp et al., 2020). Larger groups have disadvantages, such as inactivity of some students, and a loss of communication and organization (Veldkamp et al., 2020). We designed the ER for groups of three students.

The cooperation-based mode was chosen for the ER. As mentioned in section 2.2, collaboration is an important aspect of computer science education. Therefore, although improving collaboration is not an explicit goal of this ER, a cooperation-based mode is more suitable for computer science education than a competitive-based mode. Finally, the level of difficulty was considered. There has to be a match between the level of difficulty of the ER and that of the explanation and exercises in the existing Python module. Therefore, we studied the existing content of the module before designing the ER.

4.3.2 Objectives

The content of the ER covers the first three chapters of the existing Python module. The first chapter consists of running simple Python programs and drawing figures using Turtle. This chapter is covered in class before the

ER. Therefore, the ER is used as reinforcement. The students will practice with running Python programs in the remainder of the module anyway. Therefore, only drawing with Turtle is part of the ER. This results in the following learning objective:

Objective 1: The student can trace the result of Turtle code.

The second chapter is covered in class before the ER as well. Therefore, the ER is also used as reinforcement for this chapter. The content includes data types, printing information, user input and performing calculations. Therefore, the learning objective belonging to this chapter is:

Objective 2: The student can use the following concepts in a Python program: standard data types, print statements, user input and calculations

The content of the third chapter is new for the students. Before playing the ER, the students get a short explanation about variables, the topic of the chapter. The ER is their first opportunity to practice with this concept. There are two learning objectives:

Objective 3.1: The student can trace the values of variables in a Python program.

Objective 3.2: The student can use variables in a Python program.

The main focus of the ER lies on the third chapter, because it contains new content for the students. Therefore, this chapter has two learning objectives, whereas the first two chapters have only one each. Also note that, as explained in Chapter 3, larger programming tasks are not suitable for an ER. Therefore, the formulated learning objectives focus on understanding or using concepts, rather than actually creating a larger program.

Learning objectives 1 and 3.1 involve code tracing. According to Lister (2020), it is crucial to teach this skill to novice programmers. He argues that students need to acquire two skills before they will be able to write good code: tracing code and explaining code. Since explaining code is a more open task, it is difficult to incorporate it in an ER. Tasks about tracing code, on the other hand, are more suitable for an ER, since they have only one correct answer. Therefore, we decided to give code tracing a substantial place in the ER.

The other learning objectives, 2 and 3.2, are about the use of programming concepts. A possible method to achieve these objectives is to let the students

write small programs that must use these concepts. Other ERs with such small programming tasks exist (López-Pernas et al., 2019a,b). However, setting up an environment in which students can program, and which lets the students only progress to the next stage if they carried out the programming task correctly, is challenging and time-consuming. Therefore, we decided to not include actual code writing in the ER. This means that the puzzles must teach students about the use of programming concepts, but the students should not actually have to program themselves. Achieving this requires creativity in the puzzle design, which is described in section 4.3.4.

Since we focus on computer science ERs, the ER is single disciplinary. A final aspect to consider in this step is the role of soft skills, such as collaboration, and problem-solving. Of course, such skills are needed in the ER, so the ER gives students practice in these skills. As described in section 2.2, they are important in computer science education. However, it is advised to create a debriefing or ER solely on soft skills, if improving them is a goal of the ER, because 'reflection on these goals is usually lost in a reflection on other educational goals' (Veldkamp et al., 2020). We mainly focus on content in this thesis. Therefore, the ER can help in improving soft skills and general problem-solving skills, but this is not explicitly formulated as a learning objective.

4.3.3 Theme

In the ER, the students need to recover the hacked computer system of a hospital. The ER starts with the following introduction:

> The president of a hospital just called you in a panic. The computer system of his hospital was attacked by hackers. They sabotaged several components of the system, gravely endangering all patients! Patients are getting the wrong amounts of medication, measurements give incorrect results, and treatments fail. The president counts on you to recover the system. To succeed, you need to re-program the sabotaged components. Make sure you succeed in half an hour, or the patients' lives are in danger.
>
> The hackers also blocked access to the system. This means that your first task is to get into the system. Good luck!

Nicholson (2015) described themes for ERs. Our ER relates to the themes 'Modern Era' and 'Technological'. The choice for these modern themes is based on the fact that computer science and programming are relatively new. Categories of narratives were identified by Nicholson (2015) as well. The chosen narrative can be placed in the category of 'Help Create Something'.

The students are not actually locked in a room, since this is impractical in a classroom with multiple groups playing. Therefore, the 'Escape Mode', as mentioned in this step of the EscapED framework, is not used in the ER. Since the students are not escaping anything, the 'Mystery Mode' fits better. However, this is also not a perfect description for this ER, since the students do not have to solve a mystery. Instead, they must carry out a task within a time limit.

Our ER is not part of a series of ERs or other games. However, it is used as part of a larger module. It is embedded into the lesson plan. This means that, from an educational perspective, it is an integrated activity rather than a stand-alone activity.

4.3.4 Puzzles

Each part of the ER is matched with a learning objective. The number of each part corresponds to the number of the relevant learning objective. The complete puzzles can be found in Appendix C, but we will shortly explain them here as well:

- **Part 1**
  The students get three pieces of Turtle code. The result of each piece forms a number. The students need to trace the code to find the three numbers (see Figure 4.1). These form the code that gives the students access to the hospital system.

- **Part 2**
  The students now have access to the system. They have to repair several parts. Two descriptions of small programs are given, together with some separate lines of Python code (correct and incorrect lines). For both programs, the students need to find the correct lines of code and put them in the correct order. Some lines are already filled in. The programs use the concepts mentioned in the learning objective.

- **Part 3.1**
  After the students have repaired some parts of the hospital system, it turns out that some more parts are broken, which have an extra protection layer. The students are given a piece of code. They need to trace the values of the variables. If they do this correctly, they get the code that gives access to the rest of the system.

```
turtle.pendown()
turtle.left(180)
turtle.forward(100)
turtle.left(90)
turtle.forward(200)
turtle.left(90)
turtle.forward(100)
turtle.left(90)
turtle.forward(100)
turtle.left(90)
turtle.forward(100)
turtle.done()
```

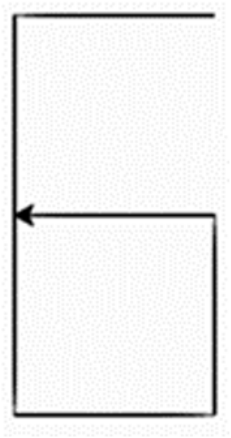


Figure 4.1: Part of a puzzle about Turtle. Tracing the Turtle code on the left results in a drawing of the number 6.

- **Part 3.2**
  The students now have access to the last parts of the system that need to be repaired. They get two descriptions of small programs with the corresponding Python code. However, there are some blanks in the places of variable names (see Figure 4.2). The students need to fill in the correct variable names.

```
WEIGHT_PILL = 500

print("How many milligrams should you take?")

input = input()

weight = int(.....)

..... = round( ..... / ....., 1)

print( "For", weight, "milligrams, you should take ", ....., "pills.")
```

Figure 4.2: Part of a puzzle about variables. Given a list of variable names, the blanks need to be filled in.

A match between the puzzles and the learning objectives is important. For Parts 1 and 3.1, this match is clear: the students trace the result of Turtle code and the values of variables, which is exactly what is mentioned in the corresponding learning objectives. Learning objectives 2 and 3.2 were more challenging, since the students do not actually write programs themselves. However, the puzzles still match the learning objectives:

- In Part 2, the students need to find the correct lines of code. This means they need to distinguish between correct lines and lines including common mistakes. This requires students to understand how to use the concepts mentioned in the learning objective.

- In Part 3.2, the students need to fill in the variables in a program. This requires them to understand, for example, what the role of each variable in the program is, which value is stored in each variable, and how calculations are done with variables. Therefore, the students need to understand how to use variables in a program in order to complete this task.

Part of this step is to create instructions and rules. The ER uses the Escapp web platform. Therefore, before playing the ER, it must be explained how this platform works. For example, it must be indicated how to submit a code and how to request a hint. It is not allowed to use the Internet or other sources during the ER. Moreover, the students get some envelopes containing printed materials; they can only be opened when indicated in the Escapp web platform. A final rule is that it is not allowed to copy-paste, compile and run Python code using some program, since this would give away answers.

Moreover, arranging hints is part of this step. We created hints for each puzzle, which are available in Appendix C (see Figure 4.3 for an example). They are delivered through the Escapp web platform. The students can request a hint every two minutes. They can indicate for which puzzle they want a hint. If students need additional support, the teacher can provide this.

1. Using a plus, you can only concatenate strings. Using a comma, you can print things of different types.
2. Length in meters is a decimal number.
3. Don't forget the parentheses when computing the BMI.
4. Lines 1, 5 and 6 are wrong.

Figure 4.3: Example of hints. These hints belong to a puzzle in Part 2.

Finally, we decided to add the preparation of a debriefing to this step. Since explaining the material on variables and playing the ER already take up most of the lesson, we need to keep the debriefing short. It is unstructured, so that we can use the questions and comments of the students as a starting point, in order to optimally connect to their experiences and learning process. The debriefing starts with a short cooling down period. Then, students get the opportunity to share their opinion on the ER, which gives us useful feedback. The learning process also gets attention, by having a class discussion around these types of questions:

- Which parts of the ER were most useful for your learning, and which were less useful?
- Did your understanding of certain concepts improve by playing the ER?
- Did you learn anything new?
- Did you encounter anything that you do not understand yet, or about which you would like some explanation?

Throughout the debriefing, the students can ask questions about the ER, the puzzles, the content, or anything else.

4.3.5 Equipment

The ER can be played in a regular classroom. It uses the Escapp web platform, as described in section 2.1.4. Because of this, computers or laptops must be present in the classroom, at least one per team. We deliberately minimized the number of physical items, because teachers should be able to use the ER without much effort or costs. Therefore, almost all materials, such as the narrative, puzzles and hints, are available digitally, in Escapp. Students do get some items on paper, such as the lines of code for Part 2 and the trace table for Part 3.1. They also get scrap paper. No other physical items are required. In this way, teachers using the ER have minimal preparation; some materials must be printed, but most of it is digital.

No real-life actors are used in the ER. Again, this choice was made to reduce the workload and preparation time for teachers. The timer and hints are handled by the Escapp web platform. The teacher is present during the ER to provide support, if needed.

4.3.6 Evaluation

After finishing the first version of the ER, two people gave feedback: the computer science teacher in whose classroom the ER will be evaluated, and a former computer science teacher, who is now working in the field of escape games. They commented on the puzzles, explanations and the narrative. Based on this, we made some small changes to the ER. The largest update was a reduction of the number of puzzles, because the ER was expected to be too lengthy. After incorporating the feedback, two test rounds of the updated version were played with a mixed group of participants, some with a computer science background and some without prior computer science knowledge. The general impression was positive, but, based on our observations, and the feedback of the participants, we made some more minor changes.

The EscapED framework includes a 're-set' in this step. This involves re-setting the ER for the next play. Because it is mostly digital, the re-set is not much work. The only preparation is to print and cut the materials that the students get on paper.

Finally, according to the EscapED framework, the learning objectives must be formally evaluated in this step. This is described, together with a more elaborate evaluation, in the next chapter.

Chapter 5

Evaluation of Learning Outcomes

The EscapED framework contains an evaluation as its last step. This includes a formal evaluation of the learning objectives. However, there may be learning outcomes that are not part of the learning objectives. Students may gain unexpected or extra knowledge. Moreover, the learning objectives focus on the curriculum content, whereas students may also learn more general skills, such as collaboration or communication skills, or problem-solving strategies. There is a wide range of possible learning outcomes. Evaluating them gives insight into the possibilities that the use of ERs provides in secondary computer science education.

5.1 Methodology

5.1.1 Data Collection

After the students played the ER, they filled in a learner report, as invented by De Groot (1980). Such a learner report distinguishes learning experiences along two dimensions. First, there is a distinction between learning about rules and learning about exceptions to rules. The second dimension distinguishes between learning about the world and learning about oneself. This results in four parts, as displayed in Figure 5.1. For each of the four parts, the students are asked to create sentences about what they learnt.

Learner reports give insight into learning that is difficult to measure objectively. This makes it a suitable method for evaluating ERs. In ERs, there is a variety of learning possibilities, including learning about content, learning soft skills, and learning about problem-solving strategies. Because

	Learning about rules	Learning about exceptions
Learning about the world	Rules about or in the world	Surprises of the world
Learning about oneself	Rules regarding oneself	Surprises regarding oneself

Figure 5.1: Grouping of learning experiences, translated from (Gerritsen van der Hoop, 1981)

of the wide range of possible learning experiences, measuring all of them objectively is difficult. Therefore, learner reports are useful, because they provide a way to get insight into the learning experiences of students. This is necessary to draw conclusions about the learning outcomes of the ER.

For that reason, we used learner reports to evaluate the ER. Evaluating an activity is indeed one of the goals of using learner reports (Van Kesteren, 1993). However, when using learner reports as an evaluation tool, it is recommended to also use more objective instruments to get information about measurable knowledge (Van Kesteren, 1993). An example of measurable knowledge is whether the learning objectives have been achieved. Therefore, the students also made some exercises after playing the ER. The exercises cover learning objectives 3.1 and 3.2. In the ER, the students practised with the material on these learning objectives for the first time. Therefore, their performance on the exercises indicates whether learning objectives can be achieved using an ER. In the first exercise, on learning objective 3.1, the students had to trace variables. In the exercise on learning objective 3.2, six programs using variables were given. In five of these, a mistake in the use of one or more variables was made. The students had to indicate the correct program.

The exercises do not cover learning objectives 1 and 2. These learning objectives are reinforcement; the students have already learnt about the material and practised with it. Therefore, including exercises about these learning objectives would not allow us to draw conclusions about the learning outcomes of the ER, since they might be attributable to the earlier lessons. In order to objectively measure the effectiveness of reinforcement, a pre- and post-test evaluation would be necessary. However, due to time constraints, we leave such an evaluation to future work, and only include exercises about learning objectives 3.1 and 3.2. Note that we can still get information on learning outcomes related to the first two learning objectives via the learner reports.

In conclusion, there are two sources of data: the learner reports, and the exercises about learning objectives 3.1 and 3.2. The results of the exercises are objective and measure achievement of the learning objectives. The

learner reports give information about a broader range of learning outcomes. Here, the focus is on student self-perception. The combination of these data sources is used to draw conclusions on the learning outcomes of the ER. The evaluation form is available in Appendix D.

5.1.2 Data Analysis

We graded the two exercises. For each exercise, there were three options: the answer was correct, incorrect, or no answer was given. In order to keep the evaluation short, the students were not obliged to give their intermediate steps or to explain their reasoning. Therefore, there are no partially correct answers, and we do not have data on the cause of mistakes. However, the correct-incorrect grading already allows us to draw sufficient conclusions on the achievement of the learning objectives. We consider a learning objective to be achieved if at least 80% of the students give the correct answer to the corresponding exercise.

We also analysed the data from the learner reports, using Atlas.ti. The first step was to digitalize the data and translate it to English. Then, we grouped similar statements and assigned these groups a label. Here, we no longer distinguish between the four parts of the learner report. The parts were useful to prompt the students to write down as many sentences as possible, but they play no role in the analysis. After the labelling, the labels were combined into more general themes. Based on the themes, we drew conclusions on the learning outcomes that students experience through the ER. The process is schematized in Figure 5.2. As indicated by the dashed arrows in the figure, the process is not linear. We regularly went back to the previous step, for example to move statements to other groups or to change a label.

5.2 Results

The ER was played in the fourth grade of both a *havo* and a *vwo* computer science class. Because of time constraints, we played a 25-minute version instead of the original 30-minute version. To save 5 minutes, we gave the students the last digit of the first puzzle (about Turtle) and the solution of the second part of the second puzzle (about computing BMI). Unfortunately, there was still no time left for a debriefing, so we omitted that.

In total, 37 students played the ER. We divided them in groups of three. There were some groups of two as well, because the number of students could not be divided by three. There were 14 teams. 9 of them 'escaped' in time, with the fastest time being 9 minutes and 13 seconds. 3 groups still completed the ER when the 25 minutes were over. The remaining 2 groups

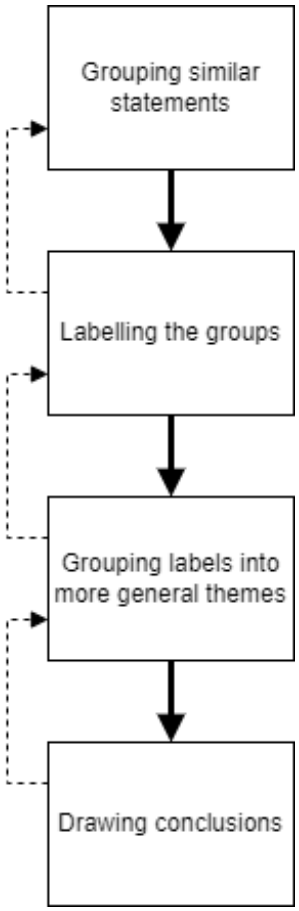


Figure 5.2: Analysis of Learner Reports

did not complete the last puzzle. Of the 37 students, 26 participated in the evaluation.

5.2.1 Exercises

The results of the exercises are summarized in Table 5.1. The first exercise, on learning objective 3.1, was answered correctly by 24 out of 26 ($\approx 92\%$) students. One student did fill in some numbers in the trace table, but did not finish the exercise. One other student gave the wrong answer. To the second exercise, on learning objective 3.2, 23 out of 26 ($\approx 88\%$) students gave the correct answer. Two students gave the wrong answer, and one student did not give an answer. We conclude that learning objectives 3.1 and 3.2 were achieved.

	Correct answers	Incorrect answers	Missing answers
Exercise 1 (learning objective 3.1)	24	1	1
Exercise 2 (learning objective 3.2)	23	2	1

Table 5.1: Results of the exercises

5.2.2 Learner Reports

The analysis of the learner reports led to the grouping of learning outcomes as depicted in Figure 5.3. There are two main themes: content and self-reflection, both of which can be divided into two parts. Then, there are two smaller themes, about problem-solving strategies and motivation, which did not fit in the two main themes. Let us go over the themes one by one.

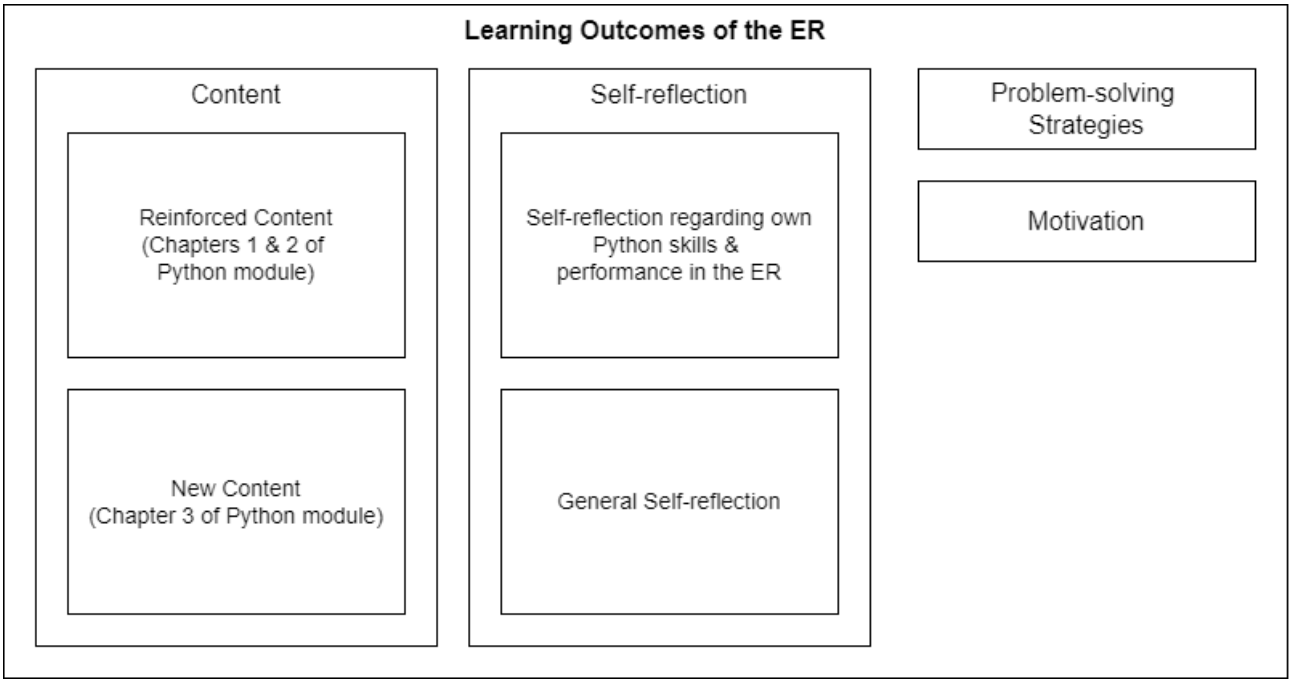


Figure 5.3: Thematized learning outcomes, according to the learner reports

Students reported learning about Python content. Before playing the ER, the students had already studied the first two chapters of the Python module. Some material of these chapters was present in the ER, as a means of reinforcement. The learner reports indicate that students still learnt new things about this material. For example, a student learnt about converting input to an integer (*'I learnt how to apply int() correctly, which I did differently before'*). Another student learnt about the existence of the *float* data type (*'I now know that there is float as well'*). Other statements are about

Turtle, printing output, and handling input.

The ER also contained new content, about variables, covering the third chapter of the Python module. Students indicate that they learnt several things regarding this new material. For example, students learnt how to trace the values of variables (*'I learnt how to trace'*, *'I learnt how to use a trace table'*), and other notions about working with variables (*'I learnt that variables need good names'*). The standard functions *min()*, *max()*, *len()* and *round()* were part of this chapter of the Python module as well, and were present in some ER puzzles. Students learnt about these functions (*'I learnt how to round a number in a specified number of decimals'*, *'I learnt that min is not subtracting'*). Similary, the short-hand operators +=, -=, *=, /= played a role in the ER, and students learnt about them (*'I learnt that x *= 3 is x times 3'*, *'I learnt *= and +='*).

Some students wrote down very general statements about learning content (*'I learnt how to program certain things in Python'*, *'I learnt how new pieces of code work'*). Because it is unclear to which part of the content these statements relate, these statements might fit in either of the content groups. However, taking only the statements into account that clearly belong in a certain group, the majority of statements about content learning concern the new content. Therefore, according to the students' perception, they learnt more things about the new content than about the reinforced content.

Self-reflection is the other main theme. Many students reflect on their Python skills (*'I noticed that I am good at Python'*). Some students indirectly reflect on their Python skills by formulating statements in terms of their performance in the ER (*'I learnt that I can make these kinds of puzzles quite easily'*). Some statements are general (*'From the escape room, I learnt that I find Python difficult'*, *'I discovered that I need to go learn more for the test'*). On the other hand, some students reflect more specifically on which topics they find easy or difficult (*'I found out that I am better with variables than with Turtle'*, *'I learnt that I am good at tracing'*, *'I learnt that it is not true that I understand Turtle'*).

Students also reflect on more general aspects, that are not specific to the topic of the ER. This includes reflection on collaboration (*'I learnt that I program better in a group'*, *'I learnt that I am good at collaborating'*). Some students learnt that they need to work more carefully (*'I learnt that I should work a bit less carelessly'*, *'I now know that I should read the exercises completely before I start'*). Other reflective statements are *'I learnt that I need to stay calm to work well'* and *'From the Escape Room, I learnt that I am competitive in games'*. It is clear that students reflect on a wide variety of aspects regarding themselves.

Finally, besides the main themes of content and self-reflection, there are two other themes. Two students wrote something about problem-solving strategies (*'I learnt that I should write problems down and only then solve them'*, *'So first writing down what you have, and then solving in steps'*). Moreover, there were two statements concerning motivation (*'I noticed that I like programming with Python'*, *'I learnt that these kinds of lessons are fun and educational, and a good substitute for regular lessons'*). However, since the number of students reporting learning regarding the themes of problem-solving strategies and motivation is much lower than the number of students reporting learning regarding the main themes of content and self-reflection, the latter two themes are more relevant.

Chapter 6

Discussion

6.1 Interpretation of Results

In Chapter 3, we established content criteria for using an ER as a teaching method. Their goal was to indicate for what kind of topics an ER is a suitable teaching method. The resulting criteria are general in nature; numerous topics can satisfy them. This is confirmed in the literature, in which previous work indicates that an ER can be themed with almost any topic (Wiemker et al., 2015). However, as reflected in the content criteria, there do exist topics for which an ER is less suitable. For example, we found that tasks with a lot of freedom, such as larger design or programming tasks, are not fitted to an ER. Also, it should be possible to define a number of small tasks with easily verifiable answers. For some topics, this is much easier than for others. Also note that the developed content criteria should be interpreted as guidelines rather than strict rules, because ERs might be used in novel and unexpected ways.

After developing the content criteria, we developed an ER, and evaluated the learning outcomes. The ER contained reinforced content, about which the students had already learnt, as well as new content. In the learner reports, students reported learning about both the reinforced and the new content. The latter group was bigger, as expected. The students had already learnt about the reinforced content and practised with it in the weeks before playing the ER. Therefore, this material should already be familiar to them, and in the ER, it should only be a reminder or simply some extra practice. For that reason, it is not surprising that students reported less learning about the reinforced content than about the new content.

Interestingly, existing work indicates that ERs are suitable to foster or assess

knowledge, but not to acquire new knowledge (Veldkamp et al., 2020). This contrasts with our evaluation, in which we found that students learnt a lot about the new content. Our conclusion is supported by both the exercises, which show that the new learning objectives had been achieved, and the learner reports, containing students' reports about learning new content. What causes the difference between our result and the result that ERs are not suitable to acquire new knowledge? The most likely cause is the short explanation about the new material, that we gave before the ER. Therefore, although the ER was the students' first opportunity to practice with the new material, it was not completely new to them when they started playing the ER.

In the learner reports, students did not report a lot of learning about general skills, such as collaboration or communication skills, even though improving such skills is one of the reasons for using an ER (Veldkamp et al., 2020; Taraldsen et al., 2020). Only three students wrote a statement regarding collaboration. However, we focused on content rather than soft skills in the design of the ER, as indicated in section 4.3.2. If the goal is to improve soft skills, it is advised to do a debriefing solely on them, as reflection on them is lost in reflection on other learning objectives otherwise (Veldkamp et al., 2020). The fact that we did not do this, and we did not focus on soft skills in the design of the ER, explains why students did not report much learning about soft skills.

Something that does occur in the learner reports is self-reflection. There is reflection on general aspects, but most students reflect on their own knowledge and skills regarding the topic of the ER. Some students specifically reflect on which parts of the material they do or do not yet understand. Self-reflection has not been mentioned as a reason to use an ER (Veldkamp et al., 2020; Taraldsen et al., 2020), but our results show that an ER can be used to let students reflect on their understanding of the content.

6.2 Implications

Of course, teachers can directly use the developed ER in their classroom. It is unique, because, if we only consider ERs published in research, it is the first one for secondary computer science education, that is used as a teaching method on a single topic. Besides using this ER directly, it can also be used as an example to create more ERs for secondary computer science education.

Besides providing an actual ER, this thesis produces more general results. The results inform teachers about when and how to use ERs in secondary computer science education. For example, the content criteria guide teachers in deciding for which topics they could use an ER. Moreover, the results

of the evaluation show teachers that an ER, in combination with an explanation of new material, can be used to achieve new learning objectives. It can also be used to reinforce content, from which some students even learn new things. Finally, teachers can use it as a self-reflection tool for students. For instance, it can be useful to play an ER at the end of a chapter or in preparation for a test. This enables students to reflect on their own knowledge and skills. Consequently, this shows them which parts they already understand, and which parts need to be studied again.

6.3 Limitations

The developed content criteria are only based on existing computer science ERs that were published in research. The number of such ERs is limited. Other sources that could influence the criteria, such as commercial educational ERs or teachers having experience using ERs, were not taken into account. We based the criteria solely on the list of ERs in Appendix B. We believe that these ERs already show a variety of content from which criteria can be extracted. However, using, in addition, other sources, would have added validity and might have resulted in slightly different criteria.

Another limitation deals with the learning outcomes of the ER. The learner reports focus on the student's perception. Unfortunately, there is a discrepancy between perceived and actual learning of content knowledge (Veldkamp et al., 2020). Therefore, we should be careful to draw conclusions on the learning of content based solely on the learner reports. We partially resolve this by also objectively measuring the achievement of learning objectives, using exercises. Only the new learning objectives are covered in the exercises, so only these are measured objectively. This means that other learning outcomes, such as learning related to the reinforced learning objectives or outside the learning objectives, are still only measured with the learner reports.

Finally, it remains an open question to what extent the learning outcomes can be attributed solely to the ER. As mentioned, we gave a short explanation about the new material at the start of the lesson. Therefore, we cannot attribute learning outcomes on this material to the ER alone. The learning outcomes are the result of the combination of an explanation and an ER. This means that we cannot draw conclusions on the learning outcomes of solely using an ER.

The fact that no debriefing was carried out, because of time constraints, might have influenced the achieved learning outcomes as well.

6.4 Recommendations

The exciting area of educational ERs is still novel. Therefore, a lot of work remains to be done, especially in secondary computer science education. For example, future work could build on the developed content criteria. As explained, they are based on a limited list of computer science ERs published in research. Other sources, such as commercial educational ERs or teachers having experience using ERs, could give more information on suitable content for ERs. Therefore, future work could improve the content criteria based on these sources. Moreover, the content criteria could be tested for correctness and completeness.

Future work could also extend the evaluation of learning outcomes. In this thesis, we intentionally looked at learning outcomes in a general sense. Students could report about various kinds of learning outcomes, such as learning content, learning soft skills, and learning problem-solving strategies. Future work could zoom in on one kind of learning, in order to draw more specific conclusions. For example, an evaluation solely on the effect of the ER on collaboration skills could be interesting.

Another point concerning learning outcomes is that we only tested achievement of the learning objectives on new content. We did not objectively evaluate whether students' understanding with regard to the reinforcement learning objectives improved due to the ER. Future research could do a pre-test and post-test to find this out. Furthermore, we only researched the learning outcomes of one ER. To allow for more general conclusions, the learning outcomes of other ERs in secondary computer science education should be evaluated as well.

Finally, some informal observations were made when students played the ER, that point towards interesting directions for future research. For example, we noticed that the automatic hint system, in the Escapp web platform (López-Pernas et al., 2021), was barely used. Instead, students asked the teachers for help when they encountered difficulties. Future work could investigate different approaches for delivering hints. We also observed that many students were highly motivated to complete the ER in time. ERs are indeed used as a means to increase motivation (Veldkamp et al., 2020; Taraldsen et al., 2020). In this thesis, we did not formally evaluate the impact of the ER on students' motivation, but future work could do this.

Chapter 7

Conclusions

In this chapter, we first answer the two sub-research questions introduced in Chapter 1. Afterwards, we answer the main research question.

7.1 Content Criteria

We studied the content of existing computer science ERs, in order to answer the following question:

What are content criteria for using an educational escape room in secondary computer science education?

Security and software development are the most occurring topics in computer science ERs. A variety of other topics, including networks, logic, algorithms and automata, are present in at least one ER as well. The similarities between characteristics of popular topics led to the following criteria, that indicate whether certain content is suitable to be taught using an ER:

- The content must contain concepts that students can apply, or concepts of which students' understanding can be tested using a puzzle. Other content, for example about factual information or creation tasks, is less suitable.

- It must be possible to easily verify correct application or understanding of the content. This means that rather than allowing for much freedom, the puzzles should be small, pre-defined tasks.

- Technical concepts are more appropriate for an ER than domains focusing on the social side of computer science.

Although these criteria are a useful indication of suitable content, they should not be used as strict rules. ERs can be used in novel and unexpected ways.

7.2 Achieved Learning Outcomes

We designed and evaluated an ER, to provide an answer to the following question:

What are the achieved learning outcomes of an educational escape room for secondary computer science education?

Using the EscapED framework (Clarke et al., 2017), we designed an ER for students in the fourth grade of *havo* and *vwo*, on the topic of programming in Python. The ER is the first one in secondary computer science education that is used as a teaching method on a single topic, and published in research. It was designed to be part of an existing module on Python, and had the following learning objectives:

- The student can trace the result of Turtle code.
- The student can use the following concepts in a Python program: standard data types, print statements, user input and calculations.
- The student can trace the values of variables in a Python program.
- The student can use variables in a Python program.

The material on the first two learning objectives had already been discussed in class, so this part was included as reinforcement. The last two learning objectives were new for the students; the ER was their first opportunity to practice with the new material on variables.

After letting students play the ER, we evaluated the achievement of the learning objectives, focusing on the last two objectives. We conclude that these were achieved. Therefore, it is possible to achieve learning objectives on content for which the ER is the first opportunity for students to practice with the material.

We also let students report about their own learning. Students report learning about both the reinforced content, and the new content, although students learnt more in the latter category. The ER also appeared to be a useful self-reflection tool for students. Many students reflected on their Python skills, and sometimes specifically on which parts they find easy or

difficult. The ER also enabled more general self-reflection on, for example, collaboration skills or the need to work more carefully.

7.3 Integration of ERs into Education

By combining the answers to the sub-questions, we answer the following main research question:

How can we integrate educational escape rooms into secondary computer science education?

We can use an ER to teach about content that satisfies the criteria described in section 7.1. Since our ER resulted in achievement of the new learning objectives and students themselves reported learning about the new content, ERs can be used as a first opportunity for students to practice with new material. Moreover, it can be used to reinforce previously discussed material, because students also reported learning about the reinforced content.

Besides an activity that teaches content, ERs can also be used as a self-reflection tool for students. Students can reflect on general aspects, not specific to one topic. Moreover, an ER generates self-reflection regarding the topic of the ER; many students reflect on their understanding of the content. Therefore, an ER can be used to give students insight into which parts they do and do not yet understand.

Bibliography

E. Barendsen and J. Tolboom. Advies examenprogramma informatica havo-vwo: inhoud en invoering. Technical report, SLO, Enschede, 2016.

E. Beguin, S. Besnard, A. Cros, B. Joannes, O. Leclerc-Istria, A. Noel, N. Roels, F. Taleb, J. Thongphan, E. Alata, and V. Nicomette. Computer-Security-Oriented Escape Room. *IEEE Security & Privacy*, 17(4):78–83, 2019.

C. Borrego, C. Fernández, I. Blanes, and S. Robles. Room escape at class: Escape games activities to facilitate the motivation and learning in computer science. *Journal of Technology and Science Education*, 7(2):162–171, 2017.

S. J. Clarke, D. J. Peel, S. Arnab, L. Morini, H. Keegan, and O. Wood. EscapED: A Framework for Creating Educational Escape Rooms and Interactive Games to For Higher/Further Education. *International Journal of Serious Games*, 4(3):73–86, 2017.

S. Combéfis and G. De Moffarts. Learning Computer Science at a Fair with an Escape Game. In *Proceedings of the 12th International Conference on Informatics in Schools: Situation, Evolution and Perspectives*, pages 93–95, Larnaca, Cyprus, Nov. 2019.

A. D. De Groot. Over leerervaringen en leerdoelen. In *Handboek Voor de Onderwijspraktijk*. Van Loghum Slaterus, Deventer, 1980.

F. A. Deeb and T. J. Hickey. Teaching Introductory Cryptography using a 3D Escape-the-Room Game. In *Proceedings of the IEEE Frontiers in Education Conference*, pages 1–6, Covington, KY, USA, Oct. 2019.

G. Dimova, M. Videnovik, and V. Trajkovik. Using Educational Escape Room to Increase Students' Engagement in Learning Computer Science. In *Proceedings of the 17th International Conference on Informatics and Information Technologies*, North Macedonia (Online Conference), May 2020.

J. Gerritsen van der Hoop. Een toepassing van het 'learner report' voor het

vaststellen van de effekten van een kursus. Technical report, Technische Hogeschool Eindhoven, Eindhoven, 1981.

A. Gordillo, D. López-Fernández, S. López-Pernas, and J. Quemada. Evaluating an Educational Escape Room Conducted Remotely for Teaching Software Engineering. *IEEE Access*, 8:225032–225051, 2020.

G. Guigon, J. Humeau, and M. Vermeulen. A Model to Design Learning Escape Games: SEGAM. In *Proceedings of the 10th International Conference on Computer Supported Education*, pages 191–197, Funchal, Madeira, Portugal, Mar. 2018.

A. Hacke. Computer Science Problem Solving in the Escape Game "Room-X". In *Proceedings of the 12th International Conference on Informatics in Schools: Situation, Evolution and Perspectives*, pages 281–292, Larnaca, Cyprus, Nov. 2019.

A. M. Ho. Unlocking Ideas: Using Escape Room Puzzles in a Cryptography Classroom. *Problems, Resources, and Issues in Mathematics Undergraduate Studies*, 28(9):835–847, 2018.

J. Kahila, T. Parkki, A. Gröhn, A. Karvinen, E. Telimaa, P. Riikonen, R. Tiitta, P. Haantio, A. Keinänen, T. Kerkkänen, I. Jormanainen, S. Penttinen, and M. Tedre. Escape Room Game for CT Learning Activities in the Primary School. In *Proceedings of the 20th Koli Calling International Conference on Computing Education Research*, pages 1–5, Koli, Finland, Nov. 2020.

C. Lathwesen and N. Belova. Escape Rooms in STEM Teaching and Learning—Prospective Field or Declining Trend? A Literature Review. *Education Sciences*, 11(6):308, 2021.

R. Lister. On the cognitive development of the novice programmer: And the development of a computing education researcher. In *Proceedings of the 9th Computer Science Education Research Conference*, pages 1–15, The Netherlands (Online Conference), Oct. 2020.

S. López-Pernas, A. Gordillo, E. Barra, and J. Quemada. Analyzing Learning Effectiveness and Students' Perceptions of an Educational Escape Room in a Programming Course in Higher Education. *IEEE Access*, 7: 184221–184234, 2019a.

S. López-Pernas, A. Gordillo, E. Barra, and J. Quemada. Examining the Use of an Educational Escape Room for Teaching Programming in a Higher Education Setting. *IEEE Access*, 7:31723–31737, 2019b.

S. López-Pernas, A. Gordillo, E. Barra, and J. Quemada. Escapp: A Web Platform for Conducting Educational Escape Rooms. *IEEE Access*, 9: 38062–38077, 2021.

T. Michaeli and R. Romeike. Investigating Students' Preexisting Debugging Traits: A Real World Escape Room Study. In *Proceedings of the 20th Koli Calling International Conference on Computing Education Research*, pages 1–10, Koli, Finland, Nov. 2020.

B. Musil, S. Gartner, I. Pesek, and M. Krašna. ICT competences assessment through ICT escape room. In *Proceedings of the 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics*, pages 622–626, Opatija, Croatia, May 2019.

S. Nicholson. Peeking behind the locked door: A survey of escape room facilities. http://scottnicholson.com/pubs/erfacwhite.pdf, 2015.

K. R. Otemaier, P. G. Zanese, N. S. Bosso, and E. E. Grein. Educational escape room for teaching Mathematical Logic in computer courses. In *Proceedings of SBGames*, pages 595–604, Recife, Brazil, 2020.

E. Sanchez and M. Plumettaz-Sieber. Teaching and Learning with Escape Games from Debriefing to Institutionalization of Knowledge. In *Proceedings of the 7th International Conference on Games and Learning Alliance*, pages 242–253, Palermo, Italy, Dec. 2018.

S. Seebauer, S. Jahn, and J. Mottok. Learning from Escape Rooms? A Study Design Concept Measuring the Effect of a Cryptography Educational Escape Room. In *Proceedings of the IEEE Global Engineering Education Conference*, pages 1684–1685, Porto, Portugal, Apr. 2020.

J. Streiff, C. Justice, and L. J. Camp. Escaping to Cybersecurity Education: Using Manipulative Challenges to Engage and Educate. In *Proceedings of the 13th European Conference on Game Based Learning*, pages 1046–1051, Odense, Denmark, Oct. 2019.

L. H. Taraldsen, F. O. Haara, M. S. Lysne, P. R. Jensen, and E. S. Jenssen. A review on use of escape rooms in education – touching the void. *Education Inquiry*, 13(2):169–184, 2020.

J. van den Akker, B. Bannan, A. Kelly, N. Nieveen, and T. Plomp. Educational Design Research: Part A: An introduction. Technical report, SLO, Enschede, 2013.

A. van der Linden, W. R. van Joolingen, and R. F. G. Meulenbroeks. Designing an Intrinsically Integrated Educational Game on Newtonian Mechanics. In *Proceedings of the International Conference on Games and Learning Alliance*, pages 123–133, Palermo, Italy, Dec. 2018.

B. J. Van Kesteren. Applications of De Groot's "learner report": A tool to identify educational objectives and learning experiences. *Studies in Educational Evaluation*, 19(1):65–86, 1993.

A. Veldkamp, L. Van de Grint, M. C. P. J. Knippels, and W. R. Van Joolingen. Escape education: A systematic review on escape rooms in education. *Educational Research Review*, 31:100364, 2020.

M. Wiemker, E. Elumir, and A. Clare. Escape Room Games: "Can you transform an unpleasant situation into a pleasant one?". In *Game Based Learning – Dialogorientierung & Spielerisches Lernen Digital Und Analog*, pages 55–68. Fachhochschule St. Pölten GmbH, St. Pölten, Austria, 2015.

Appendix A

Exam Program

This appendix contains a translated version of the Dutch exam program for computer science in secondary education (Barendsen and Tolboom, 2016). Domain A, about skills, is omitted.

Domain	Topic	Subdomain	Subtopic
B	Foundations	B1	Algorithms
		B2	Data Structures
		B3	Automata
		B4	Grammars
C	Information	C1	Objectives
		C2	Identifying
		C3	Representing
		C4	Standard representations
		C5	Structured Data
D	Programming	D1	Developing
		D2	Inspecting and adjusting
E	Architecture	E1	Decomposition
		E2	Security
F	Interaction	F1	Usability
		F2	Societal aspects
		F3	Privacy
		F4	Security
G	Algorithmics, computability and logic	G1	Complexity of algorithms
		G2	Computability
		G3	Logic

H	Databases	H1	Information modelling
		H2	Database paradigms
		H3	Linked data
I	Cognitive computing	I1	Intelligent behaviour
		I2	Characteristics of cognitive computing
		I3	Applications of cognitive computing
J	Programming paradigms	J1	Alternative programming paradigm
		J2	Choice of programming paradigm
K	Computer architecture	K1	Boolean algebra
		K2	Digital circuits
		K3	Machine language
		K4	Variation in computer architecture
L	Network	L1	Network communication
		L2	Internet
		L3	Distribution
		L4	Network security
M	Physical computing	M1	Sensors and actuators
		M2	Development of physical computing components
N	Security	N1	Risk analysis
		N2	Measures
O	Usability	O1	User interfaces
		O2	User research
		O3	Design
P	User experience	P1	Analysis
		P2	Design
Q	Societal and individual influence of computer science	Q1	Societal influence
		Q2	Legal aspects
		Q3	Privacy
		Q4	Culture
R	Computational science	R1	Modelling
		R2	Simulating

Appendix B

Computer Science ERs

Article	General Topic	Learning Objectives / Subtopics
Computer-Security-Oriented Escape Room (Beguin et al., 2019)	Security	discover ways to protect oneself from computer attacks (defender scenario: patching vulnerabilities such as weak passwords and computers left unlocked; attacker scenario: identifying and using vulnerabilities)
Room Escape at Class: Escape Game Activities to Facilitate the Motivation and Learning in Computer Science (Borrego et al., 2017)	Information and Security	LZ77 compression algorithm, public/private RSA scheme, Huffman coding
Teaching Introductory Cryptography using a 3D Escape-the-Room Game (Deeb and Hickey, 2019)	Security and Cryptography	introduce to security problems related to usage of the same password and disposal of sensitive information (Dumpster diving), introduce to basic concepts and vocabulary of cryptography (encryption/decryption, using Caesar Cipher)
Unlocking Ideas: Using Escape Room Puzzles in a Cryptography Classroom (Ho, 2018)	Cryptography	application/review of content, cryptography (Shift Decryption; Mix of Ciphers: substitution, affine, play fair, rail fence and Vigenère; Hill Ciphers)

Learning from Escape Rooms? A Study Design Concept Measuring the Effect of a Cryptography Educational Escape Room (Seebauer et al., 2020)	Cryptography	apply/revise knowledge about cryptography (cryptography methods and hash algorithms e.g. AES, RSA, SHA3)
Escaping to Cybersecurity Education: Using Manipulative challenges to Engage and Educate (Streiff et al., 2019)	Cybersecurity	cryptography (e.g. Caesar cipher, encrypt/decrypt public key), IoT Wifi/BT, secure systems, forensics
Evaluating an Educational Escape Room Conducted Remotely for Teaching Software Engineering (Gordillo et al., 2020)	Software Engineering	reinforce most important concepts of software modelling (understand and interpret UML activity/class/state/sequence/use case diagrams)
Analyzing Learning Effectiveness and Students' Perceptions of an Educational Escape Room in a Programming Course in Higher Education (López-Pernas et al., 2019a)	Front-end development	reinforce basics of web development, including HTML, CSS, JavaSript, Node.js, express and SQL
Examining the Use of an Educational Escape Room for Teaching Programming in a Higher Education Setting (López-Pernas et al., 2019b)	Back-end web development	enhance (pair) programming skills, revising main concepts (Redux architecture, HTML, CSS, JavaScript basics, react lifecycle methods, debug a React application, Flexbox layout mode)
Investigating Students' Preexisting Debugging Traits: A Real World Escape Room Study (Michaeli and Romeike, 2020)	Debugging	find and fix errors, (systematic or particle) troubleshooting process in debugging-related scenarios, topographic search strategy
Room Escape at Class: Escape Game Activities to Facilitate the Motivation and Learning in Computer Science (Borrego et al., 2017)	Computer Networks	network sniffing, TCP conversation
Educational Escape Room for Teaching Mathematical and Logic in Computer Courses (Otemaier et al., 2020)	Maths for software engineering	propositional logic, predicates

Learning Computer Science at a Fair with an Escape Game (Combéfis and De Moffarts, 2019)	Basic CS concepts	not fully specified, but mentioned examples are: Caesar cipher, notion of an algorithm, bubble sort algorithm
Using Educational Escape Room to Increase Students' Engagement in Learning Computer Science (Dimova et al., 2020)	Various topics	course material (not fully specified, but mentioned examples are: translating binary code, working with an Excel file)
Computer Science Problem Solving in the Escape Game "Room-X" (Hacke, 2019)	Problem solving	Caesar encryption, logic, finite automata
Escape Room Game for CT Learning Activities in the Primary School (Kahila et al., 2020)	Computational thinking	several computational thinking skills (e.g. program a repair robot (to practice stepwise, deterministic program execution), binary logic and bit flips, shortest-route problem, decrypt codes)
ICT competences assessment through ICT escape room (Musil et al., 2019)	ICT competences	testing following ICT competences: information and data literacy, communication and collaboration, digital content creation, safety and problem solving (activities include: connecting hardware, setting up computer, using office tools, transferring a file)

Appendix C

Material of the ER

The following pages contain all the material that was used in the ER. The material is in Dutch.

Puzzels Escape Room

Verhaal

Het computersysteem van een ziekenhuis is gehackt, waardoor alle patiënten in gevaar zijn. De hackers hebben meerdere onderdelen van het systeem gesaboteerd en de toegang tot het systeem geblokkeerd. De leerlingen moeten toegang krijgen en het systeem herstellen door de gesaboteerde onderdelen opnieuw te programmeren.

Introductieverhaal dat de leerlingen te zien krijgen

De directeur van het Radboudumc heeft jullie net in paniek opgebeld. Het computersysteem van zijn ziekenhuis is gehackt. De hackers hebben meerdere onderdelen van het systeem gesaboteerd, waardoor alle patiënten in gevaar zijn! Patiënten krijgen de verkeerde hoeveelheden medicatie, metingen geven foute uitslagen, en behandelingen gaan fout. De directeur heeft jullie hulp ingeschakeld om het systeem te herstellen. Hiervoor moeten jullie de gesaboteerde delen opnieuw programmeren. Zorg dat dit binnen een half uur lukt, anders komen de patiënten in levensgevaar...

Houd je aan de volgende regels:

- Gebruik geen internet of andere bronnen.
- Gebruik geen programma's of websites om Python code uit te voeren.
- Open de enveloppen pas als dat wordt aangegeven.

De hackers hebben ook de toegang tot het systeem geblokkeerd. Jullie eerste taak is dus om in het systeem te komen. Succes!

Deel 1

Verhaal

De hackers hebben een toegangscode op het systeem gezet. Deze moet gevonden worden om toegang te krijgen tot het systeem. Op het inlogscherm zijn wat rare codes te zien.

Puzzel

Er zijn drie stukjes Turtle code te zien. De output van elk stukje is een getal. Deze drie getallen vormen de code.

Wat de leerlingen te zien krijgen

De hackers hebben een toegangscode op het systeem gezet. Op het inlogscherm zien jullie alleen wat rare codes. Vind de code om toegang te krijgen tot het systeem!

```
turtle.pendown()
turtle.left(180)
turtle.forward(100)
turtle.left(90)
turtle.forward(200)
turtle.left(90)
turtle.forward(100)
turtle.left(90)
turtle.forward(100)
turtle.left(90)
turtle.forward(100)
turtle.done()
```

```
turtle.pendown()
turtle.left(90)
turtle.forward(200)
turtle.penup()
turtle.left(90)
turtle.forward(100)
turtle.left(90)
turtle.pendown()
turtle.forward(100)
turtle.left(90)
turtle.forward(100)
turtle.done()
```

```
turtle.pendown()
turtle.forward(100)
turtle.right(90)
turtle.forward(200)
turtle.right(90)
turtle.forward(100)
turtle.penup()
turtle.goto(0, -100)
turtle.setheading(0)
turtle.pendown()
turtle.forward(100)
turtle.done()
```

Onthoud: Turtle begint altijd op coördinaat (0, 0), kijkend naar rechts.

Hints

1. Begin voor elk stukje code een nieuwe tekening!

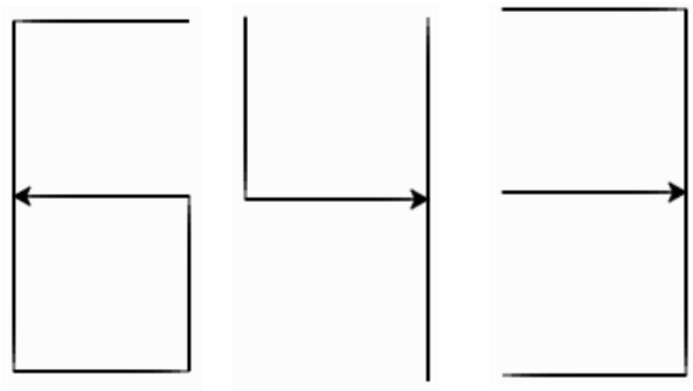
2. Dit is wat de functies doen:

turtle.pendown()	Zet pen op papier, alle bewegingen hierna worden zichtbaar.
turtle.penup()	Haal pen van papier, alle bewegingen hierna worden niet zichtbaar.
turtle.right(x)	Draai x graden naar rechts.
turtle.left(x)	Draai x graden naar links.
turtle.forward(x)	Loop x stappen vooruit.
turtle.goto(x, y)	Zet turtle op positie (x,y). Het midden van het scherm is (0, 0).
turtle.setheading(x)	Kijkrichting. 90 is naar boven, 0 is naar rechts.
turtle.done()	Klaar met tekenen.

3. Het resultaat van de stukjes code vormt een 6, een 4, en een 3.

Oplossing

Elk stukje Turtle code traceren leidt tot een getal. Van links naar rechts is dit 643.



Deel 2

Verhaal

De leerlingen hebben nu toegang tot het systeem. Ze kunnen nu verschillende gesaboteerde onderdelen van het systeem herstellen.

Puzzel

De leerlingen krijgen drie keer een beschrijving van een programma en losse regels code (zowel goede als foute regels). Voor elk programma moeten ze de juiste regels uitzoeken en in de juiste volgorde zetten. Sommige regels zijn al op de juiste plek gezet.

Wat de leerlingen te zien krijgen

Goed gedaan, jullie kunnen nu in het systeem! Er zijn twee onderdelen die hersteld moeten worden. Open envelop A om het eerste onderdeel te herstellen. In de envelop vind je een deel van een programma en losse regels code. Vul het programma aan met de goede regels code (er zitten dus ook foute regels bij!). Doe dit door de goede getallen aan de rechterkant te zetten. Daarna kan je de oplossing van boven naar onder aflezen!

<u>Onderdeel 1: Maximale hartslag</u>

Mensen hebben een maximale hartslag. Als je hartslag hierboven komt, loop je gevaar. Sommige apparaten in het ziekenhuis gebruiken dit om te detecteren wanneer iemand gevaar loopt. De maximale hartslag kan je berekenen met de formule:

$$maximale\ hartslag\ =\ 220\ -\ leeftijd$$

Dit programma moet iemands leeftijd vragen en de maximale hartslag berekenen.

invoer = input("Wat is je leeftijd?")	
print("Je leeftijd is", leeftijd)	

De leerlingen krijgen de volgende losse regels code:

leeftijd = int(invoer)	1
max_hartslag = 220 - leeftijd	5
print("Je maximale hartslag is:", max_hartslag)	3
leeftijd = invoer	2
print("Je maximale hartslag is:" + max_hartslag)	4

<u>Onderdeel 2: BMI berekenen</u>

Goed gedaan, jullie hebben het eerste onderdeel hersteld. Open envelop B voor het volgende onderdeel. De opdracht werkt hetzelfde als net!

Om informatie te krijgen over iemands gezondheid kan je het BMI berekenen. Dit wordt gedaan door het gewicht in kilo's te delen door het kwadraat van de lengte in meters:

$$BMI = \frac{gewicht}{lengte\ x\ lengte}$$

Dit programma moet de gebruiker om zijn lengte en gewicht vragen en vervolgens het BMI printen.

invoer = input("Hoeveel kilogram weeg je?")	
gewicht_in_kg = int(invoer)	
print("Je gewicht is " + str(gewicht_in_kg))	
print("Je lengte is " + str(lengte_in_m))	

De leerlingen krijgen de volgende losse regels code:

invoer = input("Wat is je lengte in meters?")	7
lengte_in_m = float(invoer)	4
BMI = gewicht_in_kg / (lengte_in_m*lengte_in_m)	2
print("Je BMI is:", BMI)	3
lengte_in_m = int(invoer)	6
BMI = gewicht_in_kg / lengte_in_m*lengte_in_m	5
print("Je BMI is:" + BMI)	1

Hints

Onderdeel 1: Maximale hartslag

1. Met een plus kan je alleen strings aan elkaar plakken. Met een komma kan je dingen van verschillende types aan elkaar plakken.
2. De invoer moet worden omgezet naar een getal.
3. De regels 2 en 4 zijn fout.

Onderdeel 2: BMI berekenen

1. Met een plus kan je alleen strings aan elkaar plakken. Met een komma kan je dingen van verschillende types aan elkaar plakken.
2. Lengte in meters is een kommagetal.
3. Denk aan de haakjes bij het berekenen van het BMI.
4. De regels 1, 5 en 6 zijn fout.

Oplossing

Onderdeel 1: Maximale hartslag

invoer = input("Wat is je leeftijd?")	
leeftijd = int(invoer)	1
print("Je leeftijd is", leeftijd)	
max_hartslag = 220 - leeftijd	5
print("Je maximale hartslag is:", max_hartslag)	3

De oplossing van dit deel is 153.

Onderdeel 2: BMI berekenen

invoer = input("Hoeveel kilogram weeg je?")	
gewicht_in_kg = int(invoer)	
print("Je gewicht is " + str(gewicht_in_kg))	
invoer = input("Wat is je lengte in meters?")	7
lengte_in_m = float(invoer)	4
print("Je lengte is " + str(lengte_in_m))	
BMI = gewicht_in_kg / (lengte_in_m*lengte_in_m)	2
print("Je BMI is:", BMI)	3

De oplossing van dit deel is 7423.

Deel 3.1

Verhaal

Er zijn nog meer kapotte onderdelen gevonden. De hackers hebben hier een extra beveiliging op gezet. De leerlingen moeten de code vinden om toegang te krijgen.

Puzzel

De leerlingen krijgen een stukje code waarin variabelen worden gebruikt. Ze moeten de waardes traceren. De code wordt gevormd door de drie gevonden waardes van variabelen achter elkaar te zetten.

Wat de leerlingen te zien krijgen

Jullie hebben de onderdelen succesvol hersteld! Helaas zijn er inmiddels nog meer kapotte onderdelen gevonden. De hackers hebben hier een extra beveiliging op gezet. Op het inlogscherm staat weer wat code. Vind de toegangscode om in de rest van het systeem te kunnen! Open envelop C voor een hulpmiddel.

```
a = 2
b = a
a *= 3
c = a / b
b = max(b, c)
a = b + c - a
```

Code = abc

In envelop C zit een lege trace-tabel:

instructie	a	b	c
a = 2			
b = a			
a *= 3			
c = a / b			
b = max(b, c)			
a = b + c - a			

Hints

1. a *= 3 is hetzelfde als a = a * 3. Oftewel, de waarde van a wordt met 3 vermenigvuldigd.
2. Eerste vier regels van de trace-tabel ingevuld.
3. Volledig ingevulde trace-tabel.

Oplossing

instructie	a	b	c
a = 2	2	-	-
b = a	2	2	-
a *= 3	6	2	-
c = a / b	6	2	3
b = max(b, c)	6	3	3
a = b + c - a	0	3	3

De code is 033.

Deel 3.2

Verhaal

De leerlingen hebben nu toegang tot de rest van het systeem. Ze kunnen nu de laatste gesaboteerde onderdelen van het systeem herstellen.

Puzzel

De leerlingen krijgen twee keer een beschrijving van een programma met bijbehorende code. Echter, de variabelen zijn op sommige plekken weggelaten. Voor beide programma's moeten ze de juiste variabelen op alle plekken invullen.

Wat de leerlingen te zien krijgen

Prima werk, jullie hebben nu toegang tot het hele systeem! Er zijn nog maar twee onderdelen die jullie moeten herstellen. De code van deze onderdelen bestaat nog, maar de variabelen zijn op sommige plekken verdwenen. Vul overal de juiste variabele in. Elke variabele is gekoppeld aan een cijfer. Door de nummers van de ingevulde variabelen achter elkaar te zetten, krijg je de oplossing.

<u>Onderdeel 3: Aantal pillen berekenen</u>

Een hoeveelheid medicatie wordt vaak uitgedrukt in aantal milligram. Het computersysteem kan dit omrekenen naar het aantal pillen dat hierbij hoort. Het gewicht van een pil ligt vast. Dit programma moet vragen hoeveel milligram iemand in moet nemen en berekent dan hoeveel pillen dit zijn. Rond af op één decimaal (pillen kunnen in kleinere delen worden gesneden).

Open envelop D. Vul de lege plekken in. Gebruik de volgende namen voor variabelen: gewicht (1), GEWICHT_PIL (2), invoer (3), pillen (4). De getallen tussen haakjes zijn de bijbehorende getallen die je nodig hebt om de code te vinden.

```
GEWICHT_PIL = 500

print("Hoeveel milligram moet je innemen?")

invoer = input()

gewicht = int(……..)

…….. = round( …….. / ……., 1)

print( "Voor", gewicht, "milligram moet je", ……., "pillen innemen.")
```

Onderdeel 4: Bloeddruk meten

Top, jullie hebben het onderdeel hersteld. Herstel nu het laatste onderdeel, op dezelfde manier als net!

Bij het meten van bloeddruk wordt de bovendruk en de onderdruk gemeten. Hiermee kunnen we de MAP (Mean Arterial Pressure = gemiddelde arteriële druk) berekenen:

$$MAP = onderdruk + \frac{1}{3}(bovendruk - onderdruk)$$

Dit programma moet de bovendruk en onderdruk vragen en daarmee de MAP berekenen.

Open envelop E. Vul de lege plekken in. Gebruik de volgende namen voor variabelen: bovendruk (1), gemiddelde_druk (2), invoer (3), onderdruk (4). De getallen tussen haakjes zijn de bijbehorende getallen die je nodig hebt om de code te vinden.

```
print("Geef de bovendruk: ")
…….. = input()
bovendruk = int(invoer)
print("Geef de onderdruk: ")
invoer  = input()
…….. = int(……..)
…….. = …….. + 1/3 * (…….. - onderdruk)
print("De gemiddelde druk is:", gemiddelde_druk)
```

Hints

Onderdeel 3: Aantal pillen berekenen

1. Deel het totale gewicht door het gewicht van een enkele pil.
2. Eerste lege plek = invoer; Tweede lege plek = pillen; Vijfde lege plek = pillen
3. Eerste lege plek = invoer; Tweede lege plek = pillen; Derde lege plek = gewicht; Vierde lege plek = GEWICHT_PIL; Vijfde lege plek = pillen

Onderdeel 4: Bloeddruk meten

1. De variabele invoer kan zowel voor het vragen van de bovendruk als onderdruk gebruikt worden.
2. Vierde lege plek = gemiddelde_druk; Vijfde lege plek = onderdruk; Zesde lege plek = bovendruk
3. Eerste lege plek = invoer; Tweede lege plek = onderdruk; Derde lege plek = invoer; Vierde lege plek = gemiddelde_druk; Vijfde lege plek = onderdruk; Zesde lege plek = bovendruk

Oplossing

Onderdeel 3: Aantal pillen berekenen

```
GEWICHT_PIL = 500
print("Hoeveel milligram moet je innemen?")
invoer = input()
gewicht = int(invoer (3))
pillen (4) = round( gewicht (1) / GEWICHT_PIL (2), 1)
print( "Voor", gewicht, "milligram moet je", pillen (4), "pillen innemen.")
```

De oplossing van dit deel is 34124.

Onderdeel 4: Bloeddruk meten

```
print("Geef de bovendruk: ")
invoer (3) = input()
bovendruk = int(invoer)
print("Geef de onderdruk: ")
invoer = input()
onderdruk (4) = int(invoer (3))
gemiddelde_druk (2) = onderdruk (4) + 1/3 * (bovendruk (1) - onderdruk)
print("De gemiddelde druk is:", gemiddelde_druk)
```

De oplossing van dit deel is 343241.

Appendix D

Evaluation Form

The following pages contain the evaluation form, consisting of two exercises and a learner report. It is in Dutch.

Opdrachten

Maak de twee onderstaande opdrachten. We gebruiken dit alleen om de Escape Room te evalueren. Maak de opdrachten individueel.

Traceren

Wat zijn de waardes van x, y en z aan het einde van het volgende programma?

Het is niet toegestaan om een programma of website te gebruiken om de code uit te voeren. Probeer het echt zelf te doen!

```
x = 5
y = x - 3
x += 2
z = x * y
y = min(x, z)
```

x = ... y = ... z = ...

Je mag onderstaande trace-tabel gebruiken als hulpmiddel.

instructie	x	y	z
x = 5			
y = x - 3			
x += 2			
z = x * y			
y = min(x, z)			

Gewicht omzetten

In sommige landen drukt men gewichten uit in *pounds* in plaats van in grammen. We willen een programma dat gewichten omzet van *pounds* naar een aantal gram. Het programma moet een

gewicht in *pounds* vragen, dit omzetten naar een aantal gram (1 *pound* = 453.59237 gram), en het resultaat (afgerond op 3 decimalen) printen op het scherm.

Hieronder staan 6 programma's. Omcirkel het juiste programma.

```
POUND_IN_GRAM = 453.59237
print("Geef een gewicht in pounds: ")
invoer = input()
gewicht_pounds = float(invoer)
gewicht_gram = gewicht_pounds * POUND_IN_GRAM
gewicht_gram = round(gewicht_gram, 3)
print("Gewicht in gram is:", resultaat)
```

```
POUND_IN_GRAM = 453.59237
print("Geef een gewicht in pounds: ")
gewicht_pounds = input()
gewicht_gram = gewicht_pounds * POUND_IN_GRAM
gewicht_gram = round(gewicht_gram, 3)
print("Gewicht in gram is:", gewicht_gram)
```

```
POUND_IN_GRAM = 453.59237
print("Geef een gewicht in pounds: ")
gewicht_pounds = float(invoer)
gewicht_gram = gewicht_pounds * POUND_IN_GRAM
gewicht_gram = round(gewicht_gram, 3)
print("Gewicht in gram is:", resultaat)
```

```
POUND_IN_GRAM = 453.59237
print("Geef een gewicht in pounds: ")
invoer = input()
gewicht_pounds = float(invoer)
gewicht_gram = gewicht_pounds * POUND_IN_GRAM
gewicht_gram = round(gewicht_gram, 3)
print("Gewicht in gram is:", gewicht_gram)
```

```
POUND_IN_GRAM = 453.59237
print("Geef een gewicht in pounds: ")
gewicht_pounds = float(invoer)
gewicht_gram = gewicht_pounds * POUND_IN_GRAM
resultaat = round(gewicht_gram, 3)
print("Gewicht in gram is:", gewicht_gram)
```

```
POUND_IN_GRAM = 453.59237
print("Geef een gewicht in pounds: ")
gewicht_pounds = input()
gewicht_gram = gewicht_pounds * POUND_IN_GRAM
gewicht_gram = round(gewicht_gram, 3)
print("Gewicht in gram is:", resultaat)
```

Leerrapport

Je gaat een leerrapport invullen, waarin je aangeeft wat je hebt geleerd van de Escape Room. Dit kan van alles zijn, bijvoorbeeld:

- Dingen over programmeren/Python
- Dingen over algemene vaardigheden (samenwerken, communiceren, analyseren, problemen oplossen, etc.)
- Dingen over jezelf
- Dingen over jullie gebruikte strategieën
- Andere dingen die niet in dit lijstje staan

Iedereen kan andere dingen geleerd hebben. Probeer zo specifiek mogelijk te zijn: maak geen vage, algemene zinnen, maar noem concrete voorbeelden.

Het leerrapport bestaat uit onderdelen A t/m D.

Deel A: “Van de Escape Room heb ik geleerd dat/hoe ...”

In deel A gaat het om voorbeelden van *dingen die je geleerd hebt waardoor je algemene regels, feiten, technieken kent*. *Je weet hoe iets in elkaar zit, of hoe iets moet.*

Voorbeeldzinnen:

- Ik heb geleerd / gemerkt/ ontdekt/ weet nu ... (dat iets zo is)
- Ik heb geleerd / gemerkt/ ontdekt/ weet nu … (dat iets zo werkt)
- Ik heb geleerd / gemerkt/ ontdekt/ weet nu … (hoe iets gedaan moet worden)
- Ik heb geleerd / gemerkt/ ontdekt/ weet nu …
  (enzovoort)

Denk nu aan voorbeelden van regels en feiten die jij hebt geleerd. Gebruik de voorbeeldzinnen hierboven om zo veel mogelijk leerzinnen te maken.

Deel B: “Van de Escape Room heb ik geleerd dat het niet waar is, dat ...”

In deel B gaat het om voorbeelden van *uitzonderingen die je geleerd hebt, dingen die niet zo zijn als je altijd dacht.*

Voorbeeldzinnen:

- Ik heb geleerd / gemerkt/ ontdekt/ weet nu dat het niet waar is, dat …. (iets altijd zo is)
- Ik heb geleerd / gemerkt/ ontdekt/ weet nu dat er ook … bestaan
- Ik heb geleerd / gemerkt/ ontdekt/ weet nu dat iets niet altijd op … manier, maar ook op … manier gedaan kan worden
- Ik heb geleerd / gemerkt/ ontdekt/ weet nu dat niet …
- Ik heb geleerd dat ook …
  (enzovoort)

Denk nu aan voorbeelden van uitzonderingen die jij hebt geleerd. Gebruik de voorbeeldzinnen hierboven om zo veel mogelijk leerzinnen te maken.

Deel C: “Van de Escape Room heb ik geleerd dat ik ...”

In deel C gaat het om voorbeelden van wat jij door de Escape Room *over jezelf* hebt geleerd.

Voorbeeldzinnen:

- Ik heb geleerd / gemerkt/ ontdekt/ weet nu dat ik … vind, omdat …
- Ik heb geleerd / gemerkt/ ontdekt/ weet nu dat ik goed (slecht) ben in …
- Ik heb geleerd / gemerkt/ ontdekt/ weet nu dat ik een hekel heb aan …, omdat …
- Ik heb geleerd / gemerkt/ ontdekt/ weet nu dat ik dat probleem het beste op … manier kan aanpakken, omdat …
- Ik heb geleerd / gemerkt/ ontdekt/ weet nu dat ik …
  (enzovoort)

Denk nu aan voorbeelden van jouw persoonlijke leerervaringen. Gebruik de voorbeeldzinnen hierboven om zo veel mogelijk leerzinnen te maken.

Deel D: “Van de Escape Room heb ik geleerd dat het niet waar is, dat ik ...”

Bij deel D gaat het om voorbeelden van *uitzonderingen en verrassingen die je over jezelf geleerd hebt.* Je mening over jezelf (daar ben ik slecht in, dat vind ik leuk, dat doe ik altijd zo), blijkt niet altijd te kloppen.

Voorbeeldzinnen:

- Ik heb geleerd/ gemerkt/ ontdekt/ weet nu dat het niet waar is, dat ik altijd goed (slecht) ben in .., omdat …
- Ik heb geleerd/ gemerkt/ ontdekt/ weet nu dat het niet waar is, dat ik nooit .., maar …
- Ik heb geleerd/ gemerkt/ ontdekt/ weet nu dat het niet waar is, dat ik altijd een hekel heb aan … omdat ...
- Ik heb geleerd/ gemerkt/ ontdekt/ weet nu dat ik dit probleem ook op … manier kan aanpakken
- Ik heb geleerd/ gemerkt/ ontdekt/ weet nu dat het niet waar is, dat ik … (enzovoort)

Denk nu aan voorbeelden van uitzonderingen en verrassingen over jezelf die jij hebt geleerd. Gebruik de voorbeeldzinnen hierboven om zo veel mogelijk leerzinnen te maken.