Radboud University

Designing unplugged methods for computer programming: functions and parameters

MASTER THESIS

MASTER'S PROGRAMME IN INFORMATION SCIENCES, SPECIALISIATION SECURITY & PRIVACY

FACULTY OF SCIENCE

THESIS SUPERVISOR:

PROF. E. BARENDSEN

2nd EXAMINER:

DR. J.E.W. SMETSERS

AUTHOR:

MART GELUK

MARTGELUK@GMAIL.COM

Abstract

Computational thinking is an important topic in the design of new curriculum and unplugged teaching seems to be a popular method. Research shows that novices show many misconceptions when learning computer programming and that the subject of functions and parameter passing is still largely undiscovered. The aim of this study is to identify obstacles in learning functions with parameters and to see whether unplugged methods are an effective tool for teaching. With the use of literature an unplugged activity was created and tested on high school students. Data was collected from the students taking part in the unplugged activity and from students following the normal curriculum. Analysis of the data discovered common misconceptions. We identified a new misconception that students have with the naming of actual and formal parameters. The results indicate that unplugged methods are useful for some learning goals and have the ability to increase interest computer science. For these reasons it is recommended to use unplugged as an additional tool next to the traditional methods for teaching computer programming. Further research is needed to provide more evidence on the real effect of unplugged teaching.

Preface

The thesis “Designing unplugged methods for computer programming” marks the end of my Master Information Sciences. Information Sciences has everything to do with how we use information, whether that is the protection of information or analysis, collection and more.

I view education as the opportunity to learn new information and I have always been interested in teaching others new things. Together with my supervisor the subject of teaching computer programming was discussed and soon an interesting research topic was found. Working with high school students during this research was a fun process. The research was not without some hiccups and writing everything down correctly took longer than I anticipated. Reading literature and collecting data can be time consuming but I soon realized that the most time-consuming process was analysis. Proper planning and design make life easier and these are two things I would improve next time. I would like to thank my supervisor Erik Barendsen for this continued support during the last year and express my gratitude towards Renske Weeda and the participants for providing me with time and effort.

Mart Geluk

Contents

1. Introduction

Computer science has become a major topic in primary and secondary education. Students are being taught how to program from a young age and the concept of computational thinking (CT), the art of solving problems by using concepts that are fundamental to computer sciences (CS), is an important topic in the current development of the curriculum.

This thesis reviews programming literature relating to the subject of functions and parameters passing. These findings are then used to develop new teaching activities while identifying difficulties in learning how to program. These difficulties are then used to identify important features of a kinesthetic learning environment that support teaching.

Computer programming is regarded as a difficult process to master, notably in consideration of all the different skills required. Novice programmers experience parameters and procedural abstraction as conceptually difficult. Studies have shown that these subjects are difficult to understand. Mostly using programming languages as the exercise tool, research has identified some difficulties and misconceptions that are common in this subject. These issues mostly relate back to novices having a misconception about the programming language, environment or by holding a wrong notional machine. The more important task of overcoming these difficulties and providing methods for bettering teaching is something left mostly untouched. Difficulties can be caused by many things but it is common for novices to have big misconceptions about some concepts, making it hard to learn and apply new things. Understanding what passing a parameter to a function entails is deemed difficult but is a skill required for an individual to process further in their programming ability. The passing of variables, returning the answer and figuring out the scope of the variables can be a hard-mental process. Furthermore, parameter passing can be made more difficult by explaining call by reference and call by value. These unexplored areas, lust for more research and understanding form the basis of this study.

2. Background

This chapter contains information about computing science education, outlining the context of this research. It starts with discussing the international perspectives and curriculum changes as well as reviewing the difficulties and misconceptions that come up when learning how to program. It expands on what this means for teachers and how the unplugged method might help.

2.1 International developments in computer education

The importance of teaching CS and more generally science is not to be understated, even for young children. Arguments for teaching science to children in kindergarten and primary education are: Children naturally enjoy observing and nature, exposing students to science develops a positive attitude which impacts their choices, early exposure to science leads to better understanding when studied later formally (Haim Eshach, 2005). Looking internationally, computer science curriculum is being introduced in K-12 education resulting in major curriculum changes in countries such as Australia, Netherland, New Zealand and the UK (Webb, et al., 2017). Australia included Digital Technologies as a compulsory subject in the curriculum in 2015. In the United Kingdom the royal society (2012) identified a need for a curriculum reform based on the unbalance in the curriculum that emphasized too much on digital skills instead of understanding concepts. The new curriculum in England that was introduced in 2014 expects students to 'understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms and data-representation' (Department for Education, 2013). Currently, the Netherlands is still using an exam program that dates from 1998, based on a time were mobiles phones, the internet and computational power was still limited. This program is being phased out. A renewal committee has created an advice for a new core program for all computer science students in high school, addressing classic and new concepts (Barendsen & Tolboom, 2016). A new curriculum based on this advice is used starting in the new school year of 2019. The new curriculum considers three skills crucial for computer science, these are: designing and developing, utilizing computing science as a perspective and working together. All these international changes in the curriculum of many developed countries emphasize the importance of teaching concepts, including abstraction to children in primary and secondary education.

Looking internationally, many countries have started with integrating computational thinking in the mandatory curriculum of primary and secondary education. A review of policy initiatives shows two reasons behind this movement: i) to prepare for future employment and fill ICT job vacancies; and ii) to enable students to think in a different way, express themselves using new media and solve real-world problems (Brackmann, et al., 2017) (Stefania Bocconi, 2016).

2.2 Learning how to program

Learning how to program presents many difficulties, this chapter uses literature to describe general programming difficulties, misconceptions and implications for teaching. First the general challenges that novice students face will be described, followed by programming concepts that are considered problematic. These concepts all retain different misconceptions that need to be identified. Al last, the implications for teachers are discussed, focusing on methods and improvements that help the effectiveness of teaching.

Programming is regarded as a difficult subject to master, with novice programmers suffering from a wide range of difficulties and deficits (Robins, Rountree, & Rountree, 2003). This earlier work reviewing the literature related to the psychological and educational study of programming identified various problems experienced by novices.

Juha Sorva describes in his doctoral dissertation that there are five challenges to learning novice programming. Cognitive load, plan schemas, misconceptions, tracing skills and notional machines (Sorva, Visual Program Simulation in Introductory Programming Education, 2012).

1. Novice programmers experience great cognitive loads when creating programs.
2. Experience programmers have schemes representing generic solutions to common problems, novices still have to acquire these.
3. Misconceptions are common with novice programmers, making programming difficult.
4. Creative and unexpected programming tasks require mental tracing of the program, something novices are not yet competent in.
5. Novices fail to make a mental model of the notional machine to understand what the program is doing.

Sorva further notices that the first four challenges seem to be affected by the fifth, the notional machine. Understanding the notional machine could help prevent or correct the misconceptions of students. The notional machine is not the single problem that causes difficulties with students, but appears to be one of the main contributors. This paper will further focus on the third and fifth challenge, misconceptions and the notional machine, not entirely disregarding the other challenges but keeping them in mind when developing new methods.

2.2.1 Conceptual difficulties

Research that looked into the learning challenges that novice programming students face when studying high level concepts, acknowledges that elements of the curriculum with a highly conceptual nature proved to be the most challenging, both from an understanding and implementation perspective. Butler and Morgan (2007) used a survey conducted on approximately 500 university students following a computer programming course to come to this conclusion.

A study of difficulties in learning programming discovered based on a survey that regarding programming concepts, students perceived the following concepts the most difficult: pointers and references, parameters, structured data types, abstract data types, error handling and using language libraries (Piteira & Costa, 2013). The survey used was adapted from literature and administered to students and teachers of programming courses. Their findings support the notion that abstract concepts are hard to understand. Research suggests that the conceptual nature of computer science makes it difficult to understand and that threshold concepts are highly related. Meyer and Land first introduced this notion in 2003 and explained threshold concepts as concepts that may represent knowledge that is conceptually difficult or as a new and previously inaccessible way of thinking (Land, 2003). Studies have been conducted since 2003 to further identify and explain threshold concepts in computer science. Concepts such as parameters and parameters passing have been identified as threshold concepts (Kallia & Sentance, 2017) (Sanders & McCartney, 2016). The studies also found that knowing that something is considered a threshold concepts impacts the way the concept is taught.

In identifying computer teachers' perspectives on threshold concepts (Kallia & Sentance, 2017) the researchers used the Delphi method to achieve consensus. They send out three online questionnaires combined with an iteration process of design, distribution, data collection and analysis. It can be clearly seen that studies focusing on difficulties in learning computer programming used mostly either literature and a survey administered to computer science students or recording, observing and think-aloud protocol students working on exercises.

An earlier mentioned study that focused on surveying the work that has been done since 2005 on threshold concepts and identifying the unexplored areas for future work mentioned in the conclusions that there is a good deal of work to be done to make threshold concepts more useful in teaching and curriculum design (Sanders & McCartney, 2016). In the conference proceedings of the Australian Society for Computers in Learning in Tertiary Education one of the future directions was focusing research on areas that contain concepts that have a high level of conceptual difficulty with the aim of clarifying exactly why students find these elements so difficult (Butler & Morgan, 2007).

It is possible for students to write working programs without understanding the concepts involved or with fundamental misconceptions. In life many things are up for debate, but most concepts in science are strictly defined and a wrong understanding of these concepts results in non-working programs.

Most of the research that has been conducted in the last decade focused on identifying difficult subjects and the general causes behind them, but lacked in making programming education more usable and improving techniques.

In the next chapter, the common misconceptions in programming will be discussed. This paper separates conceptual difficulties and misconceptions, even though they appear somewhat similar. Based on previous research, conceptual difficulties are more about defining which concepts are considered

troublesome, misconceptions dive deeper into the problem and is therefore described independently.

2.2.2 Misconceptions in programming

Since the mid ’80s many researchers centered their attention on identifying students’ misconceptions (Kallia & Sentance, 2017). The definition of a ‘misconception’ can be broad or narrow, in this article it will refer to any deviation in the intended understanding of a particular concept.

An important concept when discussing misconceptions is the notional machine. A notional machine of a device or program is not about the physical aspects but the abstraction, the capabilities provided. Many misconceptions arise because of a wrong notional machine. Recommendations from the misconception literature suggest that teacher need to help students form a model of program execution, an abstract notional machine (du Boulay, 1986) (Sorva, Notional machines and introductory programming education, 2013).

Finding misconceptions in programming is difficult to accomplish with surveys. Most studies used interviews or think-aloud protocols.

Knowing the misconceptions in these related fields allows for a complete overview. Sorva catalogued known misconceptions in his PhD thesis, the related misconceptions will be used for this research. In appendix A, a summary with the most related misconceptions can be found based on the work of Sorva (2012).

2.2.3 Implications for teaching programming

Teachers that can identify misconceptions in their students have larger classroom gains than teachers that cannot identify these misconceptions, according to a study of physics teachers (Sadler, 2013). Even if programming courses or other science subject are taught in a way that should be clear to all students, misconceptions that students have from previous courses or their environment influences their understanding. When students’ misconceptions are not identified and addressed then teaching them becomes difficult, using the existing knowledge on misconceptions in the development of curriculum could fix the underlying issue.

To effectively teach programming to students, especially novices, a combination of techniques and skills is necessary. Identifying, preventing and correcting misconceptions is one of these skills and should be done early in the development.

A relatively old but relevant work on how to make computer programming more understandable for novices, describes two pedagogical techniques (Mayer, 1981). The first technique is to provide the students with a concrete model of the concept that is being introduced, this supposedly helps novices understand the system better. The second technique is to let novices explain technical information to the instructor in their own words. Mayer analyzed the effectiveness of these techniques by measuring the ability to solve problems that where not explicitly taught.

Mayer describes that a concrete model can have a strong effect on the use of new information and that there is empirical proof that a deeper understanding of the system allows novices to use new information in a more useful way. A difficult to define line should be drawn between understanding the simple and visible parts of the systems and having to be a system expert.

The second technique focusses on the elaboration process. Having students explain the new information and relating it to other concepts may help understanding. This can also be used by teachers to gain a deeper understanding in the students' problems.

Robins, Rountree, & Rountree (2003) summarize in their work on novice programming related to teaching and learning, that novices must learn to develop models of the problem domain, the notional machine and the desired program. They also describe that having familiarity with issues identified in literature can help course design.

As mentioned earlier, the use of tangible objects can be instrumental when explaining concepts that novices don't have a vocabulary for. When using tangible objects an analogy is implicitly used. The use of analogy can also bring unwanted effects, where the link with the CS concept is quickly forgotten. To explain possible unwanted effects created by analogy or tangible objects, the following example will be used. During an introductory programming lecture, the students are taught that a variable is like a box, you can put stuff in it. When the link with the CS concept isn't made and the constraints are clarified, students start reasoning with their own mental frame of a box. This brings many unwanted results, students start thinking a variable can store multiple value's at once, just like a real box. Or that a variable can store different things, like numbers and words.

Highlighting the link with CS is therefore important. Semantic waves may be a key to development by enabling recontextualization of knowledge (Maton, 2013). When teaching technical or abstract subjects the teacher could start off with explaining the technical side before descending to the real-life implementations. To finish the wave, the teacher should ascend back-up the wave by link the concrete example to the technical explanation he/she started with. This technique ensures that the links between concepts and real-life implementations are made correctly.

2.2.4 Evaluating unplugged methods

Developing new methods that can be used for teaching is a contributing process to education. But these new methods need to be evaluated and the effect on students has to be assessed, otherwise the contributing factor is unknown. The Solo taxonomy (Biggs & Collis, 1982) and the Bloom taxonomy (Krathwohl, 2002) are two broadly used taxonomies for classifying and assessing learning outcome. A lot of computer science research uses a combination of one of these taxonomies to develop tests and analyze data. For example, research investigating if Scratch, a visual programming environment, can be used to teach CS concepts, used SOLO and Bloom's taxonomy to develop the tests used to assess the students involved in the research (Orni Meerbaum-Salant, 2013).

Bloom's taxonomy describes 6 levels for classifying learning objectives. The 6 levels cover the six major categories in the cognitive domain and Bloom believed that his taxonomy could additionally provide a common language for learning goals and a basis for determining educational goals (Krathwohl, 2002). The categories of the original taxonomy were; Knowledge, Comprehension, Application, Analysis, Synthesis, and Evaluation (Bloom, Engelhart, Hill, & Krathwohl, 1956). Krathwohl, one of the researchers that helped develop the original taxonomy revised the taxonomy in 2002. One of the reasons for this revision was based on the fundamental difference between the knowledge category and the other 5 levels. The revised taxonomy expanded the taxonomy from one dimension to two dimensions: the knowledge dimension and cognitive process dimension. The knowledge and cognitive process dimension form a two-dimensional table, which can be used to classify learning objectives.

Blooms' taxonomy is also used for classifying learning outcomes but does this in terms of their complexity. The structure of observed learning outcomes describes levels of increasing complexity in students' understanding. The SOLO model consists of the following five levels of understanding: pre-structural, Uni-structural, multi-structural, relational and extended abstract. To measure understanding and to see if there is a meaningful difference between plugged and unplugged the SOLO taxonomy will be used.

2.2.5 Unplugged teaching and learning

One of the current methods for teaching students that are not familiar with CS principles, concepts of computer science is by using "CT Unplugged". This approach applies so-called unplugged learning activities, that is, activities that do not make use of a computer (Nishida, et al., 2009). In the recent years a lot of development has taken place in the creation of unplugged activities with the goal of teaching students important computational thinking skills. The low-tech approach works well for beginner audiences because it is accessible for those with little technical background (Curzon, 2009). Another benefit of the unplugged approach is that it doesn't depend on computers or other technologies. Those that don't have access to computers, countries where a reliable power source is unavailable or where an internet connection is only for the privileged, the unplugged approach allows individuals to convey the fundamentals of computer science and computational thinking to others. Unplugged activities do not require the ability to program and thus provide a barrier free method of learning.

Many approaches have been suggested to combat the decline in interest in computing science. A somewhat successful approach is the CS unplugged method. In 2009 researcher wanted to developed more unplugged activities, but realized that they needed some design guidelines to retain the favorable properties of the original activities. The structure and design of the at that time available unplugged activities where analyzed, resulting in an unplugged design pattern. The researchers proposed this "CS Unplugged Pattern" as a guideline for developing teaching resources and a methodology for identifying components that might be used to construct a kinesthetic activity

Step 1 – Choose a CS concept

Step 2 – Identify key elements

Step 3 – Identify link with common objects

Step 4 – Turn it into a challenge

Step 5 – Evaluate the activity

Step 6 – Refine

Step 7 – Publish

Figure 1: Unplugged Design Pattern

(Nishida, et al., 2009). The researchers provided 7 steps for constructing an exercise, see figure 1.

Studies with the goal of developing unplugged activities mainly used a design study. The researchers came up with an activity based on literature, tested it out in real life and analyzed the response of the participants to further improve the activity.

Grover and Pea (2013), two researchers in the computational thinking field, described in an article the current state of computational thinking in K-12 education. They note that there remains much to be done to develop more theoretical and practical understanding of computational competencies in children and how difficulties can be addressed. They ask the questions if and what children will do better when the curriculum is designed with CT in mind.

Currently, there is a limited amount of research providing the indication that unplugged methods have a positive effect. A paper about developing CT skills through unplugged activities describes in the background that the effectiveness of the unplugged approach for developing CT skills is not heavily investigated (Brackmann, et al., 2017). Brackmann concludes by stating that there is a need for more research that provides evidence on the effect of unplugged activities. To encourage interest in computer science, a group of researchers visited several fourth-grade classes to do unplugged activities. Pre- and post-tests indicate improved confidence and interest in computer science and math (Guiffre, 2009). Although this experiment shows improved confidence and interest it did not measure actual improvements in knowledge or better scores on tests. In a study examining the effect of unplugged activities on students' views of CS, the results indicate that unplugged activities can start the process of changing students' views (Taub, Ben-Ari, & Armoni, 2009). They indicate that despite the fact that students understood what computing science is, they still perceived the computer as the essence of CS and not as a tool. One of the goals of unplugged activities is to show students that computer science is not depended on computers and that the computer is simply an efficient method to complete certain tasks. Not all research concludes with positive results for the unplugged approach. Using a quasi-experimental, nonequivalent control group design, the impact of a semester-long program based on CS unplugged was evaluated (Feaster, Segars, Wahba, & Hallstrom, 2011). Results indicate that the program had no impact on students' attitudes or understanding towards computer science. A more recent paper from 2013 affirms that computer science unplugged is an effective resource for outreach but states that whether or not the effectiveness of unplugged activities compares to that of traditional teaching methods is still unanswered (Thies & Vahrenhold, 2013). The study's conclusion is that teaching using unplugged activities can be at least as effective as the traditional approach.

In the section above we discussed the various advantages and disadvantages of the unplugged approach. Previous studies indicate that the unplugged approach could be used as a powerful resource but still provide inadequate data on the effect on understanding. So, what is this powerful resource exactly and what parts construct an unplugged activity. Unplugged learning uses a combination of different learning styles as do the more traditional teaching methods. A model, proposed by Walter, B (1979) and expanded by Neil Fleming categorizes learning into four different types. Visual learning, auditory learning, read/write and kinesthetic. Whereas programming traditionally is taught auditory and students use a combination of reading and writing, unplugged uses foremost kinesthetic learning. That is, learning by the use of physical activities.

Unplugged activities involve games, magic tricks and competitions to show children the kind of thinking that is expected from a computer scientist (Bell, Alexander, Freeman, & Grimley, 2011). In a research project mentioned earlier it was observed that students prefer learning by doing (Piteira & Costa, 2013) which can fit into the unplugged approach. Direct kinesthetic activities build upon this and uses tangible object to represent concepts, such as using a box to represent a variable. Using tangible objects makes that session memorable and the invisible visible according to (Curzon, 2009). An advantage of tangible objects is that students can explain difficulties by pointing to parts of the physical object without having to master the vocabulary or form a mental picture of the problem.

Most unplugged activities use a combination of kinesthetics, reading/writing, auditory and visual. The CS Education Group (2019) from the University of Canterbury in New Zealand has created an online repository of unplugged teaching material for primary and secondary education in collaboration with Google and Microsoft. These activities cover different subjects such as, sorting networks, binary numbers up to encryption, databases and error control. These activities use (combinations off) different techniques. Binary numbers are explained by writing and reading assignments and binary patterns are taught by coloring. More physical methods come in the form of drama and music. Physical objects are used to aid the teaching progress, for example, binary digits are explained by using cards that represent bits.

To get an impression of unplugged activities, several existing activities will be described below. A very simple activity, aimed at children in primary school, involving decomposition, requires students to break down a problem, find the solution and write down the necessary steps. This in turn creates some sort of algorithm for solving the problem. This shows students that simple and complex tasks can be broken down in simple instruction that when followed in the correct order accomplish a goal.

For general programming concepts there already are a wide variety of existing activities, in this next part the currently available activities for functions with parameters will be discussed.

Having a look at the currently available unplugged activities will help the design of our own activity. An excellent example is songwriting with parameters (Code.org, 2018-2019). This activity teaches students why combing chunks of code into functions is useful. The activity will have students locate repeating chorus in popular lyrics such as “Old MacDonald Had a Farm”, where the chorus changes slightly from verse to verse. The entire chorus will be classified as a function and the changing words will be passed as a parameter to alter the chorus.

Variables are an important part to parameter passing and many activities exist to teach the inner workings of variables. These activities can be integrated in the activity that will be developed as part of this study. An interesting concept can be found in the box variable activity (Queen Mary, University of London, 2014). Students holding boxes represent variables, and the exercise physically demonstrates the copy of values, and how storing new values destroys old values.

This example is lacking in explaining the return values that are commonly part of functions with parameters. Currently there aren't many other activities available that focus on this part of the curriculum.

2.3 Learning Functions with parameters

The previous chapter focused on the more general side of programming. In this chapter the specifics of functions with parameters will be discussed. Parameters passing and functions are considered a difficult subject, and it is closely related to other programming aspects, such as variables and storage. Using literature, we will emphasize the conceptual difficulties in this subject. This subject has specific misconceptions that are not found in other programming subjects, these misconceptions combined with other methods can be useful for teaching and designing new methods.

2.3.1 Conceptual difficulties

Two older but highly relevant works on parameter passing where carried out by Fleury (1991) and Madison and Gifford (1997). They both examined student's knowledge related to parameter passing in Pascal. Fleury interviewed students enrolled in an introductory programming course and asked them to predict if given programs worked correctly. Fleury started off with giving the students a functioning program and changed each new version just enough to only address one issue. Fleury pointed out the differences between the versions to students to decrease the overall interview time, this could potentially interfere in the students thinking process. Madison and Gifford used a variety of data sources such as observations of classroom instruction, semi-structured interviews and document examination. Fleury (1991) and Madison and Gifford (1997) both did research in the parameter passing field by having students solve programming exercises. Research into misconceptions and difficulties using unplugged activities as exercises hasn't specifically been done before.

A more recent analysis of data collected from a program simulation tool also found misconceptions related to functions and parameters (Sirkia, 2012). The research analyzed log files collected by a program simulation tool used in a programming course to find out what the most common errors where and the causes of these errors. Sirkia notes that errors can be causes by the user interface, simulation or because of a student's misconception, and includes in the discussion a comparison with previous literature.

2.3.2 Misconceptions

The previous mentioned research found that novices have many misconceptions regarding this concept. Having a deep understanding of these allows the creation of curriculum that is better suited at addressing these misconceptions.

Sorva catalogued known misconceptions in his PhD thesis, the related misconceptions will be used for this research. In appendix B, a summary with the most related misconceptions can be found based on the work of Sirkia (2012). These misconceptions related to functions and parameters will be explained here.

One of the most basic misconceptions found by Roy Pea (1986) is that novices think that the computer knows the intentions of the programmer and acts on that. Computer do not (yet) have the ability to think and reason about the meaning of code and will not deduce the intention of the programmer.

Passing parameters usually involves variables, a common misconception found by Du Boulay (1986) is that students think that a variable can store multiple values at the same time or that it can remembers 'old' values. This common misconception about variable assignment sometimes occurs in the case of two consecutive assignment statements, such as x = 5; x = 7. Some novices think that the variable can hold these multiple values simultaneously (Hermans, Swidan, Aivaloglou, & Smit, 2018). A study that compered the usage of metaphors found that explaining a variable as a box or as a label generates different results. Half of the participants received a programming lessons where a variable was explained as a box, the other half received the explanation that a variable is similar to a label. The results show that when it comes to simple questions regarding variables the box group performed better, but when faced with questions that involves two consecutive assignments, the label group performed way better. Another variable related misconception is that an assignment moves the value from a variable to another (du Boulay, 1986).

Storage related, students commonly forget to store the return value of a function (Hristova, Misra, Rutter, & Mercuri, 2003). This indicated a deep misconception about where values are stored and how these can be used later on. This misunderstanding is also acknowledged by (Ragonis & Ben-Ari, 2005).

In the analysis of log files collected by UUhistle, a program simulation tool used in a programming course, more misconception related to functions and parameters where found (Sirkia, 2012), some similar to the previous mentioned misconceptions. Some misconceptions Sirkia found are caused by the functionality and design of the tool. Executing functions instead of defining it, is one of the major errors. This seems to be caused by not understanding what happens in the parsing phase. As mentioned before the results also indicated that students forget to store the return value. He also notes that some students are successful in passing parameters to a function but simply store the finish value in a variable instead of returning it.

2.3.3 Implications for teaching

In the previous chapter about teaching it was mentioned that to effectively teach programming, especially novices, a combination of techniques and skills in necessary. For the specific subject of functions with parameters most general techniques can be applied, from a misconception's standpoint, teacher should dive in the specific problems of this subject to prevent them in their students.

3. Aim of the study

Computer programming can be a difficult process to master considering the amount of skills that is required. The last decades research has mostly been conducted with the goal of identifying programming difficulties or misconceptions in computer programming (du Boulay, 1986) (Sirkia, 2012) (Fleury A. E., 2000). The information in this chapter was mostly discussed previously, but is reiterated to emphasize the importance of this study's aim. Research has concluded that some programming subjects and concepts are harder than others, one of these subjects is functions and parameter passing. The importance of parameter passing and the appropriate use of functions shows that students understand abstraction, an important goal in many computer science curriculums.

Studies from the last decade mostly agree on that much remains to be done in this topic. Parameter passing is considered as a threshold concepts and making threshold concepts more useful in teaching and curriculum design is still an unexplored area (Sanders & McCartney, 2016). Furthermore, identifying difficulties and misconceptions becomes important because teacher that known these problems have larger classroom gains (Sadler, 2013). Grover and Pea (2013) describe that developing more theoretical and practical understanding of computation competencies in children and how to address difficulties is a subject that should be further explored.

An earlier mentioned approach that has become quite popular is the CT unplugged method. This approach uses activities that do not make use of a computer. The low-tech approach works well for beginner audiences because it is accessible for those with little technical background (Curzon, 2009). This also fits in the belief that students learn better when the curriculum makes use of 'fun' activities and tangible objects. Research has been conducted on the topic of functions with parameters but none of these studies used such kinesthetic activities when teaching this subject. This differentiates this study from previous research.

The aim of this study it to determine the difficulties that students face when learning functions with parameters such that we can address these difficulties when teaching. Not only focusing on the technique but also on the mechanism behind the concept. All past research used programming languages in some form for collecting data. Some of the results can be attributed by deficiencies in the language or tool. These studies also left the most important task of overcoming the difficulties and providing methods for teaching untouched. In chapter 2.2.5 past unplugged research is described, noticeable is that none of these studies looked into the actual effectiveness of unplugged. They mostly tried to identify an increase in interested or a change in view on computer science when learning with unplugged activities.

This main research questions that will be answered by this study is:

- Which obstacles and difficulties are important when learning the programming subject of functions with parameters?

- What are features of a kinesthetic learning environment that support teaching functions with parameters?
- What are similarities and differences between teaching using unplugged and plugged methods?

4. Methods

To achieve the aim of this study and answer the main research question, an education design research was conducted. By designing and researching an unplugged activity at the same time, new knowledge was discovered along with the creation of an activity.

4.1 Design

An unplugged activity was created using the design methods, misconceptions and areas of interest found in the literature as building stones. Determining which aspects of unplugged activities are most useful and should be included in our activity is a project by itself. Before the actual process of designing the activity started the learning objectives were set. Based on literature and previously discovered misconceptions the important areas were established and incorporated into the following general learning objectives:

- After the activity participants will be able to explain to others the practical usage of functions using real-world examples.
- After the activity participants will be able to write functions with parameters.
- After the activity participants will be able to explain the use and details of functions with parameters including the return.

As described earlier in chapter 2.2.4, previous research on activity design resulted in the CS Unplugged design pattern (Nishida, et al., 2009). This paper analyzed the structure of different unplugged activities and identified important elements. The resulting design pattern was used as a guideline for the creation of our activity. The design pattern contains 7 steps for constructing an exercise, with iteration between these steps. The design of the activity and the choices made will be described according to these seven steps. **Step one** & **two** consist of choosing a concept and identifying the key elements. The concept is, as anticipated, functions with parameters.

Following the second step of the unplugged design pattern, an inventory of the important characteristics or key elements was made based on literature. In collaboration with a computer science teacher and researcher the key elements were discussed and expanded. The CS teacher also provided insights on topics that students struggle with based on her experience in the classroom. Based on the recommendation of the CS teacher and of known misconceptions in this area it was decided that the topics return and variables would receive extra attention. Previous research by Sorva & Sirkia (2012) identified these topics as difficult for novices.

After identifying the key elements different activities were considered and elaborated. The proposed activities were discussed with a CS teacher to ensure feasibility. This process repeated itself several times until a challenging activity was created, see appendix C for the activity. This **third step** required many hours, which started with considering an appropriate theme for the activity. Getting students interested and actively participating in lectures can be challenging, this is even more challenging when the activity is boring or childish.

The goal of the **fourth step** is turning the activity into a challenge. An existing activity for explaining functions called Songwriting Unplugged, uses a song with an easy identifiable chorus as the starting point for creating functions. This is appropriate for students in primary school but might be boring for secondary school students. These factors influenced the creation of the activity. Researching the existing unplugged activity and finding out which elements are used for secondary education allowed the creation of a better activity.

The unplugged design pattern emphasizes that the difficult part of designing and activity is in making the correct part of the activity challenging. The activity should be challenging in nature but at the correct level, preferably at the concept level and not in difficult calculations or remembering. If the activity requires participants to perform difficult calculation than the challenge is mathematical, which is not the intended in this case. **Step five & six** of the design pattern are ‘evaluate’ and ‘refine’. Before using the activity for actual data collection, the activity will be tested on a small number of adult participants. The conscious choice was made to initially test the activity on adults, adults possess a bigger vocabulary which allows them to provide better feedback on the activity. The feedback will be used to improve the activity. During the data collection part, the researcher might find small improvement for the activity. The evaluation and refinement process will never be completely finished.

The developed activity consists of two parts, the first part contains a general introduction, the second part consists of playing the actual unplugged activity. The general introduction introduced the concept of functions to the participants. Explicitly stating the relation of the concept to real-life implementations helps participants make the connection. At the end of the activity the teacher refers back to the concept and states the relation between components of the activity to parts of the concept, this completes the so-called semantic wave. As stated earlier, semantic waves may be a key to development by enabling recontextualization (Maton, 2013). The actual outcome of this design is described in the results.

4.2 Research

Education design research combines the design, in this case of an unplugged activity while simultaneously researching the data that was collected by testing the activity. The activity was designed to contribute to unplugged methods and to gather data from participants. Testing the activity on students that represent the intended audience allowed the improvement of the activity before finalizing. This is also an important step in the unplugged design pattern.

4.2.1 Participants

The participants for this research are composed of students from a high school local to the researcher. In collaboration with the schools' computer science teacher, two classes were chosen. The students are around the age of 14 and are following an introductory programming course using Python as the programming language. Each class is made up of approximately 20 students with an almost even mix of males and females. Both classes follow the same curriculum and haven't been introduced to the subject of functions and parameter passing. Starting off with students that have little experience with our research subject is important because the researcher doesn't have to account for previous knowledge. Of the two classes, one class was given the 'standard' lecture by their own teacher using a PowerPoint presentation showing code samples and finishing with programming exercises. A normal lecture takes around 50 minutes, and normally starts with an introduction and examples and finishes with the students making exercises while being supported by the teacher. The second class was solely taught using the unplugged activity. Using two classes allows us to make a comparison between the unplugged and plugged teaching methods.

After the lecture the participants filled in a questionnaire; no personal data was collected in this part. Secondly, the participants were asked to be part of an additional interview. From each class two participants volunteered for the interview. The researcher provided participants taking part in the interview with consent forms one week beforehand. Each participant submitted the consent forms and participants under the age of 16 also received permission from their legal guardian.

4.2.2 Data collection

Data was collected during and after the activity. All participants were administered a written test and questionnaire and a selection of participants was interviewed.

Test and questionnaire

All participants completed a written test directly after the plugged or unplugged lesson, see Appendix D for the questions included in the test. The online survey software Qualtrics was used to administer the test. Qualtrics allows the easy creation of questionnaires with a large array of possibilities. The test was written in Dutch, the participants are all native Dutch speakers and English would add some difficulty for the participants. The test consists of 6 questions, four questions related to 3 code examples and two open questions.

The open questions should provide qualitative data about the application of functions and the usefulness of teaching programming unplugged. Participants are asked when and why functions are used and if they think learning computer science using unplugged methods (without a computer) is useful. It will be interesting to see whether participants that participated in the unplugged activity have different feelings towards unplugged compared to the group that only followed the plugged lesson.

The questions related to examples focused primarily on the key elements, these are: return, scope and the difference between print and return. The examples were written in Python by the researcher, Python is familiar to the participants.

The first program allowed us to see if the participant could trace a value when a function was called and if they are able to explain what happens at every step during execution. This should provide data about the function call, passing of parameters, calculation and return. The program contains a function definition and a main program, the definition takes two values as input, multiplies these values and returns the result. The main program defines two variables, calls the function passing the variables and printing the result. The participants are asked to write down the result and explain what happens at each line. Asking the participants to write down what happens step by step allows us to see if and when something goes wrong. Only requiring participants to write down the answer would not provide data about why participants are able to come up with a correct or incorrect answer.

```python
# Functie definitie
def berekenen(lengte, breedte):
        m2 = lengte * breedte
        return m2

#Hoofdprogramma en variabelen
afstand1 = 2
afstand2 = 5
print ( berekenen( afstand1 , afstand2 ) )
```

Table 1: Test, first program

The second program is meant to challenge the participants, they are provided with a small program and are asked to explain why it doesn't work. The program contains a function definition and a main program, the definition is the same as the previous program and takes two values as input, multiplies them and returns the result. The main program defines two variables, calls the function passing the variables but doesn't store the result. It then tries to print a non-existent variable to looks like the return. This program requires participants to break down the code in small parts, analyze and make relations. The program fails to store the return value and tries to print a variable from the wrong scope. Previous research from Sorva & Sirkia (2012) identified these two issues as common misconceptions, it would be interesting to see if participants show less signs of these misconceptions.

```python
# Functie definitie
def berekenen(lengte, breedte):
        m2 = lengte * breedte
        return m2

#Hoofdprogramma en variabelen

```

```
afstand1 = 2
afstand2 = 5
berekenen( afstand1 , afstand2 )
print ( m2 )
```

Table 2 Test, second program

The third and last program focusses on print and return. Research by Sorva (2012) and Sirkia (2012) indicates that there are many misconceptions related to the print and return. Students get confused about where the return values go and the difference between assigning and returning values. This exercise seems fairly simple, we hope this provides students with the opportunity to better explain the difference between line 2 and 3.

```
1  #Hoofdprogramma
2  print ( 13 )
3  return 13
```

Table 3 Test, third program

During the activity the researcher was busy explaining and executing the activity. For this reason, another teacher was present to write down any interesting observation. These notes will later be used to improve the activity.

Interview

Because of time constraints and willingness only two participants from each group were interviewed. This interview follows a think-aloud protocol and is similar in fashion to the written test, but allows the participants more time to expand on their answers and provides the interviewer with the possibility to ask questions, see Appendix E for the interview template. Recording were made of the interviews. The interview is scheduled to take around 20 minutes and allows the interviewer to ask questions when the participants expresses difficulty or shows interesting sings.

During the interview participants were asked to think-aloud while answering a series of questions about four small code programs. Previous studies have shown that a common method to gain better understanding in the thought process is to use a think-aloud protocol. Analyzing the programs that student write is not enough because studies have shown that students are able to create working programs without understanding the concepts.

The first program is formed based on a known misconception. Previous research indicates that students struggle with the naming of variables and parameters. Participants are asked whether there is a difference between the names of the variables and the names of the parameters in the definition. The think-aloud format allows the participants to express their thoughts more deeply and we hope this uncovers more information about this specific misconception.

```
getal1 = 5
getal2 = 8

def print_getallen ( a, b):
    print(a)
    print(b)

print_getallen( getal1, getal2)
```

Table 4 First program interview

The second program is large compared to the other examples. In this program a function is called from within another function. This is something the participants have not seen before; the interview allows the participants to explain how this program would work and why this would or wouldn't work. In this example participants have to trace the program; this allows us to exactly pinpoint where something goes wrong.

```
#FUNCTIE DEFINITIE
def bereken_inhoud( a, b, c ):
    d = a * b * c
    return d

def berekenen ( afstand1, afstand2, afstand3 ):
    print("M3: ")
    return bereken_inhoud(afstand1, afstand2, afstand3)

#HOOFDPROGRAMMA MET AANROEP VAN FUNCTIE
lengte1 = 3
lengte2 = 4
lengte3 = 2
totaal_inhoud = berekenen( lengte1, lengte2, lengte3 )
print( totaal_inhoud )
```

Table 5 Second program interview

The fourth program is formed around the scope of variables. Students struggle with scoping issues and in this example, they are asked to write down and explain which name is printed when the function is called.

```
a = "Mart"
#FUNCTIE DEFINITIE
def aanpassen( naam ):
    a = "Renske"
#HOOFDPROGRAMMA
aanpassen( a )
print (a)
```

Table 6 Third program interview

The interview finishes with writing a small program based on a predefined specification. Participants were asked to write a program that is able to calculate the total square meters based on two values, length and width. The program should contain a function that handles the calculation. The program should be executed with two self-determined values and print the result. Being able to correctly transform a specification into a program shows that a participant is able to go from abstract to concrete, which shows a higher level of understanding than for example, explaining a concept.

4.2.3 Data analysis

During analysis we initially focused on the use of the key elements by participants, the level of understanding, if students are able to make relations between concepts and if there are new or interesting misconceptions. The use of key elements was simply done by scanning the data for the key elements composed earlier. To see whether participants show understanding the SOLO taxonomy was used to classify the data. As described in chapter 2.2.5, the SOLO model describes 5 levels of understanding. Looking at the answers from participants, we were able to see if students only focus on one aspect (Uni-structural, the second lowest level of understanding) or if they are able to make relations with the different concepts (relational, the second highest level).

Recordings of the interview were transcribed and imported along with the written tests into ATLAS.ti, a qualitative analysis program. Analysis started out with carefully examining all the data, this gave an initial impression of what was collected. After the first impression, the data was again examined and classified into not relevant, relevant for context and relevant for the research question. The segments that provide context are useful to describe the research and provide information such as the total number of participants and which group someone belongs to. The segments that were relevant for the actual research questions started off with open coding.

For analysis, a combination off the deductive and inductive approach was used. In the literature phase several important concepts related to functions with parameters and programming were discussed, these concepts were used to code the data. These concepts, which were also the main focus of the test and questionnaire are: return, scope, print and return. Coding for the use of concepts allowed the researcher to see if one groups uses a specific concept more in comparison to the other group. Also, coding initially for the used concepts, the researcher was provided with the opportunity to diver deeper into the data during the next coding iterations.

In the literature phase a list of programming misconceptions was constructed based on previous work. This list allowed the researcher to code for known misconception and identify new misconceptions.

More inductively, the other open codes were primarily chosen based on the interpretation of the researcher. The lack of previous research on this subject required us to observe new categories and themes in the data. However, it is difficult to remain open minded because of the literature review.

Analysis continued by looking for patterns and connections in segments. Participants answered questions in varying difficulties and patterns might emerge within several answers. Participants might answer incorrect when questions become too difficult or the other way around.

Test and questionnaire

For the test and interview the general analysis described above was used. Some specific methods were applied for the test as for the analysis of the interview. These specifics are discussed below.

18 participants filled in the test, 11 from the plugged group and 7 from the unplugged group. The results were analysis both qualitatively and quantitatively. Using the initial coding phases as a starting point most data was coded. Specifically looking for misconceptions and correct and incorrect answers, these numbers will later be used to compare the groups quantitatively. The amount of correct and incorrect answers for the test will be used for a comparison. This allows us to see whether one groups is better at answering specific question than the other group. This could prove strong and weaknesses of unplugged and plugged methods.

Answers from individual participants will be compared to see if their vocabulary changes when questions get more difficult.

Interview

2 participants from each group were interviewed. Each interviewed took around 20 minutes and the recordings of the interviews were transcribed. The interviews were also coded using the general approach, but because the interview allowed participants to further indulge in their answer compared to the test, the analysis required a qualitative focused approach. Answers of participants were carefully analyzed for understanding and relations between different subjects. The last questions during the interview had participants write a small program, they were provided with instruction on what the program should accomplish but not how this had to be coded. This allows us to see if participants and members of a particular group are capable of translating something abstract into a ‘real’ working program.

5. Results

This research combines the design of a new unplugged activity with analyzing the data collected from testing the activity. This results in two things, the actual activity and the results from the data. First the unplugged activity is discussed after which the results of the data analysis are reviewed.

5.1 Results of activity design

Following the design principles and guidelines discussed in chapter 4.1, an unplugged activity was created based on the 7 steps of the unplugged design pattern.

The **first** & **second step** consist of choosing a concept and identifying the key elements. This research focusses on the development of unplugged methods for functions and parameter passing, consequently this concept was used for the activity. They key elements and areas of interest of functions and parameters passing are the function definition, return of a function, global and local cope and variables. In collaboration with a computer science teacher and researcher the key elements were discussed and expanded. Feedback from a computer science teacher was extremely valuable, it provided a better insight in classroom dynamics and topics that students struggle with.

The **third step** of the design was to consider the kind of elements used in the activity. For the developed activity several types and elements were considered. The kind of games or puzzles and if and what kind of physical objects would be practical were extensively discussed. Literature recommended the use of tangible objects because these objects make the session memorable and more engaging (Curzon, 2009). This recommendation was implemented in the form of paper templates and signs that represent roles and functions. Online repositories with unplugged activities were investigated to gain information and improve our activity.

An important consideration for activities and also the **fourth step** was how do we keep all participants engaged. After discussing this aspect with a CS teacher, the activity would keep all participants engaged by forming small groups and having each group perform the activity standalone. The unplugged design pattern actually describes considering a team activity where students can engage in different roles (Nishida, et al., 2009). This advice was implemented and a team activity were students depend on each other was used. Working in teams allows communication between teammates about the concept and activity helping them express their thoughts better.

Step five & **six** of the design pattern are ‘evaluate’ and ‘refine’. To evaluate the activity and work out the major flaws the activity was tested on two participants that have no prior programming experience. This process allowed for some small refinements in the activity before it was used for the data collection part. The activity was improved based on the feedback of the initial test. After using the activity for data collection small improvements were found and implemented by the researcher. After discussing how the activity came to be based on the design guidelines, the description of the actual activity is given below.

The developed activity consists of two parts, the first part contains a general introduction, the second part consist of playing the actual unplugged activity. The general introduction introduces the concept of functions to participants, this is useful for students that have no experience with or knowledge about functions and parameter passing. Classes that have prior experience with this subject can skip the introduction and commence with the activity. Starting off with an introduction allowed the participants to get used to the concept and settle in. Using a PowerPoint presentation, the teacher gives a real-life example that all participants can relate to. For the real-life example, Tikkie, a popular payment method in the Netherlands, is used. Tikkie is used to request a payment from friends. For example, after a night out and paying for drinks, you send your friends a payment request (Tikkie) for €10. Or, a more appropriate example for primary and secondary school would be sending your friend a payment request after visiting the supermarket during recess. The teacher explains to the participants that it would be a massive hassle to make a complete payment request every time someone owes you money, and that by using a function with a parameter for the amount and description it can be simplified. Before showing an example function in code, the participants were asked to think about which information would be necessary for a payment request and what information would change or stay the same in every payment request. After writing down the answers of participants on the board, the teacher discusses them and shows a simple function definition of a payment request. After this general introduction, the second part, the unplugged activity starts.

The unplugged activity mimics an actual program by assigning participants the role of a function and utilizing these functions to generate an end result. The activity is formed around the idea of graduation. Someday, most students will receive a diploma of some form. The activity makes participants work together to create and diploma. The teacher starts off with telling the participants about all the different aspects of a diploma. The activity can be altered based on the particular characteristics and grading system used in a school, the school of the participants displays the name, profile and the average of three grades on their diploma. The teacher describes the values to the participants and tells the participants that making the diploma using a computer would be very efficient. The participants are divided into groups of three with the teacher representing the central computer. Each student in a group represents one of three functions, one for making the diploma (make_diploma), one for calculating the average (calc_average) and one for writing the information on the diploma (print_diploma). See figure 2 for a schematic overview of the interactions between the different roles.

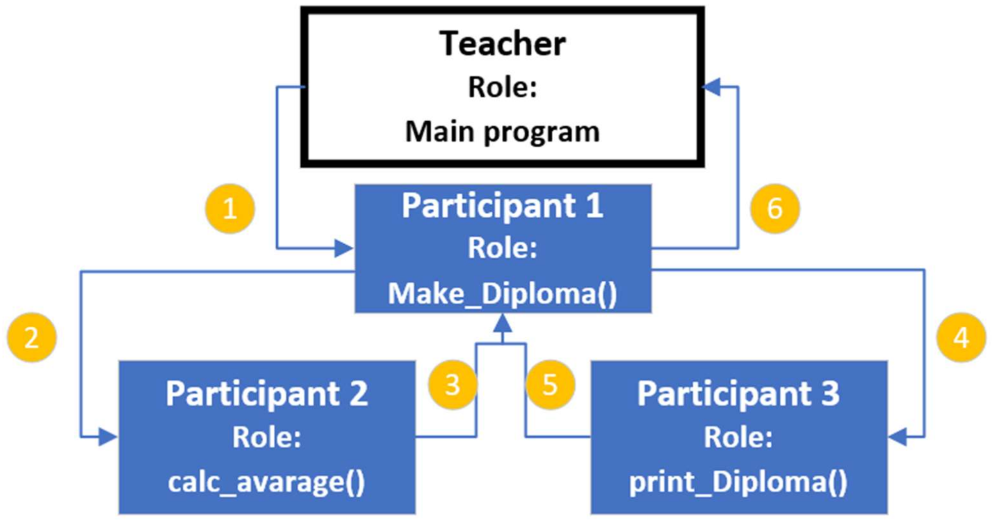

Figure 2 Schema unplugged activity

The main program asks the participant representing the make_diploma function to make a diploma and supplies the participants with five values: the name, profile, and the grades of three courses. The commands issues by the central computer (teacher) are represented by predefined forms, see appendix C. The participants playing the make_diploma function utilizes the other two participants in his groups to calculate the average of the three grades and to print this information on the diploma. The participant playing the make_diploma function has to do this in the correct order. After the group finished, they returned the diploma to the central computer. Participants were provided with templates for the diploma and for calling other functions and return information. These templates allowed students to see the storage of values and how they were being passed to other functions. After the activity, the teacher discusses the flow and how each function relates to one another. Using a whiteboard, the teacher, using input from the participants, writes down the formal definition of the functions performed in the activity. This shows the participants that it is possible to go from something concrete and real to a more abstract form completing the semantic wave.

5.2 Results of research

In the previous chapter we discussed the unplugged activity that is designed to teach the programming subject of functions with parameters. The activity was used to collected data from two classes at a local high school. After taking part in the activity all participants filled in a test and additionally a handful of participants were interviewed. The data was analyzed and the results will be described in the next section.

5.2.1 Test and questionnaire

The written test asked participants to answer questions and explain them step by step. This provided valuable information about the concepts used by students. By not only asking for an answer but having participants explain them, we should be able to better pinpoint were things get difficult. Data was initially coded using the key elements established earlier, these are *function definition*, *return*, *scope* and *variable related issues*. The results will be discussed according these elements.

Function definition

Analysis of the data shows that regarding simple functions, all participants are able to calculate the result of a simple program. But, explaining exactly how the value is calculated remains difficult for most participants. They are able to provide the correct answer, but offer no logical explanation. Participants describe in their answer the execution of functions, passing of variables and the calculation that occurs. All 18 participants were able to answer the first question correctly. There was no difference between the plugged and unplugged group. Participants from both groups are able to mental execute a simple program and calculate the value. Both groups struggle with explaining their answers, with no meaningful difference observed between the two groups.

Interestingly, participants that are able to identify the different components that construct a program still struggle with more difficult question that require deep analysis. For example, participant A, as seen below, describes the execution of the program, the passing of values to the function, and the result in his answer of the first exercise. Participant A was able to correctly answer this question which required the mental execution of a function.

> *Participant A: When line 9 is executed, the function 'calculation' is executed where for 'length' the value of variable 'distance1' is entered and for 'width' the value of variable 'distance2'.*

However, participant A failed to answered a more difficult question regarding identifying mistakes in a program. Participant A answered that the function hadn't been called. This shows a discrepancy. Participant A is able to identify a function call and compute the answer in the first question but unable to do a similar task in the second question. Confronted with a more difficult problem, Participant A answered incorrectly that the function call was missing.

> *Participant A: Line 9 causes an error because the function 'calculation' isn't called.*

Return

One of the main goals of the unplugged activity is to teach students why and how to apply functions. Often, functions are used inside a program to perform a specific task and use the outcome for something else. It is therefore important for beginners to understand the return considering its fundamental part in programming. The results show that most participant struggle with applying the return. Participants are able to explain what the return is, but applying and understanding the return remains difficult. Most participants explain the return as something that is used inside a function and as a way of giving a result back to the code.

> *Participant C: Return = The main program gets the value of the definition and can then do something with it.*
>
> *Participant A(Interview): I think that it is the result of a function and then the return is used so that you can use it to calculate with that number.*

Participants in this study also possess some previously discovered misconception. Such as thinking that using the return automatically stores the value 'somewhere' in the memory.

> *Participant J: Return stores it in memory.*
>
> *Participant B (Interview): If no return is used the definition is ignored.*

These examples are provided by the participants in the written test, it shows that participants are at least able to explain in simple terms what the return is supposed to do. Even tough participants are able to explain the use of the return, they are not able to apply the return or analyze code for return related mistakes.

Question 2 of the test asked participants to identify the mistakes in a small program. The program failed to store the return value and tried to use an undeclared variable. When we compare the answers between the plugged and unplugged group, we see a major difference. 50% of the plugged group answered correctly that the return was not stored and that the program therefore couldn't function. Only one participant from the unplugged group managed to answer this question correctly. Some participant answered that the problem was related to a scope issue which is also correct. None of the participants from the unplugged group indicated anything wrong about the scope. The scope issue is the result of the missing return and only participants from the plugged group could identify this.

Variable related issues

Variables are one of the most fundamental parts a novice should understand when learning how to program. Literature shows that novices have many misconceptions related to variables, and our data supports these misconceptions. In our data participants often describe that variables are undefined or not correctly associated with values. They also have trouble with passing parameters and get confused when variables are given arbitrary names.

Participants are able to correctly answer the first question and show that they are able to identify which variables are used inside the function. Exercise 1 and 2 both require participants to mentally compute the passing of parameters and describe the outcome, but with varying results. Even though they were able to provide the correct answer in the first exercise, when asked to explain, participants answered wrongly that variables are not defined or that it is not clear which variables are passed to the function. The second exercise received a lot of answer stating that the variables are not correctly assigned values.

> *Participant P: There are no values for length and width.*

> *Participant C: It is not clear that length and width should be 2 and 5.*

The data showed that participants not only have trouble with parameter passing and assigning values, but also with something that seems trivial. The naming of variables, this is something previous research did not indicate. Participants struggle when the name of the actual parameter is different than the name of the formal parameter. The data suggests that participants do not know the difference between actual and formal parameters and don't understand why they have different names.

> *Participant A (Interview): I don't know if a variable is the same as a parameter.*

> *Participant D (Interview): I don't understand why the variables have different names in the function call and function definition.*

The data shows no real difference between the plugged and unplugged group related to the naming of actual and formal parameters. Both groups struggle equally with this.

5.2.2 Interview

As part of the think-aloud session participants were asked to write a function based on a simple specification. Most participants were able to write a function with the specification in mind. None of the four participants managed to write it perfectly. Common mistakes identified in the written test are also found here, such as, failure to use return, not storing the return or printing variables from the wrong scope.

Participant A writes a correct function and makes a call to the function but forgets to store the return and print it. Participant B writes a correct function using the return but forgets to make a call to the function and print the result. Participant C writes a correct function with return and makes a call to the function but forgets to store the return. Participant C instead prints a variable from the wrong scope. Participant D writes a correct function but forgets to use the return. The function is called and the result is stored and printed. But because the function has no return, the programs doesn't compute.

Scope

Keeping track of the scope of different variables requires deep analyze of the code and breaking it into several parts to see what belongs where. Only remembering and understanding the scope is not always enough. In the think-aloud session the participants were introduced to a scope problem, 75% of the participants answered incorrectly. Sadly, only two participants were able to provide and explanation for their answer. Answer “Mart” is correct but none of the participants gave a correct explanation.

> *Participant A (Interview): They are both named A. But I think that. It seems more logical that that “Mart” is printed because the other (A=”renske”) hasn’t been read before executing the function.*

During the think-aloud session participants were asked about parameters. Most participants replied that because the exact name of the formal parameter is not used as the actual parameter the values will not be passed. They think that the name of the actual and formal parameter has to be the same. Some participants did answer correctly that naming is up to the programmer, but failed at providing a logical explanation. One participant even suggested that the computer decides the names for variables and as long as the computer knows what is doing that it will be fine.

> *Participant D (Interview): As long as the computer knows what it is doing, the computer can fill in names by himself and knows what he is talking about.*

6. Conclusion

With a combination of literature, research and consults an unplugged activity was created that incorporated many important aspects of the programming subject functions with parameters. The activity was tested on a high school class and data was collected from them and another programming class following normal (plugged) lectures. Analysis of the data gathered from the written test and interview provided confirmation of many previously discovered misconceptions, while also discovering new areas that are considered challenging and features that are important for teaching functions with parameters and programming in general. The data shows that both the plugged and unplugged group exhibit similar misconceptions and challenges, but at the same time show clear distinctions that can be attributed to both methods.

It seems that teaching in the more classic 'plugged' manner provides students with a better comprehension of scope, return and analyzing code. This can be explained by the fact that these students spend more time writing and looking at code during the lectures. Students taught by the unplugged method seem to be slightly better at making relations and seeing the bigger picture. Previous research (Hristova, Misra, Rutter, & Mercuri, 2003) found that students commonly forget to store the return value of a function. Ragonis & Ben-Ari (2005) also acknowledged this misconception. In our research we concluded the same misconception for both groups, with now clear contrast between the groups.

Most participants have no trouble with questions that focus on one aspect, this corresponds to the SOLO level, Uni-structural. Participants were able to successfully answer a question about a simple program, but when asked to analyze and dive deeper into the code of a more difficult program most students failed by providing wrong answers. The problem was similar to the first question but altered to included multiple aspects of functions with parameters. Here we see participants start to struggle. These more difficult questions correspond to the higher SOLO levels called multi-structural and relational. This concerns questions that require focusing on multiple aspects and integrating these into a whole picture to provide a logical answer.

Interestingly, the data shows that some participants are able to compute the correct answer in the first part of the questionnaire involving parameter passing. But later on, when asked to identify mistakes in a similar program, the participants answered wrongly. The answered that variables were not defined or that it was not clear which variables were passed to the function. This indicates that students struggle when different parts become integrated into a whole program.

The questionnaire shows another interesting result. 50% of the participants that followed the plugged lessons answered correctly that a program wouldn't function because a variable outside the scope is used. Only 1 participant from the unplugged lessons was able to answer this question correctly. All other unplugged participants answering illogical. Participants from the unplugged activity have less experience with code and applying the return, it seems that this translates into a lower score on this particular subject.

Furthermore, the results show that most participants do not know the difference between actual and formal parameters and struggle when they are assigned different names in a program. This issue is not limited to one group, both the plugged and unplugged group struggle equally with this issue.

Participants that have never experienced unplugged activities generally express more negative feeling towards unplugged. Participants with unplugged experience are positive about this method and mostly express unplugged as a method that should support teaching but not replace it. Participants from the unplugged lessons state that unplugged should be a contribution to 'real' programming exercises and that it helps to keep the lessons interesting and relatable to the real world.

To summarize, the goal of this research was to identify the obstacles and difficulties in learning the programming subject of functions with parameters while additionally trying to determine the features of a kinesthetic learning environment that supports teaching. This also allowed us to determine the similarities and differences in teaching using unplugged and plugged methods.

Student that are introduced to the subject for the first time experience many obstacles, such as, struggling with the return of a function, global and local scope and the naming of variables. Our data shows that students commonly forget to include or store the return. It seems that this is related to the storage of values and can be attributed to a wrong notional machine. This became apparent while analyzing the data gathered from the interview. A participant explained that because the computer knew what is was doing, he did not have to explicitly store the return value. Not indicated by previous research is the misconceptions students have about the naming of the formal and actual parameters. A large number of participants struggle with the names given to parameters. They have the misconception that the formal and actual parameter has to be called the same. This misconception should receive attention.

The development of an unplugged activity allowed us to determine features that could be important in a kinesthetic learning environment. When teaching students using unplugged methods, attention should be drawn to making the relation between abstract and concrete clear. Completing the semantic wave seems supportive of this. Using physical object brings greater excitement for students, this also allows them to make the link with real-life implementations.

Based on the data, similarities and differences between plugged and unplugged teaching were identified. It can be concluded that plugged and unplugged teaching are both effective tools for teaching programming concepts, but they should be applied for specific goals. Stating that unplugged is more or less effective than other methods cannot be done based on the limited number of participants in this research. However, unplugged teaching has a positive effect on the interest young students have in computer science which increases participation. We can establish that unplugged has a place in modern curriculum. Using SOLO's taxonomy, both groups can be classified as Uni-structural and multi-structural. The unplugged group does seem to have a higher level of understanding in the relations related aspect. Participants from the unplugged group seem slightly better at making relations between pieces of code and seeing the bigger picture. Participants taught by the plugged method are slightly better at code related exercises and scope related problems. For introducing the programming subject functions with parameters and getting an accurate notional machine an unplugged activity is great. It shows students the main picture we are trying to communicate while also having students think about the inner workings and relations. Plugged methods should be used later to go into detail and explain the specific programming related keywords. Performing this study on a larger scale should provide more evidence on the effect of unplugged and the specific use cases.

7. Discussion and further research

The purpose of this study was to find the obstacles and difficulties that come up when learning the programming subject of functions with parameters. We also looked for the features that supported a kinesthetic learning environment. The data shows that the previously discovered misconceptions are also emerging in the subject of functions and parameters. Students struggle with many of the common misconceptions related to computer programming. A new interesting misconception or difficulty that was not highlighted by previous research is the issue of parameter naming. Students from both groups struggle with the naming of the actual and formal parameters. When the actual and formal parameters have different names, students get confused. This is a topic that should receive more attention. This issue is related with other concepts such as scope and variables which are fundamental to programming.

When comparing the plugged participants with the unplugged participants a clear distinction can be seen. Participants from the plugged lesson are better in code related exercises and scoping related problems. Students from the unplugged do however seem to be better at making relations between pieces of code and seeing the entire picture. The general feeling towards unplugged is positive in the eyes of students that have experience with unplugged activities. Students that have not been introduced to unplugged methods have a more negative feeling towards this method. These results are important because unplugged is being integrated in the curriculum of countless nations and the impact should be investigated.

The generalizability of the results is limited due to the number of participants. To make a more general statement on the identified misconceptions and the effect of unplugged methods it should be compared with plugged methods in greater numbers. Comparing more classes, different age groups and with a variety in previous knowledge is advisable. The reliability of this data is impacted by the extra lesson given to two participants. Two participants from the unplugged group that took part in the interview received an extra plugged lesson on functions and parameters making the comparison between these participants not possible. The data from the interview was still useful for identifying difficulties but not for comparison with the plugged group.

The subject of functions with parameters contains many important programming concepts and the developed activity tried to include them all. By splitting the activity and focusing more on a specific concept the activity could be more focused and less confusing. Also, the names used for the function definition and variables in the written test could provoke a bias.

I would recommend using unplugged as an add-on to the normal way of teaching. It is extremely viable for teaching students the global picture and seeing relations. It is however not a substitute for coding exercises and ‘real’ programming. For teaching computational thinking skills in primary education this method seems to be useful. Many unplugged activities have been created in the past years but more research on the effectiveness of unplugged methods is still required.

Literature

Barbe, W. B., Swassing, R. H., & Michael N. Milone, J. (1979). *Teaching through modality strengths : concepts and practices.* Columbus, Ohio: Zaner-Bloser.

Barendsen, E., & Tolboom, J. (2016). *Advies examenprogramma informatica vwo-havo: inhoud en invoering.* Enschede: SLO.

Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2011). Computer Science Unplugged: school students doing real computing. *New Zealand Journal of Applied Computing and Information Technology*.

Biggs, J. B., & Collis, K. F. (1982). *Evaluating the quality of learning: the SOLO taxonomy (structure of the observed learning outcome.* New York: Academic Press.

Bloom, B., Engelhart, M. F., Hill, W., & Krathwohl, D. (1956). *Taxonomy of educational objectives: The classification of educational goals. Handbook 1: Cognitive domain.* New York: David McKay.

Brackmann, C. P., Robles, G., Moreno-León, J., Casali, A., Barone, D., & Román-González, M. (2017). Development of Computational Thinking Skills through Unplugged Activities in Primary School. *Proceedings of the 12th Workshop on Primary and Secondary Computing Education* (pp. 65-72). Nijmegen: ACM New York.

Butler, M., & Morgan, M. (2007). Learning challenges faced by novice programming students studying high level and low feedback concepts. *ascilite Singapore 2007.* Singapore.

Code.org. (2018-2019). *Computer Science Fundamentals.* Retrieved from curriculum.code.org: https://curriculum.code.org/csf-18/

Computer Science Education Research Group. (2019, April 14). *Computer Science without a computer*. Retrieved from CS Unplugged: https://csunplugged.org/en/

Curzon, P. (2009). Enthusing & Inspiring with Reusable Kinaesthetic Activities. *Innovation and Technology in Computer Science education*, 94-98.

Department for Education. (2013). *National curriculum in England: computing programmes of study.* London: Department for Education.

du Boulay, B. (1986). Some Difficulties of Learning to Program. *Journal of Educational Computing Research*, 57-73.

Feaster, Y., Segars, L., Wahba, S. K., & Hallstrom, J. O. (2011). Teaching CS unplugged in the high school (with limited success). *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education* (pp. 248-252). Darmstadt, Germany: ACM New York.

Fleury, A. E. (1991). Parameter Passing: The Rules the Students Construct. *SIGCSE Bulletin*, 283-286.

Fleury, A. E. (2000). Programming in Java: Student-Constructed Rules. *SIGCSE Bulletin*, 197-201.

Grover, S., & Pea., R. (2013). Computational Thinking in K–12: A Review of the State of the Field. *Educational Researcher, 42*(1), 38-43.

Guiffre, L. L. (2009). Computer science outreach in an elementary school. *Journal of Computing Sciences in colleges*, 118 - 124.

Haim Eshach, M. N. (2005). Should Science be Taught in Early Childhood? *Journal of Science Education and Technology*, 314-336.

Hermans, F., Swidan, A., Aivaloglou, E., & Smit, M. (2018). Thinking out of the Box: comparing metaphors for variables in programming education. *Workshop in Primary and Secondary Computing Education* (p. Arctile 8). Postdam: ACM New York.

Hristova, M., Misra, A., Rutter, M., & Mercuri, R. (2003). Identifying and Correcting Java Programming Errors for Introductory Computer Science Students. *SIGCSE Bulletin*, 153-156.

Kallia, M., & Sentance, S. (2017). Computing Teachers' Perspectives on Threshold Concepts: Functions and Procedural Abstraction. *12th Workshop on Primary and Secondary Computing Education.* Nijmegen, Netherlands.

Krathwohl, D. R. (2002). A Revision of Bloom's Taxonomy: An Overview. *Theory Into Practice*, 212-218.

Land, J. M. (2003). *Threshold Concepts and Troublesome Knowledge: Linkages to Ways of Thinking and Practising within the Disciplines.* Edinburgh: ETL Project, Universities of Edinburgh, Coventry and Durham.

Madison, S., & Gifford, J. (1997). *Parameter Passing: The Conceptions Novices Construct.*

Maton, K. (2013). Making semantic waves: A key to cumulative knowledge-building. *Linguistics and Education*, 8-22.

Mayer, R. E. (1981). The psychology of how novices learn computer programming. *ACM Computing Surveys*, 121-141.

Nishida, T., Kanemune, S., Idosaka, Y., Namiki, M., Bell, T., & Kuno, Y. (2009). A CS Unplugged Design Pattern. *SIGCSE 2009* (pp. 231-235). Chattanooga: ACM New Yor.

NRC. (1996). *National Science Education Standards No.* The National Research Council (1996).

Orni Meerbaum-Salant, M. A.-A. (2013). Learning computer science concepts with Scratch. *Computer Science Education* , 23:3, 239-264,.

Pea, R., Kurland, D., Clement, C., & Mawby, R. (1986). A Study of the Development of Programming Ability and Thinking Skills in High School Students. *Journal of Educational Computing Research*, 429-458.

Piteira, M., & Costa, C. (2013). Learning computer programming: study of difficulties in learning programming. *Proceedings of the 2013 International Conference on Information Systems and Design of Communication.* Lisboa, Portugal.

Queen Mary, University of London. (2014, February). *The box variable activity.* Retrieved from Teaching London Computing: https://teachinglondoncomputing.files.wordpress.com/2014/02/activity-boxvariables.pdf

Ragonis, N., & Ben-Ari, M. (2005). A Long-Term Investigation of the Comprehension of OOP Concepts by Novices. *Computer Science Education*, 203-221.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and Teaching Programming: A Review and Discussion. *Computer Science Education*, 137-172.

Sadler, P. M.-S. (2013). The Influence of Teachers' Knowledge on Student Learning in Middle School Physical Science Classrooms. *American Educational Research Journal*, 1020-1049.

Sanders, K., & McCartney, R. (2016). Threshold concepts in computing: past, present, and future. *16th Koli Calling International Conference on Computing Education Research.* Koli, Finland.

Sentance, S., Barendsen, E., & Schulte, C. (2018). *Computer Science Education: Perspectives on teaching and learning in school.* Londen: Bloomsbury.

Sirkia, T. (2012). *Recognizing Programming Misconceptions – An analysis of the data collected from.* Aalto: Aalto University.

Sorva, J. (2012). *Visual Program Simulation in Introductory Programming Education.* Aalto University.

Sorva, J. (2013). Notional machines and introductory programming education. *ACM Transactions on Computing Education*, Volume 13, Isssue 2, Article 8.

Sorva, J., & Sirkia, T. (2012). Exploring programming misconceptions: an analysis of student mistakes in visual program simulation exercises. *Koli Calling 12'* (pp. 19-28). Koli: ACM.

Stefania Bocconi, A. C. (2016). *Developing Computational Thinking in Compulsory Education.* Publications Office of the European Union.

Taub, R., Ben-Ari, M., & Armoni, M. (2009). The effect of CS unplugged on middle-school students' views of CS. *conference on Innovation and technology in computer science education* (pp. 99-103). Paris, France: ACM New York.

The Royal Society. (2012). *Shut down or restart? The way forward for computing in UK schools.* London: The Royal Society.

Thies, R., & Vahrenhold, J. (2013). On Plugging "Unplugged" into CS Classes. *Proceeding of the 44th ACM technical symposium on Computer science education* (pp. 365-370). Denver, Colorado: ACM New York.

Webb, M., Niki, D., Bell, T., Katz, Y. J., Reynolds, N., Chambers, D. P., & Sysło, M. M. (2017). Computer science in K-12 school curricula of the 2lst century: Why, what and when? *Education and Information Technologies*, 445-468.

Glossary

term	definition
computer science (CS)	Study of principles and use of computers
computational thinking (CT)	Solving problems using concepts grounded in computer science
unplugged	Without computers. Unplugged learning is learning by the use of games, activities, reading/writing all without using electronics.
programming: function (definition)	Part of a computer program used as a procedure or routine. The definition is the blueprint of the function.
return	The return can be used as a programming statement that tells the program to end the function and send back something to the main program
variable	A place to store information.
local scope	A variable that is limited in use, it can only be used in specific places in a program.
global scope	A variable that can be used in the entire program.
python	Programming language.
think-aloud	The think-aloud protocol has participants thinking aloud when they are performing a task. Participants can say anything that comes to mind, this gives the researcher information that would normally not be written down as part of the answer.
Actual parameter	The values or variables that are passed to the function when a function is called.
Formal parameter	The variables that are defined in the function definition, these variables receive the actual parameters.

Definitions gathered from https://www.lexico.com/ and/or from the researcher.

Appendix A - Misconception list Sorva (2012)

Number	Description	Source
1.	The computer knows the intention of the program or of a piece of code, and acts accordingly	(Pea, Kurland, Clement, & Mawby, 1986)
2.	The computer is able to deduce the intention of the programmer	(Pea, Kurland, Clement, & Mawby, 1986)
3.	A variable can hold multiple values at a time / ‘remembers’ old values.	(du Boulay, 1986)
4.	Assignment moves a value from a variable to another.	(du Boulay, 1986)
5.	A return values does not need to be stored (even if one needs it later).	(Hristova, Misra, Rutter, & Mercuri, 2003)
6.	A method can be invoked only once	(Ragonis & Ben-Ari, 2005)
7.	Numbers or numeric constants are the only appropriate actual parameters corresponding to integer formal parameters	(Fleury A. E., 2000)
8.	Difficulties distinguishing between actual and formal parameters. Confusion over where parameter values come from.	(Hristova, Misra, Rutter, & Mercuri, 2003)
9.	Confusion over where return values go.	(Ragonis & Ben-Ari, 2005)
10.	Parameter passing forms direct name-based procedure-to procedure links between variables with the same name (in call and signature).	(Madison & Gifford, 1997)
11.	Parameter passing forms direct procedure-to-procedure links between variables with different names (in call and signature).	(Madison & Gifford, 1997)
12.	When the value of a global variable is changed in a procedure, the new value will not be available to the main program unless it is explicitly passed to it.	(Fleury A. E., 1991)
13.	Expressions (not their values) are passed as parameters.	(Sorva, Visual Program Simulation in Introductory Programming Education, 2012)

Appendix B - Misconception list Sirkia (2012)

- Executing function instead of defining it
- Constructing new function call instead of assigning return value
- Creating parameter value to wrong frame
- Creating new function call instead of passing parameters
- Trying to start function call before evaluating parameters
- Creating a variable for return value to wrong frame
- Assign return value back to variable instead of returning it

Appendix C – Lesson plan unplugged Activity

An unplugged activity for functions with parameters

Lesson overview

Functions with parameters are one of the most used structures in computer science. Without a deep understanding of these concepts, developing bigger programs becomes difficult. With this unplugged lesson, students that already have a limited understanding of programming will gain a deeper knowledge of functions and parameters.

Teaching Summary

Introduction – 5 minutes

- Explaining lesson goals
- Real-life example - Tikkie

Unplugged Activity: Making diploma's with parameters – 15 minutes

- Making diploma's

Practicing with Python – 20 minutes

- Let Students practice with Python, examples and exercises.

Assessment - 5 minutes

- Online test/questionnaire

Lesson Objectives

Students will:

- After the activity participants will be able to explain to others the practical usage of functions using real-world examples.
- After the activity participants will be able to write functions with parameters.
- After the activity participants will be able to explain the use and details of functions with parameters including the return.

Materials

- PowerPoint slides;
- Cards used in the activity;

1 Guide

Introduction – Tikkie Payment Request

This introduction is meant to explain what functions are and more importantly show the students that this is something that is used every day in our life. We will be using the example of "Tikkie", Tikkie is a mobile application that makes it extremely easy to request money from friend and relatives. Tikkie is now used in the Netherlands as the generic term when sending a payment request. Most major banks incorporated the features of "Tikkie" in their own applications.

Ask the students who has ever used a payment request and what steps did they have to take to make a payment request. Write down on the board the values they had to supply. The outcome of this is supposed to be the amount and the description. If the student come up with different solutions try to steer them in the right direction or adapt if their input is valuable.

So, we now have two arguments we need to create a payment request: Amount and description. Explain that in programming there is a thing called functions and that these functions are used to define things that happen more than once. We want to make our life as easy as possible.

What if we had to make functions for every amount possible. Too much code, this is why it is possible to supply a function with parameters. PaymentRequest(amount, description).

Fill in a payment request on the board (PaymentRequest(50, 'Lesgeven aan vwo')) and end the introduction.

2 Unplugged Activity: Making a Certificate/Diploma

Introduction

This activity will calculate the average of three grades and will print it on a certificate, being similar to something all students will experience in the last year of high school during the exam period. The average grade should be a whole number.

We are going to let students play in a theatrical style where they will be playing different roles that represent different parts of a function. The Python equivalence is:

```
#Functies beschrijving
def maakDiploma(naam, profiel, cijferCKV, cijferProfielw,
cijferMaatsc):
        combiCijfer =
bereken_combi_cijfer(cijferMaats,cijferPWD,cijferCKV)
        diploma = schrijf_op_diploma(naam, profiel, combiCijfer)
        return diploma

def berekenCombiCijfer(cijferMaatsc, cijferProfielw, cijferCKV):
        cijfer = round((cijferCKV + cijferProfielw + cijferMaatsc) / 3)
        return cijfer

def schrijfOpDiploma(naam, profiel, combicijfer):
        #schrijf met pep en papier
        return diploma
#Hoofdprogramma
maak_diploma("janneke", "EM" , 6 ,10, 8)
```

Students will be assigned roles but will not be told how to actually perform their function, the inner workings of the function are not that important as longs as it does what it was designed for. The entire class will be divided into groups of 3 people.

Roles:

Teacher: Computer calling the function

Student 1: Plays the maakDiploma() function

Student 2: Plays the berekenCombinatiecijfer() function

Student 3: Plays the schrijfOpDiploma function

Objects used:

- Cards that show the Role, Appendix A.
- Cards for writing down the values of parameters

Directions

1. Introduce activity to students
    a. Tell students they will be forming group and that each student will be playing a role (part of the programm)
    b. The goal of the activity is to interact with eachother to get to a final result
    c. Tell the students they have access to cards and pencils to write down different values
    d. The instructor will hand out the intial values.
2. Tell the students they will be playing a computer programm aimed at creating a diploma
    a. Form groups of 3 and explain each individual will be playing a role (part of a programm)

3 Practice

Introduction

Practicing immediately after the unplugged activity creates an opportunity for addressing misunderstanding and possibly for better recollection. This exercise can be tailored to the programming language the students are most familiar with, which in this case is Python. Students will come up with different solutions to calculate the average and name the function and variables differently but is should all be similar to the code shown on the previous page. This creates an excellent opportunity to see if student have understood functions and parameters and allows the teacher to address any issues with individual students.

The students will be shown a correct Python program that corresponds to the unplugged activity they just completed. The Python program will be expanded and explained step by step allowing time for questions.

After this explanation the students will be given the following example and are asked to write down the corresponding Python program.

Exercises:

Complete the following exercises from the online environment: 8.2.4, 8.2.5

Directions

1. Let the students know we are about to start practicing in the online environment.
    a. http://course.cs.ru.nl/pythonVO/ “Oefenruimte zonder Turtle”
2. Show students the Python code belonging to the unplugged activity
    a. Expand the function step by step
    b. Explain each part of the function
3. Explain exercise 1 and give the introduction
    a. Shown the requirements for the function on the board.
    b. Asks students to write down the program in Python
4. Students can start working on assignments 8.2.1 and 8.2.4 after completing the first exercise.
5. Walk around and help students fix mistakes.

maak_Diploma

bereken_Combi_Cijfer

schrijf_Op_Diploma

DIPLOMA

Naam: ....

Profiel: ....

CombinatieCijfer ....

DIPLOMA

Naam: ....

Profiel: ....

CombinatieCijfer ....

maakDiploma(“Janneke”, “Natuur en techniek”, 9 ,7 ,8)
maakDiploma(“Pietje”, “Natuur en techniek”, 5 ,6 ,7)

berekenCombiCijfer(______ ,______ , ______)
berekenCombiCijfer(______ ,______ , ______)

schrijfOpDiploma(___________ ,__________ , ______)
schrijfOpDiploma(___________ ,__________ , ______)

Appendix D – Unplugged Activity - Questionnaire

The online questionnaire will give students some code examples that integrated the misconceptions. This questionnaire should take more than 5 minutes to complete; this will allow some open questions and one/two exercises.

Open questions:

1. Waarvoor dient een functie?
2. Denk je dat oefeningen en lessen waarbij programmeren wordt uitgelegd zonder gebruik te maken van computers kan helpen bij het leren? Vertel ook welke aspecten je belangrijk vindt aan een goede les?

Online opdracht 1:

```
1   # Functie definitie
2   def berekenen(lengte, breedte):
3           m2 = lengte * breedte
4           return m2
5
6   #Hoofdprogramma en variabelen
7   afstand1 = 2
8   afstand2 = 5
9   print ( berekenen( afstand1 , afstand2 ) )
```

In deze afbeelding zie je een Python programma. Over dit programma worden enkele vragen gesteld. In het programma zie je een functie definitie en het hoofdprogramma met de aanroep naar de functie

1. Wat is de waarde van m2?
2. Beschrijf stapsgewijs wat er gebeurt op het moment dat regel 9 wordt uitgevoerd.

Online opdracht 2:

```
1   # Functie definitie
2   def berekenen(lengte, breedte):
3           m2 = lengte * breedte
4           return m2
5
6   #Hoofdprogramma en variabelen
7   afstand1 = 2
8   afstand2 = 5
9   berekenen( afstand1 , afstand2 )
10  print ( m2 )
```

1. Leg uit waarom bovenstaand programma een foutmelding veroorzaakt?

```
1  #Hoofdprogramma
2  print ( 13 )
3  return 13
```

1. Wat is het verschil tussen print() en return. In andere woorden: Wat is het verschil tussen regel 2 en regel 3 in bovenstaande afbeelding?

Appendix E – Unplugged Activity - Think-aloud form

Interview

1. Waarvoor dienen functies? Wat zijn parameters precies?
2. Kan jij me uitleggen wat je moet doen met de **return** van een functie?
3. Maakt het verschil tussen de benaming van de parameters in de functie definitie en in de aanroep uit?

```
getal1 = 5
getal2 = 8

def print_getallen ( a, b):
    print(a)
    print(b)

print_getallen( getal1, getal2)
```

4. Vertel stapsgewijs wat er gebeurt wanneer het hoofdprogramma wordt uitgevoerd?

```
#FUNCTIE DEFINITIE
def bereken_inhoud( a, b, c ):
    d = a * b * c
    return d

def berekenen ( afstand1, afstand2, afstand3 ):
    print("M3: ")
    return bereken_inhoud(afstand1, afstand2, afstand3)

#HOOFDPROGRAMMA MET AANROEP VAN FUNCTIE
lengte1 = 3
lengte2 = 4
lengte3 = 2
totaal_inhoud = berekenen( lengte1, lengte2, lengte3 )
print( totaal_inhoud )
```

5. Wat wordt er geprint na regel 10?

```
1   # Functie definitie
2   def berekenen(lengte, breedte):
3           m2 = lengte * breedte
4           return m2
5
6   #Hoofdprogramma en variabelen
7   afstand1 = 2
8   afstand2 = 5
9   berekenen( afstand1 , afstand2 )
10  print ( m2 )
```

6. Welke naam wordt er geprint? Waarom denk je dit?

```
a = "Mart"
#FUNCTIE DEFINITIE
def aanpassen( naam ):
    a = "Renske"
#HOOFDPROGRAMMA
aanpassen( a )
print (a)
```

7. We hebben een functie nodig die op basis van de **lengte** en de **hoogte** het aantal vierkante meter berekend. De functie moet op zo'n manier geschreven zijn dat in het hoofdprogramma de berekende waarde geprint kan worden. Vertel hardop hoe je dit zou doen.

8. Wat heb je geleerd in les over functies met parameters? Wat is de kern?