Universiteit Leiden
The Netherlands

Opleiding Informatica & Economie

Reading Strategy and Reading Focus of Novice Programmers: An Exploratory Study

Jie Wu
s2315211

Supervisors:
Vivian van der Werf & Efthimia Aivaloglou

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

11/07/2021

Abstract

Motivation. Code reading is an important programming skill since it influences other programming abilities such as code writing and code comprehension. To gain more insights into the code reading process of programmers and the reasoning behind it, previous research uses the eye-tracking method and the think-aloud method. **Objective.** This thesis aims to give insights into how the reading strategies for a natural language text are related to the reading strategies for a source code among novice programmers and what they focus on based on an example code. **Method.** We approach this using semi-structured think-aloud interviews. Ten participants were interviewed about 1) their common reading strategies for a natural language text, 2) their common reading strategies for a source code, 3) the main difference between reading a text and a code, 4) their reading focus based on an example code, 5) the part of the example code they find the most important, and 6) the part of the example code they find the most difficult. **Results.** We found that majority of the participants mentioned that they use a skimming strategy to read a natural language text in order to get a general idea of the text. Furthermore, the most frequently mentioned strategy to read a source code is to follow the code's execution. Moreover, according to most of the participants, the main differences between reading a text and reading a code are the reading order and reading goal. Regarding the reading focus, the majority of the participants seem to focus on the code behavior. **Discussion.** Our results show that the reading strategies for a natural language text are only partially applicable in the reading strategies for a source code. Furthermore, our results also show that the participants focused more on the parts or elements of the example code that are more relevant to the comprehension or execution of the code.

Contents

1 Introduction

In many introductory programming courses, novice programmers are taught how to write code, but they are rarely taught how to read code. Code reading is often seen as an ability that a programmer automatically develops during the code writing process. However, according to several studies [CLT11, LAF+04, S+12], the development of this code reading skill is not as good as programming educators expected. Therefore, it might be necessary to teach students this skill explicitly as some studies [LAF+04, LFT09, LWRL08, PRW07] have concluded that code reading influences other programming skills such as code writing. In addition, it also forms an essential basis for the code comprehension process of programmers [BS13].

To gain more insights into the code reading process of programmers, eye-tracking methods are often used. Using this method, programmers' eye movements and eye fixations during code reading can be observed. These eye movements and eye fixations indicate programmers' reading order and reading focus. Some studies [BT06, BSB11, BSS+14, Gué06, TFSL14] also compared the code reading process to the code comprehension of programmers. Furthermore, Busjahn et al. [BBB+15] stated in their study that reading a source code is not the same as reading a natural language text. However, novice programmers tend to read a source code more like a natural language text, while experienced programmers tend to follow the execution flow of the program [PSA20]. Therefore, the strategies used by novice programmers to read a natural language text might have an impact on their code reading processes.

The motivation behind this thesis is to get a deeper understanding of the code reading process of novice programmers focusing on 1) the perspectives of novice programmers on the common reading strategies for a natural language text and the common reading strategies for a source code, and 2) the reading focus during the code reading process according to novice programmers. Therefore two main research questions (RQ) are addressed with corresponding sub-questions which are relevant to answer the research questions:

RQ1. *How are the common reading strategies for a natural language text related to the common reading strategies for a source code among novice programmers?*

- What are the common reading strategies for a natural language text according to novice programmers?
- What are the common reading strategies for a source code according to novice programmers?
- What are the main differences between reading a natural language text and reading a source code according to novice programmers?

RQ2. *What do novice programmers focus on during reading, based on an example code?*

- Which programming concepts/elements of the code do novice programmers focus on?
- Which parts of the code are the most important according to novice programmers?
- Which parts of the code are the most difficult according to novice programmers?

2 Background and Related Work

2.1 Relationship between programming abilities

In general, it is assumed that there exists a relationship between the different programming abilities, such as code tracing, code explaining, and code writing [LAF+04, LFT09, LST+06, LWRL08, PRW07, SCL+08, S+12, VTL09, WLT+06]. For example, Lister et al. [LAF+04] concluded that many students are not able to program at the end of their introductory programming course and one possible reason for this is that they lack the ability to trace code. In addition, Whalley et al. [WLT+06] have found that students who write successful programs are also able to give a correct explanation about the code. Sheard et al. [SCL+08] have found a positive correlation between code explaining ability and code writing ability. Furthermore, Lister et al. [LFT09] concluded that students who cannot trace the code are also unable to explain the code. They also found statistically significant relationships between code tracing, code explaining, and code writing abilities [LFT09]. Both the abilities, code tracing and code explaining, are often used to measure the code reading process of programmers.

Additionally, according to several studies [LFT09, LWRL08, S+12, VTL09, XLN+19], there exists a certain learning hierarchy between these programming skills. For example, Sorva [S+12] has mentioned that code tracing is lower ranked than code explaining, following the SOLO taxonomy. This is because code tracing requires multi-structural learning outcomes while code explaining requires relational learning outcomes [S+12]. Other studies [LFT09, VTL09] support Sorva's conclusion and add that these two skills together are again lower ranked than code writing.

Furthermore, code reading might also influence the code comprehension of programmers. In a study done by Busjahn and Schulte [BS13], it is found that overall, code reading is considered as a prerequisite to comprehending a program: "*[code reading] was conceptualized as the process of visually perceiving the source code, and as [a] prerequisite of program comprehension*" [BS13]. Moreover, Busjahn and Schulte have concluded that "*successful [code reading] enables students to understand the programs' execution and intention*" [BS13]. According to them, successful code reading will enable the students to become faster at all code reading-related tasks [BS13].

From the above, it appears that code reading is an important process that is worth further investigation. Therefore, this thesis will take a further look into the reading process of novice programmers.

2.2 Eye-tracking and think-aloud

To gather insights into programmer's cognitive process of code reading, many studies [Bed12, BBB+15, PIS17, PSA20, SFM12, UNMM06, UNMM07] used the eye-tracking method. Busjahn et al. [BSB11] have mentioned in their study that "*reading can be analytically divided into perception and comprehension. Eye-tracking is a means to observe the process of perception to obtain insights in the process of comprehension.*" [BSB11]. For example, Busjahn et al. [BBB+15] used the eye-tracking method to investigate the reading order of programmers. A similar study is also done by Peitek et al. [PSA20]. Moreover, other studies [BT06, BSB11, BSS+14, Gué06, TFSL14] also connected

programmers' cognitive process of reading to their comprehension of the code.

However, it is hard to translate these eye movements into insights about a programmer's mental state during reading and understanding a program. Therefore, to have a deeper understanding of this mental state, some studies [UNMM06, HOM+08] combined the eye-tracking method with the think-aloud method. The think-aloud method itself is also a commonly used methodology in programming education research that is focused on the cognitive aspects of learning and reasoning. For example, Whalley and Kasto [WK14] have conducted a think-aloud study to investigate "*the cognitive aspects of the early stages of learning to write computer programs*" [WK14]. Another example is a think-aloud study conducted by Izu et al. [IPW17] to investigate the reasoning strategies the students used to examine program behavior within the context of reversibility. Also, Lister et al. [LST+06] used the think-aloud method to gather data about how students and educators solve small code reading problems.

Sorva [VSBS94] described the think-aloud method as follows: "*the think aloud method consists of asking people to think aloud while solving a problem and analysing the resulting verbal protocols*" [VSBS94]. Furthermore, according to Sorva, the think-aloud method "*has applications in psychological and educational research on cognitive processes*" [VSBS94] and in many cases, this method provides a unique source of information [VSBS94]. Therefore, the think-aloud method will be a feasible instrument to gather data about programmers' code reading processes.

However, asking the subjects to think out loud while reading the code might disturb their reading processes since they will need more cognitive load to finish the task. Cognitive load is related to the amount of "*cognitive resources that are focused and used during learning and problem solving*" [CS91]. Thinking out aloud during problem-solving is called concurrent think-aloud [BSB11]. However, a version that does not interrupt reading is the retrospective think-aloud method. This method will not interfere with the subjects during problem-solving and allows the subjects to finish the reading process without disturbance. Only afterward, the subjects are questioned about their thought processes during reading [VSBS94].

Therefore, this thesis uses the retrospective think-aloud method to further investigate the code reading process of programmers.

2.3 Reading a natural language text versus a source code

In general, it is assumed that reading a natural language text is different from reading a source code. In one study done by Blascheck and Sharif [BS19], it is found that reading a natural language text partially differs from reading a source code. People tend to read a natural language text more linearly, whereas, with a source code, this linear reading order is only partially applicable. Besides following the code in a linear order, people also tend to follow the execution order of the program [BS19]. In addition, Blascheck and Sharif [BS19] also found that participants focus more on "*those areas that are important to comprehend core functionality*" [BS19] and that they "*skip unimportant constructs such as brackets*" [BS19].

Also, in another study done by Busjahn et al. [BBB+15], it is found that reading a source code

is less linear than reading a natural language text. However, the results show that novice programmers tend to follow a source code in a more linear reading order than expert programmers. Busjahn et al. [BBB+15] have found that when reading a natural language text, approximately 80% of novices' eye movements follow a linear reading order, whereas reading a source code, the percentage decreases to 70%. However, this percentage is still higher than that of the expert programmers, which is 60% when reading a source code [BBB+15]. Therefore, it is assumed that with the increase of programming experience, programmers will less likely to follow a source code linearly.

Thus, there might exist a certain relationship between the reading strategies the novice programmers use to read a natural language text and a source code. Therefore, this thesis will take a further look into these reading strategies.

3 Methods

This research aims to find the relationship between the strategies that are used to read a natural language text and the strategies that are used to read a source code among novice programmers, and also to gather more insights into programmers' reading processes. To achieve this, the reading strategies for a natural language text, reading strategies for a source code, and reading focus on a source code are investigated. The data is collected using a semi-structured think-aloud interview. In the following subsections, the participants, materials, set-up of the interview, and data processing will be discussed.

3.1 Participants

For this research, we aim to include participants who have a computer science background and have less than three years of actual programming experience or undergraduate computer science students. The reason behind this choice is because the target group for this research is novice programmers. In addition, the participants should be introduced to the Python programming language, or at least they should know about the basic concepts of Python. That is because the example code that is used during the interview is written in Python.

In total, we conducted ten interviews. The duration of the interviews varied from nine minutes to 68 minutes with an average of 27.5 minutes. All the participants had a computer science background and the majority of this group were undergraduate students specializing in computer science (N = 3) and computer science & economics (N = 4). All of them had less than two years of Python programming experience, which varied from one month to 24 months with an average of 12 months. The programming experience varied from four months to 72 months with an average of 34.6 months. The one with 72 months was an undergraduate computer science & economics student who also included a high school computer science course in his/her programming experience. Furthermore, eight participants were between the ages of 20 and 25, and two of them were older than 40. Seven participants identified themselves as being male and three being female. All of them were Dutch-speakers. Lastly, one of them declared to have dyslexia.

We recruited the participants online, whereby three of them were recruited via an online registration

form. This online registration form was spread through different platforms such as Twitter and Brightspace (an online learning platform). We also recruited participants via a WhatsApp group for computer science students and computer science & economics students.

3.2 Materials

3.2.1 Source code

We use the programming language Python to write the example code and we include different programming concepts in this code, which are: lists, conditions, for-loops, while-loops, slicing, and nesting. We have chosen these concepts because they are the basics of Python programming language that are assumed to be not too difficult and do not require other external libraries or files to execute. By combining these different concepts, the participants are forced to take more complicated reading strategies to comprehend the code. We explicitly divide the code into four parts (using comments) to facilitate the questioning part during the interview where the participants can refer to the different parts of the example code. *Part 1* contains the function *check_pattern*, *Part 2* the function *moving_average*, *Part 3* the function *process_list*, and *Part 4* the main function as you can see in Figure 1.

3.2.2 Survey

This research intends to collect background information of the participants using a Google Form survey. In this survey, we ask several questions in order to form a picture of the participants and also to make sure that we only include participants who are truly novice programmers. The questions in this survey are as follows:

- Q2. What is your gender?
- Q3. What is your age?
- Q4. What is your native language?
- Q5. How many months of experience do you have in programming in general?
- Q6. How many months of experience do you have in the Python programming language?
- Q7. Do you have dyslexia?

3.3 Interview setup

We conduct the interviews online via Zoom because of the current situation of COVID-19. A semi-structured think-aloud interview strategy is used to gather the needed data. We have chosen this strategy because, in this way, we can ask more open-ended questions and can have a discussion with the participants rather than a straightforward fixed question and answer format. By doing this in a think-aloud manner, the participants can express their thoughts freely. Besides, we also have more freedom to ask for more details or related questions to any interesting answers they give. We conduct the interviews in Dutch and record them on audio and video. Hence, the collected data and the transcribed protocols for this research are also in Dutch. The interview procedure is outlined below.

```python
# PART 1
def check_pattern(input_string, pattern):
    if len(input_string) == 0 and len(pattern) == 0:
        return True
    if len(pattern) == 0:
        return True
    while len(input_string) != 0:
        if pattern in input_string:
            for i in range(len(input_string)):
                input_pointer = i
                pattern_pointer = 0
                while pattern_pointer < len(pattern):
                    if input_string[input_pointer] == pattern[pattern_pointer]:
                        input_pointer += 1
                        pattern_pointer += 1
                    else:
                        break
                if pattern_pointer == len(pattern):
                    index = i
                    break
        else:
            return False
        input_string = input_string[0:index] + input_string[index + len(pattern):]
    return True

# PART 2
def moving_average(values, n):
    moving_average_values = []
    for i in range(len(values)):
        moving_average = None
        if i >= n-1:
            moving_average = 0
            selector = i - n
            while selector < i:
                selector += 1
                moving_average += values[selector]/n
            moving_average_values.append(int(moving_average))
    return moving_average_values

# PART 3
def process_list(list_of_lists):
    new_list = []
    for sub_list in list_of_lists:
        for item in sub_list:
            new_list.append(item.capitalize())
    return new_list

# PART 4
input_string = ['TETESTST', 'TETESTT']
pattern = 'TEST'
for sub_string in input_string:
    if check_pattern(sub_string, pattern):
        values = [10,12,9,11,14,20]
        print(moving_average(values, 3))
    else:
        super_list = [["duck", "cat"], ["flower", "tree"], ["whale", "swordfish"]]
        print(process_list(super_list))
```

Figure 1: The example code used during the interview.

3.3.1 Explaining the procedure

Before the start of the interview, we will make clear to the participants what they can expect during the interview and what the purpose is of the data. Furthermore, we will ask for participants' permission to record them on video and audio as it is necessary for the recording of their answers. Afterward, we will transcribe these answers into protocols. The recording ensures that errors in this transcription process are minimized.

3.3.2 Creating an open/safe environment

During the interview, the participants can be insecure about their answers or what they should do. Since we do not intend to steer them in a certain direction, we will ensure that the participants are aware that there are no right or wrong answers. Any thought or answer they come up with can be useful for this research. To make sure that the participants are aware of this, we will use the following instruction (translated from Dutch to English) at the start of the interview and if necessary, also during the interview: *"There are no right or wrong answers. You will not be judged by this. So, you are free to say everything that comes to your mind and try not to be afraid to do something wrong. We are interested in everything that you will say and how you will say it."*

3.3.3 Google Form survey

We will provide the participants a link to the survey about their background information so that they can fill it in. More detailed information about this survey can be found in section 3.2.2.

3.3.4 Reading strategies

In this part of the interview, we will ask the participants to answer several questions regarding the common reading strategies they use to read a natural language text, the common reading strategies they use to read a source code, and the main difference between these two. We defined these questions based on the sub-questions of the first research question. By asking these questions, a basic understanding of the reading strategies commonly used by the participants can be gathered. The questions (translated from Dutch to English) that will be asked in this part of the interview are as follows:

- Q1. What are the common reading strategies that you will use to read a difficult text, for example, a scientific paper?
- Q2. What are the common reading strategies that you will use to read a source code?
- Q3. According to you, what is the main difference between reading a text and reading a code?

In the question about the natural language text (Q1), we chose to give the participants an example of a natural language text, namely a scientific paper, so that they can have a reference point. This also means that the answers the participants give are based on the given example, i.e. how do they read a scientific paper.

3.3.5 Introducing the code

We will provide the participants a source code as shown in Figure 1. The participants have to read and comprehend this code. In the remaining parts of the interview, the participants will have access to this code snippet. Thereby they can refer to it while answering the questions. This research intends to do a retrospective think-aloud interview, which means that we will ask the participants to think out aloud after the problem-solving, thus when they are finished with the code reading process. However, during code reading, the participants can choose for themselves if they want to do it in a think-aloud manner or not, because if that is what the participants normally do, then we are not intended to disturb this normal reading process of the participants.

3.3.6 Reading focus

In this part of the interview, we will ask the participants to answer several questions regarding their reading focus based on the example code. We defined these questions based on the sub-questions of the second research question and mainly focused on gathering information about participants' reading focus and their (partially) reasoning behind it. Therefore, we will ask one question (Q1) to get a basic understanding of the reading focus of the participants. Thus, which concepts and/or elements of the example code do they mostly focus on while reading the example code. Furthermore, it is assumed that the perceived importance level can also influence the reading focus of the participants. If a certain part will be seen as more important, then the participants are likely to focus more on that part. Therefore, we will ask two questions (Q2 and Q3) to gather insights into the most important part of the example code that the participants have defined and the reasoning behind it. We will also ask two questions (Q4 and Q5) about the most difficult part of the example code according to the participants and the reasoning behind it. This difficulty level might also influence the reading focus of the participants, because if a certain concept/part is seen as more difficult, then the participants are likely to focus more on that concept/part. The questions that will be asked in this part of the interview are as follows:

- Q1. What did you mostly focus on while reading?
- Q2. Which part of this code is the most important according to you?
- Q3. Why do you find this part the most important?
- Q4. Which part of this code is the most difficult to understand according to you?
- Q5. Why do you find this part the most difficult?

3.4 Data processing

We use a qualitative inductive data analysis, which is not based on a theory or framework [YAH20], to gather insights about programmers' reading strategies and reading focus. Thus, the information is identified through the collected data [YAH20]. After the interviews are transcribed into protocols, these protocols are categorized into different themes. This categorizing procedure does not require much time because the interviews are already divided into different themes according to the sub-questions of the two research questions. The themes are as follows: 1) *common reading strategies*

for a natural language text, 2) *common reading strategies for a source code*, 3) *differences between reading a text and reading a code*, 4) *reading focus based on an example code*, 5) *important code parts according to novice programmers*, and 6) *difficult code parts according to novice programmers*. The first three themes are in line with the sub-questions of the first research question, and the other three themes are in line with the sub-questions of the second research question. This data processing procedure is similar to the one used by Yeni et al. [YAH20].

After defining the themes, open coding [YAH20] is used to define several basic labels for each theme. This is completed while reading all the transcribed protocols of each interview. Using an inductive approach means that open coding can lead to a broad identification of ideas and views for each theme [YAH20].

The next step is using axial coding [YAH20] to review and group the related labels under a more general category if necessary. Otherwise, the labels that are not related to any other label form on themselves a separate category. An example where we used the axial coding was within the theme *common reading strategies for a source code*. We grouped the labels *look at the classes and/or methods*, *look at the code structure*, and *look at the functions and imports*, defined in the open coding phase, under the category *look at the code's organization* during axial coding.

Lastly, we use selective coding [YAH20] to group different categories that are related to each other under a new sub-theme. However, this selective coding was only applicable within the theme *reading focus based on an example code*. Within this theme, the categories *execution flow*, *control flow statements*, *variable usage/changes*, and *functions/methods* were grouped together under the new sub-theme *code behavior*. Furthermore, the categories *documentation/comments* and *variable names and function names* were grouped together under the new sub-theme *code organization*.

4 Results

Each participant is assigned a number corresponding to the order in which they have been interviewed and will be referred to as P1 to P10 in the remaining sections. The quotes used in the following sections are translated from Dutch to English which may contain our interpretation, therefore in Appendix I, a table is provided with each translated quote and its original version. During translation, unnecessary fragments of the quote, such as repetition of words, are removed and stated as [...]. To protect the anonymity of the participants, personal pronouns such as he or she are not used individually, instead, we use a combination such as (s)he. The results are divided into two sections. The first section (4.1) is about the interview part before reading the example code, thus the general reading strategies the participants said that they use. This interview part is related to the sub-questions of the first research question: "*How are the common reading strategies for a natural language text related to the common reading strategies for a source code among novice programmers?*". The second section (4.2) is about the interview part after reading the example code, thus the reading focus of the participants, specific to that example code. This part of the interview is related to the sub-questions of the second research question: "*What do novice programmers focus on during reading, based on an example code?*".

4.1 Novice programmers' common reading strategies

The first research question consists of two parts: 1) common reading strategies for a natural language text, and 2) common reading strategies for a source code. During the interview, each participant was asked to answer three questions that are related to the first research question. This part of the interview was done before reading the example code, and the answers the participants gave were based on their general reading strategies. The questions asked in this part of the interview are as follows:

1. *What are the common reading strategies that you will use to read a difficult text, for example, a scientific paper?*
2. *What are the common reading strategies that you will use to read a source code?*
3. *According to you, what is the main difference between reading a text and reading a source code?*

In Table 1 an overview is shown of the categories and their presence in the answers of the participants for each question. Per question, the answers will be discussed below. Finally, we will provide a summary of our main findings in a separate section.

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	N
Common reading strategy (text)											
Skimming	X	X			X	X	X	X	X	X	8
Scanning		X	X		X	X	X		X		6
Intensive reading	X			X				X			3
Common reading strategy (code)											
Follow the code's execution				X	X	X	X			X	5
Look at the code's structure	X	X				X				X	4
Scan for function names and/or variable names								X	X	X	3
Read comments/documentation first		X	X					X			3
Determine the programming language	X					X					2
Other			X					X	X		3
Main difference between reading a text and a code											
Reading order						X		X	X	X	4
Reading goal	X	X		X							3
Reading strategy			X		X		X				3

Table 1: Overview of the categories and their presence in participants' answers regarding the questions about the reading strategies.

4.1.1 Common reading strategies for a natural language text

During the interview, the participants were asked to give the common reading strategies they use to read a difficult text such as a scientific paper. The reading strategies are summarized in Table 1 under the heading *Common reading strategy (text)*. These reading strategies can be

grouped into three categories: 1) skimming (N = 8), 2) scanning (N = 6), and 3) intensive reading (N = 3). Important to mention is that each participant could give more than one reading strategy.

4.1.1.1 Skimming. The reading strategy that was most frequently said to be used by the participants (8 out of 10) to read a scientific text is skimming. Using this strategy, the subject will not read the whole text, instead, they only look for the general or main idea of a text by reading important information, such as the title, headings, introduction, and conclusion. Two examples in which the participants mentioned the elements of the skimming strategy are: "*I look at the table of contents, so which subheadings are there, and then I usually start at the top, with something like an introduction*" (P5), and "*I would sort of reading the introduction and conclusion first*" (P7). The most mentioned text sections/elements are the headings (N = 6) and introduction (N = 4). Other text sections/elements that were mentioned are the abstract (N = 3), title (N = 3), pictures (N = 2), conclusion (N = 2), and results (N = 1).

4.1.1.2 Scanning. The second most mentioned reading strategy is scanning (6 out of 10). Like skimming, by using scanning as a strategy, the participants will not read the whole text. However, the difference between scanning and skimming; the participants will look for a fact or piece of information they are interested in, instead of searching for a general idea of the entire text. P2 stated that "*I never read from top to bottom with those kinds of texts, you really go looking for what you want to know and filter out what you already know or what you don't find relevant at that moment*", and P3 declared that "*you really unpack the keywords in there and [...] read around the keywords [...], but I am not going to read a text completely*".

Additionally, there were five participants (P2, P5, P6, P7, and P9) who said to use both strategies. They declared that they first use scanning and then skimming. They use the information they get from scanning to further search for the parts they are interested in by using skimming. For example, P9 stated the following: "*first, the title of course, and then the abstract, and then the headings, and then based on that, [I] will search for what [I] want to know and then read the entire paragraph per interesting point what I actually search for*".

4.1.1.3 Intensive Reading. The least frequently mentioned strategy to read a scientific paper is intensive reading (3 out of 10). In contrast to the other two reading strategies, by using this strategy, a reader is expected to read the entire text to get a full understanding of it. P1 stated the following: "*you say it's a difficult text and you really have to learn to make it your own and [...] then I'm going into it very intensively, which means that I highlight points paragraph by paragraph, so to speak*". P1 also expressed: "*especially if I have to read and understand the piece intensively, then I really go through it sentence by sentence, and [...] try to mark for me what the most important things are in there*". The other two participants (P4 and P8) said that they just read the whole text and when they did not understand a certain sentence or part, they will read it again.

However, both P1 and P8 mentioned that their reading strategies depend on their reading goal. P1 stated this explicitly in his/her answer: "*then I go, yes depends on how I'm supposed to read, should I read it as some sort of reviewer, should I read it to apply it myself*". P8 only said that if (s)he will not read the whole text, then (s)he will only read the introduction and the conclusion, otherwise, P8 will read the entire text line by line. Both participants mentioned the reading strategies skimming

and intensive reading.

4.1.2 Common reading strategies for a source code

During the interview, the participants were asked to answer the following question: "*What are the common reading strategies that you will use to read a source code?*". The answers to this question can be divided into five categories: 1) follow the code's execution (N = 5), 2) look at the code's structure (N = 5), 3) scan for function names and/or variable names (N = 3), 4) read comments/documentation first (N = 3), 5) determine the programming language (N = 2), and 6) other (N = 3). These answers are summarized in Table 1 under the heading *Common reading strategy (code)*. Important to mention is that each participant could mention more than one reading strategy.

4.1.2.1 Follow the code's execution. Most of the participants (5 out of 10) declared that they follow the main execution of the code. By doing this, they can see what the code is trying to do. P5 expressed the following: "*if the code is with multiple functions, I first find the main function, [the] function that calls all the other functions, and then [...] I do those functions from top to bottom and only when I really want to know, then I will look at what the sub-functions do. So, when I read [a] function, basically, [it is] how the computer itself runs through it: [I] try to follow the computer step except I don't look at sub-functions, I'll skip those*".

4.1.2.2 Look at the code's structure. The second most mentioned strategy is to look at the code's structure (4 out of 10). Two participants (P1 and P10) said that they look at the classes and/or methods and this reflects the structure of the code. They stated the following: "*when I look at code from developers, who have more experience, who also use classes and methods, yes, that means you first have to check which class, methods are actually in it, otherwise you don't know [...] how you have to read the rest*" (P1), and "*to read a long code, I would first check if there are any classes, roughly see what the class is without reading it exactly*" (P10). Further, P6 explicitly mentioned that (s)he looks at the code structure, and P2 expressed: "*you also look at which functions are defined there, and which imports are there because that also gives an idea*". This, in a way, reflects that P6 also looks at some aspects of the code's structure.

4.1.2.3 Scan for function names and/or variable names. The third category is scan for names of the functions and/or the names of the variables (3 out of 10). The participants declared that: "*I think I would scan through the function names*" (P10), "*looking at the function names, especially that*" (P9), and "*I usually try to look a bit at the function names and also the general variable names*" (P8).

4.1.2.4 Read the comments/documentation first. The fourth category is to read the comments in the source code or the documentation of the source code first (3 out of 10). P2 stated that: "*if you look at a Python program, there should be a docstring at the top [...] and I think that's the first thing you should look at*". The other two participants (P3 and P8) explicitly expressed that comments/documentation is the most important part that you should look at. P8 also explicitly described why the comments are the most important part. They said the following: "*of course, you start with the documentation, that's the most important*" (P3), and "*of course, you also look for comments, which is, of course, the most important, because that just explains how the code works*"

(P8).

4.1.2.5 Determine the programming language. Two participants (P1 and P6) mentioned the programming language in their answers. P6 explicitly described in his/her answer that (s)he looks at the programming language. However, P1 did not explicitly mention this, but did mention that (s)he changes his/her reading strategies based on the programming language: "*I do notice that I tend to adjust my reading style a bit per language, but I have to say that I don't like reading Java as much as Python*".

4.1.2.6 Other. Three reading strategies were only named once, which are 1) look at the complex parts (P3), 2) split the code into different parts (P8), and 3) look at the control flow statements and variables (P9). P3 expressed that: "*I think I mainly look at the complex parts [...] that I just say oh well look here is a complex part, I'll start with that and what does it roughly do*", P8 stated that: "*of course, if it's a good code, at least well organized in functions and parts, so try to break everything down into different pieces to make it easier to understand*", and P9 declared that: "*look at all if statements and the like, so where [...] all decisions in a code actually are and only then I look at what exactly happens with variables and the like*".

4.1.3 Differences between reading a text and reading a code

During the interview, the participants were asked to give the main difference between reading a text and reading a source code according to them. Their answers can be grouped into three categories: 1) reading order, 2) reading goal, and 3) reading strategy. This is summarized in Table 1 under the heading *Main difference between reading a text and a code.*

4.1.3.1 Reading order. The first category is the reading order which was mentioned by four participants (P6, P8, P9, and P10). According to these participants, reading a source code is not as linear as reading a natural language text. For example, P6 said that: "*the order is not always left to right, top to bottom*". P10 also made a clear comparison between the reading order of a source code and the reading order of a natural language text, (s)he expressed that: "*with a text, you go through the text from top to bottom, often from beginning to end, [whereas] with a code, through functions and the like, you don't read it from top to bottom, but sometimes you also suddenly go up a bit, go down a bit*".

4.1.3.2 Reading goal. The second category is the reading goal. Three participants (P1, P2, and P4) declared that when reading a source code, the goal is to find out what the code is trying to do. They stated the following: "*with code, I think we often tend to look at the practical use of that code, so it has to run, so to speak*" (P1), "*with code you want to understand what it does*" (P2), and "*a code must actually do something, so I think that you are looking in a code for what the code must do*" (P4). Two participants (P2 and P4) also described explicitly what the reading goal is of a text. They stated that: "*a text is primarily intended to inform*" (P2), and "*in a text, you also look for the core, but then you read, for example, [...] first the introduction and the conclusion if you want to know what the text is about*" (P4).

4.1.3.3 Reading strategy. The third category is the reading strategy. Three participants (P3, P5, and P7) described their reading strategy for a natural language text and their reading strategy for

a source code. According to them, there exists a difference between these reading strategies. P5 said that (s)he starts reading a source code from the inside, whereas with a natural language text, (s)he starts reading it from the outside. P3 said that it is easier to have a general idea of what the code is about without exactly reading it, whereas, with a scientific paper, it is more difficult to know where the important parts are before you start reading it. In contrast, P7 said that with a text, (s)he can easily scan through it to see what the text is about and what (s)he is looking for, but with a source code, it is more difficult to scan and see what it is about.

4.1.4 Summary reading strategies

Our participants mostly mentioned *skimming* as their main/preferred reading strategy for a natural language text (8 out of 10), followed up by *scanning* (6 out of 10). However, half of the participants (5 out of 10) said that they use both strategies. Further, only a small proportion of the participants (3 out of 10) said that they use the strategy *intensive reading*. One interesting observation is that none of the participants who said to use the strategy *intensive reading* mentioned the strategy *scanning*. Two of them (P1 and P8) mentioned the strategy *skimming*, and one of them (P4) only mentioned *intensive reading*. It seems that the participants who chose to read the entire text often do not try to get a general idea of the text. However, *intensive reading* is less mentioned than *scanning* and *skimming* because a scientific paper is long and often people only use a part of it instead of the entire text.

Furthermore, the most often mentioned strategy to read a source code is to *follow the code's execution* (5 out of 10). Then, in second place is the strategy *look at the code's structure* (4 out of 10). The other two strategies 1) *scan for function names and/or variable names*, and 2) *read the comments/documentation first* have the same frequency of repetition (3 out of 10). Then, a less frequently mentioned strategy is to *determine the programming language*, which is mentioned by two participants (P1 and P6). Finally, there are three strategies which are only mentioned once, which are: 1) *look at the complex parts*, 2) *split the code in different parts*, and 3) *look at the control flows and variables*. It seems that the participants mostly looked at the parts of the code that are more relevant to the execution of the code. Other parts that are less important to the execution of the code are less frequently mentioned in their answers.

Lastly, the main difference between reading a text and reading a code that was given by the participants can be divided into three categories: 1) *reading order*, 2) *reading goal*, and 3) *reading strategy*. The *reading order* is most frequently mentioned (4 out of 10). According to the participants, reading a source code is less linear than reading a natural language text. Furthermore, the *reading goal* was mentioned by three participants (P1, P2, and P4). Their main argument was that when reading a code, they try to find out what the code will do, whereas with a text this is not the case. They argued that a code should be executed and will have an output, whereas a text is deemed static, you only have to read it in order to get informed about the text. Lastly, the *reading strategy* was mentioned by three participants (P3, P5, and P7). One participant (P3) said that (s)he read a text from the outside, whereas with a code, (s)he starts reading it from the inside. Two other participants disagreed with each other. One (P5) being convinced that it is easier to get a general idea of the code without actually reading it compared to a text, whereas, the other (P7) declared that it is easier to scan a text and see what the text is about compared to a code.

4.2 Novice programmers' reading focus

This section is related to the reading focus of the participants based on an example code. During the interview, the participants were asked to answer several questions regarding their reading focus based on the source code they read. Those questions are related to the second research question. The asked questions are as follows:

1. *What did you mostly focus on while reading?*
2. *Which part of this code is the most important according to you?*
3. *Why do you find this part the most important?*
4. *Which part of this code is the most difficult to understand according to you?*
5. *Why do you find this part the most difficult?*

These questions provide different aspects about a programmer's reading process: 1) elements/concepts of the code which novice programmers focus on while reading, 2) code part which is most important according to them, and 3) code part which is most difficult according to them. In table 2 an overview is shown of the categorization and their presence in the answers of the participants for each question. Below, the answers to each question will be discussed separately. Finally, we will provide a summary of our main findings in a separate section.

	P1	P2	P3	P4	P5	P6	P7	P8	P9	P10	N
Focus on code behavior											
Execution flow	**X**		**X**		X	**X**	**X**				5
Control flow statements		X	X	**X**	X				X		5
Variable usage/changes					**X**		X		X	X	4
Functions/methods		**X**	X			X					3
Focus on code organization											
Documentation/comments		X	X					X			3
Variable names and function names								**X**	**X**	**X**	3
Important code part											
check_pattern (part 1)					X	**X**		**X**			3
moving_average (part 2)					X						1
main function (part 4)	**X**	**X**	**X**	**X**		X	**X**	X	**X**	**X**	9
Difficult code part											
check_pattern (part 1)	**X**		**X**						**X**		3
moving_average (part 2)		**X**			**X**	**X**	**X**	**X**		**X**	6
process_list (part 3)				**X**							1

Table 2: Overview of the categories and their presence in participants' answers regarding the questions about the reading focus. The most important/difficult element mentioned by each participant is shown in bold.

4.2.1 Reading focus based on an example code

The answers of the participants for this section can be grouped into two main categories: 1) code behavior, and 2) code organization. These two categories can be further categorized into different sub-categories, which are summarized in Table 2 under the headings *Focus on code behavior* and *Focus on code organization*. Important to mention is that each participant can belong to multiple categories as well as sub-categories. This is because some participants focused on different elements of the code in a certain order or explicitly include some reading focus that is not specific to the example code. If the participants mentioned multiple elements of the source code, then the element they explicitly said that is the most important, or if that is not explicitly stated, then the first element they mentioned is considered as the most important element.

4.2.1.1 Code behavior. The first category is code behavior, which consists of the following sub-categories: 1) execution flow, 2) control flow statements, 3) variable usage/changes, and 4) functions/methods. All these sub-categories show an aspect of the code behavior, i.e. how and what will the code do when it is executed. The execution flow is the order in which the different statements are executed by the computer. The control flow statements are the statements such as conditions and loops that affect the execution order of the code. The variables are a part of the code that is used to assist the execution of the code, for example, how they are used in the statements or the changes of the variables that can affect a certain order that the computer is going through. Furthermore, the functions/methods also show an aspect of the code behavior because they can be called during the execution of the code and the return value of these functions/methods might affect the further behavior of the code.

Execution flow. Most of the participants (5 out of 10) mentioned that when reading the given source code, their attention goes to the execution order of the code. Thus, in which order the computer is executing the program. P5 mentioned explicitly by this code that (s)he followed the computer steps because (s)he was trying to use the entire code, however (s)he also mentioned that when (s)he was only looking for a certain part of the code, (s)he focused on the loops, which did not apply in the given example code. Furthermore, one participant (P3) said that (s)he focused on the output of the code, thus what does the code print/show when it is executed. P3 stated the following: "*so, I start with the output myself, so what does it do, what does it actually print*".

Control flow statements. Most of the participants (5 out of 10) also mentioned that their reading focus depends on the control flow statements such as decision-making statements (if-else) and looping statements (while and for). Two participants (P2 and P9) mentioned the conditions in their answers, and one participant (P3) explicitly stated the if-else statements. P3 also mentioned the while loop. Furthermore, P4 and P5 also stated the loops in their answer. For example, P4 stated the following: "*what I focus most on, for example, with [part one] was [...] the second while loop, to understand that, and with part two it's actually also on that while loop, what happens there, so I always focus on that*".

Variable usage/changes. Four participants (P5, P7, P9, and P10) said that they focused on what happens with the variables. For example, they stated that: "*when I had to find the output, then I really started looking more closely at what happens to the values*" (P7), and "*my main focus has been on keeping track of what variables contain to see what they actually did, like if variables were being*

compared, that in my head, I'm like okay this variable is this and this is that, so what comes out of these comparisons" (P10). One special case was the answer of P5. P5 first said that (s)he focused on the execution order in which the computer goes through, however after (s)he asked for a repetition of the question, (s)he gave another answer, namely that (s)he focused on the changes of the variables.

Functions/methods. Three participants (P2, P3, and P6) mentioned the functions or methods in their answers. P2 and P6 expressed that they focused on the input and output of the functions. P2 stated that: "*on the functions and what the input and output of the functions is*", and P6 declared that: "*I will see from which moments which headers are called, and what comes in and what comes out*". Furthermore, P3 stated that (s)he scanned through the code to see which methods are important.

4.2.1.2 Code organization. The second category is code organization, which is further categorized in the following sub-categories: 1) documentation/comments, and 2) variable names and function names. Documentation/comments do not affect the behavior of the code, but they do reflect how well the code is organized. Does the code have a documentation string above the code that explains the code? Also, variable names and function names give a clear overview of how well the code is organized. Are the names clear and well organized, and are they representative of what the functions and variables should do?

Documentation/comments. In this part of the interview, one participant (P8) explicitly mentioned that he/she focused on the comments if there was any: "*in this case there are no comment but normally also [...] comments*" (P8). No other participant mentioned the comments or documentation in their answer in this part of the interview, due to the lack of it in the given source code. However, in the previous part of the interview, two participants (P2 and P3) did mention that comments/documentation is important to them.

Variable names and function names. Three participants (P8, P9, and P10) declared that they focused on the names of the variables and the names of the functions. For example, P10 stated: "*I notice that my focus first went to the variable name and the function name to see if I can already deduce something from there*".

4.2.2 Important code parts according to novice programmers

The given source code consists of four parts which are stated as Part 1, Part 2, Part 3, and Part 4 in the comments. Part 1 is the *check_pattern* function, part 2 the *moving_average* function, part 3 the *process_list* function, and part 4 the main function. During the interview, the participants were asked which part of the given source code they considered as the most important, also they were asked to explain why. For each code part, it is determined how often the part was mentioned and why the participants argued it as the most important part. This is summarized in Table 2 under the heading *Important code part*. Below, the reason for each part is discussed. Furthermore, Part 3 was never mentioned as most important by the participants. As you can see in Table 2, there are three participants (P5, P6, and P8) who mentioned two code parts in their answers, however, P6 and P8 made clear which of the two they found the most important. One special case is P5, (s)he mentioned part 1 and part 2 as important code parts, but (s)he did not explicitly mention

which of the two is the most important. In Table 2, the most important code part for each participant is shown in bold. For P5, both parts are considered as important, but not as the most important.

4.2.2.1 check_pattern (part 1). Two participants (P6 and P8) expressed that the *check_pattern* function is the most important part. The reason they gave is that this part determines the behavior of the code. For example, P6 stated: "*part one is also very much directing the whole behavior, so I'll say part one*", and P8 described: "*check pattern could, of course, be seen as a little more important than [part] two and three because that is used to determine whether it does [part] two or three, but [...] actually all code depends on the check pattern function so if I have to choose, I will choose part one*". Furthermore, P5 did not explicitly mention that part 1 is the most important part, but (s)he did mention that this part is important, together with part 3. The reason (s)he gave why part 1 is important is the same as mentioned above.

4.2.2.2 moving_average (part 2). Only one participant (P5) mentioned the *moving_average* function as an important part, because of its difficulty. (S)he explained that: "*I think the moving average is important because that was the [...] hardest to understand*".

4.2.2.3 main function (part 4). Almost all participants (9 out of 10) considered part 4 as an important code part. Seven of them (P1, P2, P3, P4, P7, P9, and P10) mentioned this part the most important. The main reason they gave is that this part is the main code where the actual execution of the code takes place. For example, P2 stated that: "*the most important is, of course, part four, because that's where the work will be done*", P7 mentioned the following: "*that last part, [...] part four, then it all comes together, that makes it clearer what happens [...] when you really [...] start running it*", and P10 declared the following: "*I think the most important part is called part four I believe, part in which other functions are called and from there it actually starts the process of which way the code goes, and also which output is shown*". Two participants (P6 and P8) also mentioned this part in their answer, but not as the most important part.

Only, one participant (P3) explicitly said that the entire code is important because "*otherwise your code won't work*" (P3). However, P3 did mention that if you want an output, then part 4 can be considered as the most important part because "*that's where they actually print it*" (P3).

4.2.3 Difficult code parts according to novice programmers

During the interview, the participants were also asked to give the part they considered as the most difficult to understand. Besides, they were asked to explain why they consider a particular part as the most difficult. For each part, the reasons are summarized in Table 2 under the heading *Difficult code part*. Below, the reasons for each part will be discussed. Furthermore, none of the participants mentioned part 4 as the most difficult.

4.2.3.1 check_pattern (part 1). Three participants (P1, P3, and P9) mentioned that part 1 is the most difficult part of the code. There are mainly three reasons why they found this part as the most difficult: 1) nested loops (N = 2), 2) many statements (N = 1), and 3) unfamiliar concepts (N = 1). The first reason was given by two participants (P1 and P9). They explained that: "*the combination of while, ifs, while, ifs, fors, things like that, that whole thing makes it more difficult to*

understand" (P1), and "*because it has a lot of layers, it's an if statement in a while statement in a for loop and an if statement in a while statement, so at some point, I kind of lose track of what does what*" (P9). The second reason was defined by P3, (s)he stated that: "*the hardest part, well that will be part one because that's where most of the statements are*" (P3). The third reason was given by P1, (s)he expressed the following: "*that behavior around a break, I sometimes struggle with it, so to speak, what happens if it is in an if-else thing, in a while, what happens then, that behavior is not so predictable to me*" (P1).

4.2.3.2 moving_average (part 2). Most of the participants (6 out of 10) said that part 2 is the most difficult to understand. There are mainly three reasons why they find this part difficult: 1) unclear variable names (N = 5), 2) unclear/difficult structure (N = 2), and 3) many variables (N = 2). The first reason was mentioned by five participants (P2, P5, P6, P8, and P10). For example, P2 stated: "*what is not very Python-like is that you are working with i and n, and I would say instead of i use element or the like, and then [...] we have yes the n, so [...] you could give the variable a somewhat obvious name*", P8 said that: "*because the only thing that is in there is moving average and so all variables are a variant of moving average, so very difficult to see*", and P10 explained that: "*because of variable names I found it difficult to link to what they were doing exactly*". Further, P2 and P6 found part 2 the most difficult because of the structure, which is unclear or difficult according to them. P2 said: "*that looks more complicated than it seems necessary to me*", and P6 said: "*there is more casting and more things put together without really clear intermediate steps*". Two participants (P7 and P10) mentioned that there were many variables in part 2 which makes it difficult to remember what each variable contained.

4.2.3.3 process_list (part 3). Only one participant (P4) expressed that part 3 is the most difficult part to understand. The reason that was given is that (s)he was unfamiliar with some concepts that are used in part 3. P4 stated that: "*with that new list, append and item capitalize, that makes the output difficult for me because I don't know exactly what it's doing there now*".

4.2.4 Summary reading focus

Almost all participants focused on the *code behavior*, except for one participant (P8), who only focused on the *code organization*. Furthermore, there are four participants (P2, P3, P9, and P10) who focused on the *code behavior* as well as *code organization*. From the participants who focused on the *code behavior*, most of them (8 out of 10) mentioned that they focused on the *execution flow* and/or *control flow statements*. Both, *execution flow* and *control flow statements*, determine the execution order of the program that the computer will go through. Other elements of the code behavior such as *variable usage/changes* and *functions/methods*, were less frequently mentioned. Further, regarding the *code organization*, the one participant that only focused on the *code organization* (P8) mentioned both *documentation/comments* and *variable names and function names*. The other four participants (P2, P3, P9 and P10) who mentioned *code organization* only stated one of those elements in their answers, whereby two of them mentioned the *documentation/comments* and the other two mentioned the *variable names and function names*.

When it comes to the most important part of the code, nearly all participants mentioned the *main function (part 4)* as an important code part. The main reason that was given is that this part is

the main function where the actual execution of the program is taking place. Furthermore, one participant (P5) mentioned the *moving_average (part 2)* as an important code part, because of its difficulty. However, (s)he also mentioned that *check_pattern (part 1)* is an important code part (P5), together with other two participants (P6 and P8) who said that part 1 is the most important part. The reason that was given for part 1 as an important part is that this part determines the code behavior. The *check_pattern* function determines whether the *moving_average* function is called or the *process_list* function is called. However, they (P6 and P8) also mentioned that the *main function (part 4)* is important, along with the seven other participants who only mentioned the *main function (part 4)*. This part is, according to them, the most important part because this is the main function where the actual execution of the program is taking place. One interesting observation is that none of the participants mentioned the *process_list (part 3)* as an important code part. Possible reasons are 1) part 3 is less relevant to the execution of the code and 2) less complicated than other parts.

Regarding the most difficult part of the code, most of the participants (6 out of 10) found the *moving_average (part 2)* as the most difficult part to understand. This part was seen as the most difficult is mainly because of the variable names, which were unclear according to the participants. The other reasons that were given are unclear/difficult structure and that there are many variables in it. In second place is the *check_pattern (part 1)*, which was mentioned by three participants as the most difficult part. The reasons that were given are: 1) *nested loops*, 2) *many statements*, and 3) *unfamiliar concepts*. Then, *process_list (part 3)* was only mentioned by one participant (P4) as the most difficult part, this is because (s)he was unfamiliar with multiple concepts that are used in this part. Lastly, none of the participants mentioned the *main function (part 4)* as a difficult part.

5 Discussion

Code reading is often seen as a granted skill for programmers. However, the development of this skill is not as good as programming educators expected. Also, it seems that novice programmers read a source code more like a natural language text compared to expert programmers [PSA20]. This thesis further explored what the reading strategies are that the novice programmers say to use when reading a natural language text versus a source code. Furthermore, this thesis also took a deeper look into the reading process of novice programmers based on an example code, i.e. what do novice programmers focus on while reading the example code. Therefore, ten novice programmers were interviewed during a think-aloud semi-structured interview.

5.1 Reading a natural language text versus a source code

According to the participants, the strategies that they primarily use to read a scientific text are *skimming* (N = 8) and *scanning* (N = 6). Skimming is a reading strategy that is used to get a general idea of a text by reading sections like the introduction and headings. Scanning is a reading strategy whereby the reader searches for an interesting part of a text. When reading a scientific text, these two strategies are often combined, according to the participants. In this research, five participants declared to use both strategies. Furthermore, the least mentioned reading strategy is *intensive reading* (N = 3). It seems that when reading a scientific text, most of the participants tend not to read the whole text, but try to get a general idea of the text first, thus what the text is

roughly about. Then based on that information, they will further search for what they want to know. One possible explanation of why they are not reading the whole text is that they are not reading it for entertainment, but because they need something from that text, for example, they need some information for their research. Therefore, they will only read the needed parts to see if there is any information from the text they can use. For example, they will look at the headings (N = 6) to see what each part is roughly about, and they will look at the introduction (N = 4) to see in which direction the text is going.

The common reading strategies that the participants said they use to read a source code are 1) to *follow the code's execution* (N = 5) and 2) to *look at the code's structure* (N = 4). Other reading strategies that are mentioned by at least two participants are: 1) *scan for function names and/or variable names*, 2) *read comments/documentation first*, and 3) *determine the programming language*. From all these reading strategies, it seems that the participants are not reading the whole code. Instead, they try to get a basic understanding of the code, i.e. what the code is about, by looking for example at the structure of the code, names of the functions and variables, and comments/documentation. This is actually in a way similar to the reading strategy *skimming*, which is also used to get a basic understanding of a text.

However, according to the participants, there are two important differences between reading a natural language text and reading a source code, which are the *reading order* and *reading goal*. The most frequently mentioned difference is the *reading order*. In their perspective, a text has a linear reading order where you start for example with an introduction and end with a conclusion, whereas, a code has an executive order where the different sub-functions are called from the main function. When the calling of these sub-functions is followed, the reading order becomes non-linear because these sub-functions can be spread through the entire script. The other important difference that the participants mentioned is the *reading goal*. With a code, we can run the script, there is an output, whereas, with a text, this is not the case. A code must do something, it is not static. For example, values can be stored in variables and changed through function calls. Therefore, participants mentioned that, while reading a code, they do not only read what is written in 'text', but also what is stored and executed behind that 'text'. In contrast, a scientific text is deemed static, everything is written down, they only need to read what is written and nothing will be changed. Thus, with a code, participants try to find out what the code will do, whereas, with a text, they only want to get informed about some topics.

The above findings are in accordance with what Blascheck and Sharif [BS19] have found in their study: reading a source code in a linear way like a natural language text is only partially applicable, even to novices, participants also tend to follow the execution order of the code.

5.2 Reading focus based on an example code

The reading focus that was mentioned by the participants can be divided into two main categories: 1) *focus on code behavior* and 2) *focus on code organization*. The category *focus on code behavior* has four sub-categories, which are 1) *execution flow*, 2) *control flow statements*, 3) *variable usage/changes*, and 4) *functions/methods*. The category *focus on code organization* has two sub-categories, which are 1) *documentation/comments* and 2) *variable names and function names*. Based on the example

code the participants have read and from their answers, it seems that most of them focused on the behavior of the code, i.e. what is the code doing and what is changed through the execution of the code. First of all, most of them mentioned that they focused on the *execution flow* (N = 5), this is the order in which a computer will go through the script, and the *control flow statements* (N = 5), e.g. looping (while and for) and decision-making statements (if-else). The *execution flow* is often followed by looking at the main function. Some of the participants also mentioned that they do not look at the sub-functions if they are not relevant. Following the main execution of the code means that irrelevant parts of the code, i.e. those that are less important to the execution of the script, would be skipped. Therefore, the participants do not need to read the whole code. Furthermore, the *control flow statements* also determines which part of the code is used and which part of the code not. The if-else statement in the example code is a good example of this: if the participants think that only the if-part is applicable, then the function *moving_average* will be called whereas the function *process_list* will not be called because that is in the else part. This means that the participants only tend to look at the *moving_average* function and would skip the *process_list* function, which the results of this study seem to confirm.

The other elements of reading focus that were mentioned by the participants are: 1) *variable usage/changes* (N = 3) and 2) *functions/methods* (N = 2). With functions, the participants especially looked at which input the function gets and which output the function returns. *Variable usage/changes* can also be relevant for the behavior of the code, because, for example, if a variable is used in the decision-making statements, then this variable can determine which decision is made by the computer when running the script. This also applies to the functions. For example, if a function is called from the main function and the return value of this function is determining the remaining code behavior, then this function is certainly more relevant. This was observed with the *check_pattern* function in the example code, which determines whether the if statement is applicable or the else statement is applicable in the main function.

The above findings are in accordance with the patterns that are found by Blascheck and Sharif [BS19]: "*participants also focus more on those areas that are important to comprehend core functionality*" [BS19] and "*they skip unimportant constructs such as brackets*" [BS19].

Furthermore, five participants said that they also focused on the *code organization*: 1) *documentation/comments* (N = 3), and 2) *variable names and function names* (N = 3). By reading these elements of the code, the participants tried to get an understanding of the code in its natural language manner without reading the actual script. An interesting observation is that the three participants who mentioned that they focused on the function names and variable names are those students from the computer science bachelor. Clear function names and variable names will make it easier for programmers to understand the code. If a function has a clear name that is representative of what it should do, then we already can have an idea of what the function will do without exactly reading it. Only looking at the name might be enough to understand what the output is of the function. This is also applicable for the comments which are used to explain certain parts of the code or the entire code. Then looking at those comments, you already know what the code is doing in general without looking at the entire code.

According to Lawrie et al. [LMFB06]: "*readers of programs have two main sources of domain*

information: identifier names and comments" [LMFB06]. Further, in several studies [LMFB06, SE19, HSH17] it is concluded that longer identifier names are better for comprehension than shorter identifier names like letters or abbreviations, but also function names that reveal the intention of the functions can improve the comprehension process of programmers when reading a source code.

Besides what the participants focused on, they were asked about which part of the example code they found the most important. Most of the participants (N = 7) said that the main function is the most important part, which is in accordance with what is discussed above. The main function is the part that determines in which order the computer executes the script and which functions are called. This is also the main reason why the main function is seen as most important. Another part that is mentioned by the participants (N = 3) is the *check_pattern* function. The *check_pattern* function is seen as (most) important because the return value of this function determines whether the if statement is executed or the else statement is executed in the main function. Also, this is in accordance with what is discussed above. The remaining two functions (*moving_average* and *process_list*) are considered less important because these two functions are not determinant for the execution order of the code. The calling of these two functions actually depends on the return value of the *check_pattern* function.

Lastly, the participants were asked to give the part which they found the most difficult in the example code. The most frequently mentioned code part is the *moving_average* function (N = 6). This is mainly because this part contains unclear variable names (N = 5). The participants explicitly mentioned that i and n in the *moving_average* function are unclear, which is in accordance with the results of several studies [LMFB06, SE19, HSH17], that concluded that shorter identifier names such as letters or abbreviations make it more difficult for programmers to understand a code. Another possible explanation for why this part is difficult for the participants to understand is that the participants may be unfamiliar with the concept "*moving average*". If they know what "*moving average*" means, it might be easier for them to understand this function. In that case, the name "*moving average*" would be a more clear name about what the function should do. However, if they do not know what the meaning of the concept "*moving average*" is, then only using the name "*moving average*" reveals almost nothing about the intention of the function. For example, the function/variable might as well have been called "*Python*". However, naming a function/variable like this makes it harder for programmers to understand the intention of the function/variable.

5.3 Other

This study also collected data about participants' explanation of the example code (summary of the code) and the possible reading order of the example code. However, this data is not further analyzed and used in this research because of three reasons. The first reason is that the majority of the participants gave an incorrect explanation about the code, which makes it hard to deduce something useful about a programmer's comprehension in relation to the reading order of the code from these code summaries. The second reason is that the reading order proved hard to detect through a retrospective think-aloud study, where the subjects might not fully remember in which order they have read the code, especially as the reading order of a code is often non-linear. The third reason is that the reading order depends on the questions that were asked in the interview which makes it inconsistent when asking the participants to give their reading order. For example, after the first time they read the code, they were asked to give a summary of the code. However,

when asking them to give the output of the code, they often went back to the code and some of the participants needed to read the code again which makes this reading order inconsistent as the reading goal differs.

6 Limitations

In this thesis, we only interviewed a small number of novice programmers. Because of this small number, the gathered data might not be representative of all novice programmers. If the data would be collected from a larger group, the results might be different than the results that are shown in this thesis. Although the results discussed here may not be generalizable, the intent of this research was to gather a qualitative, in-depth understanding of novice programmers' reading process.

Furthermore, this thesis used think-aloud interviews to gather data. A known issue with think-aloud interviews is that the subjects might have difficulties with verbalization of their thoughts, such as they do not finish some sentences and they repeat the words they already have said multiple times. This makes it difficult to clearly distinguish what the participants have said and what they intended to say. In addition, a limitation specific to the retrospective think-aloud method is that the subjects might not fully remember what their actions were during the problem solving (reading process). When they cannot actually remember what they have done, their answers might not be consistent with other given answers and with what they truly have done during the reading process.

Lastly, the general strategies participants say to use to read a natural language text are based on a scientific paper. However, this example might not be representative of other text types such as a book, news article, or blog. Each text type might have a different reading strategy adjusted to it. Some participants also explicitly mentioned in their answers that their reading strategies differ per reading goal; as in they will first define why they are reading a certain text, and based on that, they will change their reading strategies.

7 Conclusions

Code reading is an important programming ability since it influences other programming skills [LAF+04, LFT09, LWRL08, PRW07]. However, reading skills are rarely explicitly taught to novice programmers in programming education. Therefore, teaching novice programmers this skill explicitly might improve their programming skills overall. As there is little knowledge about the reading processes that novice programmers have developed themselves, we conducted a study to explore the common reading strategies novice programmers say to use when reading a source code and investigate the reading focus that novice programmers had when reading an example code. We have found that the participants mostly focused on the parts or elements of the code which are important to the comprehension or execution of the program. A source code differs from a natural language text, however, it appears that novice programmers tend to read a source code more like a natural language text [PSA20]. Therefore, we also collected data about the reading strategies novice programmers say to use when reading a natural language text and compared it to the reading strategies for a source code to determine what the relationship is between these two. We found that the reading strategies for a natural language text are only partially applicable in the reading

strategies for a source code among the participants. Most of the participants tend to follow the execution order of the program instead of the text order.

8 Further research

In this thesis, we only looked at one type of natural language text, namely scientific paper. However, there are different text types and each text type might have another reading strategy applied to it. Therefore, in further research, we can take several text types into account to see what the relations are between reading a natural language text of different types and reading a source code. In this thesis, the retrospective think-aloud method is used to explore programmers' reading focus, however, combining the eye-tracking method with the retrospective think-aloud method might reveal more about a programmer's reading focus. First using the eye-tracking method to explore which focus areas there exist and then applying the retrospective think-aloud method afterwards would reveal more information about the thinking process of programmers. Therefore, in further research, we can combine the eye-tracking method with the retrospective think-aloud method when aiming to explore more about a programmer's reading focus.

References

[BBB+15] Teresa Busjahn, Roman Bednarik, Andrew Begel, Martha Crosby, James H Paterson, Carsten Schulte, Bonita Sharif, and Sascha Tamm. Eye movements in code reading: Relaxing the linear order. In *2015 IEEE 23rd International Conference on Program Comprehension*, pages 255–265. IEEE, 2015.

[Bed12] Roman Bednarik. Expertise-dependent visual attention strategies develop over time during debugging with multiple code representations. *International Journal of Human-Computer Studies*, 70(2):143–155, 2012.

[BS13] Teresa Busjahn and Carsten Schulte. The use of code reading in teaching programming. In *Proceedings of the 13th Koli Calling international conference on computing education research*, pages 3–11, 2013.

[BS19] Tanja Blascheck and Bonita Sharif. Visually analyzing eye movements on natural language texts and source code snippets. In *Proceedings of the 11th ACM Symposium on Eye Tracking Research & Applications*, pages 1–9, 2019.

[BSB11] Teresa Busjahn, Carsten Schulte, and Andreas Busjahn. Analysis of code reading to gain more insight in program comprehension. In *Proceedings of the 11th Koli Calling International Conference on Computing Education Research*, pages 1–9, 2011.

[BSS+14] Teresa Busjahn, Carsten Schulte, Bonita Sharif, Andrew Begel, Michael Hansen, Roman Bednarik, Paul Orlov, Petri Ihantola, Galina Shchekotova, and Maria Antropova. Eye tracking in computing education. In *Proceedings of the tenth annual conference on International computing education research*, pages 3–10, 2014.

[BT06] Roman Bednarik and Markku Tukiainen. An eye-tracking methodology for characterizing program comprehension processes. In *Proceedings of the 2006 symposium on Eye tracking research & applications*, pages 125–132, 2006.

[CLT11] Malcolm Corney, Raymond Lister, and Donna Teague. Early relational reasoning and the novice programmer: swapping as the'hello world'of relational reasoning. In *Proceedings of the Thirteenth Australiasian Computing Education Conference:*, pages 95–104. Australian Computer Society, 2011.

[CS91] Paul Chandler and John Sweller. Cognitive load theory and the format of instruction. *Cognition and instruction*, 8(4):293–332, 1991.

[Gué06] Yann-Gaël Guéhéneuc. Taupe: towards understanding program comprehension. In *Proceedings of the 2006 conference of the Center for Advanced Studies on Collaborative research*, pages 1–es, 2006.

[HOM+08] Aulikki Hyrskykari, Saila Ovaska, Pävi Majaranta, Kari-Jouko Räihä, and Merja Lehtinen. Gaze path stimulation in retrospective think-aloud. *Journal of Eye Movement Research*, 2(4), 2008.

[HSH17] Johannes Hofmeister, Janet Siegmund, and Daniel V Holt. Shorter identifier names take longer to comprehend. In *2017 IEEE 24th International conference on software analysis, evolution and reengineering (SANER)*, pages 217–227. IEEE, 2017.

[IPW17] Cruz Izu, Cheryl Pope, and Amali Weerasinghe. On the ability to reason about program behaviour: A think-aloud study. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, pages 305–310, 2017.

[LAF+04] Raymond Lister, Elizabeth S Adams, Sue Fitzgerald, William Fone, John Hamer, Morten Lindholm, Robert McCartney, Jan Erik Moström, Kate Sanders, Otto Seppälä, et al. A multi-national study of reading and tracing skills in novice programmers. *ACM SIGCSE Bulletin*, 36(4):119–150, 2004.

[LFT09] Raymond Lister, Colin Fidge, and Donna Teague. Further evidence of a relationship between explaining, tracing and writing skills in introductory programming. *Acm sigcse bulletin*, 41(3):161–165, 2009.

[LMFB06] Dawn Lawrie, Christopher Morrell, Henry Feild, and David Binkley. What's in a name? a study of identifiers. In *14th IEEE International Conference on Program Comprehension (ICPC'06)*, pages 3–12. IEEE, 2006.

[LST+06] Raymond Lister, Beth Simon, Errol Thompson, Jacqueline L Whalley, and Christine Prasad. Not seeing the forest for the trees: novice programmers and the solo taxonomy. *ACM SIGCSE Bulletin*, 38(3):118–122, 2006.

[LWRL08] Mike Lopez, Jacqueline Whalley, Phil Robbins, and Raymond Lister. Relationships between reading, tracing and writing skills in introductory programming. In *Proceedings of the fourth international workshop on computing education research*, pages 101–112, 2008.

[PIS17] Patrick Peachock, Nicholas Iovino, and Bonita Sharif. Investigating eye movements in natural language and c++ source code-a replication experiment. In *International Conference on Augmented Cognition*, pages 206–218. Springer, 2017.

[PRW07] Anne Philpott, Phil Robbins, and J Whalley. Assessing the steps on the road to relational thinking. In *Proceedings of the 20th annual conference of the National Advisory Committee on Computing Qualifications*, volume 286, 2007.

[PSA20] Norman Peitek, Janet Siegmund, and Sven Apel. What drives the reading order of programmers? an eye tracking study. In *Proceedings of the 28th International Conference on Program Comprehension*, pages 342–353, 2020.

[S+12] Juha Sorva et al. *Visual program simulation in introductory programming education*. Aalto University, 2012.

[SCL+08] Judy Sheard, Angela Carbone, Raymond Lister, Beth Simon, Errol Thompson, and Jacqueline L Whalley. Going solo to assess novice programmers. In *Proceedings of the 13th annual conference on Innovation and technology in computer science education*, pages 209–213, 2008.

[SE19] Christina Sunnegårdh and Klara Eserstam. Intention-revealing function names and small functions to facilitate code comprehension, 2019.

[SFM12] Bonita Sharif, Michael Falcone, and Jonathan I Maletic. An eye-tracking study on the role of scan time in finding source code defects. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 381–384, 2012.

[TFSL14] Rachel Turner, Michael Falcone, Bonita Sharif, and Alina Lazar. An eye-tracking study assessing the comprehension of c++ and python source code. In *Proceedings of the Symposium on Eye Tracking Research and Applications*, pages 231–234, 2014.

[UNMM06] Hidetake Uwano, Masahide Nakamura, Akito Monden, and Ken-ichi Matsumoto. Analyzing individual performance of source code review using reviewers' eye movement. In *Proceedings of the 2006 symposium on Eye tracking research & applications*, pages 133–140, 2006.

[UNMM07] Hidetake Uwano, Masahide Nakamura, Akito Monden, and Ken-ichi Matsumoto. Exploiting eye movements for evaluating reviewer's performance in software review. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 90(10):317–328, 2007.

[VSBS94] MW Van Someren, YF Barnard, and JAC Sandberg. The think aloud method: a practical approach to modelling cognitive. *London: AcademicPress*, 1994.

[VTL09] Anne Venables, Grace Tan, and Raymond Lister. A closer look at tracing, explaining and code writing skills in the novice programmer. In *Proceedings of the fifth international workshop on Computing education research workshop*, pages 117–128, 2009.

[WK14] Jacqueline Whalley and Nadia Kasto. A qualitative think-aloud study of novice programmers' code writing strategies. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*, pages 279–284, 2014.

[WLT+06] Jacqueline L Whalley, Raymond Lister, Errol Thompson, Tony Clear, Phil Robbins, PK Ajith Kumar, and Christine Prasad. An australasian study of reading and comprehension skills in novice programmers, using the bloom and solo taxonomies. In *Conferences in Research and Practice in Information Technology Series*, pages 243–252. Australian Computer Society, 2006.

[XLN+19] Benjamin Xie, Dastyni Loksa, Greg L Nelson, Matthew J Davidson, Dongsheng Dong, Harrison Kwik, Alex Hui Tan, Leanne Hwa, Min Li, and Andrew J Ko. A theory of instruction for introductory programming skills. *Computer Science Education*, 29(2-3):205–253, 2019.

[YAH20] Sabiha Yeni, Efthimia Aivaloglou, and Felienne Hermans. To be or not to be a teacher? exploring cs students' perceptions of a teaching career. In *Koli Calling'20: Proceedings of the 20th Koli Calling International Conference on Computing Education Research*, pages 1–11, 2020.

A Appendix I: Translations

Section	Dutch (original version)	English (translated version)
4.1.1.1	"*ik kijk naar de inhoudsopgave, dus welke tussenkopjes er allemaal zijn, en dan begin ik meestal bovenaan, bij introductie achtig*" (P5)	"*I look at the table of contents, so which subheadings are there, and then I usually start at the top, with something like an introduction*" (P5)
	"*ik zou eerst soort van de introductie en de conclusie lezen*" (P7)	"*I would sort of reading the introduction and conclusion first*" (P7)
4.1.1.2	"*ik ga nooit van boven naar onder lezen bij dat soort teksten, je gaat echt op zoek naar dat wat je te weten wil en wegfilteren wat je al kent of wat je op dat moment niet zo relevant vindt*" (P2)	"*I never read from top to bottom with those kinds of texts, you really go looking for what you want to know and filter out what you already know or what you don't find relevant at that moment*" (P2)
	"*dat je wel echt de kernwoorden d'r uitpakt en, zeg maar, om de kernwoorden heen leest zeg maar, maar ik ga niet een tekst volledig doorlezen*" (P3)	"*you really unpack the keywords in there and [...] read around the keywords [...], but I am not going to read a text completely*" (P3)
	"*eerst de titel natuurlijk en dan de abstract en dan de kopjes en dan gebaseerd daarop ga je dan zoeken wat je wil weten en dan de hele alinea lezen per interessant punt waar ik zoek eigenlijk*" (P9)	"*first, the title of course, and then the abstract, and then the headings, and then based on that, [I] will search for what [I] want to know and then read the entire paragraph per interesting point what I actually search for*" (P9)
4.1.1.3	"*je zegt van het is een lastige tekst en je moet echt, echt helemaal leren om het echt helemaal eigen te maken en dan ga ik heel, dan ga ik heel intensief op in, dat betekent dat ik alinea voor alinea bij wijze van spreken punten highlight*" (P1)	"*you say it's a difficult text and you really have to learn to make it your own and [...] then I'm going into it very intensively, which means that I highlight points paragraph by paragraph, so to speak*" (P1)
	"*zeker als ik het stukje intensief moet lezen en begrijpen, dan dat ik echt zin voor zin er doorheen ga, en ja probeer te markeren voor mij wat daar de belangrijkste dingen zijn*" (P1)	"*especially if I have to read and understand the piece intensively, then I really go through it sentence by sentence, and [...] try to mark for me what the most important things are in there*" (P1)
	"*vervolgens ga ik, ja hangt van af hoe ik moet lezen, moet ik het lezen als een soort van reviewer, moet ik het lezen om het zelf toe te passen*" (P1)	"*then I go, yes depends on how I'm supposed to read, should I read it as some sort of reviewer, should I read it to apply it myself*" (P1)

4.1.2.1	"*als de code is met meerdere functies, zoek ik eigenlijk eerst de hoofdfunctie, dus die functie die alle andere functies aanroept, en dan loop ik er, naja dan doe ik die functies van boven naar beneden en pas als ik echt wil weten, dan kijk ik naar wat de sub-functies doen, dus bij, als ik functie lees, in principe, hoe de computer er zelf doorheen loopt, dus ja probeer de computer stap te volgen behalve dat ik niet naar sub-functies kijk, die zegmaar, die sla ik over*" (P5)	"*if the code is with multiple functions, I first find the main function, [the] function that calls all the other functions, and then [...] I do those functions from top to bottom and only when I really want to know, then I will look at what the sub-functions do. So, when I read [a] function, basically, [it is] how the computer itself runs through it: [I] try to follow the computer step except I don't look at sub-functions, I'll skip those*" (P5)

4.1.2.2	"*terwijl als ik code van developers kijk, die dus meer ervaring hebben die ook classes en methods gebruiken, ja, dat betekent dat je eerst eventjes moet kijken welke klasse, methods zitten daar eigenlijk in, anders weet je niet, ja, hoe je, hoe je de, hoe je de rest moet lezen*" (P1)	"*when I look at code from developers, who have more experience, who also use classes and methods, yes, that means you first have to check which class, methods are actually in it, otherwise you don't know [...] how you have to read the rest*" (P1)
	"*om een lange code te lezen, ik zou eerst bekijken naar of er eventuele classes zijn, deze even kijken wat het class ongeveer is zonder hem exact te lezen*" (P10)	"*to read a long code, I would first check if there are any classes, roughly see what the class is without reading it exactly*" (P10)
	"*verder kijk je wat voor functies d'r gedefinieerd zijn en wat voor imports daar zijn want dat geeft ook al een idee*" (P2)	"*you also look at which functions are defined there, and which imports are there because that also gives an idea*" (P2)

4.1.2.3	"*ik zou daarna denk ik even door de functienamen heen scannen*" (P10)	"*I think I would scan through the function names*" (P10)
	"*naar de functienamen kijken dat vooral*" (P9)	"*looking at the function names, especially that*" (P9)
	"*dan probeer ik meestal een beetje te kijken naar de functienamen en ook de algemene variabel namen*" (P8)	"*I usually try to look a bit at the function names and also the general variable names*" (P8)

4.1.2.4	"*als je naar een Python programma kijkt dan staat daar als het goed is, aan de bovenkant staat er een docstring en daar ga je dan denk ik als eerst naar kijken*" (P2)	"*if you look at a Python program, there should be a docstring at the top [...] and I think that's the first thing you should look at*" (P2)
	"*je begint natuurlijk met de documentatie, dat is het belangrijkste*" (P3)	"*of course, you start with the documentation, that's the most important*" (P3)
	"*natuurlijk ook zoek je naar commentaar, dat is natuurlijk wel het belangrijkst, want dat legt gewoon uit hoe de code werkt*" (P8)	"*of course, you also look for comments, which is, of course, the most important, because that just explains how the code works*" (P8)

4.1.2.5	"*ik merk wel dat ik per taal wel geneigd ben om mijn leesstijl een beetje aan te passen zeg maar, ik moet zeggen dat ik Java niet zo graag lees als Python*" (P1)	"*I do notice that I tend to adjust my reading style a bit per language, but I have to say that I don't like reading Java as much as Python*" (P1)

4.1.2.6	"*ik denk dat ik vooral kijk naar de complexe delen, dat ik, gewoon direct alle variabelen enzo, dat ik daar niet eens naar kijken en dat ik dan gewoon zeg van oh nou kijk hier is een complexe deel, daar begin ik wel van wat doet het ongeveer*" (P3)	"*I think I mainly look at the complex parts [. . .] that I just say oh well look here is a complex part, I'll start with that and what does it roughly do*" (P3)
	"*natuurlijk als het goede code in ieder geval goed opgedeeld in functies en gedeeltes, dus probeer wel alles op te splitsen in verschillende stukken om makkelijker te begrijpen*" (P8)	"*of course, if it's a good code, at least well organized in functions and parts, so try to break everything down into different pieces to make it easier to understand*" (P8)
	"*alle if statements en dergelijk kijken, dus waar alle, alle beslissingen in een code eigenlijk en daarna ga ik pas kijken naar wat er precies gebeurt met variabelen en dergelijk*" (P9)	"*look at all if statements and the like, so where [...] all decisions in a code actually are and only then I look at what exactly happens with variables and the like*" (P9)

4.1.3.1	"*de volgorde is niet altijd van links naar rechts, boven naar beneden*" (P6)	"*the order is not always left to right, top to bottom*" (P6)
	"*bij een tekst ga je van boven naar beneden door de tekst heen, vaak van begin tot einde, waarbij bij de code door functies en dergelijke is het niet van boven naar beneden lezen, maar ga je ook soms ineens een stukje naar boven, stukje naar onder*" (P10)	"*with a text, you go through the text from top to bottom, often from beginning to end, [whereas] with a code, through functions and the like, you don't read it from top to bottom, but sometimes you also suddenly go up a bit, go down a bit*" (P10)

4.1.3.2	"*bij code denk ik toch wel vaak geneigd om het richting het praktische gebruik van die code te kijken, dus hé, het moet, het moet runnen zeg maar*" (P1)	"*with code, I think we often tend to look at the practical use of that code, so it has to run, so to speak*" (P1)
	"*bij code wil je doorgronden wat het doet*" (P2)	"*with code you want to understand what it does*" (P2)
	"*een code moet daadwerkelijk iets doen, dus ik denk dat je in een code juist op zoek gaat naar datgene wat de code moet doen*" (P4)	"*a code must actually do something, so I think that you are looking in a code for what the code must do*" (P4)
	"*een tekst is vooral bedoeld om te informeren*" (P2)	"*a text is primarily intended to inform*" (P2)
	"*in een tekst, ga je ook wel op zoek naar de kern, maar dan lees je bijvoorbeeld eerst de alinea, eerst inleiding en de slot als je wil weten waar de tekst over gaat*" (P4)	"*in a text, you also look for the core, but then you read, for example, [. . .] first the introduction and the conclusion if you want to know what the text is about*" (P4)

4.2.1.1	"*nou ik begin dus zelf bij de uitvoer, zo van wat doet, wat print die daadwerkelijk af*" (P3)	"*so, I start with the output myself, so what does it do, what does it actually print*" (P3)
	"*waar ik dan meest op focus bij dan bijvoorbeeld die eerste was die, even kijken, die tweede while loop om daar te begrijpen, en bij part twee is het eigenlijk ook op die while loop van wat gebeurt daar, dus daar focus ik steeds op*" (P4)	"*what I focus most on, for example, with [part one] was [...] the second while loop, to understand that, and with part two it's actually also on that while loop, what happens there, so I always focus on that*" (P4)
	"*ik focus het meest op verandering van variabelen*" (P5)	"*I focus most on the change of variables*" (P5)
	"*toen ik de output moest gaan vinden, toen ging ik echt meer kijken wat er met de waardes gebeurt*" (P7)	"*when I had to find the output, then I really started looking more closely at what happens to the values*" (P7)
	"*daarna mijn meest focus lag op het bijhouden wat variabelen bevatten om te kijken wat ze daadwerkelijk deden, van als bijvoorbeeld variabelen vergeleken werden dat ik in mijn hoofd van oké deze variabele is dit en dit is dat, dus wat komt er uit deze vergelijkingen*" (P10)	"*my main focus has been on keeping track of what variables contain to see what they actually did, like if variables were being compared, that in my head, I'm like okay this variable is this and this is that, so what comes out of these comparisons*" (P10)
	"*op de functies en wat de input en de output van de functies is*" (P2)	"*on the functions and what the input and output of the functions is*" (P2)
	"*dan ga ik gewoon een beetje scannen van hé wat zijn de belangrijke methodes*" (P3)	"*I will see from which moments which headers are called, and what comes in and what comes out*" (P6)

4.2.1.2	"*in dit geval is er geen commentaar maar normaal ook echt commentaar*" (P8)	"*in this case there are no comment but normally also [...] comments*" (P8)
	"*van deze code, ik merk wel dat mijn focus eerst, wel ging ook naar de variabelen naam en de functienaam om te kijken of ik daar al wat kan afleiden*" (P10)	"*I notice that my focus first went to the variable name and the function name to see if I can already deduce something from there*" (P10)

4.2.2.1	"*in ieder geval de check pattern functies, functie, vind ik belangrijk, die bepaalt wel welke andere functie wordt uitgevoerd*" (P5)	"*in any case, I find the check pattern [...] function [...] important, which determines which other function is executed*" (P5)
	"*deel één is ook heel erg leidinggevend voor het gehele gedrag, dus ik zal toch zeggen deel één*" (P6)	"*part one is also very much directing the whole behavior, so I'll say part one*" (P6)
	"*check pattern zou je natuurlijk wel iets belangrijker kunnen zien dan twee en drie want die wordt gebruikt om te bepalen of het twee of drie doet, maar nou ja eigenlijk alle code hangt af van de check pattern functie dus als ik moet kiezen zal ik part één*" (P8)	"*check pattern could, of course, be seen as a little more important than [part] two and three because that is used to determine whether it does [part] two or three, but [...] actually all code depends on the check pattern function so if I have to choose, I will choose part one*" (P8)

4.2.2.2	"*de moving average vind ik belangrijk, maar dat was de moeilijke, [...] moeilijkst te begrijpen vond*" (P5)	"*I think the moving average is important because that was the [...] hardest to understand*" (P5)

4.2.2.3	"*het belangrijkste is natuurlijk part four want daar gaat het werk gedaan worden*" (P2)	"*the most important is, of course, part four, because that's where the work will be done*" (P2)
	"*dat laatst deel, zeg maar deel vier, dan komt allemaal bij elkaar, dat maakt het wel duidelijker wat er gebeurt soort van ofzo als je het echt soort van gaat uitvoeren*" (P7)	"*that last part, [...] part four, then it all comes together, that makes it clearer what happens [...] when you really [...] start running it*" (P7)
	"*het belangrijkst is denk ik toch wel het part vier heet het geloof ik, stuk waarin andere functies worden aangeroepen en vanuit daar begint het eigenlijk het proces van welke kant de code opgaat en ook welke output d'r geshowd wordt*" (P10)	"*I think the most important part is called part four I believe, part in which other functions are called and from there it actually starts the process of which way the code goes, and also which output is shown*" (P10)
	"*ik zou zeggen deel vier omdat die de hoofdfunctie is*" (P6)	"*I would say part four because that is the main function*" (P6)
	"*part vier is de main functie, ja voor de rest het is part één, twee en drie dat zijn gewoon verschillende andere functies*" (P8)	"*part four is the main function, [...] part one, two and three are just different other functions*" (P8)
	"*anders werkt je code niet*" (P3)	"*otherwise your code won't work*" (P3)
	"*want daar druk die dat daadwerkelijk af*" (P3)	"*that's where they actually print it*" (P3)

4.2.3.1	"*de combinatie van while, ifs, while, ifs, fors, dat soort dingen, dat hele gelag zeg maar maakt het lastiger om het te doorgronden*" (P1)	"*the combination of while, ifs, while, ifs, fors, things like that, that whole thing makes it more difficult to understand*" (P1)
	"*omdat het heel veel lagen heeft, het is een if statement in een while statement in een for loop en een if statement in een while statement, dus op een gegeven moment raak ik een beetje de draad kwijt van wat wat doet, ja wat wat doet*" (P9)	"*because it has a lot of layers, it's an if statement in a while statement in a for loop and an if statement in a while statement, so at some point, I kind of lose track of what does what*" (P9)
	"*het moeilijkst, nou dat wordt het dan sowieso deel één, want daar staan gewoon de meeste statements in*" (P3)	"*the hardest part, well that will be part one, because that's where most of the statements are*" (P3)
	"*dat gedrag rond een break, daar zit ik soms wel mee te stoeien zeg maar, wat gebeurt er nou als die in een if else ding zit, in een while, wat gebeurt er dan, dat gedrag is voor mij niet zo voorspelbaar*" (P1)	"*that behavior around a break, I sometimes struggle with it, so to speak, what happens if it is in an if-else thing, in a while, what happens then, that behavior is not so predictable to me*" (P1)

4.2.3.2	"*wat niet erg Python like is, is dat je met i en n werkt, en dan zou ik zeggen gebruik in plaats van i element ofzo, en dan vervolgens hebben we ja de n, dus je zou de, het variabele zou je een wat duidelijke naam kunnen geven*" (P2)	"*what is not very Python-like is that you are working with i and n, and I would say instead of i use element or the like, and then [...] we have yes the n, so [...] you could give the variable a somewhat obvious name*" (P2)
	"*want het enige een beetje daar zit is moving average en dus alle variabelen zijn een variant van moving average, dus heel lastig om te zien of de vervolgen*" (P8)	"*because the only thing that is in there is moving average and so all variables are a variant of moving average, so very difficult to see*" (P8)
	"*doordat er variabelen namen vond ik lastig te koppelen aan wat ze exact deden*" (P10)	"*because of variable names I found it difficult to link to what they were doing exactly*" (P10)
	"*dat ziet er ingewikkelder uit dan het mij nodig lijkt*" (P2)	"*that looks more complicated than it seems necessary to me*" (P2)
	"*er wordt meer gecast en meer dingen achter elkaar gezet zonder echt duidelijke tussenstappen*" (P6)	"*there is more casting and more things put together without really clear intermediate steps*" (P6)

4.2.3.3	"*met die new list, append en item capitalize, dat maakt voor mij de output zeg maar wat die print moeilijk omdat ik niet precies weet wat die daar nu doet*" (P4)	"*with that new list, append and item capitalize, that makes the output difficult for me because I don't know exactly what it's doing there now*" (P4)

B Appendix II: Interview Transcriptions

Participant 1
Datum: 18 mei 2021
Duur: 1 uur en 8 minuten

Interviewer:
Ik zal een aantal vragen stellen tot jouw algemene leesstrategieën die je zou gebruiken bij het lezen van een normale tekst en het lezen van een code. Dus kun je me even vertellen wat zijn jouw algemene leesstrategieën die je zou gebruiken om een moeilijk tekst te lezen, bijvoorbeeld een wetenschappelijk artikel?

Participant 1:
Oké, ja goed, ik heb hier een aantal wetenschappelijke artikelen, wat ik doe als ik ze moet lezen is dat ik altijd eventjes gewoon scan zeg maar de letterlijk gewoon van hoeveel bladzijdes überhaupt, waar gaat het eigenlijk over, dus eerst de headline zeg maar, dus ik begin altijd gewoon daar, vooraan, ik kijk altijd eerst even scannend wat er allemaal in staat, waar ik aan begin zeg maar, dus ik blader d'r blader geheel door om te zien van ja waar zitten de hoofdstukken, waar, hoe is de indeling van tekst, waar vind ik verwijzingen zeg maar, dat soort [...], en vervolgens ga ik, ja hangt van af hoe ik moet lezen, moet ik het lezen als een soort van reviewer, moet ik het lezen om het zelf toe te passen, kijk als je, je zegt van het is een lastige tekst en je moet echt, echt helemaal leren om het echt helemaal eigen te maken en dan ga ik heel, dan ga ik heel intensief op in, dat betekent dat ik alinea voor alinea bij wijze van spreken punten highlight, hier ook doe [*participant verwijst hier naar een stukje tekst van een wetenschappelijk artikel die hij heeft gelezen*], dus en met die gele markers geef ik aan wat eigenlijk de belangrijkste zinnen zijn zeg maar in de alinea's en vanuit daaruit ga ik gewoon ja heel, ook doorheen zeg maar, het probleem is dat als ik een lange tekst zeg maar zeker een lange ingewikkelde tekst meteen helemaal lees zonder dat ik daar iets van zeggen, en markers of aantekeningen zet, dat ik dan toch vaak terug moet bladeren om te kijken van oké wat was in het begin belangrijk, waar ging het ook alweer over, ja dus ik gebruik makers en ik gebruik soms ook verschillende kleuren, ik weet niet of het te zien is [*de participant verwijst naar de wetenschappelijk artikel*], hier zit er een rode, rode ding, hier de tekst zeg maar, en als ik dingen vind die voor mij onduidelijk zijn dan zet ik meestal in de kantlijn, zet ik voor mezelf aantekeningen, dus, dus of het, of het, ja of ik daar vragen bij heb, of het voldoende beantwoord is, ja, dus ik probeer eerst zeg maar een globale structuur te krijgen, globale indruk van het stuk en dan zeker als ik het stukje intensief moet lezen en begrijpen, dan dat ik echt zin voor zin er doorheen ga, en ja probeer te markeren voor mij wat daar de belangrijkste dingen zijn, vaak hebben tekst ook wel al van een structuur die je kan aanhouden, en natuurlijk afbeeldingen, van afbeeldingen word ik altijd heel blij van, dit soort dingen, ah leuk, weet je dan, vaak zegt een afbeelding meer dan duizend woorden, daar ben ik altijd [...] van, structuren en afbeeldingen helpt wel.

Interviewer:
Oké, dan is mijn volgende vraag: heb je ook wat algemene leesstrategieën die je zou gebruiken om een code te lezen?

Participant 1:

Nou ja de, ik, ik werk zeg maar als business analist, dus ik ben veel bezig met business teksten bezig, ook wel enigszins technische documentatie, maar het is niet heel vaak dat ik echt scripts lees voor mijn studie, voor big data, data science en data-analyse ben ik wel veel bezig met code, maar wat je vaak bij data science ziet dat het best wel, ja moet zeggen, een beetje spaghetti code achter elkaar wordt geschreven, dus je begint gewoon van boven en regel voor regel vul je het aan zeg maar, terwijl als ik code van developers kijk, die dus meer ervaring hebben die ook classes en methods gebruiken, ja, dat betekent dat je eerst eventjes moet kijken welke klasse, methods zitten daar eigenlijk in, anders weet je niet, ja, hoe je, hoe je de, hoe je de rest moet lezen zeg maar, ja, dat zijn de soort van, de klasse en de method zijn wat mij betreft een soort van hoofdstukindeling waarbinnen weer dingen kunnen staan, hier later waarnaar, ja, verwezen kan worden zeg maar.

Interviewer:
Oké, mijn laatste vraag voor dit onderdeel is: wat is volgens jou het belangrijkste verschil tussen het lezen van een code en het lezen van een tekst?

Participant 1:
Even kijken hoor, het belangrijkste verschil tussen code en tekst.

Interviewer:
Ja, qua lezen.

Participant 1:
Ja qua lezen, dus als je zo review zeg maar, bij code denk ik toch wel vaak geneigd om het richting het praktische gebruik van die code te kijken, dus hé, het moet, het moet runnen zeg maar, het moet niet omvallen, en dan merk ik dat het ja het belangrijkst [...], ik weet dat het onderzoek ook al door Felienne werd, werd aangehaald zeg maar, en ik ben op zich wel een fan van wat zij, wat zij zegt, dus het in principe gewoon zo intuïtief mogelijk te lezen zou moeten zijn, [...], maar ja goed, wat is het belangrijkst punt, ja ik doe nu veel met Python en ik ben daar wel blij mee want ik deed, hiervoor deed ik wel een aantal dingen voor mijn studie met Java en Java vind ik een iets minder fijne taal om te lezen dan Python en Python lees wat meer, ja, functies en de methods, die zijn meestal wel enigszins leesbaar zeg maar, je hebt natuurlijk wel spaties bij Python maar bij Java heb je veel meer van die interpuncties, dus code braces en dat soort zaken, hier dan weer heel erg waar je goed op moet letten, ja ik, als ik code lees, dan helpt het mij als het, als het, als het op is gedeeld in stukjes, net zo met een hoofdstukindeling of alinea indeling in een tekst dus dat je tekst leest dat je wel snel zeg maar een soort van indruk kan krijgen, dat er, wat we met ons groepje doen is dat er, dat we belangrijke onderdelen in je script dat je aangeeft waar het voor dient, bij wijze van spreken een multi-line comment erin zet om te markeren van oké hier begint wat en wat is het dan, ja dus eigenlijk verschilt dat wat dat betreft niet zo heel veel met een artikel ofzo, ik merk wel dat ik per taal wel geneigd ben om mijn leesstijl een beetje aan te passen zeg maar, ik moet zeggen dat ik Java niet zo graag lees als Python.

Interviewer:
Ja, oké. Dat zijn dan mijn vragen. Ik weet niet of je zelf nog opmerkingen hebt over jouw algemene leesstrategieën.

Participant 1:
Nou ja, ik heb een visual studio code, vind ik wel een fijne functie die erin zit is dat je kan uitzoomen zeg maar, dus je ziet de code, zie je staan in je hoofd scherm zeg maar, waar in de kantlijn aan de rechterkant zie je heel erg sterk uitgezoomd de hele structuur van de tekst, dus je kunt eigenlijk aan de vorm zien waar alle, bijvoorbeeld alle libraries importeren zitten en waar het stukje zit waar veel comments in zitten, soms kun je ook aan de structuur zien of er heel veel strings in zitten of dat het juist meer, ja hoe moet je zeggen, een methode is waar je dus meer inspringt, dus op die manier kun je eigenlijk al een beetje structuur krijgen van oké, ziet hier heel veel linkjes binnen, binnen, binnen het, binnen het script of is het gewoon spaghetti van boven naar beneden zeg maar.

Interviewer:
Dan zijn we klaar voor dit onderdeel. Je gaat nu dus een Python code lezen die ik heb geschreven. Straks zal ik dus een aantal vragen, hier vind je dus die code, heb ik in Google Forms gedaan omdat het makkelijker te openen is. Straks zal ik dus nog een paar vragen stellen over jouw interpretatie en jouw leesstrategieën die je hebt gebruikt om deze code te lezen en hierbij mag je natuurlijk aantekening maken als je het wilt. Het gaat uiteindelijk om jouw leesstrategieën, dus je mag zelf kiezen, maar wel als je wel aantekening maakt, dat je dat op jouw notepad op jouw laptop of computer doet, dat je dat straks ook naar me kan sturen, want dat hoort natuurlijk ook bij jouw leesstrategieën die ik nodig heb voor mijn onderzoek. Ja, is dat goed?

Participant 1:
Ja, dus je vraagt nu aan mij om, hier heb je die link gestuurd, die heb ik geopend, je vraagt mij wat ik, wat ik daar als leesstrategie op los zal laten.

Interviewer:
Ja, dus je mag de code lezen zoals je wilt en dan zal ik dus straks een paar vragen stellen tot jouw leesvolgorde en jouw lees focus en nog interpretatie van de code. En je mag het zolang over doen totdat je de code begrijpt. Dus geef een seintje wanneer je klaar bent met lezen. Je mag ook jouw camera uit doen als je dat fijner vindt, dus maakt niet uit.

Participant 1:
Nee, maakt niet uit, maakt niet uit, want ik bedoel het, wat moet ik doen met die knop verzenden, moet ik ergens mee doen.

Interviewer:
Niks, dat was gewoon om het makkelijker te maken, want ik wilde eerst de code gewoon in de chat sturen, maar dat moet gedownload worden.

Participant 1:
Ja, dat is een beetje tricky, want dan vaak is je, is je, wordt indenting, klopt niet meer.

Interviewer:
Ja.

[*De participant begint nu met het lezen van de code, de opmerkingen die de participant heeft gemaakt*

tijdens deze leesprocedure is hieronder genoteerd.]

Participant 1:
Ik zie in ieder geval dat je drie belangrijke onderdelen hebt, nee vier belangrijke onderdelen, ik zie wel dat die onderdelen een beetje grijzige zijn, dus ik zou eerst zeg maar, ik zou direct op het eerst oog gezien, zou ik beginnen waar overal def staat, zou ik beginnen, zou ik zegmaar, ja oke, dus ik zie vier belangrijke blokken, het zijn er puts die daar schrijft, d'r heet eentje check pattern, d'r heet eentje moving average, d'r heet eentje process list, en d'r heet eentje substring, nee, dat is nou geen def, dat is een for loopje, oke, drie code ja, precies, ik had eventjes wat er in gaat, dus wat je input is en wat je output is, per method om te zien van hé de output die bij de ene wordt gebruikt, door de andere weer ingaat ofzo, wat doe je bij input [...], wat staat daar allemaal, dat hele lange [...] ofzo, moet ik nu iets doen dat ik de werking van dit script uitlegt of wat, wat wil je dat ik uitlegt aan jou.

Interviewer:
Zeg maar, wat je hebt geïnterpreteerd van deze code en straks ook de output van deze code om te kijken of je de code echt snapt.

Participant 1:
oké, meteen over het brugje springen, oké, in ieder geval, bij die eerste methode zie ik dat er een input string ingaat en een pattern gaat er in, van bovenaan, dus als de lengte van de input string is, gelijk is aan nul en de lengte van de patroon is gelijk aan nul, dan komt er true uit, als de lengte van de patroon is nul, dan is het, dan komt er ook true uit, dan een while ding, stel dat je lengte van je input string is niet nul, gaat die het in, als pattern in input string, ja precies, dus er komt ook een true of false uit, als dat true is, dan gaat die de for loop in voor een range length puts input string, input string, daar pak je de lengte van, de range van, even kijken hoor, ja je brak dus binnen dat bereik van die lengte, ja en doe je dat dus, stel je input string weet ik wat vier karakters is, daar de lengte is vier, range is dan vier, en dan i, ja die wordt gevuld, [...] pattern pointer begint bij nul, while pattern pointer is kleiner dan de lengte van de patroon, oké, oh wacht sorry, ja, ik zie hem, oké, schakelen waar die pattern pointer, die input pointer vandaan kwam, zit natuurlijk in die for loop, gezet, het stukje met die square bracket heb ik even moeite mee om dat te lezen, if input string, square bracket input pointer is equal to pattern square bracket pattern pointer, weet ik niet precies wat daar het gedrag is, zeg maar, ik gebruik zelf niet echt van die square bracket, dus weet niet precies wat het doet, array dan [...], als die conditie waar is, dan telt die d'r eentje bij op bij die input pointer en de pattern pointer, doe die zolang pattern pointer kleiner is dan de lengte van pattern, wat is pattern pointer ook alweer, ja oke, begint dus bij nul, telt op zolang dat kleiner is dan de lengte van de patroon, en anders dan breekt die eruit, oké, dan krijg je daarna volgend if loop, if ding, als de pattern pointer is gelijk aan de lengte van de pattern, lengte ja, dan index, i wordt gelijk gesteld aan de index en dan breek die eruit, oke, input pointer, die pattern pointer die tellen op totdat je, dat je gelijk lengte, even kijken, while, dus op het moment dat je pattern pointer gelijk is aan de lengte van de pattern, dan gaat die eruit, dan gaat die niet meer in, en dan even kijken hoor, [...] dan doe die voor de volgens mij in range, van die lengte en die range, ik heb dat wel eens eerder gezien, maar ik heb het niet heel veel gebruikt, range en lengte lijken een beetje op elkaar, maar ik weet dat het net iets anders is, maar, gokken, input pointer die groeit ook in die while loop, en wat je index waarde wordt overgenomen, i wordt overgenomen naar

index, ja, dus die index, die groeit ook mee, nee je index wordt gevuld op het moment dat i, while, groter is dan die, groeit, tevens dat die index gelijk aan i, dus ja, zolang dat patroon in die input string zit of past, dan blijft die nog in die if en die groeit zolang die range lengte van die input string zit, stel dat het patroon, pattern niet in de input string zit, dan gaat die naar die else, krijg je return false, je krijgt false d'r uit, en dan ja, [...] input string, dus bij het eerste character, bij, tot aan index, waar wordt i, oh dat is de i, dus input string, index plus de lengte van de patroon, oh ja, ja die krijg je aan het begin [...], index, ik zat te denken wat je op regel drieëntwintig nou het eigenlijk doet zeg maar, dat ding dat kan ik lezen, maar je moet daarvoor dat hele while if for while ding, dat moet je ook in elkaar sleutelen zegmaar mentaal, input string nul tot index en die index, die krijg je, i, die for loop wordt gedaan, oké, dus de input string, waarbij je vanaf index plus de lengte van het patroon, het patroon, dat krijg je binnen bij je method, we hebben de lengte van het patroon, dus dat patroon, weet ik veel, drie karakters is ofzo, dan telt die er drie bij op, en dat, [...] heb natuurlijk een range, die begint bij nul en je gaat tot en met het eind, lijkt het alsof je dat, oh wacht even, nee ik snap het , ja ja, hij slaat dat stukje over wat je dus met lengte van pattern aan karakters opschuift zeg maar, oké dat snap ik oké, en dan hebben we nog in die while dinges, en dan is dat op een gegeven moment klaar, ding is, oh wacht even, ja ja oké, ja, precies, uiteindelijk zou die dus, ik zit na te denken wanneer die uit die while loop gaat, die eerste def, ik gebruik zelf niet heel veel van die breaks om eerlijk te zijn, dus ik weet niet precies dan helemaal uit die, uit dat die dan naar die return true gaat of dat die, of dat die die iteratie afbreekt en dan naar de volgende iteratie gaat, dat moest ik eventjes naar kijken, [...], krijg je in drie gevallen, krijg je true als de input string is en de lengte is van het patroon, is die nul, als het nul is, oké dus, if lengte blablabla, krijg je de, in drie gevallen krijg je true als het input string is het en de lengte is het, van patroon is het [...] dan een nul is, en, als het patroon niet in de input string voorkomt, dan krijg je een false, kijken of het, vaak condities of de input string en de pattern niet helemaal leeg zijn, en of de lengte niet, van de input string groter is dan nul of niet nul is, ja groter is dan nul, en, voorkomt, als het niet voorkomt in de input string, dan krijg je een false, wel voorkomt in de input string, dan gaat die, beetje lastig voor te stellen waar die pattern pointer, oh ja, ja ik snap hem, die square bracket is me duidelijk nu, want dat geeft aan, dus aan op welke index je van die input string of pattern je bezig bent, blijkt, [...] dan schuift die eentje op van de pattern pointer naar de volgende en gaat die weer kijken of de character op de input pointer plaats zelfde is, als de character op het pattern, ja, dat die dan zo door, ze kijken gewoon of het patroon voorkomt in, de pattern voorkomt in die input string,en even kijken hoor, ja dat doe die tot aan zegmaar de complete lengte van de pattern, ja precies tot aan de lengte van de pattern, [...] dan breekt die eruit, dan gaat die naar de if, [...], dat het gelijk aan de lengte van de patroon, is het pattern poiter gelijk aan de lengte van de patroon, [...] van die for loop zet die index, oke, ja als ik dit soort dingen zou gaan uitwerken zeg maar, dan of ik zou het gaan overtypen en dan kijken hoe het zich gedraagt zegmaar, maar ja dat is natuurlijk niet zo hele slimme manier, of ik zal het gaan uittekenen, dus dat je bijvoorbeeld een heel simpel woord pakt, vier letters bij wijze van spreken, niet veel, en dan zo'n pattern pakt van bij wijze van spreken twee of drie characters en dan kijken wat die doet, wat het dan levert, eerst methode, dat als ik helemaal wil doorgronden het gedrag zeg maar van als er nou daarboven boven die lengte uitkomt van pattern, wat doet die dan en als die gelijk is of als die eronder zit, wat doe die dan, en dat is best wel ingewikkeld om allemaal op te slaan zeg maar, om allemaal bij te houden, dus dat is wat beetje gebeurt op de achtergrond, dat ik heel hard aan het nadenken ben, maar dat begrijp je wel, oké, check pattern vandaan, boven [...] ja ja, false uitkomt, true uitkomt, waarde, min tien moving average, [...] ja precies ja, array [...], min

tien, even kijken, is, selector plus i is een shortcode voor een, selector is selector plus een, [...], oh wacht even, gedeeld door, oh daar ja ja, oké ja ja, append nou die voegt die weer toe aan de array, dan wat uit de moving average komt, waarde gedeeld door [...], ja, [...] waardes oké, process list, ja list, dus je krijgt hier, letters, dan moet er wel een true uitkomen, de vraag is wat er nou uitkomt, komt er nou een, wordt print moving average geprint, of wordt print process list geprint, toch.

Interviewer:
Ja, dat moet je zelf even bepalen.

Participant 1:
Ja precies, oké, [...], dat moet ik zelf bepalen, even kijken hoor, wat zullen we zeggen, wat ik dat tot nu toe ben, ik moet een beetje gaan afronden, maar wat ik tot nu toe ben is dat ik de samenhang nu redelijk goed begrijp zeg maar, ik ben nog niet zover dat ik een uitspraak heb op wat er nou precies uitkomt, ik weet wel de samenhang, dat weet ik wel, ik weet wel ja wat er, wat er links en rechts wordt berekend, bepaald of dat soort dingen, maar ik heb even wat meer tijd nodig om helemaal te zeggen wat er nu uitkomt, ja, ik weet er wel hé, als het een, als het, als bij de if check pattern, als dat true is, dan weet ik wel wat, wat die, wat die moving average ongeveer doet en ik weet wat bij de else, wat er, wat er bij de process list wordt gedaan, dat kan ik gewoon uitleggen, maar de uitkomst, die heb ik er nog niet, je moet hier even mee doen.

[*Eind leesprocedure. Deze leesprocedure heeft 41 minuten geduurd.*]

Interviewer:
Dus kunt u de functies even samenvatten? Kunt u de code, wat u nu begrijpt, even samenvatten wat de code doet?

Participant 1:
Ja, de check pattern die heeft een input string en een patroon als, als inputs, die kijkt of, die checkt [...], oké, dus ja, dus dat is de input en wat er uitkomt is d'r, een true of false, afhankelijk van of dat patroon, ja of dat niet nul is of dat patroon en de input string niet nul zijn en of de, ja die input string is, [...] nou dat tweede ding, dus de patroon, nou ze komen, de patroon komt erin voor, zo snel gezegd, ja dus d'r komt een false uit, ja daar komt een false uit want de lengte input string is niet nul, de patroon is niet nul, kijk nou goed, tot daar, tot daar ben ik eventjes voor die part één, en part twee is de moving average, die heeft een, krijgt een, ja values krijgt die d'r in, dat is een array in al een integer, en wat doet die, even kijken hoor, hij berekent telkens het, hij berekent het, het gemiddelde, wacht even hoor, even kijken, ja dus hij kijkt naar de lengte van die array en dan gaat die doorheen itereren, en daar komt op een gegeven moment komt het average values uit terug en wat doet die, dat is, dat is een leeg array en daar voert die de gemiddeldes aan toe, ja die voegt die daaraan toe als waarde in die array, dan part drie is, dat is dus een lijst van hoofdletters, met hoofdletters van maak, capitalize, en dat onderst stuk, die part vier, dat eigenlijk wat dat allemaal samenbrengt, even de substring even kwijt, wacht sorry ja, natuurlijk, ja dus de input string bestaat uit twee, uit twee waardes in je array, voor ieder waarde ga die kijken of het voldoet aan het eerste dingetje hè, dus de check pattern, of het dat het voldoet en anders is het, is het, ja anders dan pakt die hoofdletter waardes uit die super list, zet die, die zet er dan dus in, begint bij de eerste, daar zit, daar zit die, daar zit die string in, bij die

eerste uit die array, even kijken hoor, ja ik denk dat er uitkomt dat er een array uitkomt met de, met de moving average values als een array, ja, ik heb hardop gedacht, ik heb het waarschijnlijk fout.

Interviewer:
Nou maakt niet uit hoor. Maar ik heb nog wat algemene vragen tot jouw lees focus en jouw leesvolgorde. Dus waar denk je dat je op focust tijdens het lezen, dus waar focus je het meest op van deze code?

Participant 1:
Ik focus het meest op zeg maar in volgorde begrijpen wat er gebeurt zeg maar, dus als ik iteraties doorloop, wat gebeurt er dan, dat vind ik eigenlijk het lastigste, daarom houd ik het liefst nog een papiertje bij, wat gebeurt er nou als het optelt tot, als ik een woord string gebruik, wat gebeurt er nou als de pattern is str, wat gebeurt er dan als ik dan str voorbij ben, of dat je dan, dus dat gedrag probeer ik zeg maar uit te splitsen om te zien van tot waar gaat iets en vanaf waar geldt een andere, gaat die een ander pad in zeg maar, ziet een beetje als een proces zeg maar wat die kan doorlopen, maar ik heb helemaal niet gekeken naar die input string beneden, daar kwam ik pas veel later achter, dat is niet zo slim, wat ik beter eigenlijk van begin kunnen zien van hé wanneer gegeven deze methodes, als je nou die input strings hebt, wat gaat dan gebeuren, dan heb je volgens mij veel overzichtelijke verhaal in plaats van dat je het helemaal van scratch moet opbouwen zeg maar.

Interviewer:
Oké, dus welk deel van deze code is volgens jou het belangrijkst om het goed te kunnen snappen?

Participant 1:
Wat zeg je, de, welk deel is het belangrijkst.

Interviewer:
Ja, om goed te kunnen snappen.

Participant 1:
Ja, nou ja, goed dat hele proces, dat zijn die methodes en het samenbrengen van die verschillende processen dat staat onderin en eigenlijk bij bovenaan, net bovenaan bij part vier staat input string en pattern, dat is eigenlijk hetgeen wat je, als dat echt de input hebt zeg maar, maar dat staat dan ergens op zeg maar helemaal aan het eind, dus dat vind ik soms wel lastig dat je, ik ben een beetje een spaghetti lezer zeg maar, ik begin boven en ik eindig onderaan, en daardoor zijn sommige dingen die halverwege of onderaan staan, die doe ik pas eigenlijk het laatste terwijl het soms juist belangrijkste zijn zeg maar.

Interviewer:
Ja, oké. Waar heb je het meest moeite mee tijdens het lezen van deze code?

Participant 1:
Nou de combinatie van while, ifs, while, ifs, fors, dat soort dingen, dat hele gelag zeg maar maakt het lastiger om het te doorgronden, dus als je meer ervaring hebt met geneste for loops, dat je dat dan makkelijker gewoon vanuit de routine zeg maar, oh ik zie hier een for loop, ik kijk gewoon even wat de eind conditie is, zoals dat van java gewend ben, met Python heb ik niet heel veel met for loops

gedaan, maar dan kun je veel beter eigenlijk gewoon aan de kleuren van de code zien, oh wacht even nou dan gaat het ongeveer die kant op uit, terwijl ik ga dat nu allemaal, helemaal lopen beredeneren zeg maar hè, dus ik ben een beetje, ga ik heel wiskundig, ga ik alle opties door om te kijken wat er met je, met je range aan getallen of met je range qua string characters gebeurt, dat zal, ik denk dat als ik meer ervaring zou hebben, zou ik daar ook wel sneller in willen worden om dat, om dat goed te kunnen leren zonder dat ik meteen helemaal laat afleiden van oké, stel nou dat dit nou een string zou zijn van het, van hè, het woordje string, wat gebeurt er dan, dat je veel meer als concepten beschouwd zeg maar, nou ik weet niet of dat realistisch is, maar dat zou, dat zou denk ik wel goed zijn.

Interviewer:
Dus dit is een concept dat je moeite mee hebt. Zijn er andere Python concepten die je ook best wel lastig vindt?

Participant 1:
Even kijken, nou kijk dat, dat gedrag rond een break, daar zit ik soms wel mee te stoeien zeg maar, wat gebeurt er nou als die in een if else ding zit, in een while, wat gebeurt er dan, dat gedrag is voor mij niet zo voorspelbaar, ik bedoel ik ken vanuit Java, ken ik dat vanuit cases dat je op een gegeven moment uit kan breken, nou bij Python zit dat ook ongeveer hetzelfde, maar hé, als je nog niet heel veel ervaring hebt, dan is het, dan kun je zeg maar denken van oké gaat die nou nog, waar breekt die eruit zeg maar, is dat de else conditie, is dat de while loop waar die uitgaat, is dat hé wat gebeurt er dan, waar sprint die dan naar terug, dat is soms lastig voor te stellen zeg maar, ja.

Interviewer:
Dat zijn mijn vragen tot jouw lees focus. Ik heb ook nog een paar vragen tot jouw leesvolgorde, dat zijn echt hele kleine vraagjes. Ik weet niet of je zelf nog iets wilt zeggen over jouw lees focus?

Participant 1:
Nou ik ken mezelf wel een beetje, ik weet wel dat ik, als ik echt goed moet doorgronden, dan ga ik echt, echt tot op de punt in de komma uit zitten, terwijl waarschijnlijk meer ervaren lezer [...], wat ik net ook zei van die geneste for loop, dat makkelijker kunnen uitzoomen zeg maar zonder dat ze meteen helemaal de draad kwijt zijn, dus dat spel daartussen, tussen extreem inzoomen en alleen maar zeg maar van een afstandje kijken, daartussen zitten volgens mij nog wel wat, wat advances die ik gewoon nog niet, niet, niet goed in me.

Interviewer:
Ja, oké. Weet je nog bij welke line je bent begonnen met het lezen?

Participant 1:
Ja, ik ben begonnen, nou ik ben gewoon bij iedere, iedere part één, twee, drie ben ik begonnen, dus ik ben eerst gewoon eventjes de blokjes gaan bekijken, alleen pas later werd me echt duidelijk wat die blokjes waren.

Interviewer:
Dat is dus de scannen fase, toch?

Participant 1:
Ja.

Interviewer:
En met het echt lezen, dat was bovenin begonnen?

Participant 1:
Ja, met het echt lezen ging ik inderdaad bovenin beginnen, maar ik denk eigenlijk achteraf had gezien dat ik bijvoorbeeld beter eerst die het gebruikt, dus vanaf regel eenenvijftig kunnen pakken en dan met name dan regel tweeënvijftig kunnen beginnen, omdat dat gewoon, ik bedoel ja, je laadt die methodes wel, maar je gebruikt ze in de manier zoals je vanaf regel eenenvijftig en verder beschrijft, dus op die manier had ik het beter kunnen aanvliegen denk ik.

Interviewer:
Wat ik nu heb opgemerkt is dat je gewoon de hele code eerst gewoon heel globaal even scant en dan ben je gewoon helemaal bovenin begonnen en dan ga je dus elke functie één voor één analyseren, dat is wat ik heb opgemerkt. Dus je bent geëindigd met het laatst deel?

Participant 1:
Ja.

Interviewer:
Dus je bent vanaf boven begonnen en dan naar beneden gegaan?

Participant 1:
Ja.

Participant 2
Datum: 19 mei 2021
Duur: 32 minuten

Interviewer:
Nu ga ik een aantal vragen stellen tot jouw algemene leesstrategieën die je zou gebruiken om een normale tekst te lezen, maar ook om een code te lezen. Dus mijn eerst vraag is dan: wat zijn jouw algemene leesstrategieën die je zou gebruiken om een moeilijk tekst te lezen, bijvoorbeeld een wetenschappelijk artikel?

Participant 2:
Dat is een open vraag?

Interviewer:
Ja, dat is een open vraag.

Participant 2:

Open vraag.

Interviewer:
Dus vertel me alles wat je denkt dat het is.

Participant 2:
Ja, een moeilijke tekst, eerst de abstract lezen en vervolgens de koppensneller, dus kopjes lezen en aan de hand daarvan inzoomen op dat wat je vooral interessant vindt, ja, ik denk dat daar wel op neerkomt, plaatjes, ik kijk graag naar plaatjes als er plaatjes in een artikel staan, dat is belangrijk, ja, en verder gebruik maak van de pointers die in een tekst aanwezig zijn, zoals kopjes of andere structuurelementen, ik ga nooit van boven naar onder lezen bij dat soort teksten, je gaat echt op zoek naar dat wat je te weten wil en wegfilteren wat je al kent of wat je op dat moment niet zo relevant vindt.

Interviewer:
Ja, oké, dat is het dan. Mijn tweede vraag is dan: wat zijn jouw algemene leesstrategieën die je zou gebruiken om een code te lezen?

Participant 2:
Ja, dus even kijken, als je naar een Python programma kijkt dan staat daar als het goed is, aan de bovenkant staat er een docstring en daar ga je dan denk ik als eerst naar kijken, ja, en verder kijk je wat voor functies d'r gedefinieerd zijn en wat voor imports daar zijn want dat geeft ook al een idee, het is moeilijk om dat zo in zijn algemeenheid te beantwoorden, maar we gaan zo meteen vast nog wel voorbeelden zien.

Interviewer:
Oké, maar heb je al wel een idee van wat het belangrijkst verschil is tussen het lezen van een tekst en het lezen van een code volgens jou?

Participant 2:
Ja, een tekst is vooral bedoeld om te informeren en bij code wil je doorgronden wat het doet, dat is wel een essentieel verschil, dus het gaat veel meer om het begrip van wat een programma gaat opleveren, wat het doel ervan is, bij een wetenschappelijke tekst kijk je meer naar wat ze gedaan hebben en wat het betekent, bij code kijk je of wil je bedenken wat het doet, dat vind ik wel een essentieel verschil.

Interviewer:
Ja, oké. dan zijn we klaar met dit deel. Nu ga je dus een Python code lezen die ik heb geschreven. Het belangrijkst is dat je de code zo lees dat je wel de code gaat begrijpen wat het doet, want straks zal ik namelijk een aantal vragen stellen tot jouw interpretatie van de code, ja, wat de code doet enzo. Maar ook jouw leesstrategieën die je hebt gebruikt bij het lezen van deze code. Dus tijdens het lezen van de code mag je natuurlijk ook aantekeningen maken als je dat wilt. Het gaat uiteindelijk om jouw leesstrategieën, dus jij mag het bepalen hoe je het doet. Dat is wat ik vooral wil vertellen. Hier is de link, het is ook in Google Forms, maar je hoeft niks te submitten of zoiets, het is puur om makkelijker te maken om de code te kunnen openen.

Participant 2:
Ja, wacht even hoor, ik moet eventje een beetje helder zetten.

Interviewer:
Ja, is goed.

[*De participant begint nu met het lezen van de code, de opmerkingen die de participant heeft gemaakt tijdens deze leesprocedure is hieronder genoteerd.*]

Participant 2:
Ik zie, je wil even horen wat ik nu bekijk, ik zie een functie check pattern, ik zie een functie moving average en een functie process list, en dan een stukje code onderaan met een input string, pattern is test, twee inputs voor, even kijken, man, man wat doe dit, ik mag hier wel even de tijd voor nemen hé.

Interviewer:
Jahoor, je mag zolang over doen als je het wilt.

Participant 2:
Moving average, nou dat geloof ik wel wat dat doet, dat zou wel een voortschrijdend gemiddelde berekenen, en het check pattern die returned een true of een false, afhankelijk van een input string in de pattern, oké, dus onderaan if substring in input string, als het pattern inderdaad in de input string zit, dan maakt die een list van values, en dan print die die moving average van de eerste drie waarden, en anders als de test string niet in de, als de test string niet in de input string zit, dan maakt die een list van lists, en daar gaat die dan wat mee doen, die gaat die dan processen en maakt een nieuwe list, en voor elk item, maakt die, oh zit die daar, maak die d'r hoofdletters van, oké, ik kan me niet helemaal bedenken of ik dat dan voor alle dingen doet, maar als je dit gewoon zo gaat runnen, dan heb je een substring test, test, test, het is even de vraag hoe vaak die dan een hit heeft op die substring, de input string, maar die input string, dat is helemaal geen string, dat is een list, ah oké, dus hij itereert over de elementen van de input string en dan checkt die of daar test in zit, nou dat is altijd het geval, en dan berekent die de, twee keer de moving average, nou ik ga het niet helemaal uitpluizen, maar hij geeft een list terug die gemaakt is door moving averages, want die moving averages, daar wordt iets aan ge-append, dus daar begint met een lege list en dan maakt de list langzamerhand groter.

Participant 2:
Hoe ver wil je dat ik ga?

Interviewer:
Het liefst is dat u echt de code snapt wat die doet.

Participant 2:
Nou wat ik zie is dat die met wat je aan input string en pattern geeft, dat die dus dat pattern checkt, if check pattern, dan zet die de values, maakt die een list van bepaalde values, tien, twaalf, negen, elf, veertien en achtentwintig, kan slecht lezen, en dan print die de output van moving averages, en die

else onderaan, daar komt die niet aan toe, want die conditie die geldt niet, dus dat hele process list of lists, die wordt niet geraakt in deze input, moving averages, die geeft ook wel een list terug, waarom heb je nou moving averages is none, oh die wordt verder niet gebruikt, op regel dertig, die moving averages vind ik nog even lastig te doorgronden, nou, ik snap niet helemaal wat dat moving averages doet, jij zou mij een beetje van de indexen i en n en selector, en ik snap sowieso ook niet waarom je iets met een selector wil doen, selector begint, die begint, i begint bij nul en n begint bij, waar waren we, drie, dus de, die selector is min drie, dan i, ja, natuurlijk is die kleiner dan i, nou, puzzeling, ik kom er niet uit, ik heb het idee dat dit een stukje korter opgeschreven zou kunnen worden, mag ik dat zeggen [...], het is moeilijker gemaakt dan nodig is [...], de hoofdloop is mij redelijk duidelijk, je checkt of die substring, of dat patroon voorkomt in de input string, nou dat is het geval en dan ga je iets met die moving averages doen, en nou, wat dan, dan precies doet, dat blijft mij onduidelijk [...].

[*Eind leesprocedure. Deze leesprocedure heeft 14 minuten geduurd.*]

Interviewer:
Oké, dan gaan we door naar de volgende stap als u denkt dat u klaar bent. Ja, eigenlijk wil ik nu vragen wat jouw interpretatie is van de code, dus kan je het samenvatten wat je net hebt gelezen. Dus dat je nog in mooiere woorden kan samenvatten wat de code precies doet?

Participant 2:
Nog mooiere, code checkt of een bepaald patroon in een input voorkomt en afhankelijk van of dat zo is doe die het één óf het ander, en het één is het berekenen van een voortschrijdend gemiddelde op een onnavolgbaar wijze en in het andere geval gaat die een stel strings kapitaliseren, wat met de gegeven input overigens niet het geval is.

Interviewer:
Ja, dan mijn tweede is eigenlijk of je de output van deze code kan geven, maar in dit geval niet.

Participant 2:
Nou ja, nee, daar heb ik al een antwoord opgegeven volgens mij, ik kom er niet uit.

Interviewer:
Oké, dan gaan we door naar de volgende vraag dan.

Participant 2:
Ja.

Interviewer:
Oké, waar focus je meest op tijdens het lezen van deze code?

Participant 2:
Ja, op de functies en wat de input en de output van de functies is, ja en daarbinnen dan naar de verschillende condities, waar ik begin met de input en de output, die moving averages op regel zevenentwintig die verwacht een stel values en een waarde en dan uiteindelijk return die een moving averages values en dat is verder een, dat is ook weer een list, en process list op eenenveertig die

krijgt een list binnen en die return ook weer een andere list, en die eerste functie check pattern die krijgt een input string en een pattern en die returned true of false afhankelijk van, dat ziet er trouwens ook vreselijk ingewikkeld uit, het zijn pointer, ik zal zeggen for string in of if string, if pattern in string return true, else return false zou ik zeggen, en gebeurt hier nog heel veel meer, maar of het heel veel, heel veel meer oplevert vraag ik me af, ook dit ziet er heel ingewikkeld uit voor wat die doet, wat ik verwacht dat die doet.

Interviewer:
Dan is mijn volgende vraag eigenlijk: welk deel van deze code is volgens jou het belangrijkst?

Participant 2:
Het belangrijkste is natuurlijk part four want daar gaat het werk gedaan worden, daar heb je een input string en een test en een pattern en dan ga je kijken of de, of het patroon, of het pattern voorkomt in de verschillende elementen van die input list, dus regels eenenvijftig tot vierenvijftig zou ik zeggen, dat is het belangrijkste en dan vervolgens roept die check pattern aan.

Interviewer:
Oké, je heeft ook mijn volgende vraag denk ik gewoon beantwoord, want mijn volgende vraag is: waarom vind je dat deze deel het belangrijkst is? Dan de volgende vraag is dan: welk deel van deze code vind je het moeilijkst om te begrijpen?

Participant 2:
Ja, dat heb ik ook al wel gezegd, de moving averages vind ik moeilijk, nou had ik daar eerst vooral op gefocust omdat dat door de code geraakt worden en die check pattern daarvan denk ik van nou ik snap wel wat die geacht wordt te doen, maar nu ik nog zo weer eens naar de code kijk, dan denk ik [...] wat is dat, wat ziet dat er ingewikkeld uit, maar nogmaals ik had me eerst gefocust op een functie die een string en een pattern krijgt en dan true of false teruggeeft, wat er tussenin staat heb ik nou niet zo naar gekeken en nogmaals dat ziet er ingewikkelder uit dan het mij nodig lijkt, ik ga maar dan, ik zit dan wel te denken van waarom is het zo ingewikkeld opgeschreven, wat, wat mis ik hier dan [...].

Interviewer:
Ja, oké. Mijn vraag is: waarom vind je dat dit deel het moeilijkst is? Dus je zegt nu dat moving average eigenlijk het moeilijkst is, maar wat maakt deze code volgens jou dat het moeilijker is?

Participant 2:
Nou wat ik in ieder geval, wat niet erg Python like is, is dat je met i en n werkt, en dan zou ik zeggen gebruik in plaats van i element ofzo, en dan vervolgens hebben we ja de n, dus je zou de, het variabele zou je een wat duidelijke naam kunnen geven, en dan wordt daar gewerkt met een selector, nee nee nee, de structuur is onduidelijk, verder vind ik ook, valt me nu ook op dat er geen docstring is, wat goed gebruik is bij Python dat je onder de definitie dat je aangeeft wat de functie geacht wordt te doen en wat de input en de output is, maar kan ik dat gewoon zien dat de input en output is, maar, en dat is op zich natuurlijk voldoende om een functie te begrijpen, maar ik zou in die docstring dan wel willen zien hoe die dan dat voortschrijdend gemiddelde berekent, want ik zie nu alleen een input list en een integer waarover die het voortschrijdend gemiddelde moet berekenen en geeft die ook weer een list terug, maar hoe die dan precies tot die output komt dat zou dan

graag in die docstring willen lezen, ik kan dan veel beter begrijpen wat dat ding doet en hoe die er dan intern doet, dan eigenlijk niet eens te weten, en hetzelfde kan ik voor check pattern vertellen.

Interviewer:
Welke Python concepten die je in deze code bent tegengekomen vind je eigenlijk het moeilijkst?

Participant 2:
Op zich vind ik dat er geen moeilijke Python concepten in zitten, ik ben onzeker over wat capitalize doet, want dat zou moeten werken op een, op een list en ik vraag me af of capitalize een functie is van een list item, maar goed dat is iets dat zou ik dan even checken en verder zitten er geen rare of onbekende Python concepten in, alleen ja wat ik zei de structuur is niet helder.

Interviewer:
Dat zijn de vragen voor de lees focus die ik wil weten. En ik heb ook nog een paar vragen over jouw lees volgorde. Weet je nog bij welke line je precies bent begonnen met het echt lezen?

Participant 2:
Ja, ik had gescand en ik had gezien part één is een functie, part twee is een functie, part drie is een functie en bij part vier begint dan het eigenlijk programma met een stukje input en dan gaat die iets met de input doen, en in die input of in die verwerking roept die dan de betreffende functies aan die daar boven aan gedefinieerd zijn, dus ik check ah een functie, nog een functie, nog een functie en hier hebben we een stukje programma in part vier waar die iets met die verschillende functies gaat doen, dus zo lees ik hem, zo heb ik hem gelezen.

Interviewer:
Oké, dus eerst de code gescand en dan begonnen met verschillende delen lezen.

Participant 2:
Ja.

Interviewer:
Dat is alles wat ik wil weten over jouw lees volgorde. Ik weet niet of je nog wat opmerkingen hebt over jouw leesstrategieën die je hebt gebruikt?

Participant 2:
Nee, niet anders dan wat ik al verteld heb.

Participant 3
Datum: 19 mei 2021
Duur: 9 minuten

Interviewer:
Ik ga een aantal vragen stellen tot jouw algemene leesstrategieën die je zou gebruiken om een normale tekst te lezen, maar ook om een code te lezen. Dus mijn eerste vraag is dan: wat zijn jouw

algemene leesstrategieën die je zou gebruiken om een moeilijke tekst te lezen, bijvoorbeeld een wetenschappelijk artikel?

Participant 3:
Nou ik denk dat, dat over het algemeen scannend lezen is, dat je wel echt de kernwoorden d'r uitpakt en, zeg maar, om de kernwoorden heen leest zeg maar, maar ik ga niet een tekst volledig doorlezen.

Interviewer:
Ja, oké. Mijn tweede vraag is dan: wat voor leesstrategieën, algemene leesstrategieën die je zou gebruiken om een code te lezen?

Participant 3:
Ja, ik denk hetzelfde, ik denk dat ik vooral kijk naar de complexe delen, dat ik, gewoon direct alle variabelen enzo, dat ik daar niet eens naar kijken en dat ik dan gewoon zeg van oh nou kijk hier is een complexe deel, daar begin ik wel van wat doet het ongeveer, en je begint natuurlijk met de documentatie, dat is het belangrijkste.

Interviewer:
Oké. Mijn derde vraag is dan: wat is volgens jou het belangrijkst verschil tussen het lezen van een code en het lezen van een tekst?

Participant 3:
Nou, dat je in code heb je heel veel delen waar je gewoon al weet wat er staat zonder dat je daadwerkelijk ook naar hoeft te kijken, waar je bij een wetenschappelijke tekst ook gewoon onderdelen overal kunnen staan en waar je niet precies weet wat de belangrijke delen zijn, voordat je eraan begint met lezen.

Interviewer:
Dat zijn mijn vragen. Ik weet niet of je zelf nog iets wil toevoegen aan jouw leesstrategieën, algemene leesstrategieën?

Participant 3:
Ik zou het zelf niet weten.

Interviewer:
Dan kunnen we gewoon door. De volgende stap is dan, je krijgt een Python code te lezen die ik zelf heb geschreven en straks zal ik nog een aantal vragen stellen tot jouw interpretatie van deze code en nog de genomen leesstrategieën die je hebt gebruikt bij het lezen van deze code. Hierbij mag je natuurlijk ook aantekening maken als je het wil, maar het gaat om jouw leesstrategieën, dus je mag zelf kiezen wat je doet. Ja oké, hier in de link vind je dus de code. Ik heb het ook in Google Forms gedaan omdat het makkelijker is te verspreiden is, dus je hoeft niks te submitten en je mag het zolang over doen als je het wilt. Zeg maar wanneer je klaar bent met het lezen.

[*De participant begint nu met het lezen van de code. De participant heeft in stilte gelezen en geen*

opmerking gemaakt tijdens het lezen.]

[*Eind leesprocedure. Deze leesprocedure heeft half minuut geduurd.*]

Participant 3:
Ik ben zover hoor.

Interviewer:
Dus snap je de code nu ook, want ik ga een aantal vragen stellen tot jouw interpretatie en wat je denkt dat de output is van deze code.

Participant 3:
Oké, ja prima.

Interviewer:
Oké. Wat is dan jouw interpretatie van deze code? Kan je dat kort samenvatten?

Participant 3:
Het lijkt erop dat die gewoon strings gaat vergelijken met bepaalde, ja zeg maar, lijstjes die je al in hebt staan, en als die bepaalde lijstjes ziet, dan gaat die, ja de pointer aanpassen, ik weet niet precies wat hier een pointer doet, maar voor de rest ja, ja gaat die nieuwe items capitalizen.

Interviewer:
Oké. Wat denk je dat de output is van deze code?

Participant 3:
Ik verwacht dat die een lijst afdrukt van, nou in dit geval, dus dier of planten of zoiets.

Interviewer:
Oké. Is dat alles wat je denk dat het is?

Participant 3:
Ik zie hem nergens, ja ik zie hem weinig print doen, dus de uitvoer moet een print zijn, en die gaat dan een average afdrukken ofzo, moving averages, ja values.

Interviewer:
Oké. Heb je nog iets toe te voegen?

Participant 3:
Niet echt.

Interviewer:
Dan gaan we door. Je was heel snel met het lezen. Dus ik heb nu een aantal vragen tot jouw lees focus, daar ben ik wel benieuwd naar. Dus waar focus je het meest op tijdens het lezen?

Participant 3:
Nou ik begin dus zelf bij de uitvoer, zo van wat doet, wat print die daadwerkelijk af en dan ga ik gewoon een beetje scannen van hé wat zijn de belangrijke methodes, en dan ga ik pas daadwerkelijk verder kijken naar de if statement en de else statement en de while statement, dat soort dingen.

Interviewer:
Oké, je hebt dus nu de code helemaal gezien. Welke deel van deze code is volgens jou het belangrijkst?

Participant 3:
Nou het is allemaal belangrijk, anders werkt je code niet, dat ten eerste, maar als je uitvoer wilt, dan is je onderdeel vier wel belangrijkste, want daar druk die dat daadwerkelijk af.

Interviewer:
Ja, oké. En welke deel van deze code vind je het moeilijkst om te begrijpen?

Participant 3:
Het moeilijkst, nou dat wordt het dan sowieso deel één, want daar staan gewoon de meeste statements in.

Interviewer:
Oké, want hier zijn natuurlijk verschillende Python concepten erin gestopt, dus welke Python concepten die je hier zijn tegengekomen vind jij het moeilijkst?

Participant 3:
Sorry, nog een keer.

Interviewer:
Python concepten, zoals list, condition en dat soort dingen. Welke heb je moeite mee, zeg maar.

Participant 3:
Ik zal heel eerlijk zijn, ik heb er zo niet op die manier naar gekeken, ja, geen idee.

Interviewer:
Oké, dus gewoon niet, gewoon vanzelfsprekend allemaal.

Participant 3:
Ja.

Interviewer:
Dat zijn mijn vragen. Heb je zelf nog iets om toe te voegen over je lees focus?

Participant 3:
Ja nou ja, weet je op het moment dat je de vragen over gaat stellen, dan kom je het er toch wel achter dat je niet helemaal goed die dingen doorleest, sowieso, dat denk je van oh ik begrijp het wel ongeveer, maar dan heb ik het dus toch nog niet helemaal.

Interviewer:
Ja, oké. Ik heb wel nog een aantal vragen tot jouw leesvolgorde. Dus weet je nog bij welke line je precies bent begonnen met het lezen?

Participant 3:
Vierenvijftig, dat is, dat is de eerste print die ik zag staan.

Interviewer:
Oké. En bij welke line ben je gestopt met het lezen? Weet je dat nog?

Participant 3:
Ergens in het midden, dus bij deel twee, bij eind deel één, begin deel twee.

Interviewer:
Oké, daar. Ik zie dat je ook gewoon eerst begonnen was met het scannen van de code, toch?

Participant 3:
Ja.

Interviewer:
En dan pas begonnen met het echt lezen?

Participant 3:
Ja.

Interviewer:
Ja, kan je nog ongeveer herinneren in welke volgorde door al die delen, ben je ongeveer doorheen gegaan?

Participant 3:
Nou ik ben dan dus sowieso begonnen bij deel vier en dan ga ik gewoon van boven naar beneden, en dan nog een keer als ik het niet begrijp begin ik weer bovenaan en dan ga ik weer naar beneden.

Interviewer:
Oké, ja, dat zijn mijn vragen tot jouw leesvolgorde. Heb je zelf nog iets toe te voegen?

Participant 3:
Niet echt, nee.

Participant 4
Datum: 20 mei 2021
Duur: 25 minuten

Interviewer:
Ik heb nu een aantal vragen over jouw algemene leesstrategieën die je zou gebruiken om een normale tekst te lezen en een code te lezen. Dus je mag alles zeggen. Mijn eerste vraag is: wat zijn jouw algemene leesstrategieën die je zou gebruiken om een normale tekst te lezen, bijvoorbeeld een wetenschappelijk artikel?

Participant 4:
Oké, nou ik heb sowieso dyslexie, dus ik moet me veel beter concentreren op aparte zinnen zeg maar, dus ik begin meestal met de inleiding lezen en die lees ik dan over het algemeen twee keer omdat ik anders vaak dingen mis ofzo, en dan de rest van de tekst lees ik iets minder goed als in ik lees hem nog steeds wel goed, maar ik ga niet elke stuk twee keer lezen, maar basically ga ik gewoon lezen, als ik een zin niet snap, dan lees ik alleen die zin opnieuw, maar ik denk niet dat ik verder echt een strategie heb van hoe ik door de tekst lees, als ik het echt op moet slaan, dan markeer ik nog wel belangrijk dingen, dat doe ik bijvoorbeeld bij een toets, als ik een vraag heb, dan moet ik ook, dan markeer ik de belangrijke dingen d'r uit, ook om dus dat uit laat te springen met mijn dyslexie, meer strategie heb ik denk ik niet echt.

Interviewer:
Oké, ja dat is goed. Dan is mijn tweede vraag: heb je algemene leesstrategieën die je zou gebruiken om een code te lezen?

Participant 4:
Echt puur de code, dus als je in Python zit, gewoon de aanroepen enzo [...], en gewoon de statement, ja, zo dat is wel een goede vraag, het is lang geleden, weet ik eigenlijk niet hoe ik dat las, ik weet wel dat ik vooral focus op de grote aanroepen en wat die doen, want die meestal het meest doen, en dat ik de dingen d'r omheen niet heel overdreven in me op [...], maar meer om echt de grote dingen die iets deden en niet de bepaalde aanroepen ofzo, die las ik over het algemeen niet, dat is waarschijnlijk ook waardoor het zo slecht ging, want de kopjes en zo en waarin het verdeeld, dat las ik niet, ik las dan echt alleen maar de grote aanroepen die wat deden en dan ging ik daar opmaken wat ze deden.

Interviewer:
Oké, ja. Wat is volgens jouw het belangrijkst verschil tussen het lezen van een tekst en het lezen van een code?

Participant 4:
Een code moet daadwerkelijk iets doen, dus ik denk dat je in een code juist op zoek gaat naar datgene wat de code moet doen, als in er zijn altijd heel veel aanroepen omheen, [...] in een code eerder op zoek gaat naar wat doet het, en dat je in een tekst, ga je ook wel op zoek naar de kern, maar dan lees je bijvoorbeeld eerst de alinea, eerst inleiding en de slot als je wil weten waar de tekst over gaat, terwijl je bij [...] meer naar de grootste, de meest uitvoerende cast van die code gaat kijken.

Interviewer:
Oké, dat zijn vragen tot jouw algemene leesstrategieën. Heb je zelf nog opmerkingen over?

Participant 4:

Nee.

Interviewer:
Nee, oké, dat is prima. Dan is de volgende stap. Ik heb dus zelf een Python code geschreven en die ga je dus lezen. Probeer wel zo goed mogelijk te lezen dat je echt probeert te snappen, want straks zal ik namelijk een aantal vragen stellen tot jouw interpretatie van de code, dus wat de code doet enzo, en ook een aantal vragen tot jouw genomen leesstrategieën, en dat heeft betrekking tot jouw leesvolgorde en jouw lees focus. Je mag hierbij ook aantekening maken, maar als je aantekening maakt, dat je dat in de notepad op jouw laptop of computer doet dat je dat straks ook naar me kan sturen, want die is namelijk ook een deel van jouw leesstrategieën die is nodig voor mijn experiment en voor mijn onderzoek. Maar het gaat uiteindelijk om jouw leesstrategieën, dus jij mag zelf bepalen hoe je het doet. Oké, is dat duidelijk?

Participant 4:
Ja.

Interviewer:
Dan vind je nu in de chat een link naar de code, die is ook in Google Forms, maar je hoeft niks te submitten, en je mag alle tijd nemen als je nodig hebt. Dus, zeg maar wanneer je klaar bent.

[*De participant begint nu met het lezen van de code, de opmerkingen die de participant heeft gemaakt tijdens deze leesprocedure is hieronder genoteerd.*]

Oh het ziet heel anders uit dan de Python die ik heb gehad, ik had echt op middelbare school Python in zo'n speciaal mooie programma, pointer, zolang de pointer kleiner is dan de lengte van de string, oké, dus je gaat het patroon van een string checken en je loopt er door middel van een pointer loop je d'r doorheen, zolang waar de punt waar je bent kleiner is dan de totale lengte, oké, en in die tweede ga je bewegen, oh nee, je gaat niet bewegen, je gaat, even kijken hoor, en in de tweede gedeelte ga je de average berekenen, dit doe je door de string heen te lopen en bij elke stap de value van dat punt toe te voegen en te delen door het nieuwe totaal van aantal zeg maar, dus n is aantal, en in de derde ga je een lijst maken [...], tweede ga je een lijst maken in alle lijst van lijstjes, oké, oh en de vierde gebruik je weer die dingen uit die eerste gedeelte, oké, gedeelte vier ga je het patroon checken van wat daar is ingevoerd met de values die daar achter staan, en van die values bereken je ook de average, dus dan begin je bij de tien en dan gedeeld door één en dan plus twaalf en dat gedeeld door twee, dus negen gedeeld door drie enzovoorts, maar dat doe je alleen, oh je doet het alleen als die string dat is, en anders ga je het met woorden doen en dan maak je er een lijst van, oké, zoals patroon, tien, twaalf, negen, elf, veertien en twintig is, ga je de moving average berekenen en printen, en als dat niet zo is, dan heb je een lijst, en ga je die lijst in een nieuwe lijst plakken, oké, moet ik nog wat over zeggen.

Interviewer:
Denk je dat je ongeveer de code snapt wat die doet?

Participant 4:
Ongeveer wel ja.

Interviewer:
Want het liefst wil ik weten wat je denkt dat de output is van de code.

Participant 4:
Oké, de input string is die test test test, dus het is niet die tien, twaalf, negen, elf, veertien, dus dan gaan we er naar de else, dus dat betekent dat we een superlist is duck, cat, flower, tree, whales, swordfish, en dat we daar wat mee gaan doen, en dat betekent dat bij part three gaan uitvoeren, dan gaan we d'r lijst superlist gaan we processen, in een sublist en voor elke item wordt een sublist gemaakt, dus dat zou betekenen dat dus die duck en cat een sublist is, dat die flower en tree een sublist is, en dat die whale en swordfish een sublist is, [...] oké, nou ik denk dat dus dat een sublist is van [...], dat die dus die lijstjes gaat printen.

[*Eind leesprocedure. Deze leesprocedure heeft 11 minuten geduurd.*]

Interviewer:
Kan je precies zeggen wat die gaat printen?

Participant 4:
Nou, gebruikt de superlist, for item in sublist, [...] capitalize is dat hoofdletters van maken, new list, ja ik denk een soort van nieuwe sublist, de lijst, [...] ik denk dat zoiets dat je een lijst hebt van elk onderdeel in die superlist.

[*De participant heeft de volgende output gegeven in de chat.*]
- DUCK CAT
- FLOWER TREE
- WHALE SWORDFISH

Interviewer:
Ja prima, als je klaar bent, dan kunnen we door naar de volgende deel.

Participant 4:
Ja.

Interviewer:
Nu ga ik dus een aantal vragen stellen tot jouw interpretatie van de code. Dus kan je misschien samenvatten wat de code doet?

Participant 4:
Samenvatten wat de code doet, nog een keer, oké, dus niet hoe ik het heb gelezen, maar wat het doet.

Interviewer:
Ja.

Participant 4:

Oké, nou het gaat eerst checken of er een patroon is van cijfers, en als dat zo is, dan gaat die door die cijfers heen lopen en elke keer een volgende cijfer optellen bij het totaal en dan delen door het aantal cijfer dat die tegen is gekomen en dat is dan het gemiddelde van die hele string die die doorloopt, en als die string niet gelijk is aan de string die is ingevoerd, dan gaat die een lijst maken, een sublijst in een andere lijst van de invoer die die krijgt.

Interviewer:
Ja, oké. De output heb je al gezegd, dus die vraag skippen we nu. Heb je nog andere opmerkingen tot jouw interpretatie van deze code?

Participant 4:
De manier waarop of.

Interviewer:
Gewoon wat je denkt dat je nog kan toevoegen.

Participant 4:
Wat ik net niet zei dat die dus, hij checkt of de input string gelijk is aan wat die wil checken en als dat niet gelijk is, dan gaat die naar de else, dus dat die tetestst staat zegmaar, daarmee checkt die.

Interviewer:
Oké, nou dat zijn mijn vragen tot jouw interpretatie. Ik heb nu een aantal vragen tot jouw lees focus. Dus waar focus je meest op tijdens het lezen?

Participant 4:
Nou, ik focus sowieso per stukje, dus je hebt die part één, part twee, part drie, part vier en per stukje ga ik het lezen en probeer te begrijpen, en waar ik dan meest op focus bij dan bijvoorbeeld die eerste was die, even kijken, die tweede while loop om daar te begrijpen, en bij part twee is het eigenlijk ook op die while loop van wat gebeurt daar, dus daar focus ik steeds op.

Interviewer:
Oké. En welk deel van deze code is volgens jou het belangrijkst?

Participant 4:
Nou ja, part vier is echt wat het uitvoert, maar zonder die aanroepen van daarvoor kan part vier niet werken, dus dat is een hele goede vraag, maar in principe is part vier de uitvoering, dus ik ga voor part vier.

Interviewer:
Ja, oké. En welk deel van deze code vind je het moeilijkst om te begrijpen?

Participant 4:
Het makkelijkst was part twee en het moeilijkst is denk ik part drie.

Interviewer:

Oké. Waarom vind je dat dit deel het moeilijkst is?

Participant 4:
Nou met die new list, append en item capitalize, dat maakt voor mij de output zeg maar wat die print moeilijk omdat ik niet precies weet wat die daar nu doet.

Interviewer:
Oké. Door de hele code heb je natuurlijk verschillende Python concepten die je bent tegengekomen. Welke Python concepten vind je het moeilijkst?

Participant 4:
Als in for loop, while loop.

Interviewer:
Ja, list, conditions, for loop, while loop, slicing, nesting, wat je kan opnoemen.

Participant 4:
Zo, ik weet niet hoe dat heet.

Interviewer:
Je kan ook regelnummer zeggen.

Participant 4:
Drieëntwintig.

Interviewer:
Ja, zijn er nog andere concepten?

Participant 4:
Zevenendertig en ja vijfenveertig met die new list zeg maar.

Interviewer:
Oké, dat zijn mijn vragen tot jouw lees focus. Heb je zelf nog iets toe te voegen?

Participant 4:
Nee.

Interviewer:
Dan gaan we door. Nu ga ik een aantal vragen stellen tot jouw leesvolgorde. Dus weet je nog bij welke line je precies bent begonnen met het lezen?

Participant 4:
Ik denk bij line zeven, bij die eerste while loop zeg maar, omdat daar dat dat werkelijk dat die iets gaat doen, start als in daarvoor worden er meer dingen gegeven, en bij die while, dan gaat er echt wel iets gebeuren.

Interviewer:
Bij welke line ben je ongeveer gestopt met het lezen? Weet je dat nog?

Participant 4:
Op het hele eind?

Interviewer:
Ja.

Participant 4:
Even kijken, ja want ik had part vier dan wel gelezen, maar dan ben ik terug gegaan naar part drie, ja dan zesenveertig, want ik heb dan vier gelezen en toen ben ik drie, voor die else van vier ben ik drie opnieuw gaan lezen, en toen ik drie had gelezen, was ik klaar.

Interviewer:
Oké, ja, mijn volgende vraag is dan: ben je gelijk begonnen met het lezen van de code, of heb je eerst de code gescand? Want dat is me niet duidelijk.

Participant 4:
Dat is wel een hele goede vraag, volgens mij ben ik gelijk begonnen met het lezen, ja, nee ik ben gelijk begonnen met het lezen, want ik had nog niet gezien dat er part één, twee en drie stond, dus ik ben gelijk begonnen met het lezen.

Interviewer:
Oké, prima. Weet je nog precies in welke volgorde je door de code delen bent heen gelopen.

Participant 4:
Eerst één, toen twee, toen ben ik even teruggegaan naar één, toen heb ik drie gelezen, toen vier, en toen, even kijken, heb ik één nog een keer gelezen, deels dan, toen was ik weer bij vier en toen drie.

Interviewer:
Oké. Dat zijn mijn vragen. Heb je zelf nog iets toe te voegen tot jouw leesvolgorde?

Participant 4:
Nee.

Participant 5
Datum: 21 mei 2021
Duur: 25 minuten

Interviewer:
Oké, dan ga ik nu een aantal vragen stellen tot jouw algemene leesstrategieën die je zou gebruiken om een normale tekst te lezen en een code. Dus mijn eerste vraag is: wat zijn jouw algemene

leesstrategieën die je zou gebruiken om een moeilijk tekst te lezen, bijvoorbeeld een wetenschappelijk artikel?

Participant 5:
Ik kijk naar de inhoudsopgave, dus welke tussenkopjes er allemaal zijn, en dan begin ik meestal bovenaan, bij introductie achtig, en dan lees ik, eigenlijk, oh dit klinkt, dit is echt precies die leesstrategieën die je bij Nederlands leert, dat je gewoon altijd eerste en de laatste twee regeltjes ongeveer lees van elke alinea, dat het, en als ik denk van oké dat is echt interessant, dan lees ik de hele alinea, en meestal sla ik alle wiskunde over, eigenlijk alle getallen sla ik over.

Interviewer:
Ja, mijn tweede vraag is: wat zijn jouw algemene leesstrategieën die je zou gebruiken om een lange code te lezen?

Participant 5:
Nu als ik echt echt een nieuwe stukje code voor me krijg, dan zoek ik eigenlijk altijd eerst naar de, noem maar, als de code is met meerdere functies, zoek ik eigenlijk eerst de hoofdfunctie, dus die functie die alle andere functies aanroept, en dan loop ik er, naja dan doe ik die functies van boven naar beneden en pas als ik echt wil weten, dan kijk ik naar wat de sub-functies doen, dus bij, als ik functie lees, in principe, hoe de computer er zelf doorheen loopt, dus ja probeer de computer stap te volgen behalve dat ik niet naar sub-functies kijk, die zegmaar, die sla ik over.

Interviewer:
Oké, ja. Wat is dan volgens jou het belangrijkst verschil tussen het lezen van een tekst en het lezen van een code?

Participant 5:
Ja, ik denk trouwens dat ik nog een andere strategie ding heb, dat inderdaad wat het verschil is, als bij ik lees een functie niet van boven naar beneden, ik kijk altijd eerst naar of d'r loopjes in staan, want dat is meestal waar het meest gebeurt, dus ik kijk altijd eerst naar de loops, dan kijk ik naar if statements, die bepalen [...] ook wat er gebeurde in een code, maar, wat grappigs is, is dat loops en if statements vaak in het midden van een functie staan, dus eigenlijk werk ik soort van, van binnen naar buiten, in plaats van wat ik bij normale tekst doe, dat ik de eerst, dus eigenlijk, dus twee buitenste randjes eerst lees, doe ik bij code eigenlijk eerst naar het middelste stukje kijk.

Interviewer:
Ja, oké. Dat zijn mijn vragen tot jouw algemene leesstrategieën. Heb je zelf nog wat opmerkingen over?

Participant 5:
Nee, ik denk dat dat het is.

Interviewer:
Ik heb zelf dus een Python code geschreven, dus die ga je lezen. Hierbij mag je natuurlijk aantekening maken als je dat wilt, het gaat uiteindelijk om jouw eigen leesstrategieën,dus je mag zelf bepalen.

Alleen als je aantekening maakt dat je ergens aantekening maakt dat je dat ook naar me kan sturen of foto's maakt en zoiets, want dit hoort natuurlijk ook bij jouw leesstrategieën en die kan ik dus gebruiken voor mijn onderzoek. Hier in de chat vind je dus de link naar de code, die is ook in zo'n Google Forms, maar je hoeft niks in te leveren of zoiets. En je mag het zolang over doen, dus probeer de code echt te snappen naar jouw idee. Zeg maar wanneer je klaar bent.

Participant 5:
Ja.

[*De participant begint nu met het lezen van de code. De participant heeft in stilte gelezen en geen opmerking gemaakt tijdens het lezen.*]

[*Eind leesprocedure. Deze leesprocedure heeft 9 minuten geduurd.*]

Participant 5:
Oké, ik denk dat ik hem nu wel helemaal door heb.

Interviewer:
Dan kunnen we gewoon door naar de volgende stap. Ik ga nu dus een aantal vragen stellen tot jouw interpretatie van de code. Dus kan je misschien de code samenvatten wat die doet?

Participant 5:
Ja, je kijkt naar, je krijgt een lijst van strings en jij hebt een bepaalde pattern en voor elk van die strings in die input lijst kijk je of de pattern daarin voorkomt, en als die d'r wel in voorkomt, dan print je een, je print een lijst met gemiddeldes van, de gemiddeldes van de getallen die in die array values staan, daarbij steeds de gemiddelde van drie getallen, dus eerst het gemiddelde van tien, twaalf en negen, en dan twaalf, negen, elf, enzovoorts tot aan het einde, als die, als dat patroon niet in die string zit, dan print je ook een lijst, maar dit keer een lijst met alle elementen die in die superlist staan, maar dan in hoofdletters.

Interviewer:
Oké, ja. Mijn volgende vraag is eigenlijk: wat is volgens jou de output van deze code? Kan je in de chat typen wat de output kan zijn van deze code?

[*De volgende is gegeven door de participant in de chat.*]
[25, 49.6666667, 61, 76]

Participant 5:
Oh wacht, dat dan twee keer.

[*De volgende is gegeven door de participant in de chat.*]
[25, 49.6666667, 61, 76]

Interviewer:
Oké. Dat zijn mijn vragen voor deze gedeelte. Heb je zelf nog opmerking over jouw interpretatie

van de code?

Participant 5:
Nee, dat is het denk ik.

Interviewer:
Ja, oké. Als je geen opmerking hebt, dan gaan we door naar het volgend deel. Dan ga ik nu een aantal vragen stellen tot jouw lees focus. Dus mijn eerst vraag is: waar focus je het meest op tijdens het lezen?

Participant 5:
Nou dat is opzich wel grappig, want ik, zeg maar het vorige keer dat ik eerst op dit for loops en while loops focuste, dat merkte ik wel tijdens mijn bachelor project, maar dan was ik heel gericht aan het zoeken naar, was ik meer aan het zoeken naar bepaald onderdeel van de code, nu probeer ik de hele code te gebruiken, dus nu volg ik echt zeg maar de computer, dus welke stappen de computer door zal lopen, dus dat [...], oh wacht wat was je vraag.

Interviewer:
Waar focus je het meest op tijdens het lezen bij deze code?

Participant 5:
Ja, ik focus het meest op verandering van variabelen.

Interviewer:
Oké. En welk deel van deze code is volgens jou het belangrijkst?

Participant 5:
In principe is denk ik, vind ik de, in ieder geval de check pattern functies, functie, vind ik belangrijk, die bepaalt wel welke andere functie wordt uitgevoerd, en de moving average vind ik belangrijk, maar dat was de moeilijke, [...] moeilijkst te begrijpen vond.

Interviewer:
Ja, oké. Ja, eigenlijk heb je mijn volgende vraag ook wel beantwoord, ik wilde vragen waarom je dit deel het belangrijkst vindt, maar dat heb je al gezegd.

Participant 5:
Ja.

Interviewer:
Mijn volgende vraag was: welk deel van deze code vind je het moeilijkst om te begrijpen, maar dat heb je net ook al gezegd, moving average, right?

Participant 5:
Ja.

Interviewer:
En waarom vind je dat dit deel het moeilijkst is?

Participant 5:
Ik raakte daar in de war door die i en die n, daar, dat die geen duidelijk namen hadden, en voor mijn gevoel, zeg maar niet, tenminste geen vaste functie, zegmaar, dus ik gebruik wel vaker i of n, maar dan is n in mijn hoofd altijd een aantal, maar dat was dit keer eigenlijk niet, en i, ja gebruik ik altijd om elementen in een array op te halen en op die manier werd deze keer ook niet gebruikt, daarom vind ik die het moeilijkst is.

Interviewer:
Oké. Nou, ja, mijn volgende vraag is dan: welke Python concepten die je in deze code bent tegengekomen vind je het moeilijkst?

Participant 5:
Concepten van Python niet, nee.

Interviewer:
Dat zijn mijn vragen tot jouw lees focus. Heb je zelf nog opmerkingen over?

Participant 5:
Nee, dat niet.

Interviewer:
Oké, dan ga ik naar het volgend deel. Nu ga ik een aantal vragen stellen tot jouw leesvolgorde. Dus mijn eerste vraag is gewoon: bij welke line ben je begonnen met het echte lezen?

Participant 5:
Helemaal begonnen bij eenenvijftig.

Interviewer:
Oké. En bij welke line ben je ongeveer gestopt met het lezen?

Participant 5:
Dat was bij line zesenveertig.

Interviewer:
Ja, oké. Ben je dan gelijk begonnen met het lezen van de code, of heb je eerst de code gescand?

Participant 5:
Ik heb eerst gescand als in eerst scan oh er zijn drie functies en de code [...] wordt eerst uitgevoerd vanaf regel negenenveertig.

Interviewer:
Oké. Weet je nog in welke volgorde je door de code delen bent heen gelopen?

Participant 5:
Ik begon dus bij eenenvijftig, toen ging ik omhoog naar negenenveertig, daarna naar beneden gegaan met tweeënvijftig, toen ben ik weer omhoog gegaan naar vanaf regel twee, die hele functie van boven naar beneden, toen ben ik bij regel drieënvijftig en verder gegaan, en daar regel zevenentwintig, die hele functie, toen vijfenvijftig weer verder gegaan tot zevenenvijftig, en toen regel eenenveertig tot en met zesenveertig, van boven naar beneden.

Interviewer:
Dus dan ben je begonnen met deel vier als ware en dan ben je naar deel één gegaan en dan naar deel twee en dan deel drie?

Participant 5:
Ja.

Interviewer:
En toen ben je weer terug bij deel vier, heb ik dat goed begrepen?

Participant 5:
Ja, nou ja, eigenlijk ben ik, nee, ik ben niet echt meer teruggegaan naar deel vier.

Interviewer:
Welke deel was je gegaan na deel drie.

Participant 5:
Ik was geëindigd.

Interviewer:
Oh je was geëindigd, dus gewoon deel vier naar deel één, twee en drie en klaar.

Participant 5:
Ja.

Interviewer:
Oké, top, is goed. Dat zijn mijn vragen. Heb je zelf nog opmerkingen over jouw leesvolgorde?

Participant 5:
Nou ja, opzich ben ik tijdens het lezen van die functies, ging ik wel altijd toen even terug naar deel vier, als ik bijvoorbeeld achter kom van dat, welke inputs is er geweest zeg maar, dus daar, in principe begin ik dan wel een klein beetje terug.

Participant 6
Datum: 24 mei 2021
Duur: 10 minuten

Interviewer:
Nu ga ik een aantal vragen stellen tot jouw algemene leesstrategieën die je zou gebruiken om een normale tekst te lezen, maar ook een code. Dus mijn eerste vraag is dan: wat zijn jouw algemene leesstrategieën die je zou gebruiken om een moeilijk tekst te lezen, bijvoorbeeld een wetenschappelijke artikel?

Participant 6:
Titel, kopjes, abstract, dan de resultaten en dan wat me nog interesseert.

Interviewer:
Oké. Mijn tweede vraag is dan: wat zijn jouw algemene strategieën die je zou gebruiken om een lange code te lezen?

Participant 6:
Eerst kijken van wat de taal is en wat de code probeert te doen, dan een beetje naar de structuur kijken en ja daarna ga ik proberen te zoeken wat de functionaliteiten zijn door te kijken naar de main code, zeg maar gewoon het hoofd stukje en dan daar de functie, stap voor stap waar ze heen verwijzen door probeer te lopen.

Interviewer:
Oké. Wat is volgens jou het belangrijkst verschil tussen het lezen van een tekst en het lezen van een code?

Participant 6:
De volgorde is niet altijd van links naar rechts, boven naar beneden.

Interviewer:
Ja, oké. Dat zijn mijn vragen. Heb je zelf nog opmerkingen over je leesstrategieën?

Participant 6:
Nee ja, ik zou zeggen zolang mensen gewoon best-practices van code bijhouden dat dan gigantische verschil maakt in je leestempo voordat je structuur, voordat je iets begrijpt.

Interviewer:
Oké, dan gaan we door naar de volgende stap. Ik heb zelf dus een Python code geschreven en dat ga je dus nu lezen. Belangrijk is dat je wel de code echt probeert te begrijpen, want ik zou straks nog vragen stellen over jouw interpretatie van de code, dus wat de code doet, en nog een paar vragen over jouw leesstrategieën die je hebt genomen om deze code te lezen, en dat heeft vooral betrekking tot de volgorde die je hebt genomen en de focus die je hebt genomen bij het lezen van de code. Oké, nou dan kunnen we gelijk beginnen. In de chat vind je de link naar de code zeg maar als een afbeelding. Je hoeft niks te submitten. Je kan alle tijd nemen, dus zeg maar wanneer je klaar bent.

[*De participant begint nu met het lezen van de code. De participant heeft in stilte gelezen en geen opmerking gemaakt tijdens het lezen.*]

[*Eind leesprocedure. Deze leesprocedure heeft 1 minuut geduurd.*]

Participant 6:
Oké, ik begrijp het wel wat het doet.

Interviewer:
Oké, helemaal top. Dan ga ik dus een aantal vragen stellen tot jouw interpretatie van de code. Dus kan je de code misschien samenvatten wat die doet?

Participant 6:
Ja, in deel vier wordt de werkelijk functie van de totale code bepaald, hierbij wordt eerst een aantal input strings gegeven en de test parameter wordt meegegeven, dus waarin zometeen een onderdeel van je input string gedaan moet worden, hier wordt doorheen ge-loopt en wordt gekeken van oké voldoet, zit er patroon in je input string, zo ja, dan gaat die het gemiddelde bepalen van een aantal waardes, en als het niet zo is, dan heeft die een andere lijst waarin vervolgens andere gedrag gaat uitvoeren.

Interviewer:
Oké, ja. Wat is volgens jou de output van deze code dan?

Participant 6:
In beide gevallen, zou die, als de def doet wat zegmaar de def zeggen, dan gaat die in allebei de gevallen, gaat die de if statement in, omdat de test in de input string staat, dus, dan zou die met de value die meekrijgt het gemiddelde, hier zal die het gemiddelde bepalen van die waardes die worden meegegeven en die zal die vervolgens uitprinten.

Interviewer:
Oké, dus het gemiddelde van die lijst, tien, twaalf, negen, veertien en twintig.

Participant 6:
Ja, en twee keer.

Interviewer:
Twee keer, ja is goed. Nou dat zijn mijn vragen tot jouw interpretatie. Heb je zelf nog andere opmerkingen over?

Participant 6:
Nee, ik ga nu wel vanuit dat de code precies zo gedraagt zoals de headers benoemt hebben, dus, zolang ze zich gewoon daadwerkelijk descriptive namen gekozen hebben in deze code, dan zal het wel kloppen maar.

Interviewer:
Oké, ja, nu ga ik een aantal vragen stellen tot jouw lees focus. Waar focus je het meest op tijdens het lezen?

Participant 6:
Eerst kijken waar daadwerkelijk de code begint, dus op zoek waar geen definition voor staat, in dit geval dus deel vier, en daarna ga ik kijken van welke momenten welke headers wordt aangeroepen, en wat dan inkomt en wat dan uitkomt, wat de code zou moeten uitvoeren en dan kijken gewoon zonder het daadwerkelijk de functie echt probeer door te lezen, kijken van, oké ja, als die input erin komt, wat komt er dan uit, en hoe gaat dan alleen van de hoofdfunctie, wat die gedraagt dan, mocht er dan nog iets niet heel duidelijk zijn, dan pas zou ik het werkelijk de kleinere header functies verder gaan lezen.

Interviewer:
Oké. Welk deel van deze code is volgens jou het belangrijkst.

Participant 6:
Ik zou zeggen deel vier omdat die de hoofdfunctie is, alleen deel één is ook heel erg leidinggevend voor het gehele gedrag, dus ik zal toch zeggen deel één.

Interviewer:
Oké, ja, dus welk deel van de code vind je het moeilijkst om te begrijpen?

Participant 6:
Deel 2, sowieso.

Interviewer:
En waarom vind je dat dit deel het moeilijkst is?

Participant 6:
Er staan meer variabelen in zonder echte toelichting, er wordt meer gecast en meer dingen achter elkaar gezet zonder echt duidelijke tussenstappen.

Interviewer:
Oké. En welke Python concepten die je in deze code bent tegengekomen, vind je het moeilijkst?

Participant 6:
Zelf ben ik niet bekend met capitalize, maar ik neem aan dat het gewoon doet wat die zegt, maar ja, ik zal toch gewoon op de term capitalize houden, item punt capitalize, want dat is de enige die niet bekend is voor mij.

Interviewer:
Oké. Dat zijn mijn vragen tot jouw lees focus. Heb je zelf nog andere opmerkingen over?

Participant 6:
Nee.

Interviewer:
Oké. Dan ga ik door naar de volgende stap. Ik ga dus een aantal vragen stellen tot jouw leesvolgorde

die je hebt genomen. Weet je nog bij welke line je bent begonnen met het echt lezen?

Participant 6:
Negenenveertig.

Interviewer:
Negenenveertig, oké. Bij welke line ben je gestopt met lezen?

Participant 6:
Zevenenvijftig.

Interviewer:
Oké. En ben je gelijk begonnen met het lezen van de code, of heb je eerst de code gescand?

Participant 6:
Eerst gescand.

Interviewer:
En in welke volgorde ben je door de code delen heen gelopen? Weet je dat nog?

Participant 6:
Ja, dus ik begon bij deel vier, toen zag ik dus eerst de check pattern, dus dat is deel één ging ik terug daarna, en daarna keek ik dus naar moving average van deel twee is, en als laatst deel drie.

Interviewer:
Oké, ja, dat zijn mijn vragen tot jouw leesvolgorde. Heb je zelf nog andere opmerkingen over?

Participant 6:
Nee.

Participant 7
Datum: 25 mei 2021
Duur: 40 minuten

Interviewer:
Ik ga nu een aantal vragen stellen tot jouw algemene leesstrategieën die je zou gebruiken bij het lezen van een normale tekst en een code. Dus mijn eerste vraag is dan: wat zijn jouw algemene leesstrategieën die je zou gebruiken om een moeilijk tekst te lezen, bijvoorbeeld een wetenschappelijk artikel?

Participant 7:
Ik zou eerst soort van de introductie en de conclusie lezen, denk ik, ja verder, lastig, ik zou vooral, als ik bijvoorbeeld een wetenschappelijk artikel lees, denk ik waarschijnlijk niet het hele artikel lezen, maar meer de eerste zinnen van de alinea of dat soort dingen, zeg maar de kerninformatie

proberen op te zoeken en dan als ik denk van oh dit is iets wat ik nodig heb waar ik meer over wil weten, dan de hele alinea ga lezen ofzo, denk ik zoiets.

Interviewer:
Oké. Wat zijn jouw algemene leesstrategieën die je zou gebruiken om een lange code te lezen.

Participant 7:
Dat zou ik wel dan echt gewoon helemaal afgaan zegmaar, gewoon stap voor stap kijken wat er gebeurt ofzo, soms misschien dingen opschrijven om te zien wat er gebeurt zeg maar, en kijken wat er uitkomt enzo.

Interviewer:
Oké. En wat is volgens jou het belangrijkst verschil tussen het lezen van een tekst en het lezen van een code?

Participant 7:
Dat is een lastige vraag, ik weet niet, dat is wel een goede vraag, bij een tekst, ik weet niet, ik heb persoonlijk denk ik bij een tekst dat ik makkelijker soort van kan scannen en soort van dan als ik scan, zie waar het over gaat en waar ik naar op zoek ben ofzo, en bij een code, heb ik zelf dat ik dat minder makkelijk kan scannen en dan zien waar het over gaat, dan moet ik echt meer stap voor stap kijken wat er gebeurt om te weten waar het over gaat zeg maar, dat denk ik.

Interviewer:
Oké. Nou dat zijn mijn vragen tot jouw algemene leesstrategieën. Heb je zelf nog opmerkingen over?

Participant 7:
Nee, denk ik niet.

Interviewer:
Oké, nou dan gaan we gewoon door naar de volgende stap. Ik heb zelf dus een Python code geschreven, die ga je dus lezen. Het is wel belangrijk dat je het zo leest dat je de code ook echt probeert te begrijpen, want ik zal straks nog vragen stellen tot jouw interpretatie van de code, dus wat de code doet enzo, en ook nog over jouw genomen leesstrategieën, maar dan focus ik vooral op jouw leesvolgorde en jouw lees focus. Hierbij mag je natuurlijk ook aantekeningen maken zeg maar als dat voor jou relevant is, maar dit mag je gewoon zelf kiezen, want het gaat om jouw leesstrategieën natuurlijk. Oké, dan zie je nu in de chat de link naar de code, je hoeft niks te submitten, gewoon de code lezen. Je mag gewoon alle tijd nemen, dus zeg maar wanneer je klaar bent.

Participant 7:
Mag ik een vraag stellen tussendoor.

Interviewer:
Nou als het gaat om jouw leesstrategieën.

Participant 7:
Nee, nee, niet echt specifiek daarover, maar meer dat ik me afvroeg, als je zo meteen een vraag ga stellen, moet ik dan alles heb onthouden wat ik bekijk, of kan ik dan het ook bekijken.

Interviewer:
Ja, je kan het bekijken, straks ook, dus de code mag je bijhouden.

Participant 7:
Oké, oké, dus ik hoeft niet precies te onthouden denk ik, top.

[*De participant begint nu met het lezen van de code. De participant heeft in stilte gelezen en geen opmerking gemaakt tijdens het lezen.*]

[*Eind leesprocedure. Deze leesprocedure heeft 15 minuten geduurd.*]

Participant 7:
Ik ben soort van klaar.

Interviewer:
Ja, oké, als je klaar bent, dan kunnen we door naar de volgende stap. Nu ga ik een aantal vragen stellen die betrekking hebben tot jouw interpretatie van de code. Kan je dus de code samenvatten wat die doet?

Participant 7:
Eerst checkt die een patroon, bij het eerst deel, voor in een tekst, en dan bij tweede deel, oké wacht even, ik moet heel even kijken want oké, ja bij tweede deel verplaatst die een waarde in een lijst met waardes, en dan bij derde deel, maakt die denk ik hoofdletters van, een sublijst maakt die een lijst daarvan met hoofdletters, als dat capitalize dat is, en bij vier voer je deel één, twee en drie uit met inputwaardes gewoon eigenlijk.

Interviewer:
Oké. Wat is volgens jou de output van deze code?

Participant 7:
Dat heb ik niet helemaal uitgevoerd, kan ik nu dat nog even doen?

Interviewer:
Ja.

Participant 7:
Oké, dat moet ik even dan helemaal gaan bekijken, want ik heb het niet helemaal uitgevoerd, dus ja.

[*De participant heeft de code nog eens doorgelopen, heeft 14 minuten geduurd*]

Participant 7:
Ik denk dat er dan een string uitkomt met vier waardes erin, en de eerste waarde is dan negen gedeeld door drie, met de tweede dan elf gedeeld door drie, maar dan uitgerekend natuurlijk, en de derde veertien gedeeld door drie, en vierde twintig gedeeld door drie.

Interviewer:
Oké, ja. En dat is dan volgens jou de output van deze code?

Participant 7:
Ja.

Interviewer:
Oké. Dat zijn mijn vragen tot jouw interpretatie van de code. Heb je zelf nog andere opmerkingen over?

Participant 7:
Nee.

Interviewer:
Oké, dan gaan we door naar de volgende stap. Nu ga ik een aantal vragen stellen tot jouw lees focus. Dus mijn eerste vraag is: waar focus je meest op tijdens het lezen?

Participant 7:
Nou, eerst focuste ik, bij deze code toch?

Interviewer:
Ja, bij deze code ja.

Participant 7:
Ja, eerst focuste ik gewoon op wat er gebeurt zegmaar, dus gewoon wat er stap voor stap gebeurt en wat dat ongeveer doet zeg maar, en toen ik de output moest gaan vinden, toen ging ik echt meer kijken wat er met de waardes gebeurt, dus eerst meer over het algemeen wat er gebeurt en daarna specifieker.

Interviewer:
Oké. En welk deel van deze code is volgens jou het belangrijkst?

Participant 7:
Misschien het laatst deel ofzo zodat je een uitkomst krijgt, ik weet niet specifiek, voor mijn begrip of.

Interviewer:
Ja, dus welke code is volgens jou het belangrijkst voor jouw leesstrategieën en dat je ook wel een beetje kan uitleggen waarom dat is.

Participant 7:

Dat laatst deel, zeg maar deel vier, dan komt allemaal bij elkaar, dat maakt het wel duidelijker wat er gebeurt soort van ofzo als je het echt soort van gaat uitvoeren, dus daarom dat het misschien als het belangrijkst deel.

Interviewer:
Ja, en welk deel van deze code vind je het moeilijkst om te begrijpen?

Participant 7:
Misschien deel twee, vond dat wel veel waardes ofzo, soms zit je wel, waardes moest onthouden en welke waarde nou wat returned enzo, dus dat vind ik wel lastig.

Interviewer:
Oké. Dus omdat hier heel veel waardes worden gebruikt?

Participant 7:
Ja, veel verschillende waardes worden gebruikt dan de andere.

Interviewer:
Dus dat is de reden waarom je dit deel het moeilijkst vindt?

Participant 7:
Ja.

Interviewer:
Oké dan. En welke Python concepten die je in deze code bent tegengekomen vind je het moeilijkst?

Participant 7:
Misschien niet een specifiek Python concept, maar dat er heel veel dingen, dat er veel loops in elkaar staan, dat is, vind ik wel lastig, maar dat is niet misschien specifiek voor Python.

Interviewer:
Dus nested loops, dat vind je moeilijk?

Participant 7:
Ja, dat ja, want de waardes veranderen er nu in de loop en dan ga je terug uit de loop en dan veranderen de waardes weer, na wat ze eerst waren zegmaar, dus dat is wel lastig te volgen soms.

Interviewer:
Oké. Dat zijn mijn vragen tot jouw lees focus. Heb je zelf nog andere opmerkingen over?

Participant 7:
Nee, eigenlijk niet.

Interviewer:
Oké, dan gaan we door naar het volgend deel. Ik ga nu dus nog een aantal vragen stellen tot jouw

leesvolgorde die je hebt genomen. Weet je nog bij welke line je bent begonnen met het lezen?

Participant 7:
Ja, bij gewoon echt het begin, nou ik heb wel eerst kort gescand, zegmaar van oké [...] het is opgedeeld in vier delen zegmaar, maar daarna ben ik gewoon bij deel één echt begonnen, bij de eerste regel.

Interviewer:
En bij welke line ben je gestopt met het lezen?

Participant 7:
Niet bij de allerlaatste, wel toen ik soort van scande bij de allerlaatste, maar toen ik echt ging uitvoeren, ben ik soort van gestopt bij regel vierenvijftig dan zeg maar, omdat ik dan de else variabele, misschien dan bij de else, want toen dacht ik dat de else is niet meer nodig.

Interviewer:
Ja, oké. Dus je hebt de code eerst gescand en dan begon je pas met het lezen?

[*Participant heeft beaamd*]

Interviewer:
Weet je nog in welke volgorde je bent door de code delen heen gelopen?

Participant 7:
Bedoel je specifiek binnen deel één of eerst deel één, dan deel twee, dan deel drie.

Interviewer:
Ja, de tweede.

Participant 7:
Oh ja, ik heb gewoon eerst deel één, dan deel twee, dan deel drie, toen deel vier.

Interviewer:
Oké. Dus lineair, van boven naar beneden?

Participant 7:
Op volgorde, ja.

Interviewer:
Dat zijn mijn vragen tot jouw leesvolgorde. Heb je zelf nog andere opmerkingen over?

Participant 7:
Nee, niet specifiek.

Participant 8
Datum: 26 mei 2021
Duur: 27 minuten

Interviewer:
Nu ga ik een aantal vragen stellen tot jouw algemene leesstrategieën die je zou gebruiken om een tekst te lezen en ook om een code te lezen. Dus mijn eerste vraag is dan: wat zijn jouw algemene leesstrategieën die je zou gebruiken om een moeilijk tekst te lezen, bijvoorbeeld een wetenschappelijk artikel?

Participant 8:
Ik moet wel even nadenken, als ik een wetenschappelijk artikel lees, van meestal doe ik wel gewoon, als ik echt alleen niet alles zou lezen, dan alleen de inleiding en de conclusie natuurlijk, en dan van, na het, regel van regel, ik weet niet echt hoe ik het, ik weet niet precies wat voor antwoord je zoekt, even denken, ik denk niet bewust de strategie die ik toepas in ieder geval, ik weet niet of dat telt als een antwoord.

Interviewer:
Ja, ik heb natuurlijk ook geen exacte antwoord.

Participant 8:
Nee ja, nee, volgens mij ja, het is niet dat ik echt specifiek, specifiek iets speciaals doen met de tekst, want ik lees het gewoon, en als ik iets niet begrijp, dan lees ik het opnieuw, weet je wel, dus, ja dat is het gewoon.

Interviewer:
Oké. Heb je ook algemene leesstrategieën die je zou gebruiken om een lange code te lezen?

Participant 8:
Nou ja ik zit wel naar de, natuurlijk als het goede code in ieder geval goed opgedeeld in functies en gedeeltes, dus probeer wel alles op te splitsen in verschillende stukken om makkelijker te begrijpen en natuurlijk ook zoek je naar commentaar, dat is natuurlijk wel het belangrijkst, want dat legt gewoon uit hoe de code werkt, maar als er geen [...] commentaar is, dan probeer ik meestal een beetje te kijken naar de functienamen en ook de algemene variabel namen, dus ik denk dat ik minder snel naar de structuur kijk en meer naar de namen van alles, want dat, ik vind dat toch meer bij een goede programmeur in ieder geval zegt over de code dan de structuur zelf, ook al is het structuur belangrijk hoor, dus niet dat ik dat helemaal negeer, maar ja.

Interviewer:
Ja, oké. En wat is volgens jou het belangrijkst verschil tussen het lezen van een tekst en het lezen van een code?

Participant 8:
Nou, sowieso bij het lezen van een code is het, het is gewoon, het is niet gestructureerd als tekst natuurlijk, van, ik kan van code van boven naar beneden en van weer naar onder springen, dus dat

sowieso, dat is geen, hoe heet dat, van links naar rechts, van boven naar onder, dus ja, ja ik denk bij code moet je echt commentaar bij lezen, de documentatie, en ja, zelfs naar andere webpagina's om te begrijpen hoe het werkt, dus het belangrijkst verschil is dat je niet lineair van een tekst kan volgen.

Interviewer:
Oké, dat zijn mijn vragen tot jouw algemene leesstrategieën. Heb je zelf nog andere opmerkingen over?

Participant 8:
Nee, ik denk dat ik wel alles heb gezegd.

Interviewer:
Oké. Dan gaan we door naar het volgend gedeelte. Ik heb dus een Python code geschreven en die ga je dus lezen. Dus het is wel zo dat je echt probeert te begrijpen wat de code doet, want straks zal ik nog vragen stellen tot jouw interpretatie van de code en wat de code doet enzo. Maar ik ga ook vragen stellen tot jouw genomen leesstrategieën en het gaat vooral om jouw leesvolgorde en lees focus. Hierbij mag je natuurlijk aantekeningen maken als je dat wilt, het gaat om jouw leesstrategieën. Dan gaan we gewoon gelijk beginnen. In de chat vind je de link naar de code, je hoeft niks te submitten, gewoon lezen. Neem je tijd, zeg maar wanneer je klaar bent.

[*De participant begint nu met het lezen van de code, de opmerkingen die de participant heeft gemaakt tijdens deze leesprocedure is hieronder genoteerd.*]

Oké, dus de functie check pattern, dus gaat waarschijnlijk iets van een patroon vinden in een string als in de variabele input string heeft, als het lengte nul is, dan heb je natuurlijk niet, oh wacht, de pattern is ook een s, oh, pattern is hetgeen dat we in de input string willen vinden, en die mogen natuurlijk beide niet nul zijn, dan loop je gewoon over de, hoe heet dat, over de input string heen, even kijken, nee dat is niet super eenvoudig, if pattern in input string, als je dat gewoon zo kunt gebruiken, dan weet ik niet waarom je over de hele string moet heen loopen, dan ga je toch wel naar de letter, nee, even kijken, input string ja, ja ja, moet ik alleen part één of alle drie hoeft te doen.

Interviewer:
Alles.

Participant 8:
Moet ik nu zeggen want anders ga ik het straks vergeten wat één deden, is het wel makkelijkst als ik alle vier bekijk.

Interviewer:
Wat jij liever wil.

Participant 8:
Oké, dan zeg ik gewoon, dus één, één heeft gewoon twee inputs, input string en pattern, en loop je over de letters van de input string en de letters van pattern en checkt die gewoon of pattern in input string zit, en zo ja, dan return je de index waar de pattern in input string zit, denk ik

in ieder geval, oké, twee, moving average, moving average, wat zal dat betekenen, groter gelijk aan n min één, oh die tweede is lastig, oké wacht ik ga hardop nadenken, dus er loopen over, over de lengte van de values, ik weet niet, ik denk dat het een array is, dat lijkt me het logisch in ieder geval, if i groter of gelijk is aan n min één, maar dat is raar, want dan begin je altijd pas bij de laatste n volgens mij, maar je doet het alleen niet zoals als je bij de laatste dingen in een for loop [...], of wacht, oh nee, huh, oh oké, selector is i min n, als selector is kleiner dan i, dit is een lastige stuk code, moving average plus is, oh value is een array ja, dus voor alles in values, moving average plus is, values, maar als selector kleiner dan i, dat is sowieso volgens mij kan dat een for loop zijn, maar dat maakt niet heel veel uit, omdat je nu altijd selector plus een doet, maar dat is, maar ja, eigenlijk is een while loop en een for loop hetzelfde, selector is i min n, i is altijd groter dan n natuurlijk, nee ik heb het echt geen idee wat de tweede doet, sorry maar ik, moving average, nee, ik weet gewoon niet wat de tweede doet, is dat dan een acceptabel antwoord of niet.

Interviewer:
Ja, het is ook acceptabel als je het echt niet begrijpt.

Participant 8:
Ik, ik zit er naar te kijken, maar moving average, [...] van de bovenste helft van de value is een array, deel je door n, maar ja, wat zou n moet zijn, weet je wel, en wat het dan, je [...] nee ik weet niet, ik ga gewoon naar part drie toe, list of lists, dus die is als goed is gewoon je krijgt, de functie krijgt een lijst van lijsten en loop die over alle lijst delen en daar binnen loop je over alle items binnen elke lijst die in de lijst van lijst zit, en dan, dan voeg je, dus je hebt nieuwe variabele, nieuw list aangemaakt, en dan, capitalize, dat betekent hoofdletters maken toch, kapitaliseren, volgens mij is dat het, ja met een hoofdletter schrijven, dus, oh waar is de code, hier, dus je gooit gewoon in een nieuwe list alle items volledig met hoofdletters van alle lijsten gooi je gewoon in een nieuwe lijst, nou maak je hoofdletters van, en dan return, dus je krijgt, je ziet dan een lijst, en dat is nieuwe list, en dan part vier, input string is test test, oh pattern is test, okay dat is wel iets logischer, for substring in input string, substring is dan gewoon een letter, als je over de string heen loopt, if check pattern oh huh, oh dat is de functie van een, oh nee, hier je gebruikt wel al die functies, oké, als het, nee in ieder geval, als het patroon test in de input string zit, oh in de substring, maar de substring is toch gewoon een, oh wacht, input string is natuurlijk een array, sorry, dus substring is gewoon de string zelf, dat is veel logischer, nee als het patroon in de substring zelf zit, dan doen we, dan doen we moving average dat ik nog steeds niet, [...], en anders doen we process list en dat was, dat was gewoon alles hoofdletters maken, moving average, even kijken [...], dus we krijgen [...], values is een array, het nadeel aan Python is van weet je niet dan zien wat type variabelen zijn, vind zo irritant, oh for i in range dus we loopen over de array heen, oh n, wat is n precies, dat is gewoon een index in het array, dus als we na die index min één zijn [...] bij nul, min een, of na of op, dan selector is i min een dus, oh dus selector wordt meteen een, [...] i min één, een beetje vergeleken met n min één, wat even kijken, dus het is twee, dus als, oh hij doet meteen plus één, oké dat kan het er, dus, dus we beginnen gewoon op nul met selector, moving average is dus de nieuwste waarde gedeeld door, oh je neemt volgens mij gewoon de average van de eerste n waarde, even checken hoor, maar je bent, oh wacht maar je krijgt natuurlijk een, selector is [...], nou geen nul en twee, even kijken maar daarna loop je eentje verder, [...] een average van, wat een vage functie, even kijken, int moving average, ja je krijgt dus gewoon, dus stel n is twee dan krijg je gewoon de averages van elke drie stukken van het array, dus je pakt gewoon een, doorsnede van het

array van n breed en voor alle doorsnedes en daar maak je een, een lijst van, volgens mij is dat het, dat is mijn antwoord, dus dan ben ik klaar volgens mij.

[*Eind leesprocedure. Deze leesprocedure heeft 14 minuten geduurd.*]

Interviewer:
Ik ga nu dus een aantal vragen stellen tot jouw interpretatie van de code, maar dat heb je net ook wel ongeveer gezegd. Maar ik wil je nog eens vragen of je de code even kan samenvatten van wat je hebt begrepen.

Participant 8:
Oké dus, part vier daar begint de code want de rest is allemaal functies, dus we hebben een input string die gewoon, daar zit een patroon in of niet en we hebben een pattern die we willen vinden in de input string en dan voor beide van die strings checken we of het patroon dat we zelf gedefinieerd array inzitten, zo ja dan nemen we dus, dan printen we de moving average en de moving average is dus elk tijd de doorsnede van dat array zelf of stukken van het gemiddelde van stukken van de doorsnede van zelf hoe groot je het wilt, en als het patroon niet inzit dan printen we ook van een zelf gedefinieerde list, lijst, en alles alleen dan als hoofdletters, even kijken hoor [...] oh waar is die functie, oh daar is die, ja dan een, dan een list, oh ja klopt, oké dat was mijn antwoord.

Interviewer:
Oké. Wat is volgens jou de output van deze code?

Participant 8:
Oh dan moeten we nog gaan rekenen, oh wacht dat is trouwens helemaal niet, oh het is wel hetgeen, oké dus, dus in die eerste zit het woord test in, in beide zit het woord test trouwens in, dat is als goed is tenzij er een bug in zit dat je twee keer de moving average printen, want test test, als goed is, oh je maakt me bang als het twee keer dezelfde is, oh dat was, oh [...], ja als goed, ja dus dat is dan het gemiddelde van tien, twaalf en negen, het gemiddelde van twaalf, negen en elf, gemiddelde van negen, elf en veertien, het gemiddelde van elf, veertien en twintig, en dat print die als goed is twee keer.

Interviewer:
Oké. Ja, dat zijn dan mijn vragen tot jouw interpretatie van de code. Heb je zelf nog iets toe te voegen?

Participant 8:
Nee, ik heb niet het gevoel dat ik het goed heb gedaan.

Interviewer:
Nou dan gaan we door naar het volgend deel. Ik ga nu een aantal vragen stellen tot jouw lees focus. Waar focus je meest op tijdens het lezen?

Participant 8:
Van code of in het algemeen?

Interviewer:
Van code.

Participant 8:
Nou dat is echt wel de namen van de variabelen en functies en in dit geval is er geen commentaar maar normaal ook echt commentaar, want ik vind dat gewoon super veel [...] kan toevoegen van je code van als je bijvoorbeeld kan uitleggen, ik doe altijd super lang over variabelen namen, een beetje te lang, maar dan, ik heb zo [...] dat ik liever een beetje te lang maar wel duidelijk uitlegt dan een zo van die korte dat je eigenlijk alleen maar i en de j's gebruiken.

Interviewer:
Ja, oké. En welke deel van deze code is volgens jou het belangrijkst?

Participant 8:
Belangrijkst, oh nou ja het ligt natuurlijk aan wat er met deze code moet worden gebeurd, maar part vier is de main functie, ja voor de rest het is part één, twee en drie dat zijn gewoon verschillende andere functies, daar zit er niet echt een, nou ja check pattern zou je natuurlijk wel iets belangrijker kunnen zien dan twee en drie want die wordt gebruikt om te bepalen of het twee of drie doet, maar nou ja eigenlijk alle code hangt af van de check pattern functie dus als ik moet kiezen zal ik part één[...], maar dus niet echt, het ligt gewoon aan wat het doel is van de code normaal, normaal gezien.

Interviewer:
Dus part één vind je het belangrijkst?

Participant 8:
Ja, de check pattern functie.

Interviewer:
Mijn derde vraag was eigenlijk waarom vind je dat dit deel het belangrijkst, maar dat heb je al ongeveer beantwoord.

Participant 8:
Ja.

Interviewer:
En welk deel van deze code vind je het moeilijkst om te begrijpen?

Participant 8:
Moving average, want het enige een beetje daar zit is moving average en dus alle variabelen zijn een variant van moving average, dus heel lastig om te zien of de vervolgen en die while loop kan volgens mij, oh, en volgens mij kan die while loop gewoon een for loop zijn, maar dat weet ik niet zeker.

Interviewer:
Oké. En welke Python concepten die je in deze code bent tegengekomen vind je het moeilijkst?

Participant 8:
Dus het ding wat ik heel irritant vind aan Python is dat je dus niet mag zeggen dat selector bijvoorbeeld een integer is of dan, dat persoonlijk haat ik dat gewoon aan Python maar dat is echt mijn persoonlijk [...].

Interviewer:
Als je niet echt iets moeilijkst vind, kan je natuurlijk ook zeggen.

Participant 8:
Was dat dan deze code of van Python zelf.

Interviewer:
Van deze code.

Participant 8:
Oké sorry oké, ja ik weet niet of het dan, nee dan niet iets specifieks aan deze code dat ik echt, want ik bedoel append, capitalize, for loops, while loops die zijn ja, zijn allemaal je weet wel, dus het is makkelijk te zien wat er gebeurd, maar het is moeilijk te zien om te begrijpen wat er gebeurd, je weet wel, dus niet specifieks aan die code zelf.

Interviewer:
Oké. Dat zijn mijn vragen tot jouw lees focus. Heb je zelf nog iets toe te voegen?

Participant:
Nee.

Interviewer:
Oké, dan gaan we door. Nu ga ik een aantal vragen stellen tot jouw leesvolgorde. Dus weet je nog bij welke line je bent begonnen met het lezen?

Participant 8:
Bij part één ja, maar dat is omdat ik dacht dat, dat het, dus ik dacht dat part één, twee, drie en vier dus ik dacht oké dan begin ik bij één want hoe heet het, maar part vier is natuurlijk de main functie, maar volgens mij kan je in Python ook, naja ik begon echt gewoon bij één, ik moet niet te veel praten.

Interviewer:
Oké. Bij welke line ben je gestopt met het lezen? Weet je dat nog?

Participant 8:
Nou uiteindelijk moet ik natuurlijk zeggen hoe de code liep dus dan eindig ik in moving average, dus ik denk dan bij achtendertig gestopt omdat daar de vragen begint.

Interviewer:
Oké. En ik zie dat je gelijk was begonnen met het lezen van de code. Dus niet de code eerst gescand?

Participant 8:
Ja klopt, nou ja ik heb wel de part één, twee, drie en vier gezien in het begin.

Interviewer:
Alleen de commentaar?

Participant 8:
Ja, eigenlijk alleen de commentaar ja want toen ben ik dat gewoon naar part één gegaan, want in mijn hoofd is dat deel één.

Interviewer:
Dus eigenlijk heeft commentaar ook wel invloed, want daar focus je ook wel op?

Participant 8:
Ja zeker.

Interviewer:
Ja, oké, dat is prima. Volgens mij ben je gewoon deel één, deel twee, deel drie, deel vier in deze volgorde gelopen, toch?

Participant 8:
Ja klopt.

Interviewer:
Oké, ja. Dat zijn mijn vragen tot jouw leesvolgorde. Heb je zelf nog andere opmerkingen over?

Participant 8:
Nee niet perse, enige is dus ik dacht, het is gewoon op rechterlijk dacht dat het vier verschillende puzzeltjes waren in plaats van dat het een gehele code was dat ik daarom echt bovenaan begon.

Interviewer:
Ja, want anders zou je bij welk deel begonnen zijn?

Participant 8:
Part vier sowieso, ja als ik meteen, dan zet ik ook meteen de input string en de pattern en dat ja, dus als ik nu nog een keer zou beginnen dan zou ik bij part vier beginnen.

Interviewer:
Ja, en dan part vier, part één, twee, drie, zo zou jij gelopen zijn?

Participant 8:
Even kijken, check pattern is één, ja want dat is gewoon weer in die volgorde ja.

Interviewer:
Ja, de volgorde wat in de main functie staat zeg maar?

Participant 8:
Ja.

Interviewer:
Ja, als je geen opmerking hebt, dan zijn we klaar met dit.

Participant 8:
Nee, geen opmerkingen.

Participant 9
Datum: 26 mei 2021
Duur: 21 minuten

Interviewer:
Ik ga dus eerst een aantal vragen stellen tot jouw algemene leesstrategieën die je zou gebruiken om een normale tekst te lezen en een code. Dus mijn eerst vraag is gewoon: wat zijn jouw algemene leesstrategieën die je zou gebruiken om een tekst te lezen, bijvoorbeeld een wetenschappelijk artikel?

Participant 9:
Eerst de titel natuurlijk en dan de abstract en dan de kopjes en dan gebaseerd daarop ga je dan zoeken wat je wil weten en dan de hele alinea lezen per interessant punt waar ik zoek eigenlijk, en ook met gewoon nieuws artikelen en dergelijk doe ik het.

Interviewer:
Oké. Wat zijn jouw algemene leesstrategieën die je zou gebruiken om een lange code te lezen?

Participant 9:
Naar de functienamen kijken dat vooral en daarna alle if statements en dergelijk kijken, dus waar alle, alle beslissingen in een code eigenlijk en daarna ga ik pas kijken naar wat er precies gebeurt met variabelen en dergelijk.

Interviewer:
Oké. En wat is volgens jou het belangrijkst verschil tussen het lezen van een tekst en het lezen van een code?

Participant 9:
Wat je bij een tekst niet heel veel heen en weer hoeft te schakelen tussen de bestanden natuurlijk wat je bij een code dan zie je iets en dan refereer naar een andere functie die je gewoon op zoek ben en dan daar kort iets gebruikt, dus ik denk vooral dat het feit dat er heel veel scriptjes is, is stuk onduidelijker.

Interviewer:
Oké. Dat zijn nou mijn vragen tot jouw algemene leesstrategieën. Heb je zelf nog andere opmerkin-

gen over?

Participant 9:
Nou behalve dat de code lezen dus niet lineair is, niet heel veel.

Interviewer:
Prima, dan gaan we gewoon door naar de volgende stap. Ik heb dus zelf een stukje Python code geschreven en dat ga je dus lezen. Het is wel zo dat de bedoeling is dat je zo leest dat je echt de code probeert te begrijpen wat die doet, want straks zal ik namelijk nog een aantal vragen stellen tot jouw interpretatie van de code en wat de code doet. Maar ik zal ook wat vragen stellen over jouw leesstrategieën, dan focus ik vooral op jouw leesvolgorde en jouw lees focus. Tijdens het lezen mag je natuurlijk ook aantekening maken als je dat wilt, het gaat uiteindelijk om jouw leesstrategieën natuurlijk. Oké, dat is wat ik wil zeggen, dus nu in de chat vind je de link naar de code. Je hoeft niks te submitten. Zeg maar wanneer je klaar bent met het lezen.

[*De participant begint nu met het lezen van de code. De participant heeft in stilte gelezen en geen opmerking gemaakt tijdens het lezen.*]

[*Eind leesprocedure. Deze leesprocedure heeft 8 minuten geduurd.*]

Participant 9:
Oké, ik denk dat ik het ongeveer wel weet.

Interviewer:
Ja, oké, top.

Participant 9:
Die eerste functie check pattern die kijkt of de string, input string een bepaalde patroon vast die een pattern zet, en die returned uiteindelijk waar of niet waar en dat is het, moving average die krijgt een lijst van getallen mee en die returned een lijst van getallen met de verschillende gemiddeldes van bijvoorbeeld de eerst drie getallen en dan de vier getallen en dan weer vijf getallen, zes getallen gebaseerd op welk n je meegeeft, en process list die krijgt een lijst van lijst mee en die voegt allemaal samen, alle items van een sublijst samen in een lijst en die returned die dan, en uiteindelijk bij het laatst deel hebben we een lijst met twee strings en die bevat niet dat woordje test en dan weer wel, we hebben een patroon dat test bevat, en dan voor elke string in de lijst met de strings die op test lijken maar wel of niet bevatte, daarvoor wordt gekeken of de string test bevat, als dat zo is, dan wordt het bewegende gemiddelde geprint van de values lijst, als dat niet zo is, dan wordt de superlijst geprint als een lange lijst, dus met duck, cat, flower, tree, whales en swordfish achter elkaar, denk ik.

Interviewer:
Ja, ik heb dus een aantal vragen tot jouw interpretatie van de code, net heb je ook natuurlijk de code soort van samengevat, dat was ook mijn eerste vraag eigenlijk, of je de code kan samenvatten wat die doet.

Participant 9:
Wat bedoel je dan met interpretatie.

Interviewer:
Ja, wat je hebt begrepen van de code. Maar, oké, als ik nog een keer vraag kan je de code samenvatten, zou je hetzelfde doen als wat je net hebt gezegd?

Participant 9:
Ja, ik denk het wel, ik kan de interpretatie gewoon iets duidelijker geven, wat ik lastig vond waar de alle if statement en dergelijke en alle variabelen, oh sorry, niet de if statement maar de variabelen assignment gebaseerd op andere variabelen, dus zoals daar staat selector is i min n, en daarna moet je weer gaan kijken wat is i, wat is n, ja de if statement, de while statement spreken vaak wel voor zich, maar als het dan een lastige variabele heeft, dan moet je ook iets terug gaan kijken wat het is, dus dat vond ik lastig aan de code, ik vond het onnodig moeilijk geschreven volgens mij op sommige plekjes omdat het technisch is met break point en dergelijk, maar ja verder was het niet super lastig om te lezen behalve dat comments zijn er niet waren maar natuurlijk, maar dat is denk ik onderdeel van een, van het experiment.

Interviewer:
Oké, ja. Wat is dan volgens jou de output van deze code?

Participant 9:
Even kijken hoor, als eerst de duck, cat, flower, tree, whales en swordfish in een lijst en op de volgende rij het gemiddelde dat wordt de moving average output van tien, twaalf, negen, elf, veertien, twintig en in dit geval denk ik dat het als eerst het gemiddelde van tien, twaalf en negen is, dan het gemiddelde van tien, twaalf, negen en elf, dan het gemiddelde van tien, twaalf, negen, elf en veertien, en dan het gemiddelde van tien, twaalf, negen, elf, veertien tot twintig.

Interviewer:
Ja, oké. Dat zijn mijn vragen tot dit gedeelte. Heb je zelf nog andere opmerkingen over?

Participant 9:
Nee.

Interviewer:
Oké. Dan gaan we gewoon door naar het volgend gedeelte. Nu ga ik nog een aantal vragen stellen tot jouw lees focus. Dus waar focus je meest op tijdens het lezen?

Participant 9:
In dit geval?

Interviewer:
Ja, in dit geval.

Participant 9:

Variabelen namen en de namen van de functies, in het begin had ik eerst helemaal de namen van de functies gekeken om te kijken wat het algemeen ze zijn voor programma of samenhang waarvan het bleek dat die los functies zijn, ja verder variabelen namen als ze makkelijk zijn te herkennen dus als ze gewoon goed zijn genoemd, dat doet ze maar waar een keer goed na te denken over wat het is en als het goed is hoop ik [...] vanaf daarna moet je weten wat het betekent, alweer is dat niet altijd want soms raak je in de war van wat alles betekent, maar goed ja, vooral gefocust op variabelen namen, en nu condities ertussen, dat is het eigenlijk, niet zo zeer de if statement en de break points hebben en dan type.

Interviewer:
Oké. En welke deel van deze code is volgens jou het belangrijkst?

Participant 9:
De part vier.

Interviewer:
En waarom?

Participant 9:
Omdat die uiteindelijk alles aanroept, anders zou het losse codes zijn die niks doet.

Interviewer:
Oké. En welk deel van deze code vind je het moeilijkst om te begrijpen?

Participant 9:
Part één, check pattern.

Interviewer:
En waarom?

Participant 9:
Omdat het heel veel lagen heeft, het is een if statement in een while statement in een for loop en een if statement in een while statement, dus op een gegeven moment raak ik een beetje de draad kwijt van wat wat doet, ja wat wat doet.

Interviewer:
Oké. En welke Python concepten die je in deze code bent tegengekomen vind je moeilijk om te begrijpen?

Participant 9:
Ik vond regel drieëntwintig onduidelijk omdat ik nooit zelf die notatie gebruik van getal dubbele punt getal, maar ik neem wel aan dat dat betekent dat je dat deel van de string pakt, en ik denk dat dat het enig is wat ik ben tegengekomen wat ik nog niet snapte.

Interviewer:

Oké, ja. Dat zijn dan mijn vragen tot jouw lees focus. Heb je nog iets toe te voegen?

Participant 9:
Nee, de kleuren zijn belangrijk, dat is handig, ik zei net vooral variabelen namen natuurlijk, maar dat is eigenlijk gewoon alle blauwe dingen waar ik op focus, en de oranje dingen die liet ik vooral weg, want daar moest je maar een keer naar te kijken, ja dat was het eigenlijk.

Interviewer:
Dan gaan we gewoon door naar de volgende stap. Dan ga ik een aantal vragen stellen tot jouw leesvolgorde. Dus bij welke line ben je begonnen met het lezen? Weet je dat nog?

Participant 9:
Bij regel twee en toen daarna regel zevenentwintig en toen eenenveertig en toen negenenveertig, waarschijnlijk heb ik het daarna [...] dat is geen functie, dan ga ik eerst naar de rest kijken.

Interviewer:
Oké. En bij welke line ben je ongeveer gestopt met het lezen?

Participant 9:
Ik denk iets van drieëntwintig, had part één overgeslagen omdat ik het niet helemaal uit kwam, dus heb ik eerst even de rest doorgekeken, en toen daarna nog even part één goed bekeken.

Interviewer:
Oké. Je was dus eerst begonnen met het scannen van de code toch? En dan ben je pas begonnen met het lezen?

Participant 9:
Ja precies.

Interviewer:
En weet je nog in welke volgorde je bent door de code delen heen gelopen? Dus van welk deel naar welk deel enzo?

Participant 9:
Ja, eerst de namen van de functies.

Interviewer:
Nee, zeg maar met parts, dus ben je begonnen met part één en daarna part twee naar part drie enzo.

Participant 9:
Oh op die manier, ik heb eerst part één, twee, drie en vier gedaan, maar part één en twee heb ik uiteindelijk nog een keer gedaan, en toen uiteindelijk part één als laatste [...] en toen was ik klaar.

Interviewer:

Oké, ja, prima. Dat zijn mijn vragen tot jouw leesvolgorde. Heb je zelf nog iets toe te voegen?

Participant 9:
Nee, denk ik niet.

Participant 10
Datum: 27 mei 2021
Duur: 18 minuten

Interviewer:
Nu ga ik een aantal vragen stellen tot jouw algemene leesstrategieën die je zou gebruiken om een tekst te lezen en ook om een code te lezen. Dus mijn eerste vraag is gewoon: wat zijn jouw algemene leesstrategieën die je zou gebruiken om een moeilijk tekst te lezen, bijvoorbeeld een wetenschappelijk artikel?

Participant 10:
Mijn strategie zou zijn om eerst de titels van subtitels en titels te bekijken, daarna de inleiding wat beter te lezen en over de andere stukken meer heen te scannen.

Interviewer:
Oké. En wat zijn jouw algemene leesstrategieën die je zou gebruiken een lange code te lezen?

Participant 10:
Om een lange code te lezen, ik zou eerst bekijken naar of er eventuele classes zijn, deze even kijken wat het class ongeveer is zonder hem exact te lezen, en ik zou daarna denk ik even door de functienamen heen scannen en dan beginnen bij de main functie om te bekijken waar die heen leidt.

Interviewer:
Oké. En wat is volgens jou het belangrijkst verschil tussen het lezen van een tekst en het lezen van een code?

Participant 10:
Ik denk de volgorde begrijpen waarin, bij een tekst ga je van boven naar beneden door de tekst heen, vaak van begin tot einde, waarbij bij de code door functies en dergelijke is het niet van boven naar beneden lezen, maar ga je ook soms ineens een stukje naar boven, stukje naar onder en die volgorde begrijpen is vind ik belangrijk.

Interviewer:
Oké. Dat zijn dus mijn vragen tot jouw algemene leesstrategieën. Heb je zelf nog andere opmerkingen over?

Participant 10:
Nee, ik denk het niet.

Interviewer:
Oké, dan gaan we gewoon door. Ik heb zelf een stukje Python code geschreven, dus die ga je nu ook lezen. Hierbij wil ik wel even zeggen dat het belangrijk is dat je echt probeert te begrijpen wat de code doet, want straks zal ik vragen stellen over jouw interpretatie van de code, dus wat de code doet en zo. Ik zal ook nog wat vragen stellen over jouw leesstrategieën die je hebt genomen tijdens het lezen van deze code, dus het gaat vooral om jouw leesvolgorde en jouw lees focus. Hierbij mag je natuurlijk wel aantekeningen maken als je dat wilt, maar het gaat om jouw leesstrategieën, dus je mag zelf bepalen. Nu in de chat vind je de link naar de code, je hoeft niks te submitten, maar het is gewoon handiger om het zo in de form te doen, dan kan je gelijk openen. Zeg maar wanneer je klaar bent met het lezen.

[*De participant begint nu met het lezen van de code. De participant heeft in stilte gelezen en geen opmerking gemaakt tijdens het lezen.*]

[*Eind leesprocedure. Deze leesprocedure heeft 8 minuten geduurd.*]

Participant 10:
Oké, ik denk dat ik hem helemaal doorheen ben gelezen en dat het redelijk begrijp.

Interviewer:
Oké, helemaal top. Dan gaan we gewoon door. Nu ga ik een aantal vragen stellen tot jouw interpretatie van de code. Dus kan je de code samenvatten wat die doet?

Participant 10:
De code die krijgt een input string en een pattern string, heeft het en probeert het pattern te vinden binnen die input string en als het pattern vindt dan gaat het met een lijst met getallen aan de slag, waarvan ik niet volledig uit het code kon halen wat er precies gebeurde, maar het gaat met een deel van de getallen in de rij, gaat het de waarde aanpassen, volgens mij om het gemiddelde te verleggen, en als het patroon niet heeft gevonden in die string, dan neemt het een lijst waarin koppels van woorden staan, en die verwerkt het in een lijst achter elkaar waarbij elk woord een hoofdletter krijgt, en volgens mij volledig in hoofdletters komen te staan.
Interviewer:
Oké, ja. Mijn tweede vraag was dan eigenlijk van: kan je ongeveer zeggen wat de output is van deze code?

Participant 10:
Even kijken, is het mogelijk om dan de stukje code even bij te pakken om te kijken wat de input string was of is het volledig gewoon.

Interviewer:
Ja, wat je van de code hebt begrepen zeg maar. Wat denk je dat de output is van deze code.

Participant 10:
Ik denk dat de output de array met die waardes is, alleen weet ik niet zeker dus wat er met die waardes exact gebeurd in die lijst, maar volgens mij worden er de laatste vier of de laatste drie

getallen aangepast.

Interviewer:
Oké, ja. Dat zijn mijn vragen tot jouw interpretatie van de code. Heb je zelf nog andere opmerkingen over deze code?

Participant 10:
Nou ja, ik merkte wel dat de eerste functie, dat de derde functie ging het makkelijkst, wel dat was ik redelijk snel doorheen, de eerste ging ook wel redelijk, met die tweede had ik wel de meeste moeite om te begrijpen wat er exact gebeurde, weet niet of het komt doordat er veel variabelen waren of omdat ik niet in een oogopslag zag oh dit gebeurt ongeveer, bij die eerste kwam uit de naam ook wel een beetje naar voren wat er gebeurde en begreep ik daardoor al misschien makkelijker, ik denk dat.

Interviewer:
Oké, dan gaan we gewoon door. Nu ga ik dus een aantal vragen stellen tot jouw lees focus. Ja, waar focus je meest op tijdens het lezen?

Participant 10:
Tijdens het lezen van teksten in het algemeen of code?

Interviewer:
Van deze code.

Participant 10:
Van deze code, ik merk wel dat mijn focus eerst, wel ging ook naar de variabelen naam en de functienaam om te kijken of ik daar al wat kan afleiden, en dat daarna mijn meest focus lag op het bijhouden wat variabelen bevatten om te kijken wat ze daadwerkelijk deden, van als bijvoorbeeld variabelen vergeleken werden dat ik in mijn hoofd van oké deze variabele is dit en dit is dat, dus wat komt er uit deze vergelijkingen.

Interviewer:
Oké. En welk deel van deze code is volgens jou het belangrijkst?

Participant 10:
Het belangrijkst is denk ik toch wel het part vier heet het geloof ik, stuk waarin andere functies worden aangeroepen en vanuit daar begint het eigenlijk het proces van welke kant de code opgaat en ook welke output d'r geshowd wordt.

Interviewer:
Ja, oké. Mijn derde vraag was gewoon waarom vind je dat dit deel belangrijkst is, maar dat zeg je natuurlijk al. Kan je nog een klein beetje toelichten?

Participant 10:
Ja, ik denk dat die, dat stukje het belangrijkst is omdat je vanuit daar dus, het is een goed

startpunt ook om te beginnen met waar de code heen gaat, dan handig misschien om de code, om de sub-functies van te voren al gezien te hebben, maar vanuit daar kan je kijken van oké weet je de code gaat eerst naar deze functie en dan ga ik zelf ook eerst bij deze functie kijken wat daar gebeurt voordat ik de andere functies dieper ga bekijken, dus vandaar dat ik dat belangrijk vind omdat voor mijn aanpak goed startpunt is.

Interviewer:
Oké. En welk deel van deze code vind je het moeilijkst om te begrijpen?

Participant 10:
Dat was part twee geloof ik, het stukje met de moving average, doordat er variabelen namen vond ik lastig te koppelen aan wat ze exact deden en doordat er veel variabelen waren om bij te houden maakt het voor mij lastig om het te interpreteren.

Interviewer:
Oké. Welke Python concepten die je in deze code zijn tegengekomen vind je moeilijk?

Participant 10:
Ik merk dat ik even moest schakelen met het if x in y omdat ik heel erg in mijn hoofd zat in het begin met dat x in y met in als een soort for loop eerst las van dat arrange ding, maar dat is het niet, maar dat was wat ik even in de war was, het scheelt voor de rest omdat ik in mijn scriptie ook met Python doet dat ik op dit moment iets meer in Python thuis ben dan ik ook voorheen was.

Interviewer:
Oké. Nou dat zijn mijn vragen tot jouw lees focus. Heb je zelf nog iets om toe te voegen?

Participant 10:
Nee, niet perse, het was interessante code om zo inderdaad even probeer door te lezen.

Interviewer:
Oké, dan gaan we door naar het volgend gedeelte. Nu ga ik een aantal vragen stellen tot jouw leesvolgorde die je hebt genomen om deze code te lezen. Dus bij welke line ben je begonnen met het lezen? Weet je dat nog?

Participant 10:
Bij welke regel.

Interviewer:
Ja.

Participant 10:
Dat is de regel van, regel twee, de functie definitie van part één, dat is eerst wat ik gelezen heb.

Interviewer:
Oké, ja. En bij welke regel ben je dan gestopt met het lezen?

Participant 10:
Helemaal aan het einde gestopt of?

Interviewer:
Ja.

Participant 10:
Dat is geweest bij regel vierenvijftig, ja regel vierenvijftig.

Interviewer:
Oké. En was je gelijk begonnen met het lezen van de code of heb je eerst gescand?

Participant 10:
Ik heb eerst gescand, ja ik ben begonnen met de functienamen van de eerst drie functies doorlezen.

Interviewer:
Oké, weet je nog in welke volgorde je bent door de code delen heen gelopen?

Participant 10:
Ja, even kijken, ik ben bij part één heb ik de functienaam gelezen, toen de functienaam van part twee, toen van drie, toen ben ik, heb ik eerst de hele part vier gelezen.

Interviewer:
Oké, ja. Dat zijn mijn vragen tot jouw leesvolgorde. Heb je zelf nog iets toe te voegen?

Participant 10:
Nee, ik denk het niet, ja misschien een kleine verduidelijkings, nadat ik part vier helemaal gelezen had, had ik daarna dus naar de verschillende functies ben gegaan, eerst naar één, toen naar twee, toen naar drie, en toen ben ik even terug naar twee omdat ik toch niet helemaal begreep en toen ben ik geëindigd weer bij de main om te kijken wat de output was.