The Effect of Reading Code Aloud on Comprehension
An Empirical Study with School Students

Swidan, Alaaeddin; Hermans, Felienne

DOI
10.1145/3300115.3309504

Publication date
2019

Document Version
Final published version

Published in
CompEd'19

Citation (APA)
Swidan, A., & Hermans, F. (2019). The Effect of Reading Code Aloud on Comprehension: An Empirical Study with School Students. In *CompEd'19 : Proceedings of the ACM Conference on Global Computing Education* (pp. 178-184). Association for Computing Machinery (ACM). https://doi.org/10.1145/3300115.3309504

The Effect of Reading Code Aloud on Comprehension: An Empirical Study with School Students

Alaaeddin Swidan and Felienne Hermans
Delft University of Technology
Delft, The Netherlands
{alaaeddin.swidan,f.f.j.hermans}@tudelft.nl

ABSTRACT

In recent times, programming is increasingly taught to younger students in schools. While learning programming is known to be difficult, we can lighten the learning experience of this age group by adopting pedagogies that are common to them, but not as common in CS education. One of these pedagogies is Reading Aloud (RA), a familiar strategy when young children and beginners start learning how to read in their natural language. RA is linked with a better comprehension of text for beginner readers. We hypothesize that reading code aloud during introductory lessons will lead to better code comprehension. To this end, we design and execute a controlled experiment with the experimental group participants reading the code aloud during the lessons. The participants are 49 primary school students between 9 and 13 years old, who follow three lessons in programming in Python. The lessons are followed by a comprehension assessment based on Bloom's taxonomy. The results show that the students of the experimental group scored significantly higher in the Remembering-level questions compared to the ones in the control group. There is no significant difference between the two groups in their answers to the Understanding-level questions. Furthermore, the participants in both groups followed some of the instructed vocalizations more frequently such as the variable's assignment (is). Vocalizing the indentation spaces in a for -loop was among the least followed. Our paper suggests that using RA for teaching programming in schools will contribute to improving code comprehension with its effect on syntax remembering.

KEYWORDS

Reading Aloud (RA), Programming Education, Primary School, Bloom's Taxonomy

ACM Reference Format:
Alaaeddin Swidan and Felienne Hermans. 2019. The Effect of Reading Code Aloud on Comprehension: An Empirical Study with School Students. In *ACM Global Computing Education Conference 2019 (CompEd '19), May 17–19, 2019, Chengdu, Sichuan, China.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3300115.3309504

CompEd '19, May 17–19, 2019, Chengdu, Sichuan, China

ACM ISBN 978-1-4503-6259-7/19/05…$15.00
https://doi.org/10.1145/3300115.3309504

1 INTRODUCTION

Programming is increasingly taught to younger students, in some countries as part of the curriculum of primary and secondary schools [19]. We know, however, that learning programming is difficult [8, 24, 41]. The question arises on how do we make learning programming less difficult for younger students? One way could be applying pedagogies we know work for this age group but are uncommon in programming education.

Young children start to learn how to read by learning the connection between symbols, one or more letters in this case, and sounds and then combining them into words and sentences. Reading text aloud is encouraged for beginners since it focuses thoughts, help memorization and improves comprehension of text [9, 12, 37]. Also in mathematics, the same approach to reading aloud can be noticed in vocalizing simple operations and equations, or when introducing a new symbol [14, 35].

Although in later development stages and adulthood silent reading becomes the norm, our brains seem to be always ready for reading aloud. Studies have shown that the brain sends signals to the primary motor cortex, controlling the lips and the mouth, during silent reading [28, 33]. This brain activity is called *subvocalization*, which is used in particular when learners face long and new words. In programming education, educators seem to spend little effort on reading code aloud to, or with the students. The lack of this *phonology* knowledge leaves students with an extra cognitive load when reading code to understand functionality. In this regard, one study measured the subvocalization of experienced developers during programming tasks and showed that the subvocalization signals could differentiate the difficulty of the programming task [28]. Therefore, we hypothesize that training students in reading code aloud will lead them to spend less cognitive effort on the reading mechanics and thus improve their comprehension of code.

Therefore, the purpose of this paper is a first quantification of the effect of reading code aloud during lessons on school students' comprehension of basic programming concepts. Furthermore, we investigate how students benefit from the practice of Reading Aloud (RA) by following it as a sort of a guideline later.

To this end, we design and execute a controlled experiment in which 49 primary school students receive three lessons of programming in Python. The students are divided into two groups which get the same teaching materials and times. The students in the experimental group, however, are asked to repeat reading the code aloud following the instructor. We assess students' learning based on Bloom's taxonomy. Since the participants are absolute beginners in programming, we focus our assessment on the first two levels of the taxonomy: the Remembering-level and the Understanding-level. In this paper, we answer the following research questions:

RQ1 What is the effect of reading code aloud on the performance of students in the Remembering-level questions?
RQ2 What is the effect of reading code aloud on the performance of students in the understanding-level questions?
RQ3 How do students follow the vocalization guideline when they read code later?

Results show that the students in the experimental group scored significantly higher in the Remembering-level questions compared to the students in the control group. There is no significant difference between the two groups in their answers to the Understanding-level questions. The analysis shows that particular code vocalizations, such as the variable's assignment, are common among the two groups. On the other hand, the participants in both groups least vocalize the spaces needed for indentation in a for loop and list brackets. The following sections contain the details of the experiment's design and results.

2 BACKGROUND AND RELATED WORK

We provide an overview of research related to Reading Aloud (RA), particularly, the RA role in reading education for young students (Section 2.1) and previous literature involving the use of voice in programming environments (Section 2.2). We also overview selected prior research on the use of Bloom's taxonomy in assessing programming comprehension (Section 2.3).

2.1 RA and Comprehension: Natural Language Perspective

Most psychologists nowadays believe that reading is a process of sounding out words mentally even for skilled readers [33]. Brain studies [28, 29, 33] show that the primary motor cortex is active during reading, "presumably because it is involved with mouth movements used in reading aloud" [33, p. 90]. Therefore, it becomes highly important for beginner readers to learn the connection between sounds and symbols, or phonics. Previous research found that systematic phonics instruction produces higher achievement for beginning readers, where they can read many more new words compared to students following other approaches. For these reasons, in the United States, phonics has been included in reading programs in schools nationwide [33]. As a verbal approach, reading aloud (RA) helps in focuses thoughts and transforming it in specific ways, causing changes in cognition [12]. Takeuchi et al. [37] highlight that RA is effective for children language development in *"phonological awareness, print concepts, comprehension, and vocabulary"* [37]. Bus et al. [9] reports that reading books aloud brings young children *"into touch with story structures and schemes and literacy conventions which are prerequisites for understanding texts"*. Several experiments related to comprehension report that students identified the sounding out of words, or loudly repeating text as a means to regulate their understanding while reading [22, 25]. When comparing RA to silent reading, research has found that students comprehend significantly more information when they read aloud versus reading silently [27, 31]. Although other studies showed opposite results [17], there seems a consensus exists among researchers that the effects of reading aloud may differ based on the reading proficiency of the students: beginning readers, regardless of age, benefited from reading aloud rather than silently [17, 31]. Finally, Santoro et al. [34] stress the importance of careful planning when reading aloud is aimed at improving the comprehension of students. RA activities, in this case, should be combined with *"explicit comprehension instruction"* and *"active, engaging in discussions about the text"*.

2.2 The Role of Voice in Programming and CS Education

One main use of code vocalization is as an assistive technology that helps programmers who suffer from specific disabilities or stress injuries (RSI) to program in an efficient matter [4, 13, 16, 36]. Another area where code vocalization is essentially practiced is the remote peer-programming [10]. Vocalizing code can also be an element in some teaching strategies especially the direct instruction, modeling and think-aloud [2, 38]. However, in all of these cases, the way in which people vocalize code is not systematic, standardized, or agreed upon. In addition, there is some ambiguity over what to vocalize and on what granularity level: tokens, blocks or compilation units [16]. These factors lead to challenges for professional programmers and learners alike [6]. For example, [4, 36] mention the problematic issue of how to vocalize symbols, and when to speak out or leave specific symbols. Price et al. [30] mention the effect of natural language's flexibility on the difficulty of vocalizing programming commands, as multiple words could be used to do the same thing (for example begin class or create a class). Another effect of natural language is the ambiguity of the meaning of some words in different contexts, for example, *add* value to a variable and *add* a method to a class. These challenges show that the use of natural language in programming needs more attention from programming designers and educators. Recent work of Hermans et al. [20] calls for the programming languages to have phonology guidelines that specify how a construct should be vocalized. Finally, related is the work of Parnin [28] who investigated the role of subvocalization on code comprehension. Subvocalization is the process of the brain sending electrical signals to the tongue, lips, or vocal cords when reading silently. Silent reading is a relatively new technique for humanity. Therefore, when reading, especially the complicated segments or even words, the brain instructs the lips and the tongue to perform the read-aloud but without a voice. Their experiment on code reading showed that measuring the subvocalization signals can be an indication of the difficulty of a programming task.

2.3 Bloom's Taxonomy in CS Education

When it comes to the assessment of learning processes, Bloom's taxonomy is one of the common frameworks educators follow [23, 39, 41]. In this framework [1, 7], Bloom identifies six levels of cognitive skills that educators should aim at fulfilling with their students. The levels are Remembering, Understanding, Applying, Analyzing, Evaluating, and Creating. These cognitive levels are ordered from low to high, simple to complex and concrete to abstract, and each is a prerequisite to the next. This classification is combined with a practical guideline that educators could use to evaluate the learning outcome of their students by forming questions with certain verbs. In this way, it stimulates the cognitive process of the required level of the taxonomy. In CS education there appear two main usages of the taxonomy. First, the use of Bloom's taxonomy

as a tool to measure the learning progress of students and how they perform in introductory courses in particular. Some research chose to build the assessment from scratch depending on the taxonomy. This includes the work of Whalley et al. [41] who assessed the reading and comprehension skills of students in introductory programming courses, creating a set of questions that conform to Bloom's taxonomy. The results show that the students performed consistently with the cognitive difficulty levels indicated by the taxonomy. Similar is the work of Thompson et al. [39] who created another set of programming questions per the main categories of the taxonomy, discussing each item and showing educators how to interpret the results. Both works have been insightful to our research. Second, other researchers have applied the taxonomy to evaluate existing programming exams or courses. Lister [23] argues that the taxonomy should be used as a framework of assessment, not learning since it provides a reliable tool with a standard level of assessment. Despite its popularity among researchers, there seems a consensus that applying the taxonomy to programming questions is challenging since a programming problem consists of several building concepts which makes isolating the problem to one cognitive category a hard job to do [15, 39, 41]. Although the challenging task to map the assessment items to Bloom's cognitive levels will always depend on the interpretation of the educators, the taxonomy should still provide a valuable tool to explore the cognitive processes involved in any programming exercise.

3 METHODOLOGY

The goal of this study is to answer an overarching research question: how does reading code aloud during lessons affect the students' learning of programming concepts? To this end, we designed and ran a controlled experiment with primary school students. In this section, we describe the setup and design of the experiment in addition to the theoretical basis we use for the assessment.

3.1 Setup

We provided Python lessons to 49 primary school students in the Netherlands. We split the participants into a control and an experimental group. Both groups received the same lessons: three lessons of 1.5 hours each given by the authors of this paper, one lesson per week. We gave the lessons to the groups subsequently: first the experimental group, followed by a break, followed by the control group. The students knew they were going to learn programming during the lessons but they were not aware of the experiment's goal. We asked the consent of the parents to collect the anonymous data needed for the research.

3.2 Participants

Participants are 49 students of one primary school in Rotterdam, the Netherlands. The programming lessons are provided as part of extracurricular activities arranged by the school, taking place during school days in a computer lab at the school. As shown in Figure 1, a total of 49 school children between 9 and 13 years with an average age of 11.12 years participated in the study. Participants were 28 boys, 20 girls, and 1 participant who chose not to specify their gender. The control group consisted of 24 children (age average=11.167 years, 6 girls - 17 boys - 1 unspecified), while the experimental group consisted of 25 children (age average=11.08 years, 14 girls, 11 boys). We could not control the split of groups since they are school classes hence the non-balance in gender.

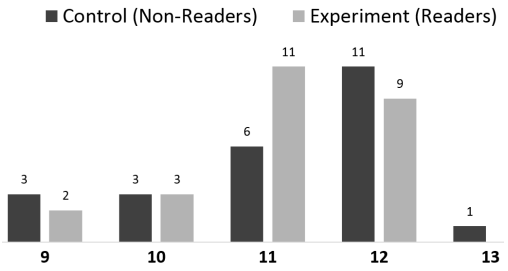

Figure 1: Age in years versus count of participants per group, mean=11.12 years. Both groups have equal age means

3.3 Lesson Design and Materials

Each lesson starts by introducing a small working program. One teacher shows a program on the interactive white-board explaining the code per line and highlighting the concepts included. The lessons include the following concepts primarily:

Variables Setting and retrieving a variable's value
Lists Creating lists of integers and strings, accessing and modifying lists through built-in functions
For-Loops Using loops for repeating certain operations
Function use Calling built-in functions and using functions from packages.

During the program explanation, the teachers encourage the students to express their thoughts on what the code does via interactive questions, such as *What do you think happens if we change this value?* According to [34], reading text aloud aiming at improving the comprehension should be combined with *"active and engaging discussions about the text"*. Following, the students are instructed to work in pairs to carry out specific exercises according to the lessons' material. During the lessons, an online compiler for Python (Repl.it[1]) was used. The final assessment questions are on-line [2].

3.4 RA Design and Implementation

Understandably, there exists no guideline on how to read code. When reading code, however, people tend to find that there are ambiguous words, symbols, and even punctuation, and vocalizing them is both challenging and subjective [20]. Consider an example as simple as the variable assignment $a = 10$, is it vocalized as *"a is ten"*, *"a equals ten"*, *"a gets 10"* or *"set a to ten"*. In this experiment, we follow a similar approach to [5, 20] where the code is read as if the person is telling another beginner student what to type into a computer. For both the experimental and the control group, the instructor read the code aloud to the students during and following the explanation of a concept within the code, a for-loop for instance. Only the students in the experimental group, however, were asked to repeat the reading activity: all-together and aloud. We consistently read all keywords, symbols identifier names and punctuation marks that are essential to the working of a program, for example quotation marks, brackets, colons and white spaces necessary for

[1] https://repl.it/repls
[2] http://bit.ly/2EhAmB0

indentation. The full list of what and how we vocalized code during the lessons are presented in Table 1. We call this list the vocalization *guideline*, and we use it later to answer RQ3 which investigates the extent to which the students follow the taught guideline later during the assessment. We do this investigation for both groups since all the students in both groups listened to the teacher vocalizing the code during the lesson, but only the experimental group's participants performed it themselves.

3.5 Assessment

We choose to assess only the two basic levels of Bloom's taxonomy: Remembering and Understanding. According to Lister [23] the two categories are sufficient for beginners when we want to assess the effectiveness of their code reading. When relating these two categories to programming assessment, Thompson et al. [39] provide useful insights into how to interpret them into programming assessment terms. Remembering can be related to activities centered around identifying a programming construct or recalling the implementation of a concept in a piece of code. For example by *"recall the syntax rules for that construct and use those rules to recognize that construct in the provided code"*[39]. For the Understanding category, it includes translating an algorithm from one form to another, plus explaining or presenting an example of an algorithm or a design pattern. For example, tracing a piece of code into its expected output. Multiple choice questions are suitable to assess these two basic levels for beginners [23, 41]. We developed an 11-questions final assessment exam: 9 are multiple choice questions, one is of fill-in type in addition to vocalizing the code snippet, and one only requires the student to vocalize a code snippet. We aimed that the questions cover i) all of the programming concepts we taught (see section 3.3), and ii) for each concept to have a question assessing the two targeted levels of Bloom's taxonomy. Table 2 shows the questions and their mappings to Bloom's levels.

3.5.1 Following the Vocalization Guideline. We ask the students in both groups to answer two vocalization questions (Question 9 and 11). The students need to write down in words how they would vocalize a code snippet to another beginner student. Although the students in the control group did not read the code aloud themselves, they listened to the instructor performing the RA. Therefore, we ask both groups to answer these questions. We use the students' answers to address RQ3.

4 RESULTS

In this section, we provide the answers to our research questions.

4.1 RQ1: What is the effect of RA on the Remembering-level?

To answer this question, we investigate the answers to the questions in the Remembering-level (7 questions) (see Table 2). The control group has a mean of 3.58 while the experimental group has a mean of 4.56. To test the equality of means we use the Mann-Whitney U Test since the sample size is relatively small and the presence of some outliers. The results (Table 3) show that the difference between the control and experimental groups is significant (p =0.003). The effect size r= 0.42 which indicates a large effect [11, 18].

Age factor: There is no relationship between the student's age and the Remembering-level score across the two groups. All age groups have equal means in the two experimental groups.

4.2 RQ2: What is the effect of RA on the Understanding-level?

To answer this question, we investigate the answers to the questions in the Understanding-level (3 questions) (see Table 2). The control group has a mean of 0.92 while the experimental group has a mean of 0.90. Similar to RQ1, we use the Mann-Whitney U Test to check the equality of the means. The test results (Table 3) show that the difference between the control and experimental groups is not significant (p = 0.93).

4.3 RQ3: How do students follow the vocalization guideline when they read code later?

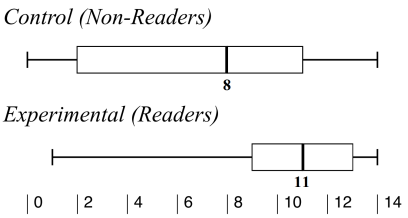

Figure 2: The vocalization score means by group

To answer this question we analyze students' answers to the vocalizing questions (Question 9 and 11 in Table 2), where we asked students to write down, in the answer paper, how would they vocalize two small code snippets.

The vocalization guideline is the way we chose to vocalize the code snippets provided during the lessons. It is summarized in Table 1. We grade the student's answers following the guideline; a point is given every time the guideline is followed, and the maximum possible is 14 points.

4.3.1 Following the Guideline: As expected there exists a significant difference between the two groups in following the vocalization guideline (see Figure 2). This is expected because of the intervention we did in the experimental group. The experimental group who read the code aloud themselves scored an average of 10.20, while the students in the control group, who only listened to the code being read, scored an average of 6.79. The Mann-Whitney test suggests the difference between the two means is significant (U=168.5, p= 0.009, r= 0.375 (a medium to large effect)).

4.3.2 Most and Least Followed Vocalizations. We analyzed the followed vocalization guidelines observed in both groups (Table 4. We notice that some vocalizations are frequent in both control and experimental groups especially the variable assignment (is), comma and single quotation mark. However, the colon in for-loop goes from one of the most frequent, in the experimental group, to the one of the least frequent in the control group. This difference can be linked to the intervention exercise making a lasting memory for the participants in the experimental group.

Table 1: The vocalization guideline used during the lessons

Vocalization Item	Description	Code	How Code was Vocalized
V1	Setting a variable value	temperature = 8	temperature **is** eight
V2	Function-calling with round brackets	for i in range(10): temperature = temperature + 1	for i in range **open round bracket** ten **close round bracket**
V3	For-loop colon		**colon**
V4	For-loop indentation space		**space space**
V5	Plus sign in expressions		temperature is temperature **plus** one
V6	Symbols in identifiers (underscore)	healthy_food = ['apple', 'banana']	healthy **underscore** food is
V7	List square bracket (open)		**open square bracket**
V8	Strings single quotation (begin)		**single quotation** apple **single quotation**
V9	Comma separation between list items		**comma**
V8 (Repeated)	Strings single quotation (end)		**single quotation** banana **single quotation**
V7(Repeated)	List square bracket (close)		**close square bracket**
V10	Function use: calling from a package with dot	food = random.choice(healthy_food)	food is random **dot** choice **open round bracket** healthy underscore food **close round bracket**

Table 2: The list of questions and their corresponding Bloom's cognitive level

#	Concept(s)	Bloom's level	Prerequisite Knowledge	Student's Action(s) to Answer
1	List Create/Modify	Remembering	The syntax to create a list of string literals	Replace syntactically incorrect line by a correct option
2	Variables	Remembering	The syntax to increase an integer variable's value	Replace an empty line with a syntactically correct option
3	Function use	Remembering	The syntax to call a function with a variable parameter	Replace syntactically incorrect line by a correct option
4	Function use	Remembering	The syntax to call the print function with a string literal	
5	Function use & Variables	Remembering	The correct syntax to print a variable's value	
6	Sequential execution & Variables	Remembering	Same indentation for each line of a Python block	Identify/recognize/locate the cause of the error from choices
7	For-loop	Understanding	For-loop syntax Indentation effect on lines being within/outside a loop	Trace and predict the outcome of a for-loop with a print statement within, followed by another print statement
8	For-loop	Remembering	For-loop syntax	Identify/recognize the syntactically correct for-loop to get a specific outcome
9	For-loop & Variables (with Vocalize)	Understanding	For-loop syntax Indentation effect on lines being within/outside a loop	Trace and predict the outcome of a loop that increases the value of a variable. Then write in words how you would vocalize the code.
10	List Create/Modify & Function use	Understanding	The syntax of List creation, List access & modification using built-in functions	Trace code and interpret its use in one of low-,medium- or high natural language descriptions
11	Vocalize only	-	Vocalize code as if your are reading it to a friend	Write, in words, on the answer sheet how you would vocalize the code snippet

Table 3: The difference by group in the answer score means to each category of questions.

	Remembering-level Score 7 Questions		Understanding-level Score 3 Questions	
	Readers	Non-Readers	Readers	Non-Readers
	n=25	n=24	n=25	n=24
Mean	4.56	3.58	0.90	0.92
Std. Dev.	1.00	1.28	0.76	0.75
Mann Whitney *U*	156.5	443.5	304.5	295.5
z-score	2.97		0.09	
(2-tailed) *p*	0.003		0.93	
Significant	Yes (r = 0.42)		No	

4.3.3 Effect of Following the Guideline: Within one group, we analyzed whether following the guideline affects the answers to either Remembering- or Understanding-level questions. Results so far showed that students in the experimental group are more likely to score higher in following the vocalization guideline, and at the same time are more likely to score higher in the Remembering-level score, than the students in the control group. However, comparing the students' score within the experimental group itself does not show a relationship between following the vocalization guideline and the score in neither the Remembering- nor Understanding-level category. The control group, however, reveals a different behavior. Results show that students within the control group who followed the vocalization guidelines scored higher on the Understanding-level questions (see Figure 3). According to the ANOVA test, the vocalization varies across the quantiles of the Understanding-level score (F=8.232, p=0.002). We highlight again that students in this group were not instructed to repeat the reading of the code, they only listened to the instructor reading aloud the code snippets.

5 DISCUSSION

5.1 Reflection and explanation of the results

The main finding in this study is the significant effect RA has on remembering the syntax of the programming constructs taught to the students. We believe this result encourages teachers in primary schools to practice code vocalization as pedagogy in their programming classes. While learning how to program is unique and known to be difficult, it is still a learning process. We can, therefore, use pedagogies from other domains to help make programming easier to learn for younger students in particular. With that in mind, we can explain the effect of RA we observe in this study from two

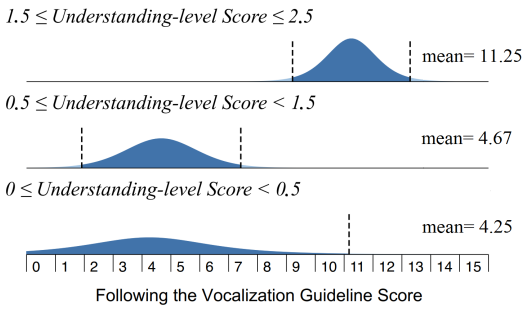

Figure 3: The variance of following the vocalization guideline among the participants in the control group and and the relation with the score on Understanding-level questions

Table 4: The most and least followed vocalizations following the guideline in Table 1

Most Followed		Least Followed	
The Experimental Group (n=25)			
V3 - Colon in for-loop (:)	22	V7 - List square brackets	11
V1 - Variable assignment (is)	21	V4 - Space indentation in for-loops	16
V9 - Comma (,) V8 - String single quotation (') V10 - Dot in function call	19	V5 - Plus sign	17
The Control Group (n=24)			
V1 - Variable assignment	14.25	V4 - Space indentation in for-loops	5
V6 - Underscore in variable names	14	V7 - List square brackets	6
V9 - Comma (,) V8 - String single quotation (')	13	V3 - Colon in for-loop (:) V2 - Round bracket for function call V10 - Dot in function call V5 - Plus sign	11

angles. First, RA improves the learning environment by utilizing a familiar technique to young students. This subsequently raises focus and attention of the students. When attention is gained and sustained learning can happen as *"attention is a prerequisite for learning"* [21, p. 3]. Secondly, RA helps in automating the retrieval of basic knowledge required for cognitive development [3]. According to the neo-Piagetian theories of cognitive development [26], students in their initial phase of learning programming are at the sensorimotor stage. At that stage, students mostly struggle in interpreting the semantics of the code they read, which affects their performance in tracing tasks in particular [38, 40]. Practicing RA could potentially help in reducing the struggle because it automates the remembering of the language constructs, and helps the student moving faster to the next development phases.

5.2 RA and the granularity of the vocalization

There is currently no standard guideline specifying how constructs of Python, or other programming languages, should be vocalized. As presented in Section 2.2, there are various granularities and strategies one can read code with. In this study, however, we follow a specific technique to vocalization (Section 3.4) which can be considered of a low granularity, focusing on syntax rather than semantics or relations within the code. Nevertheless, when teaching young and novice students, the RA method we follow could help teachers create a benchmark where students know how to call all the elements in their programming environment. We see from the answers of the control group students that there exist some variances in calling specific symbols. For example, calling the single (') a *"single quotation"*, *"apostrophe"*, or *"upper comma"*. This variance shows the challenges that beginners face to identify symbols in the first place. An extra cognitive effort is spent on remembering rather than on conceptual understanding. We hypothesize that higher granularities of vocalization of code structures or semantics can be integrated into the following phases. To determine the best vocalization method teachers start with, however, is out of this study's scope and is an opportunity for future research.

5.3 Threats to validity

Our study involves some threats to its validity. First, the split of the participants into the two groups might have influenced the results. The split was introduced by the school structure; i.e., per class. However, we randomly selected one class as the experimental group and the other as the control group. The second threat, the authors being the teachers at the same time could introduce a bias in favor of the experimental group. To reduce the effect of such bias we ensured that both groups studied the same materials over the same amount of time with the same teacher. The main teacher was accompanied by another teacher who among other things observed the teaching given to the two groups. By these steps, we ensured that the only difference between the two groups would be the reading-aloud method. Third threat, a wrongful assignment of a question to one of Bloom's cognitive levels by the authors. This is a common challenge for researchers in similar studies [39, 41], and future experiments will lead to refining this process. Finally, a threat to the external validity of our study is the difficulty to generalize its results. This is, however, an inherent issue in similar studies with small sample size [32]. To overcome this threat we should replicate the study across different participants in the future.

6 CONCLUSIONS AND FUTURE WORK

Our paper aims at measuring the effect of reading code aloud during programming lessons on comprehension. We perform a controlled experiment with 49 school students aged between 9 and 13. We assess the students' comprehension of basic programming concepts after three Python programming lessons. The assessment is based on Bloom's taxonomy and focused on the of remembering and understanding levels. The results show that the participants in the experimental group score significantly higher in remembering-level questions. However, the two groups perform similarly in understanding-level questions. Furthermore, we observe that the participants in both groups vocalize specific constructs more often than others. For example, the variable's assignment (is) and punctuation symbols (comma, underscore and quotation mark). Our paper suggests that using RA for teaching programming in schools will contribute to improving comprehension among young students. In particular, it will improve remembering the syntax, paving the way to spending more cognitive effort on the higher level understanding of the concepts. For future work, we aim at experimenting with different RA approaches with different code granularities to find the best approach to improve code comprehension at this age.

REFERENCES

[1] Lorin W Anderson and David Krathwohl. 2001. *A Taxonomy for Learning, Teaching, and Assessing*. New York.

[2] Richard Arends. 2012. *Learning to teach*. McGraw-Hill.

[3] Craig Barton. 2018. *How I wish I'd taught maths*. John Catt Educational Ltd.

[4] Andrew Begel. Programming By Voice: A Domain-specific Application of Speech Recognition. (????).

[5] A. Begel and S.L. Graham. Spoken Programs. *2005 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC'05)* (????). DOI: http://dx.doi.org/10.1109/vlhcc.2005.58

[6] A. Begel and S.L. Graham. 2006. An Assessment of a Speech-Based Programming Environment. *Visual Languages and Human-Centric Computing (VL/HCC'06)* (2006). DOI: http://dx.doi.org/10.1109/vlhcc.2006.9

[7] B.S. Bloom. 1956. *Taxonomy of Educational Objectives: The Classification of Educational Goals*. Number v. 1 in Taxonomy of Educational Objectives: The Classification of Educational Goals. D. McKay. https://books.google.nl/books?id=hos6AAAAIAAJ

[8] B. Du Boulay. 1986. Some Difficulties of Learning to Program. *Journal of Educational Computing Research* 2, 1 (1986), 57–73. DOI: http://dx.doi.org/10.2190/3LFX-9RRF-67T8-UVK9 arXiv:https://doi.org/10.2190/3LFX-9RRF-67T8-UVK9

[9] Adriana G. Bus, Marinus H. van IJzendoorn, and Anthony D. Pellegrini. 1995. Joint Book Reading Makes for Success in Learning to Read: A Meta-Analysis on Intergenerational Transmission of Literacy. *Review of Educational Research* 65, 1 (1995), 1–21. DOI: http://dx.doi.org/10.3102/00346543065001001

[10] G. Canfora, A. Cimitile, and C.A. Visaggio. Lessons learned about distributed pair programming: what are the knowledge needs to address? *WET ICE 2003. Proceedings. Twelfth IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.* (????). DOI: http://dx.doi.org/10.1109/enabl.2003.1231429

[11] Barry H Cohen. 2008. *Explaining Psychological Statistics* (3 ed.). John Wiley & Sons.

[12] Ryan Deschambault. 2011. Thinking-Aloud as Talking-in-Interaction: Reinterpreting How L2 Lexical Inferencing Gets Done. *Language Learning* 62, 1 (2011), 266–301. DOI: http://dx.doi.org/10.1111/j.1467-9922.2011.00653.x

[13] A. Desilets. 2001. VoiceGrip: A Tool for Programming-by-Voice. *International Journal of Speech Technology* 4, 2 (01 Jun 2001), 103–116. DOI: http://dx.doi.org/10.1023/A:1011323308477

[14] Richard Fateman. 1998. How can we speak math. *Journal of Symbolic Computation* (1998).

[15] Sue Fitzgerald, Beth Simon, and Lynda Thomas. 2005. Strategies that students use to trace code. *Proceedings of the 2005 international workshop on Computing education research - ICER '05* (2005). DOI: http://dx.doi.org/10.1145/1089786.1089793

[16] Joan M Francioni and Ann C Smith. 2002. Computer science accessibility for students with visual disabilities. In *ACM SIGCSE Bulletin*, Vol. 34. ACM, 91–95.

[17] Andrea D. Hale, Renee O. Hawkins, Wesley Sheeley, Jennifer R. Reynolds, Shonna Jenkins, Ara J. Schmitt, and Daniel A. Martin. 2010. An investigation of silent versus aloud reading comprehension of elementary students using Maze assessment procedures. *Psychology in the Schools* 48, 1 (2010), 4–13. DOI: http://dx.doi.org/10.1002/pits.20543

[18] John Hattie. 2009. *Visible learning*. Routledge.

[19] Felienne Hermans and Efthimia Aivaloglou. 2017. Teaching Software Engineering Principles to K-12 Students: A MOOC on Scratch. In *Proceedings of the 39th International Conference on Software Engineering: Software Engineering and Education Track (ICSE-SEET '17)*. IEEE Press, Piscataway, NJ, USA, 13–22. DOI: http://dx.doi.org/10.1109/ICSE-SEET.2017.13

[20] Felienne Hermans, Alaaeddin Swidan, and Efthimia Aivaloglou. 2018. Code phonology. *Proceedings of the 26th Conference on Program Comprehension - ICPC '18* (2018). DOI: http://dx.doi.org/10.1145/3196321.3196355

[21] John M. Keller. 1987. Development and use of the ARCS model of instructional design. *Journal of Instructional Development* 10, 3 (1987), 2–10. DOI: http://dx.doi.org/10.1007/bf02905780

[22] Sherry Kragler, Linda Martin, and Virginia Schreier. 2015. Investigating Young Children's Use of Reading Strategies: A Longitudinal Study. *Reading Psychology* 36, 5 (2015), 445–472. DOI: http://dx.doi.org/10.1080/02702711.2014.884031

[23] Raymond Lister. 2005. Computer Science Teachers as Amateurs, Students and Researchers. In *In Proceedings of the 5 th Baltic Sea Conference on Computing Education Research. (Koli.* 3–12.

[24] Raymond Lister, Elizabeth S. Adams, Sue Fitzgerald, William Fone, John Hamer, Morten Lindholm, Robert McCartney, Jan Erik Moström, Kate Sanders, Otto Seppälä, Beth Simon, and Lynda Thomas. 2004. A Multi-national Study of Reading and Tracing Skills in Novice Programmers. *SIGCSE Bull.* 36, 4 (June 2004), 119–150. DOI: http://dx.doi.org/10.1145/1041624.1041673

[25] Linda E. Martin and Sherry Kragler. 2011. Becoming a Self-Regulated Reader: A Study of Primary-Grade Students' Reading Strategies. *Literacy Research and Instruction* 50, 2 (2011), 89–104. DOI: http://dx.doi.org/10.1080/19388071003594697

[26] Michael F. Mascolo. 2015. Neo-Piagetian Theories of Cognitive Development. *International Encyclopedia of the Social & Behavioral Sciences* (2015), 501–510. DOI: http://dx.doi.org/10.1016/b978-0-08-097086-8.23097-3

[27] R. Steve McCallum, Shannon Sharp, Sherry Mee Bell, and Thomas George. 2004. Silent versus oral reading comprehension and efficiency. *Psychology in the Schools* 41, 2 (2004), 241–246. DOI: http://dx.doi.org/10.1002/pits.10152

[28] Chris Parnin. 2011. Subvocalization - Toward Hearing the Inner Thoughts of Developers. *2011 IEEE 19th International Conference on Program Comprehension* (2011). DOI: http://dx.doi.org/10.1109/icpc.2011.49

[29] M. Perrone-Bertolotti, L. Rapin, J.-P. Lachaux, M. Baciu, and H. LÂŻvenbruck. 2014. What is that little voice inside my head? Inner speech phenomenology, its role in cognitive performance, and its relation to self-monitoring. *Behavioural Brain Research* 261 (2014), 220–239. DOI: http://dx.doi.org/10.1016/j.bbr.2013.12.034

[30] David E Price, DA Dahlstrom, Ben Newton, and Joseph L Zachary. 2002. Off to See the Wizard: using a" Wizard of Oz" study to learn how to design a spoken language interface for programming. In *Frontiers in Education, 2002. FIE 2002. 32nd Annual*, Vol. 1. IEEE, T2G–T2G.

[31] Suzanne M Prior and Katherine A Welling. 2001. " Read in Your Head": A Vygotskian Analysis of the Transition from Oral to Silent Reading. *Reading Psychology* 22, 1 (2001), 1–15.

[32] Katherine E. Purswell and Dee C. Ray. 2014. Research With Small Samples. *Counseling Outcome Research and Evaluation* 5, 2 (2014), 116–126. DOI: http://dx.doi.org/10.1177/2150137814552474

[33] Keith Rayner, Barbara R. Foorman, Charles A. Perfetti, David Pesetsky, and Mark S. Seidenberg. 2002. How Should Reading be Taught? *Scientific American* 286, 3 (2002), 84–91. DOI: http://dx.doi.org/10.1038/scientificamerican0302-84

[34] Lana Edwards Santoro, David J. Chard, Lisa Howard, and Scott K. Baker. 2008. Making the Very Most of Classroom Read-Alouds to Promote Comprehension and Vocabulary. *The Reading Teacher* 61, 5 (2008), 396–408. DOI: http://dx.doi.org/10.1598/rt.61.5.4

[35] Marcel Schmeier and Ruud Bijman. 2017. *Effectief rekenonderwijs op de basisschool* (1st edition ed.). Uitgeverij Pica.

[36] Lindsey Snell and Mr Jim Cunningham. 2000. An investigation into programming by voice and development of a toolkit for writing voice-controlled applications. (2000).

[37] Osamu Takeuchi, Maiko Ikeda, and Atsushi Mizumoto. 2012. Reading Aloud Activity in L2 and Cerebral Activation. *RELC Journal* 43, 2 (2012), 151–167. DOI: http://dx.doi.org/10.1177/0033688212450496

[38] Donna Teague and Raymond Lister. 2014. Longitudinal Think Aloud Study of a Novice Programmer. In *Proceedings of the Sixteenth Australasian Computing Education Conference - Volume 148 (ACE '14)*. Australian Computer Society, Inc, Darlinghurst, Australia, Australia, 41–50. http://dl.acm.org/citation.cfm?id=2667490.2667495

[39] Errol Thompson, Andrew Luxton-Reilly, Jacqueline L. Whalley, Minjie Hu, and Phil Robbins. 2008. Bloom's Taxonomy for CS Assessment. In *Proceedings of the Tenth Conference on Australasian Computing Education - Volume 78 (ACE '08)*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 155–161. http://dl.acm.org/citation.cfm?id=1379249.1379265

[40] Jacqueline Whalley and Nadia Kasto. 2014. A qualitative think-aloud study of novice programmers' code writing strategies. *Proceedings of the 2014 conference on Innovation & technology in computer science education - ITiCSE '14* (2014). DOI: http://dx.doi.org/10.1145/2591708.2591762

[41] Jacqueline L. Whalley, Raymond Lister, Errol Thompson, Tony Clear, Phil Robbins, P. K. Ajith Kumar, and Christine Prasad. 2006. An Australasian Study of Reading and Comprehension Skills in Novice Programmers, Using the Bloom and SOLO Taxonomies. In *Proceedings of the 8th Australasian Conference on Computing Education - Volume 52 (ACE '06)*. Australian Computer Society, Inc., Darlinghurst, Australia, Australia, 243–252. http://dl.acm.org/citation.cfm?id=1151869.1151901