Qualitative Evaluation of Visualizations for List-based Iteration

Ms. Molly Rebecca Domino, Virginia Polytechnic Institute and State University
Ms. Margaret O'Neil Ellis

Associate Professor of Practice, Computer Science Department, Virginia Tech

My research interests include examining ways to improve engineering educational environments to facilitate student success, especially among underrepresented groups.

Dr. Dennis Kafura

Qualitative Evaluation of Visualizations for List-Based Iteration

Molly Domino
mollydomino@vt.edu
Computer Science
Virginia Tech

Dennis Kafura
kafura@cs.vt.edu
Computer Science
Virginia Tech

Margaret Ellis
maellis1@cs.vt.edu
Computer Science
Virginia Tech

Jieun Chon
g471000@gmail.com
Computer Science
Virginia Tech

Luke Guzukuma
lukesg08@vt.edu
Computer Science
Virginia Tech

Abstract

Interactive visualizations were developed to improve the learning of list-based iteration by students in our university's Computational Thinking course for non-majors. An initial quantitative evaluation of the visualizations raised questions about their long-term effectiveness and ease of use. This study represents a qualitative study done to gain deeper insight into the experiences of students. The results of this study demonstrated students were engaging with course materials in unexpected ways but frequently referred back to the visualizations. Additionally, students had an approach to understanding the visualizations that was both helpful and problematic. These findings help to inform visualization and curriculum designers about student attitudes and strategies in using course materials.

1 Introduction

Due to the abstract nature of Computer Science it is not uncommon for novice students to struggle in understanding some of the foundational structures and semantics in programming. This struggle is even more evident for the growing population of university students in other majors seeking to learn about computing and programming. It is this population of students which is served by our course,[1] providing students with a perspective for how programming could be useful in a broad variety of disciplines.

An important topic with which novice students struggle is iteration. From an early study[2] to more recent ones[3,4] iteration has persisted as a consistent problem for students. Iteration, specifically list-based iteration, was a challenge for the particular population of students in our non-majors course. In addition to anecdotal reports from instructors, an evaluation at the end of the iteration unit showed that students overall scored only 77% on 12 multiple choice questions and only 55%

on the subset of 3 questions that dealt with the dynamic execution of iteration. Also, only about one third (32.5%) were able to complete an open-ended programming problem focused on lists and iteration.[5].

Previous work created interactive visualizations to support student learning in a block-based environment of list-based iteration. A quantitative study[5] of the impact of these visualizations had two anomalies that we wanted to explore more deeply. First, the study found a statistically significant improvement with a medium effect size at the time when the visualizations were used. However, this near-term improvement was not sustained[5]. Second, while the large majority of students on the survey rated the visualizations as either "Very Helpful" or "Helpful" they were lukewarm on their ratings for its ease of use.

The goal of this study was to gain deeper insight into students' experience so that more informed decisions could be made to improve the interactive visualizations. We conducted a qualitative study to answer three key research questions:

- **RQ1:** How were students engaging with these visualizations?
- **RQ2:** Did students perceive these visualizations to be clear and helpful?
- **RQ3:** Were these visualizations helping to foster a sound understanding of list-based iteration?

A qualitative study, with a focus on non-numerical data and deeply specific conclusions, offered the opportunity to more completely understand students' perspectives. Students across two semesters were interviewed shortly after using these visualizations both as classwork and homework. The interview transcripts were qualitatively coded for meaning.

The contributions of this paper come from answering the three research questions as follows:

- Students initially engaged with these visualizations in a variety of unexpected ways, though they did consistently refer back to these visualizations later as a way to help them when unsure or stuck.
- Students often expressed that the visualizations were clear and helpful.
- When shown a new problem in this visualization format, students demonstrated a flawed understanding of list-based iteration.

These observations are relevant to instructors developing visualizations for courses with non-computer science majors or courses that have a significant number of non-majors.

This paper will begin with a brief review of related work to set the context for our study outlined in section two. Section three will discuss the methodology of the study and relevant course context to understand some terminology used. The results of that process are presented in section four. An analysis of the pedagogical implications is examined in section five. This paper will end with an examination of some threats to validity in section six and future work in section seven.

2 Background and Related Work

This work falls at the crossroads of research in Computer Science Education and Program Visualization. Visualizations were initially chosen as an intervention in line with Mayer's *multimedia principle*. In his analysis of evidence-based learning techniques, he defines this principle as the idea that learning is enhanced from using visuals and text together, rather than just one or the other[6]. Given the abstract nature Computer Science instruction, it is perhaps not surprising how frequently adding a visual model can help[7]. Indeed, using visualizations in the context of Computer Science education is far from a novel idea.

Educational visualizations for Computer Science courses typically fall into one of three categories: Algorithm Visualizations, Program Visualizations, and other tools (for visualizations that don't fit neatly into either of the other categories).

The visualizations developed for this course fall into the category of Program Visualizations. This type of visualization focuses on the specific rather than the generic as they allow a student to trace a program, rather than an abstract algorithm.[8]. PythonTutor[9] is a popular example of this form of visualization practice. Program visualizations have been frequently used in an educational context to facilitate the learning of foundational Computer Science material among novice programmers[10,11].

Program Visualizations have been used in a number of ways to facilitate teaching looping structures to novice programmers. Game-like visualizations have been used in multiple studies[12,11] as have other forms of visualization[13,14]. However all of these studies were primarily quantitative in nature and none appear to use a qualitative approach. Indeed, while quantitative work abounds when studying either Program or Algorithm Visualizations, less has been done to examine student experiences through a qualitative lens.

Within the context of Computer Science Education, qualitative methods have also been used for a variety of applications. Examinations of the experiences of students[15], marginalized populations of students[16] and especially novice students[17,18,19,20] have offered tremendous insight into how learners comprehend their Computer Science lessons. This study seeks to examine novice and specifically non-major experiences with a Program Visualization through qualitative measures.

3 Course and Visualizations

This university's Computational Thinking course is designed as an opportunity for students with no previous computing background to gain an understanding of CS fundamentals. Students use the block-based programming environment, BlockPy, to practice basics concepts like conditionals, abstraction, and iteration.

A study in Spring 2018 showed that, compared with other topics taught, students were struggling with iteration. Jien Chon created interactive visualizations in an effort to help students' understanding of iteration. These visualizations were designed with three goals: be visually compatible with BlockPy, clearly represent list-based iteration, and provide some scaffolding for students to more effectively learn[5]. These design goals largely informed the look and use of the visualizations discussed in this paper. Chon's work demonstrated that the interactive visualizations resulted in an increased understanding of list-based iteration, but a significant number of students surveyed at the end of the year reported not remembering the visualizations.

Broadly, these interactive visualizations can be categorized by how they were used in the course. *Textbook Visualizations* were integrated into the e-textbook, while *Exercise Visualizations* were assigned online graded exercises.

3.1 Textbook Visualizations

Figure 1 shows an example of a visualization embedded in the textbook. These visualizations were integrated in the e-textbook and offered students the chance to see aspects of iteration demonstrated immediately after the relevent paragraph.

The design of the visualization reflected the appearance of the block-based language the students were using on their first encounter with iteration. The horizontal green segmented rectangle is the list which moves from right to left on each iteration so that a single list item becomes the value of the iteration variable ("price" in this example).

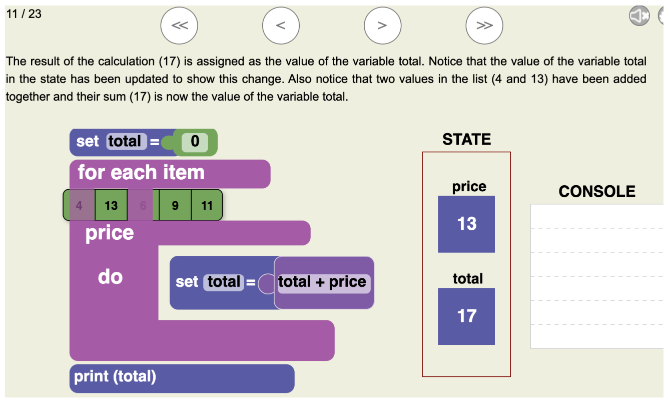


Figure 1: Example of a Textbook Visualization

The result of the calculation (17) is assigned as the value of the variable total. Notice that the value of the variable total in the state has been updated to show this change. Also notice that two values in the list (4 and 13) have been added together and their sum (17) is now the value of the variable total.

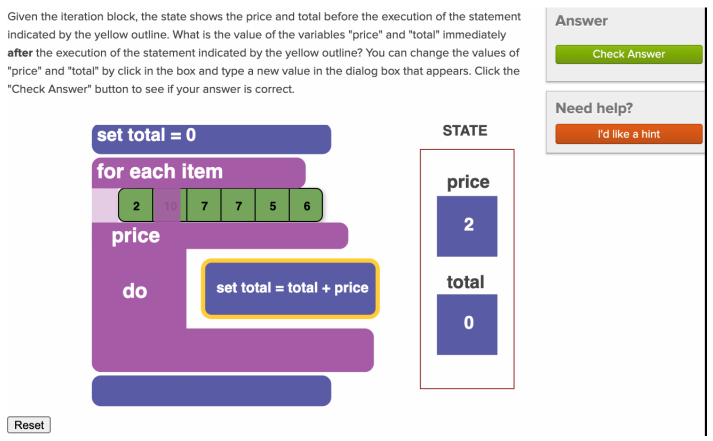
```
set total = 0
for each item 4 13 6 9 11 price
do
    set total = total + price
print (total)
```

To interact with these visualizations, students clicked on the four arrow icons seen at the top of the figure. Clicking the '¿' button triggers an animation in which sections of the image would light up, allowing students to visually trace through the iteration. When appropriate, the "STATE" and "CONSOLE" areas are updated, demonstrating the change in the value of variables and any console output the code might produce.

3.2 Exercise Visualizations

Some interactive visualizations were graded exercises to be completed as classwork shortly after the students were introduced to iteration. An example can be seen in Figure 2. To successfully complete this exercise, students needed to correctly fill in the boxes marked "price" and "total" with what those variables would be after the line `set total = total + price` executes (in Figure 2 price is 2 and total is 0).

Figure 2: Example of an Exercise Visualization


Given the iteration block, the state shows the price and total before the execution of the statement indicated by the yellow outline. What is the value of the variables "price" and "total" immediately **after** the execution of the statement indicated by the yellow outline? You can change the values of "price" and "total" by click in the box and type a new value in the dialog box that appears. Click the "Check Answer" button to see if your answer is correct.

```
set total = 0
for each item
  2 10 7 7 5 6
price
do
    set total = total + price
```

3.3 Study Description

In order to better understand how students were engaging with these visualizations, qualitative semi-structured interviews were conducted, and the transcripts of those interviews were manually coded for meaning. In discussions about when different events occurred in class, this paper will refer to a specific class by number, for example ”Day 3”. This does not mean the 3rd day of the semester, but rather the 3rd time a given section of Intro to Computational Thinking met in the semester.

In the semesters studied, students were informally observed on Day 8 of class: when they were interacting with exercise visualizations. While some cursory notes were taken, these observations were primarily done for the interviewer to get a better understanding of the student population. After a brief introduction from the interviewer about the study, students were directed to sign up online for a 45 minute time slot if interested. A scheduling conflict caused this presentation to be given on different class days in the Fall and Spring semesters. However, all interviews occurred after students had been exposed to the iteration module and before the post-quiz was administered.

In the fall semester, all time slots were filled before the second section had a chance to sign up. To mitigate this in the spring semester, a different plan was implemented. Two different sign-up sheets were created: one for each section of the class. In the case students from different sections signed up for the same time slot, the interviewer reached out and asked if their schedules could accommodate a different time. If no alternate time could be scheduled, the student would be thanked for their time but no interview would occur. Students were informed of this plan (and its reasoning) both in writing and verbally before they signed up. In the case of this semester of interviews, all 12 students who signed up were able to be accommodated.

Demographic information of the 25 recruited participants can be seen in Table 1. Students were predominantly (though not exclusively) in their first year of college and did not have any previous experience with Computer Science.

Table 1: Demographic Information of Participants

Fall Semester				Spring Semester				
Codename	**Year**	**Major**	**Prev. CS experience**	**Codename**	**Section**	**Year**	**Major**	**Prev. CS experience**
AK	1	Neuroscience	No	AG	1	1	Information Technology	No
BS	2	Criminology and Psychology	No	AG2	2	1	Int. Relations	No
CJ	1	Undecided Liberal Arts	Yes	AS	1	3	Geography	Yes
CP	1	Nat. Security and Foreign Affairs	No	BH	2	4	Management and Consulting Analysis	No
GJ	1	Nat. Security and Foreign Affairs	No	HS	1	1	Comp. and Systems Neuroscience	No
GL	1	Nat. Security and Foreign Affairs	Yes	HSh	1	1	Public Relations	No
IE	3	Chemistry	No	JS	1	1	Comp. Modelling and Data Analytics	No
JC	1	Undecided	No	KU	1	1	Int. Relations	Yes
JD	3	Int. Studies	Yes	LC	1	3	Psych. Analysis	No
MB	2	Property Management	No	LM	1	2	Geography	No
MP	1	Dairy Science	No	SD	2	3	Biochemistry	Yes
MS	2	Criminology, Psychology, and Sociology	No	TG	1	1	Communications	No
PS	1	Int. Relations	No					

3.3.1 Interviews

The interviews themselves were qualitative and semi-structured in nature. While a 45 minute block of time was set aside, interviews typically lasted between 15 and 30 minutes. All interviews took place in the same room with access to the same materials. The interviewer and participant were seated at a small round table. A laptop displayed a new visualization (Figure 3) and a microphone was next to it. While the interview was in progress, the screen of the laptop was recorded to capture any cursor movements over the visualization. The voices of both participant and interviewer were also recorded but no video recordings were taken of any people involved.

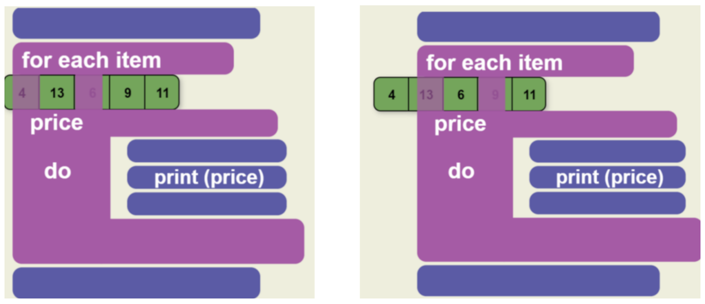
Before the interviews started, students were given an IRB consent form to read and sign. After obtaining their consent the interviewer provided some general guidelines. Since participants would be discussing their understanding of Computer Science concepts, they were told the interviewer would not correct them if they described something incompletely or incorrectly. Instead, they were encouraged to treat their understanding of material as correct in all cases. Also, in an effort to ensure a good rapport between interviewer and participant, all students were told that some questions may feel repetitive. This was not because the interviewer was not paying attention to their responses, but instead to ensure the interviewer had the most complete understanding of the experience.

The interview consisted of four basic sections. First, students were asked about their background. Their year in school, major, previous experience with computer science or and their comfort level with list-based iterations were all discussed here.

Next, the interviewer described the visualizations she was referring to and asked if the student had used them and if so how they were used within the greater context of the course. This related both to how students used the course materials and where they used them (in class versus at home for example). Students were also asked about if and why they looked back on these visualizations once they had gone on to other material. Students were also asked questions about the clarity and helpfulness of the visualizations. To ensure both participant and interviewer were understanding each other, the interviewer first asked only about the textbook visualization, then the exercise visualizations.

Then, the interviewer would present participants with a new visualization (Figure 3). Notably, while similar to the textbook and exercise visualizations, this interview visualization was concerned with printing out the elements in the list, rather than calculating the total of the list as seen in Figure 1 or Figure2.

Figure 3: Interview Visualization



The interviewer asked students to describe what was happening and discuss what the relationship (if any) existed between the left and right images. This was in an effort to understand how students used and interpreted the visualizations used in class. They were asked follow up questions concerning their understanding of the state of each image. Then they were asked what they thought the output of running such code would look like.

Finally, participants were asked for their final thoughts and asked for suggestions as to how these visualizations could be improved.

4 Analysis and Contributions

In total, 59 codes were generated from all interviews and stratified into 6 themes. Those themes were ”study strategies”, ”overall feelings on class and visualizations”, ”comments on textbook visualizations”, ”comments on exercise visualizations”, ”non-visualization resources” and ”analysis of interview visualization”. In understanding these themes, we gained insight into the student experience as discussed in this section. ”Study strategies”, ”comments on textbook visualizations” and ”comments on exercise visualizations” all helped answer RQ1. ”Overall feelings on class and visualizations” and ”non-visualization resources” helped answer RQ2. ”Analysis of interview visualization” helped answer RQ3.

4.1 RQ1: How were students engaging with these visualizations?

4.1.1 Engagement with Textbook Visualizations

When it came to engaging with the online course materials, no two students had an identical approach. Some students reported using the resources in the order laid out by the module; reading and taking notes before trying visualization exercises and then trying programming problems.

For example, one student reported: *"...I typically go through the readings for like the reading quizzes beforehand and take notes with that"* (GL). In contrast some students avoided online textbook readings if at all possible: *"I really try not to open the textbook"* (JD) one student succinctly summarized. Most students found themselves between these two extremes when it came to usage. Many would report "going through" the readings in one form or another and using them as a reference if they needed to on a problem. *"I kind of skim through them and then if I'm doing like the classwork, I will look back at them"* (IE), one student said. Outside of the students who avoided the textbook materials completely, all students reported at least skimming through the reading and integrated textbook visualizations, even if they did not read them thoroughly.

While there was a great diversity in how students initially engaged with the online textbook, referring back to the textbook and its visualizations was a very common strategy used by nearly every student. Many described keeping two windows open: one for the exercise they were working on and one for the textbook. When struggling with an exercise, students would look back to the text as a guide for what to do. When stuck, many students returned to the textbook in order to better understand the concepts they'd initially skimmed through. One student described using both of these strategies together: *"actually the other day I was in a class doing that the classworks and homeworks and I had the reading [and textbook visualizations] up on one side of my screen and I was doing the [coding] and it was helping me a lot cause I mean some of these they show like in the readings like a [solution] of like a completed algorithm correctly"* (MB).

Here, visualizations were a way to look at correctly formatted code and read through a description for what the code did. This provided students who were stuck with a frame of reference with which to understand novel exercises.

Students would use this referring back strategy after working through questions on their own. Here, students weren't stuck on how to complete the exercise. Instead, the textbook was a means of double checking that they were on the right track. One participant described this strategy when he said: *"I don't think there's been anything where I'm like, oh, I don't know what's going on. Usually if I've done my reading, gone through the animation. I'll be like what's this again? And then I double checked like with the reading and then I'm like, okay, that's what's going on."* (JS) This strategy was less frequently reported than that of referring back when stuck, but still was brought up by several participants.

4.1.2 Engagement with Exercise Visualizations

In contrast to the textbook visualizations, students engaged with the exercise Visualizations in a fairly uniform way. All students interviewed said they did the exercises, even those that avoided using the textbook. Students frequently did more than simply try the exercises as well. When asked if they completed all five exercises in the lesson, or just enough to get a passing grade, many students seemed to insist on a score of five out of five. Most students described trying and re-trying visualization exercises until all 5 problems were answered correctly. Only a few students accepted a lower score. It is important to note that while all students interacted with these exercise visualizations, this does not imply they were necessarily better than the visualizations in the textbook. There were many factors that could have contributed to the differences in engagement: for one thing the completion of these exercises impacted the student's grade directly while readings were assessed through a short quiz which did not directly ask about the interactive visualizations.

Much like the textbook visualizations, students used the exercises as a resource to refer back to. However, while students did say they referred back to these exercises if asked, this strategy was brought up less by students outside of the direct question. The exception to this trend being the students who avoided using the textbook entirely. When asked if they used either type of visualization, one student said: *"I would go back to some of my classworks ¡visualizations¿ to help me with the homework. I didn't really go back to the textbook"* (BS). This demonstrates how a student used referring back to the exercises as an alternative to using the textbook.

These observations show part of a larger phenomenon described in these interviews. Students reported using many resources within the class; referring back to these visualizations being only one strategy among many. Those interviewed described many strategies for completing problems such as asking a teaching assistant, conferring with peers, and referring back to previous class material or lectures. Each student seemed to have their own order for which resource to use first and which to try after if they were stuck. For example one student said that if stuck they: *"... try to like stick it out because most of the time you can kind of figure it out. At least I in the past have been able to and if I truly don't understand I'll wait until the next class. Sometimes if it's homework I'll wait until the next class. I'll ask either my cohorts which I usually ask them first and then that one of the TAs" (AG1).* Thus, while students did describe using both the textbook and exercise visualizations to help them learn, they were not used in isolation or (typically) to the exclusion of other resources available to them.

RQ1: While a variety of engagement patterns were described, referring back to these visualizations was a common study strategy most students utilized.

4.2 RQ2: Did students perceive these exercises to be clear and helpful

When asked about the clarity of interactive visualizations in the textbook, many participants said that they were only eventually clear. As one student put it: *"I mean it is confusing when you first see it, but once you like try it and you'll understand what the textbook was trying to do"* (AG2). Notably, it is worth making the distinction here between "clear" and "immediately intuitive". For example when asked about clarity one student responded: *"Um, I think it was more clear after we did the classwork because it first like- I was just- Like I'd like we haven't really used it yet. So I didn't know like how it would apply specifically until we actually did something like that"* (MP). This type of response was emblematic of many students. Some students would simply say they found the textbook visualizations clear, then add that it took some work to understand or contextualize them.

This was not the case for every participant, though it was for the majority. While one did report that: *"if I had no idea of computational thinking and I read through that ¡textbook visualization¿ I would be able to teach myself"* (KU), it was much more common that contextualizing these visualizations with other materials was a critical step. This was demonstrated in the reports of participants who found them unclear even after working at them. For example as one student reported, *"I think that even when I like a read it ¡the textbook¿ it didn't directly be like, oh here is something that you can use to practice like here is a directions of like how work your way through this"* (LC). This student was struggling to understand the textbook visualizations because they could not relate them to the text surrounding them. The student's description of their experience is an example where the clarity of the visualizations could not be answered by a simple yes or no. Rather, the students understood the visualizations not in a vacuum but in the context of other course materials.

Even more frequently than with the textbook, students reported the exercise visualizations being clear only after some struggle. Again, the notions of immediately clear and eventually clear are distinct here. When asked about it, one student said: *"Yeah I didn't understand it. And then it was really easy wasn't like anything complex, but the TA had to kind of explain it to me"* (JC). This student shows how once they were able to understand the visualization within the context of other course materials, they found the exercises "easy" to complete. However, many students needed that additional context from other resources in order to achieve that. Students also mentioned the instructions for the interactive exercise being unclear. One student described looking at the exercise, and not understanding what task needed to be completed: *"I had to figure out like that took me a while to like understand what it was asking because the first question… it had a table of price and total and I was like, well, what is it do you mean to do?"* (AK). Here, clarity was an issue in terms trying to understand what the exercise was asking for. Students frequently asked for the help of their cohort or teaching assistants and then had little difficulty completing the exercises.

Some students did find the textbook visualizations to be highly helpful while many found them moderately helpful. One student put it clearly when they said: *"I guess it would help me like figure out how to get to the end and figure out the problem, but it wouldn't necessarily. I guess like...teach.... help me understand it better if that makes sense?"* (JC). Here, the visualizations were a tool to help them work through a given exercise or problem, but were less helpful in fully cementing the concept of list-based iteration. Only the students who avoided reading the textbook completely found the visualizations to be entirely unhelpful.

RQ2: Generally, students reported finding both types of visualizations to be clear and helpful, though some initial struggle making sense of the visualizations was common.

4.3 RQ3: Were these visualizations helping to foster a sound understanding of list-based iteration?

Throughout the interview students had a view of a new visualization similar to, but distinct from those students had seen before. This was the *Interview Visualization* shown in Figure 3. The third section of the interviews focused on students discussing the iteration in these figures. This was used to gain some insight into participants' schema of list-based iteration.

Many were quick to relate the new image to previous coursework. One student, when asked about the image on screen in their interview said: *"Yes. We've had like a couple quizzes with like these visualizations and like saying like typing in what the price is or what the total is sum and total stuff like that"* (AK). This student was one of many to relate the on screen image to the exercise visualization seen in Figure 2, where the goal was for students to trace through an iteration as it added up a list of integers. When the students refers to 'sum' and 'total' in these passages, they mean two variables inside the iteration seen in Figure 2. These terms were notably not present in the interview visualization seen in Figure 3. While some students also compared the interview visualization to other exercises, this comparison between the interview visualization and exercise visualization was especially frequent. It was perhaps a contributing factor in their descriptions of this new image.

When asked to describe the function of the interview visualization, students overwhelmingly responded that it illustrated the sum of the list of integers. Interestingly, this is not what the interview visualization computes: as described before, this visualization prints out the list of numbers (4, 13, 6, 9 and 11), but does not calculate a sum. What is notable about this consistent response was the lack of "clues" to imply that this visualization was adding numbers. No variables with names like "sum" or "total" were on screen. No adding operators were used in the visualization either. Even when students noticed the print statement in the body of the for loop, it did not seem to change their answer. As one student said: *"It's going to print the price. This is the only thing it says but it's probably like adding them up or something"* (MB).

Indeed, many seemed confident in their answer too. One participant, who had previously described iteration as an easy concept for them to understand, reported that this on screen visualization generated *"probably like a sum of some sort"* (HSh). While some students did provide the correct answer, they were very much in the minority compared to those who incorrectly said it calculated a sum.

RQ3: Students almost universally displayed a flawed understanding of the interview visualization, assuming it was identical in function to Figures 1 and 2.

5 Pedagogical Observations

5.1 Student Patterns in Comprehending Material

As described in the Results section, students were quick to compare the interview visualization with one of the exercise visualizations they had tried in class. Some participants described this process as recognizing patterns between exercises. For example one student suggested improving the exercises by making these ”patterns” more clear: *”...because there’s a pattern here, right? So kind of like specify more the pattern because I know there’s a pattern where you have... you name the list, you equal your sum to zero, and you write the iteration, and then you put these pieces together and then you evaluate”* (AG2). While many participants described the strategy slightly differently, these “patterns” seemed to be an attempt to more abstractly conceptualize list-based iteration.

Generally, students seemed to have a reasonably correct understanding of the concept of iteration. For instance, every student could accurately describe that an iteration applies the same code to all items in a list one at a time. They could also identify where looping was occurring in these visualizations. Also, they could identify what order the values in the list would be operated on. Furthermore, these aspects of iteration could be correctly identified both by students who reported feeling confident with the material and by those who still felt they did not fully understand.

However, this ”pattern finding” did have some significant pitfalls as well. Most notably, nearly all participants incorrectly said the interview visualization would output a sum. As discussed in the Results section, no variables were named things like “total” or “sum”, nor were there any addition operators present. However, students had been exposed to the exercises seen in Figure 2 before the interview. Thus, it’s possible participants assumed they were looking at their previous exercise as they were visually similar. Whether or not that is true, however, students seemed to rely on the patterns they had seen rather than applying a cognitive understanding to a novel experience. Many participants even noticed the print statement in the Figure 3, but still said the code outputted a total of the numbers. For example, one student mentioned when pointing out the components of the visualization that repeat said: *”I feel like it would be possibly printing with numbers”*(AG1). However, when asked what running this code would output said: *”...after it is executed it should give you a sum of all the numbers”*(AG1). This student’s description is notable as they did notice the interview visualization was different from their in-class exercise, but still guessed the code produced a sum. Thus this interview visualization highlighted both what participants understood though pattern finding and what misconceptions they had.

Through a Constructivist lens, participants’ approach to comprehending the interview visualization makes some sense. The National Research Council explains this in their book on effective instruction in undergraduate science and engineering: *”This depth and organization of knowledge enables experts to notice patterns, relationships, and discrepancies that elude novices. It allows them to quickly identify the relevant aspects of a complex problem or situation, make inferences, and draw conclusions. . . .The knowledge that novices possess, by contrast, is often disconnected, unorganized, and therefore less usable...they may focus on aspects of a problem, such as superficial details, that make it more difficult rather than easier to solve*[21].

While a more focused investigation may be needed to fully understand the behavior seen here, this may explain how few participants seemed to focus or analyze the code inside the iteration structure. Rather, they were capable of describing iterations in the abstract, but not what the interview visualization might output. It may also be the case that the additional text above the exercise and iteration visualizations, may have helped students contextualize what they represented.

The pattern finding strategy points toward the development of exercises or problems that have student identify the similarities and differences between related but distinguishable artifacts. For example, posing questions like ”How is this iteration similar to and different from this other iteration?” may help students better distinguish between the core concept and problem-specific details.

Student pattern finding is a common strategy, but instructors should keep in mind that students may have trouble forming the right pattern. Curriculum designers should consider including an instructional approach that assists students with correct pattern finding.

5.2 Student Engagement Patterns and Pedagogical Implications

Through these interviews we came to realize that our expectations of a ”successful” student did not match the behavior patterns seen here. Before starting this experiment, we hypothesized that in order to do well in the class, a student would need read thoroughly, perhaps take notes, then work on exercises. While some participants did seem to take this approach reported skimming or avoiding the textbook and only referring back if they found it necessary. Many students seemed to treat any resources that were not directly graded as more of a set of tools to be employed. They would exhibit preferences for some tools and avoid others as they saw fit. This shows students not simply completing rote tasks put in front of them, but metacognitively evaluating resources at hand and their own understanding of material. This in itself stands as an important reminder to instructors to be cognizant about potential differences between their expectations with the actual strategies of their students.

In terms of Bloom’s revised taxonomy, this would imply students are at the ”Apply + Metacognitive” Stage of knowing as they are able to evaluate how they learn best and use those techniques[22]. This analysis may also assume the Felder-Silverman model of learning styles in which students exhibit preferences in how they best take in information[23].

However, though understanding the interview visualization is by no means a precise measurement of student success, it was noteworthy that no one pattern of interaction correlated with correctly explaining the interview visualization. Ultimately, student descriptions of the interview visualization revealed an issue that goes deeper than correctly or incorrectly knowing what the code did. Nearly every student assumed the interview visualization (Figure 3) functioned identically to the exercise visualization (Figure 2) even though no plus signs or variables were visible. Students were missing key signposts in the visualization that would have helped them think through the problem.

Thus, students are analyzing and using the course materials that work best for their specific learning style yet it is not helping them look critically at these visualizations. In terms of next steps, there seem to be two different options informed by one’s pedagogical approach.

One possible conclusion we could draw is that few participants were employing a truly successful study strategy. The behavior seen in these interviews could imply students behaving in ways that get them a good grade without genuinely taking in or mastering the information. This sort of behavior seems distinctly possible among any population of non-majors in an introductory class. For example, in Fouh’s examination of interactions with an e-textbook in a Computer Science course[7], the majority of students ignored the readings; only doing what was required to receive credit for reading. Students may or may not be exhibiting the same ”credit seeking”[7] behavior seen in this paper, but they are likely not reading as thoroughly as an educator might expect when the textbook was written.

From this point of view, the visualizations created were at best somewhat successful at communicating the underlying ideas of list-based iteration. It is possible changes to the visualizations could help improve this epistemic fidelity[24]. Additionally, educators could change the class to encourage better patterns of study behavior like grading full engagement with the visualizations.

Alternately, one could conclude that educators need to lean into this tools-based approach. Here, it may be best to examine different frameworks of learning styles, such as the Felder-Silverman sensory-focused model[23]. Examining approaches to learning such as Entwistle, et al.'s framework[23] may also be informative. Providing many resources across these different categories would allow learners to tailor their own experiences. Here, the goal would be less focused on getting student behavior to conform to a certain style and more focused on providing a variety of tools for students to pick and choose from.

From this perspective, the visualizations were arguably highly successful. Some students interviewed found them to be very clear and helpful – and creating a resource for this audience was all they needed to do. Instead of considering how best to improve these visualizations, it may be better to consider how to better serve those students who struggled with them.

For this course, as the student population is large and diverse, we tend to find ourselves more in the latter interpretation. We judge these visualizations to be a success for the students that use them and are likely to find or develop other resources for other learning styles.

Analysis of the success of instructional visualizations varies based on the theoretical viewpoint. Visualizations are successful tools for some learners and instructors should consider providing a variety of “tools” that students could use in a variety of ways rather than believing that all students will use the course materials in the same way.

6 Threats to Validity

As is the case with all qualitative research, the work can be highly interpretive. All interviews were conducted, transcribed and coded by the first author leading to potential bias. Thus it is a possibility that this single viewpoint biased the analysis and results.

Participants could also bring some biases into the study. While efforts were made to mitigate this issue, questions could have been misinterpreted by participants during the interview. For example, while the interviewer did their best to clarify, a student may have mixed up the terms textbook and exercise visualizations which could have impacted their answers. As mentioned in section 5, the additional text may have helped students contextualize the visualizations which might have impacted their understanding of the interview visualization. Additionally, participants could have tried to present the best possible side of themselves, rather than being fully candid in their reports.

All participants did also volunteer to be interviewed, thus the perspectives seen in interviews may not be shared by all students who take the course.

Additionally, a small rephrasing of the directions for the exercise visualization was done between the fall semester of interviews and the spring semester. It is possibly due to this change that slightly fewer participants in the spring semester of interviews reported having trouble with the directions in completing that work. However, as the change was only slight, it is difficult to determine exactly what it occurred if a reason exists at all.

7 Future Work

The insights offered by these interviews shows many opportunities for deeper study. Most immediately, it would be useful to know if these findings were applicable outside of this highly specific context. Additional quantitative and qualitative studies of other courses will be needed to know if, or to what extent, our findings apply to other contexts.

Additionally, evaluating the interview visualization could be seen as a more complex task than completing an exercise visualization to novice programmers. It would be useful to apply existing educational theory, such as Webb's Depth of Knowledge levels[25], to the visualizations described in this study. Hopefully, that may allow for more conclusions to be drawn on how students made sense of Figure 3.

8 Acknowledgments

This work was funded by the National Science Foundationgrants DUE 1624320, DUE 1444094, and DUE 0822220.

References

[1] Dennis Kafura, Austin Cory Bart, and Bushra Chowdhury. A computational thinking course accessible to non-stem majors. *J. Comput. Sci. Coll.*, 34(2):157–163, December 2018. ISSN 1937-4771. URL `http://dl.acm.org.ezproxy.lib.vt.edu/citation.cfm?id=3282588.3282611`.

[2] Benedict Du Boulay. Some difficulties of learning to program. *Journal of Educational Computing Research*, 2 (1):57–73, 1986. doi: 10.2190/3LFX-9RRF-67T8-UVK9. URL `https://doi.org/10.2190/3LFX-9RRF-67T8-UVK9`.

[3] Alireza Ebrahimi. Novice programmer errors: language constructs and plan composition. *International Journal of Human-Computer Studies*, 41(4):457 – 480, 1994. ISSN 1071-5819. doi: https://doi.org/10.1006/ijhc.1994.1069. URL `http://www.sciencedirect.com/science/article/pii/S107158198471069X`.

[4] Nell B. Dale. Most difficult topics in cs1: Results of an online survey of educators. *SIGCSE Bull.*, 38(2):49–53, June 2006. ISSN 0097-8418. doi: 10.1145/1138403.1138432. URL `http://doi.acm.org.ezproxy.lib.vt.edu/10.1145/1138403.1138432`.

[5] Jieun Chon. Iteration Visualization for Novice Learners. Master's thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, 2019.

[6] R. E. Mayer. Applying the science of learning: Evidence-based principles for the design of multimedia instruction. *American Psychologist*, 63(8):760–769, November 2008.

[7] Eric Fouh, Monika Akbar, and Clifford Shaffer. The role of visualization in computer science education. *Computers in the Schools*, 29:95–117, 01 2012. doi: 10.1080/07380569.2012.651422.

[8] Rossevine Artha Nathasya, Oscar Karnalim, and Mewati Ayub. Integrating program and algorithm visualisation for learning data structure implementation. *Egyptian Informatics Journal*, 20(3):193 – 204, 2019. ISSN 1110-8665. doi: https://doi.org/10.1016/j.eij.2019.05.001. URL `http://www.sciencedirect.com/science/article/pii/S1110866518302603`.

[9] Visualize code execution learn python, java, c, c , javascript, and ruby. URL `http://www.pythontutor.com/`.

[10] Raghvinder S. Sangwan, James F. Korsh, and Paul S. Lafollette. A system for program visualization in the classroom. *ACM SIGCSE Bulletin*, 30(1):272–276, 1998. doi: 10.1145/274790.274311.

[11] Nouf M. Al-Barakati and Arwa Y. Al-Aama. The effect of visualizing roles of variables on student performance in an introductory programming course. *ACM SIGCSE Bulletin*, 41(3):228, 2009. doi: 10.1145/1595496.1562949.

[12] Zhang Jinghua, Mustafa Atay, Rebecca Caldwell, and Elva Jones. Visualizing loops using a game-like instructional module. pages 448–450, 07 2013. doi: 10.1109/ICALT.2013.137.

[13] John P. Dougherty. Concept visualization in cs0 using alice. *J. Comput. Sci. Coll.*, 22(3):145–152, January 2007. ISSN 1937-4771.

[14] Marie Olsson, Peter Mozelius, and Jonas Collin. Visualisation and gamification of e-learning and programming education. *Electronic Journal of e-Learning*, 13, 01 2016.

[15] Michael T. Rücker and Niels Pinkwart. Towards a grounded theory of how students identify computing. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*, WiPSCE '17, page 111–112, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450354288. doi: 10.1145/3137065.3137071. URL `https://doi-org.ezproxy.lib.vt.edu/10.1145/3137065.3137071`.

[16] Katherine Deibel. Studying our inclusive practices: Course experiences of students with disabilities. *SIGCSE Bull.*, 39(3):266–270, June 2007. ISSN 0097-8418. doi: 10.1145/1269900.1268861. URL `https://doi-org.ezproxy.lib.vt.edu/10.1145/1269900.1268861`.

[17] Judy Sheard, S. Simon, Margaret Hamilton, and Jan Lönnberg. Analysis of research into the teaching and learning of programming. In *Proceedings of the Fifth International Workshop on Computing Education Research Workshop*, ICER '09, page 93–104, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605586151. doi: 10.1145/1584322.1584334. URL `https://doi.org/10.1145/1584322.1584334`.

[18] 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*, ITiCSE '14, page 279–284, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450328333. doi: 10.1145/2591708.2591762. URL `https://doi.org/10.1145/2591708.2591762`.

[19] Ma. Mercedes T. Rodrigo, Ryan S. Baker, Matthew C. Jadud, Anna Christine M. Amarra, Thomas Dy, Maria Beatriz V. Espejo-Lahoz, Sheryl Ann L. Lim, Sheila A.M.S. Pascua, Jessica O. Sugay, and Emily S. Tabanao. Affective and behavioral predictors of novice programmer achievement. *SIGCSE Bull.*, 41(3):156–160, July

2009. ISSN 0097-8418. doi: 10.1145/1595496.1562929. URL
`https://doi-org.ezproxy.lib.vt.edu/10.1145/1595496.1562929`.

[20] Raymond Lister, Tony Clear, Simon, Dennis J. Bouvier, Paul Carter, Anna Eckerdal, Jana Jacková, Mike Lopez, Robert McCartney, Phil Robbins, Otto Seppälä, and Errol Thompson. Naturally occurring data as research instrument: Analyzing examination responses to study the novice programmer. *SIGCSE Bull.*, 41(4):156–173, January 2010. ISSN 0097-8418. doi: 10.1145/1709424.1709460. URL
`https://doi.org/10.1145/1709424.1709460`.

[21] National Research Council. *Reaching Students: What Research Says About Effective Instruction in Undergraduate Science and Engineering*. The National Academies Press, Washington, DC, 2015. ISBN 978-0-309-30043-8. doi: 10.17226/18687. URL
`https://www.nap.edu/catalog/18687/reaching-students-what-research-says-about-effective-`

[22] Revised bloom's taxonomy. URL
`https://www.celt.iastate.edu/teaching/effective-teaching-practices/revised-blooms-taxon`

[23] Richard M. Felder and Rebecca Brent. Understanding student differences. *Journal of Engineering Education*, 94(1):57–72, 2005. doi: 10.1002/j.2168-9830.2005.tb00829.x.

[24] Christopher D Hundhausen, Sarah A Douglas, and John T Stasko. A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages & Computing*, 13(3):259–290, 2002.

[25] Bloom's taxonomy and webb's depth of knowledge. URL
`https://www.synergiseducation.com/blooms-taxonomy-and-webbs-depth-of-knowledge/: :text=`
`Depth of Knowledge (DoK,level of complexity in thinking.`