Using Visual Aids to Strengthen Student Understanding in CMSE 201
Mentored Teaching Project
Certificate in College Teaching

Emily Bolger
Mentor: Dr. Devin Silvia

March 10, 2023

Contents

1 Introduction

In Fall 2020, I started my Teaching Assistantship with the Computational Mathematics, Science, and Engineering (CMSE) Department at Michigan State University teaching an introductory to data analysis and computational modeling course. During my first year, the course was taught remotely as we were in the height of a global pandemic. In the virtual environment, I quickly realized the challenges of trying to explain code logic to introductory students without ease of drawing or demonstration. Further, it was not only challenging as an educator to make these visualizations, it hindered the students' ability to deeply understand the tools we were teaching them. While there is often a disconnect between the syntax of the code and the conceptual understanding of the code for introductory students, this was only exacerbated by the virtual environment.

The disconnect I noticed in students was most apparent in coding tools such as lists, data frames, and two-dimensional arrays. Students often had trouble visualizing the structure of these objects and as a result, struggled with learning how to retrieve information from them using indexing. In some instances, instructors use analogies to describe the composition of these objects. However, it would also be valuable for students to have visual representations that they can reference at any time. It would help novice students grasp initial concepts that persist through various coding tools and strengthen their ability to apply algorithms to real-life problems [Qian and Lehman, 2017, Rudder et al., 2007]. Further, we know that these visualizations are most beneficial when they require active involvement from the students [Naps et al., 2002].

Beyond deciphering the structure of the object, the flow of a code piece or full algorithm is challenging for students to breakdown. This is occassionally referred to as tracing, and it was shown by Lopez et al. [2008] to be positively related to effectively writing code. The authors highlight challenges that students faced explaining code chunks specifically related to loops and arrays. Additionally, Izu et al. [2019] analyzed how students approach tasks that require them to think about the direction of incrementation in their loop. In general, students defaulted to using loops with forward incrementation, even when it was not optimal to solve the task. They claim that this is a result of students being exposed more frequently to forward loops as well as the students starting to code before planning out their solution. Similar studies analyze how students think about indexing as it relates to using for loops [Cherenkova et al., 2014, Miller and Settle, 2021, Rigby et al., 2020]. Specifically, Miller and Settle [2021] noticed that many students defaulted to using a loop by value approach when a loop by index was most appropriate. Further, when students loop by index, they struggle with setting the upper bound of the loop and often attempt to access elements beyond the length of the list [Rigby et al., 2020]. Loops that are created to access elements in a data frame or a two-dimensional array are very similar to those created for elements of a list. Thus, student misunderstandings in loops with lists are likely to persist with more complex coding objects.

In my experience, when presented with coding problem, many students were able to recognize which coding tools would be useful in solving the problem, but did not understand the tool at a deep enough level to know how to update the code to match the specific situation. This inability to transfer knowledge to new applications has been studied by computationalists and is sometimes referred to as inert knowledge [Davies, 1993, Pea, 1986, Perkins and Martin, 1985]. In fact, after conducting think aloud interviews with students in their first or second year of coding, Fitzgerald et al. [2005] noted that while students use a range of strategies for solving computational problems, they often used pattern recognition and code syntax identification.

Related to this, students need practice articulating their code implementations and algorithms. This skill strengthens the students' understanding of the code, while also encouraging them to practice justifying their work to another audience, which is a practice that is useful beyond the classroom walls [Berland and McNeill, 2012, Osborne and Patterson, 2011]. While some students have this experience by working in groups on coding projects, it is crucial that they practice this more consistently through written tasks in course work, guided discussion with peers, and conversations with instructors. The ability to communicate and collaborate with a team are highly requested

skills in academia and industry [Council, 2012, Halwani et al., 2021].

Thus, in this work, we integrate visual aids in course materials to help students in their understanding of coding practices like lists, data frames, and two-dimensional arrays, particularly in relation to indexing with those tools. The goal of the visual aids is to help students conceptualize the structure of the object as well as walk through examples that show information retrieval from these objects. We also embed guided questions for students to practice explaining their coding choices and implementations. Specifically, we are looking to study how students think about and explain code using descriptive language to their instructors, fellow coding classmates, and non-coders, particularly when thinking about for loops, data access, and array manipulation. We want to analyze whether this practice of visualization and explanation deepens student understanding of the course content.

As we will discuss, many students did not partake in the tasks that required them to share their understanding and explain their work, thus we focus on student reception to the material, and note places for improvement in the materials to encourage students to have these experiences.

2 Modifications to CMSE 201

I explored my research question within the context of Introduction to Computational Modeling and Data Analysis I in the Computational Mathematics, Science, and Engineering Department (CMSE 201) [Silvia et al., 2019]. The course is targeted towards students in all disciplines and requires no prior background in coding. The course seeks to provide students with basic data analysis and computational modeling skills that they can use in their own fields throughout their career. Students learn the basics of Python as well as many libraries in Python that focus on data visualization, data analysis, and modeling techniques.

The course is taught in a flipped classroom style, like other recently developed computational courses [Gong et al., 2020, Irving et al., 2017, Ortega-Alvarez et al., 2020, Pipitgool et al., 2021]. Students complete pre-class assignments before attending class that introduce the current topic through videos and examples. In class, students work in groups to complete a guided problem set about the same topic. In addition, students individually complete bi-weekly homework assignments that are graded by the teaching assistants based on a rubric provided by the instructor that created the assignment. In addition to the homework assignments, students complete an end of semester project on a topic of their choice using tools from the course. Finally, the course has a midterm exam and final exam.

There are typically 6-8 sections of this course per semester with about 40-60 students in each section. Thus, making large scale changes to the course materials is very challenging. For this reason, the changes that I implemented were modifications to existing course content.

By focusing on for loops, data access, and array manipulation, I modified 10 assignments for CMSE 201, at varying levels. For each section, I list the modifications that were made. In general, the changes included aids for the students to help visualize different pieces of code and conceptual questions aimed to deepen student understanding of coding tools as well as give them practice explaining code pieces to others.

Full course assignments that I modified can be seen in the portfolio.

2.1 For Loops and Indexing

Early in a first coding course, students are introduced to for loops. These are frequently used in coding algorithms to iterate through a set of objects. However, students are often struggle with understanding the use of indices and how it relates to object selection from a list in the for loop. To aid this development, I created gifs to walk through the steps of a for loop. It highlights the item in the list that is being selected, the associated index of that item, and the output of the loop (see Figure 1).

To further clarify the use of indices in for loops (as seen on the right of the Figure 1), we contrast with a syntactically different for loop, typically referred to as for loop by value (on the left side). By putting the different types

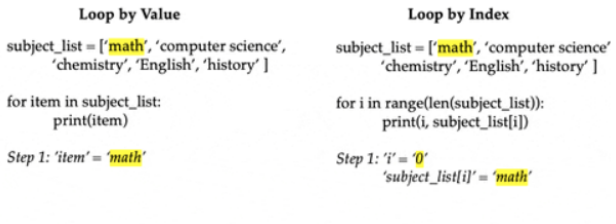


Figure 1: Snapshot of gif used in a pre-class assignment of CMSE 201 to help reinforce the concept of indexing when using for loops.

of for loops side-by-side, we seek to highlight the similarities in the output, but difference in syntax.

In the same pre-class assignment as well as the in-class assignment, students are given application problems that require them to use a for loop. In both assignments, student are tasked with first explaining the information they need to complete their task and how they will use that information to solve the given problem. In addition, students are asked to explain the idea of an index to a non-coder. These components of the assignment give the instructors further insight to the student's understanding of for loops rather than simply looking at their code. It also encourages the students to think more deeply about the purpose of the coding tool instead of mapping the syntax to a type of problem.

2.2 Data Access

The challenge that students encounter with indices in for loops often permeates in using indices to retrieve information from a data frame. To help counteract this, we created a second set of gifs that show a task retrieving information from a data frame, the code used to complete that task, and the information in a data frame that the piece of code retreives (see Figure 2). The visual aids show examples of different data retrieval techniques used in the Pandas library from Python.

In addition to the gifs, students were asked to complete various data retreival tasks. For a given question, they were to describe the information they needed to complete the task (e.g. a column, a row, some combination of

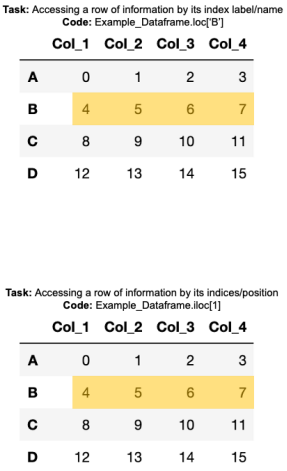


Figure 2: Snapshot of gif used in a pre-class assignment of CMSE 201 to help reinforce the concept of indexing to access pieces of a data frame.

both), the available coding options to complete the task, and the code they would use to complete the task. Breaking down the task on this level of detail encourages the students to think more criticallly about the information they wanted to retrieve and the coding tools available to them. Similar to the questions in the for loop section, we wanted students to understand the coding techniques, rather than simply memorize a mapping between a specific syntax and a specific problem.

Similar to the critical questions presented in the pre-class assignment, students were tasked with questions in the following two in-class assignments that guided them towards thinking about data retrieval and referred them back to the gifs in the pre-class assignment.

2.3 Array Manipulation

Finally, we used visualizations to represent indexing in two-dimensional NumPy arrays. In my experience, the challenges that students struggle with when using arrays are related to both indexing in for loops and in data frames. In Python, arrays can be thought as a list comprised of lists or a mathematical matrix. To iterate through all the

elements in the array, one uses nested for loops - one that iterates through the lists and one that iterates through the objects in the selected list. Thus, if students struggle with indexing in lists, it affects their ability to extend it to two-dimensional arrays. Further, in order to access elements of an array, we use similar slicing techniques that can be used for data frames. Once again, if slicing in data frames is difficult for students it usually continues in arrays.

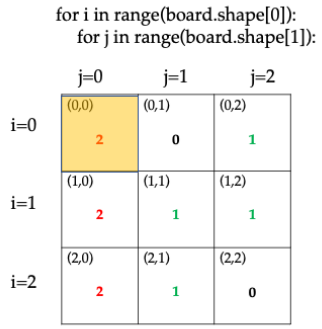


Figure 3: Snapshot of gif used in a pre-class assignment of CMSE 201 to highlight 2-dimensional indexing.

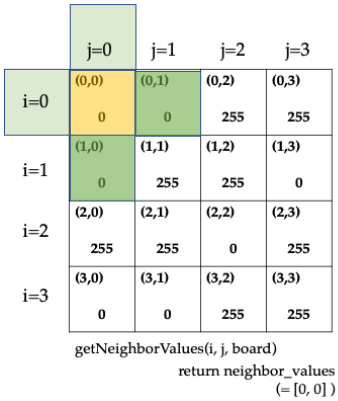


Figure 4: Snapshot of gif used in a pre-class assignment of CMSE 201 to highlight 2-dimensional indexing and the indices of neighboring elements

Keeping all this in mind, we created a couple visualizations (Figures 3-4). Each gif highlights a few different aspects of arrays. Figure 3 shows the code typically used for iterating through a two-dimensional array and accessing each element in the array. The numbers in the top left corner of each box represent the row and column indices for that element, which are further emphasized through row and column labels. For example, in the top left box, the numbers are $(0, 0)$ emphasizing that both $i = j = 0$ in the first iteration of the loop. The yellow box highlights which element is selected based on the values of i and j. In addition to which element they are selecting, we emphasize the difference between the indices and the value at the index - similar to the for loop visualization. The element at indices $i = j = 0$ is 2. It is colored red as a reminder that 2 represents a fire, as this gif comes from an assignment where students are modeling the movement of a fire through a forest.

In Figure 4, we show similar information. For a task in the in-class assignment, students needed to design a function that returns the values of the neighbors for a given element. This gif was created to help them visualize what those neighbors are for different elements in the array, the indices of those neighbors, and the values of those neighbors. As the gif evolves, students see the neighbors (green boxes) of a given element (yellow box).

Following suit to the for loops and data frames, we task students with explaining their solution to problems related to array manipulation. They are asked to explain what information they were presented with, what information they needed to complete the task, how they retrieved that information, and how they used that information for solving the larger problem. We seek to help students more critically think about the coding tools they are employing, while additionally giving them practice seeing where their smaller tasks fit into the larger solution.

3 Assessment Technique

To assess the modifications implemented in the CMSE 201 curriculum, we used three methods: bi-semesterly survey, in-class student observations, and class assignment observations.

3.1 Bi-Semesterly Survey

We distributed a bi-semesterly survey to the students to gauge self-reported student confidence in completing various tasks related to critical concepts in CMSE 201. It also was used as a self-check study tool for the mid-term and final exams, and thus, included more topics than this project aimed to study.

Using a Likert Scale (1: Not Confident → 5: Extremely Confident), we gather information on six topics: Lists and Indexing, For Loops, Functions, Plotting, Data Retreival and Usage, and Array Retreival and Usage. We assess student confidence in completing tasks related to these topics in course assignments as well as outside of the classroom. Additionally, they report their confidence in explaining these topics to fellow classmates and non-coders.

The survey can be viewed here. Only students that completed both surveys were included Section 4.

3.2 In-Class Student Observations

For each in-class assignment that was modified, I documented field notes about student reception to the material as they completed it in class [Phillippi and Lauderdale, 2018]. This included student group discussions around the material, individual student progress based on curriculum changes, and conversations that were instigated through these changes. For each piece of the assignment that I changed, I noted how it seemed to affect student understanding of the material as well as where it could have been improved. The notes reference groups of students for privacy reasons.

3.3 Class Assignment Observations

In addition to the observations on the in-class assignments, I also took detailed notes on student responses to pre-class and homework assignments. Again, I noted specific questions related to student reception, rate of completion, level of detail, and uncompleted components.

4 Analysis

To interpret the results from the bi-semesterly survey, we use lollipop and parallel plots as well as conducted a Wilcoxon Signed Rank Test. We compare the student reported results in the plots and the statistical test with the coursework observations.

4.1 Plotting Techniques

To show the change in the student's responses before the mid-term and before the final, we chose lollipop and parallel plots. In Figures 5-10, we show the student survey responses regarding confidence describing a list or index, for loop, and data analysis task to a classmate or non-coder. We do not have information on self-reported confidence regarding arrays before the mid-term, since students do not learn two-dimensional arrays until the last third of the semester.

In the Lollipop Plots, for each student we compare their mid-term self-assessment score with their final self- assessment score. The distance of the line between the two points indicates the change in student ranking. Looking at Figure 5, we see many students start and end with a higher confidence in explaining the concept of an index to a classmate than a list to a noncoder. Of course, these are not direct analogs, but interestingly about one third of the students indicated at both points in the semester that they could very confidently explain an index to a classmate.

Further, most student showed in increase in confidence by 1-2 levels. However, about three students in the class indicated an increase in their confidence of explaining indices by 3. Pinpointing when this shift in confidence occurred during the semester would be interesting as this same jump does not appear in the bottom plot of Figure 5. In both plots, we notice a few students who decreased their confidence level as the semester progressed. These cases are particularly intriguing and may indicate that students overestimated their confidence in understanding early in the semester.

We look at the same type of plots, in Figure 6, regarding for loops. In this set, we are asking students to state

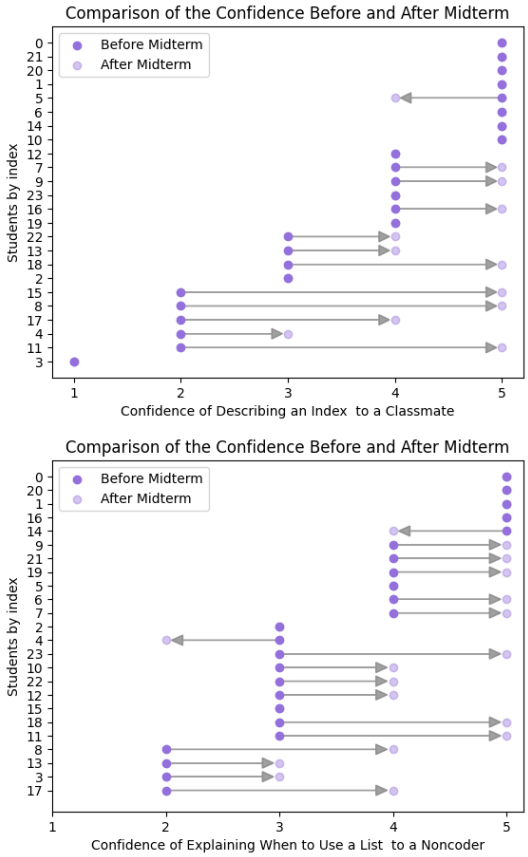


Figure 5: Student survey responses to explaining lists and indexing to coders and non-coders.

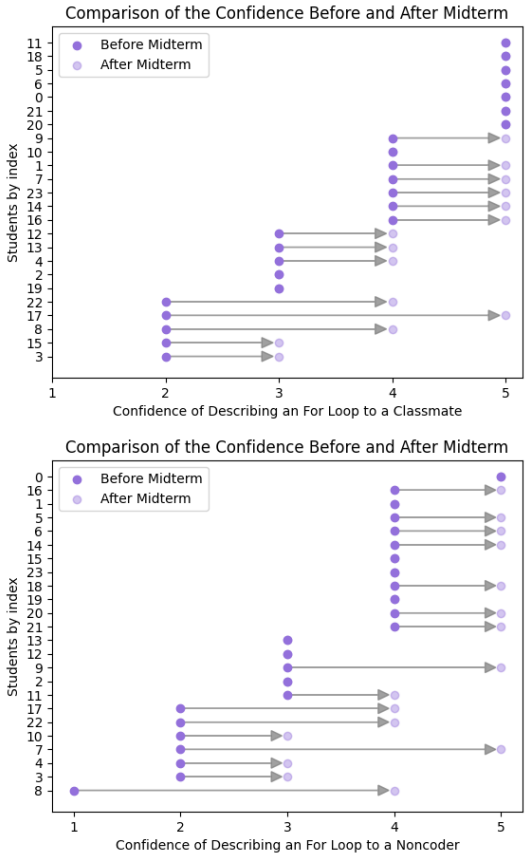


Figure 6: Student survey responses to explaining for loops to coders and non-coders.

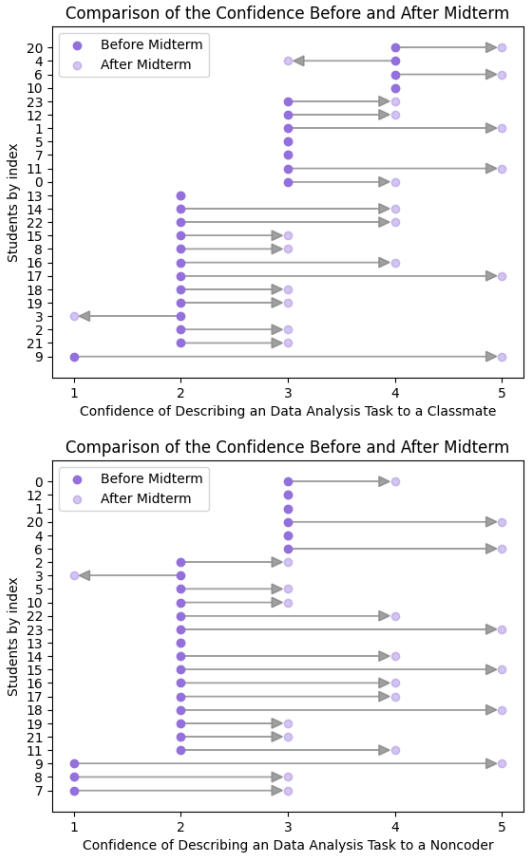


Figure 7: Student survey responses to explaining data to coders and non-coders.

their confidence in describing a for loop to a classmate and a for loop to a noncoder. We notice many similarities between the two plots across number of students in each confidence level and changes in confidence levels, which are all increasing. The most striking difference between them is the number of students who were Extremely Confident (5) about their ability to describe a for loop to a classmate both before and after the midterm, but did not have that same confidence when asked about explaining it to a non-coder.

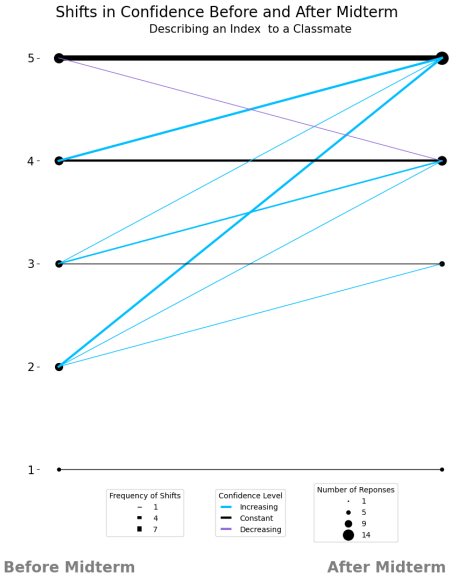
Finally, we compare student confidence regarding data analysis tasks in Figure 7. While we see a few students who confidence decreases throughout the semester, most students shown an increase in confidence. Further, most of those increased were very large, particularly for communicating with a noncoder. This makes sense as students start to learn about data analysis tasks shortly before the midterm.

In Figures 8-10, we present the same information in a parallel plot. In this format, we can more easily see the number of level changes that occur. Again, each line represents the change in confidence level. More specifically, a blue line indicates an increase in confidence, a black line indicates no change in confidence, and a purple line indicates a decrease in confidence. The width of the line indicates how many students follow that path. The size of the nodes indicate the number of responses for that confidence level.

Looking at Figure 8, the number of students whose confidence level remained at a 5 for explaining an index to a classmate compared to a list to a noncoder is clear. Additionally, we see many students report an increase in confidence regarding explaining lists to a noncoder, particularly from levels 4 to 5.

In Figure 9, we notice there are many more students who remain at the same confidence level when describing for loops to classmates and noncoders, which for some students is at a 3. In both plots, we again notice many students shifting from 4 to 5. Further, we see a similar branching from level 2 to levels 3, 4, and 5.

The similarities between the classmate and non-coder confidence levels continue in Figure 10. With the exception of a few lines, the plots are nearly identical in the changes



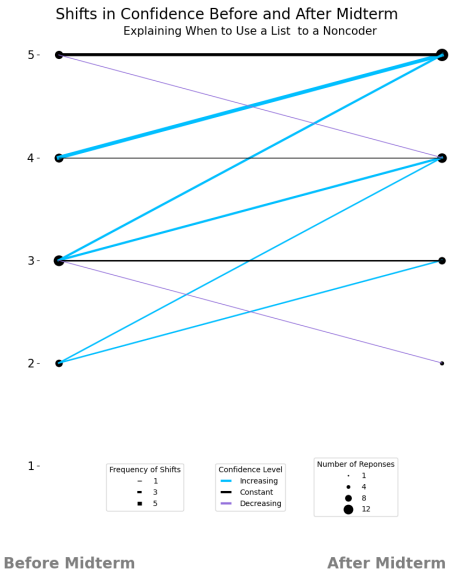
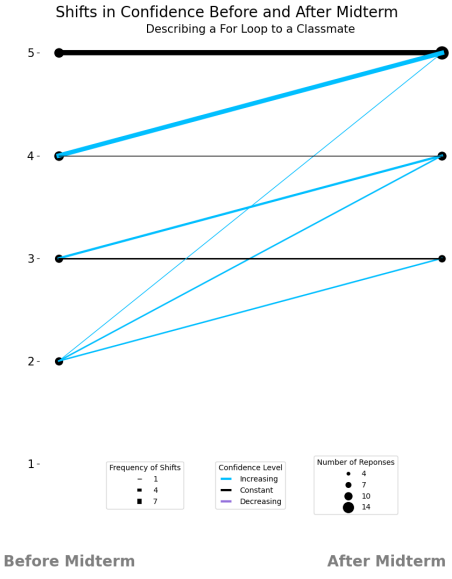


Figure 8: Student survey responses to explaining lists and indexing to coders and non-coders.



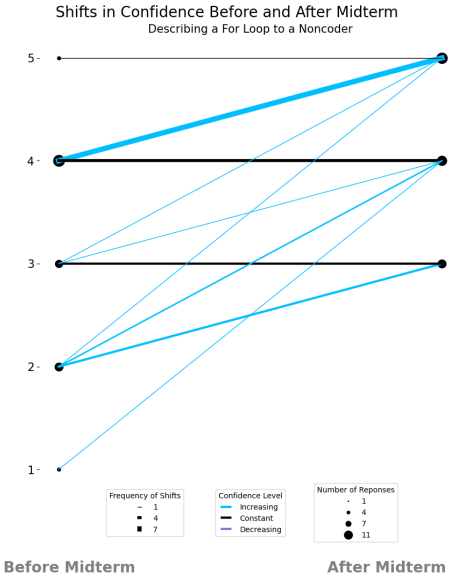


Figure 9: Student survey responses to explaining for loops to coders and non-coders.

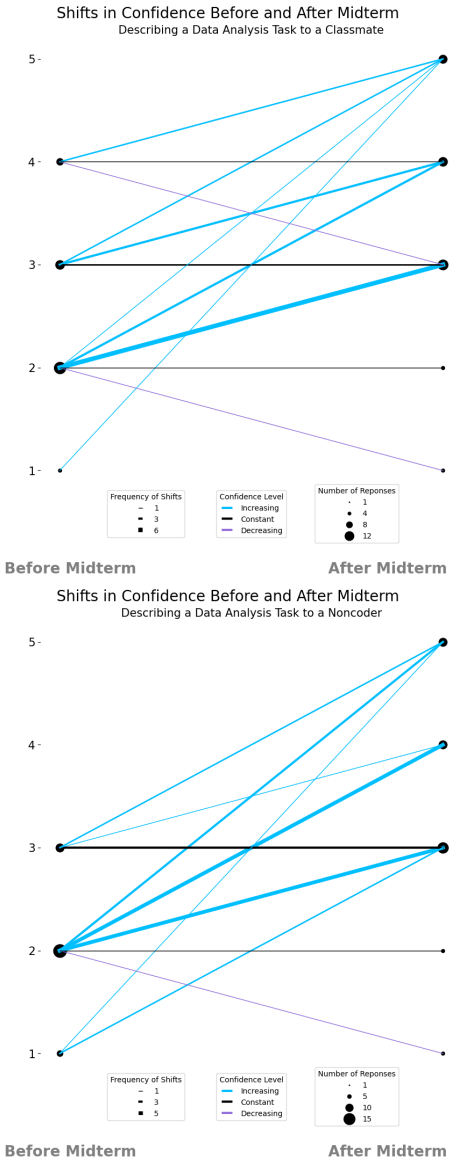


Figure 10: Student survey responses to explaining data to coders and non-coders.

that are present and the frequency in those changes.

Generally speaking, student confidence increased over the semester for explanatory tasks with fellow classmates and non-coders. However, many students indicated a higher confidence in describing these concepts to classmates over non-coders. Further, that confidence level increased at the most frequency particularly for the list and for loop tasks. It would be interesting to interview students about how they view these two populations differently. For example, how would explaining a coding concept to a noncoder make them change their approach, if at all? What would those changes be? On the same token, do the students' increase in confidence relate to increase in understanding of the material?

Finally, we briefly discuss student survey responses to questions regarding arrays. Students learned two- dimensional NumPy arrays after the mid-term exam and thus, they were only asked about their confidence regarding using them before the final. Immediately, we notice that most of the responses are a 3 or higher. The category with the largest about of responses below a 3 are explaining an array to a non-coder.

4.2 Statistical Test

To statistically analyze the differences in a single students' response, we used a Wilcoxon Signed Rank Test. The non-parametric test determines if the distribution of paired differences is symmetric around 0. This test is often used in place of a paired sample t-test when the data is ordinal, interval, or ratio scale as we have with the Likert scale responses. For this test, the differences need to be approximately symmetric, which we found to be true.

To calculate the test statistic, we assign ranks, R_i, to the absolute differences, $|D_i|$, of the student responses, $i = 1, \cdots, n$. We omit any differences of 0 and update the number of samples, n'. For differences with the same value, we use an average ranking method and subsequently, reduce the standard error by $\frac{t^3-t}{48}$ for each group of t tied ranks. The test statistic, W, is the sum of the ranks of the initial positive differences:

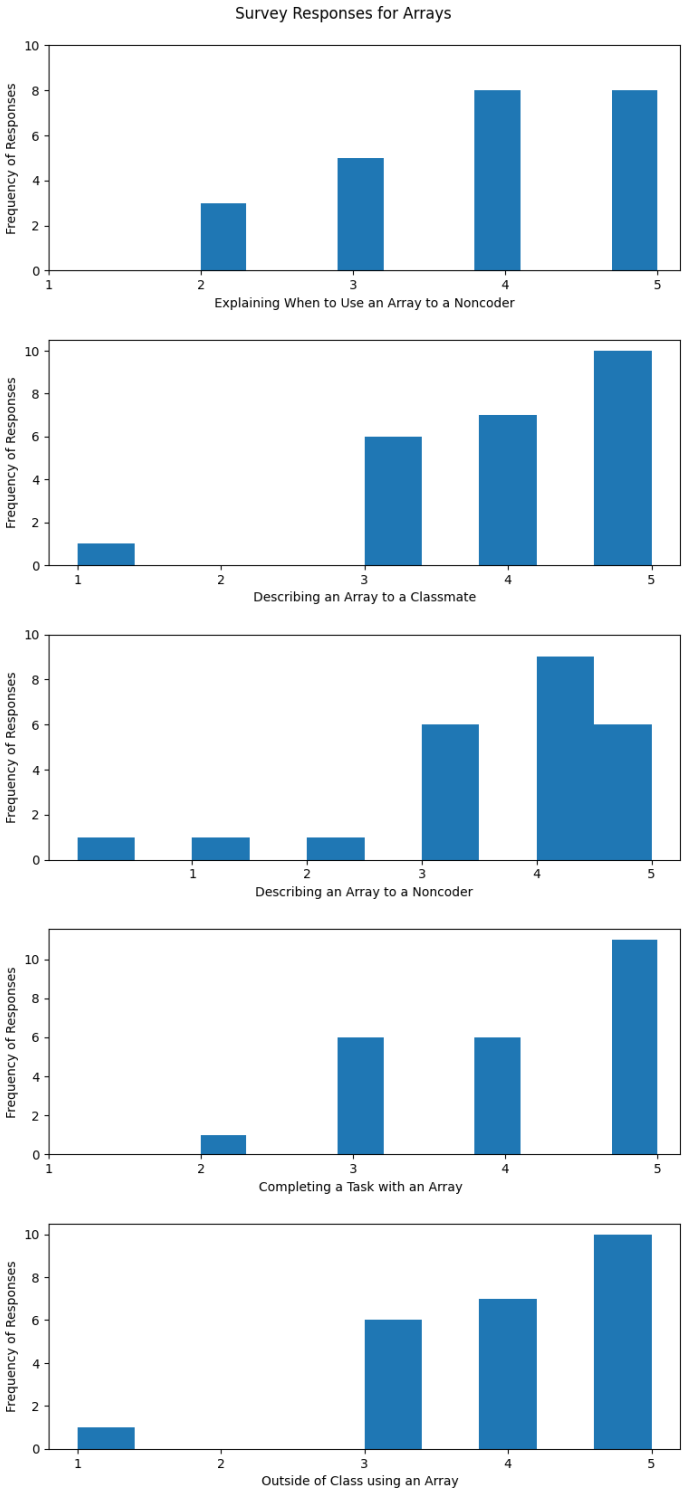


Figure 11: Student Responses for NumPy Array.

$$W = \sum_{i=1}^{n'} R_i^{(+)}.$$

For a large enough sample, $W \sim N(\mu_W, \sigma_W)$ for $\mu_W = \frac{n'(n'+1)}{4}$ and $\sigma_W = \sqrt{\frac{n'(n'+1)(2n'+1)}{24}}$.

We use scipy's implementation to calculate the p-values, which are presented in the following table.

Task	To Whom	p-value
Index	Classmate	0.0049
List	Non-Coder	0.0017
For Loop	Classmate	0.00048
For Loop	Non-Coder	0.00026
Data Analysis	Classmate	0.00044
Data Analysis	Non-Coder	0.00013

We notice that all the values are extremely small, indicating that the paired differences are statistically different from 0. There is enough evidence to claim that the students confidence levels were different between the two points of data collection for each of the survey questions considered. However, we note a few things. First, our total number of participants is 23, which is a small sample to start with. Additionally, for some of the questions, there are a fair amount of users who did not report any changes to their confidence level. For example, in the top graph of Figure 7, there are 7 users who remained at a 5. These get removed from the sample. Looking at the same graph, there are many users with the same increase in confidence. There are 6 users who move from a 4 to a 5. While we use the average ranking method for these users and adjust the variance accordingly, the amount of ties affects the test statistic and hence the p-value calculation.

4.3 Field Notes and Course Observations

For the sake of brevity, we choose a few noteworthy observations for each of the major changes that were made to highlight the general undertone of the class and their reception to the changes in the material.

4.3.1 List and For Loops

With the changes that were implemented, we wanted to deepen student understanding of indexing as it relates to lists and for loops as well as strengthen their ability to explain these concepts to others. The student responses to the pre-class assignments and their conversations in class indicate they grasp the concept of an index as it relates to retreiving a single value from a list, but they do not follow the extension of this process in for loops. For example, their knowledge of using 'list[0]' to obtain the first element in the list is clear. Further, when asked to explain the idea of indexing to a non-coder, many students were able to use descriptive langauge referring to the index as a label or location of an item in the list. Some students even used analogies to aid their thoughts. However, translating this idea to use indexing in a for loop with a list is not as clear.

After using the gifs to demonstrate the differences between looping by item and looping by index, we asked students to describe any differences they saw as well as implement the two types for a given task. Most students were not able to articulate any major differences between the two types and had trouble implementing two different versions of the for loop. While the two types arguably accomplish the same broad task, there are differences syntactically and in usage. This is the crux of a larger problem. Students assume the two different styles are identical as they return the same information, so they default to the for loop by item as it contains less syntax to understand. This is further exacerbated by built-in Python functions such as enumerate that allow them to access the index and item together in a loop by item format. While it may be challenging for students to articulate why one for loop might be used in a certain situation over the other when they initially learn it, there are still differences in the code syntax that should be apparent to students.

Even with this in mind, I would say that the inclusion of the gifs and the question were successful. While it is not completely apparent in their responses in the pre-class assignment, it did result in many conversations with students about the topic. By asking them if there was a difference, it indicated to them that there should be, even if they did not discern the difference yet. This reaction makes me curious about their responses to the survey. We did not specify the type of for loop in the survey. Since the students have a strong grasp on the for loop by item, I wonder if this is the reasoning for the high amount of confidence we saw from many of the students. It would be interesting to write this question more clearly and analyze if there are any differences between the student responses for the different types of for loops.

We also asked them to explain their thought process in completing a question that required the use of multiple for loops, but most students skipped these types of questions. This continued throughout the semester. When they did complete them, they often used jargon specific to coding and avoided deeper explanations.

While students still struggle with indexing related to lists and for loops as in previous semesters, the modifications to the course materials challenged them to think about these concepts in ways they wouldn't have otherwise. By asking them about the differences in the for loops, they were pointed towards noticing that a difference does exist, even if they weren't quite sure what it was yet. By asking them to explain the idea of an index, it encouraged them to think about how the seemingly small idea contributes to many larger coding tools. With that said, I think a bit more work could be done to increase student thinking in these assignments. For example, we need to create more situations where students are required to loop by index as well as give them practice explaining these implementations to others. Through doing so, and talking to instructors along the way, it will challenge their own understanding.

4.3.2 Data Analysis

The changes in the data analysis course materials appeared to strengthen student undestanding of retrieving information from a data frame using indices, but did not aid in their ability to explain their process, mostly due to lack of completion.

In the pre-class assignment, when students were asked to practice their skills with indexing a data frame, nearly all students were able to complete the task with success, but did not explain what information they were trying to

access or why they used the tools they chose. While there is chance students did not complete these questions due to lack of comprehension, it is more likely that they did not complete them to save time, especially considering they completed the coding tasks with success.

This type of success continued throughout the in class assignments. It was refreshing to see students refer to the visual aids in class to discuss their ideas with fellow classmates and instructors. Further, these tools seemed to give students a better visual of data frames in general. Compared to previous semesters, there was a shift in the types of questions I received related to data retrieval tasks. Student questions were more targeted. They were asking about specific syntax to complete a certain task, rather than broadly asking about how to start thinking about completing the task. It gave students the framework for understanding the process they needed to take, while only struggling slightly with the code to complete that task.

Unfortunately, the modifications throughout the pre-class and in-class assignments to guide students to explain their thinking were not met with success. This continued in the homework as well as for similar types of questions regarding pseudocode (writing the main ideas of an algorithm without specific coding language syntax) and results from data analysis. Students often struggled to explain their ideas in the level of detail that is needed to share their ideas with someone who is not familiar with the jargon. While this is not a simple feat, students do not take advantage of the chances in the pre-class and in-class assignments, and thus struggle to perform well on this style of question on homework.

Overall, the modifications that were made seemed to increase student understanding of tools available to them to access information from a data frame as well as provide them with visuals to refer back to at any time. However, I have learned that asking students to write down their thinking process in an assignment that does not get marked for accuracy is ineffective in getting students to explore this practice. As instructors, it is necessary to create these experiences verbally in class, so students learn the value of this skill and continue to develop it through various written assignments.

4.3.3 Array Usage

Since two-dimensional arrays are deeply related to both lists and data frames, specifically obtaining information from them using indexing, it is helpful to study student understanding of arrays to gather further information on their perception of list and data frames.

When prompted, students were able to describe how two-dimensional arrays were related to things like matrices, data frames, and nested lists with a few additionally providing great analogies. In general, students were more successful in noticing the similarities of the row representations between data frames and two-dimensional arrays rather than the column representation. About half of the students accurately described that the general framework was the same, but the syntax was slightly different. In addition, they recognized the similarities in information retrievel from both types of objects, specifically the similarities of slicing and iloc [1] /loc [2].

Further, with the aid of the visual representations, the students were able to distingiush between the index and the value of an object in a two-dimensional array more clearly than in previous semesters and more clearly than they did with lists earlier in the semester. This was crucial to understanding a few of the in-class assignments. Students needed to access the neighbors of a given object and with the visuals, students seem to have a better grasp of what the neighbors of a cell were, which they showed by actively discussing with their groupmates. While the students were still challenged with obtaining the neighboring values, many more students more clearly understood the task and once again had the tools to ask more targeted questions of the instructors. This is a large improvement from previous semesters where students struggled to grasp what they were trying to acheive. As they worked through the problem, they were able to more clearly conceptualize the difference between the value and the index of the array element.

Again, students were asked to explain their thinking through the process of constructing the forest fire. This semester,

[1] Pandas iloc documentation: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.iloc.html

[2] Pandas loc documentation: https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.loc.html

the assignment was spread over two class periods. At the end of day one, students were asked to articulate the next stages in their progress which they were able to do with success. Many were not sure what their ideas would look like in code, or at least did not write that, but they were able to describe what they had done in the first day and how that was going to be used in the second day. However, at the end of the second day, students were asked to explain the entire process to a non-coder and this was not as successful. Many students were broad in their explanations and used language specific to the code them implemented.

As with the previous sections, the gifs gave students the visualizations they needed to conceptualize various coding tasks. With the visuals, they were able to discuss potential solutions with their groupmates and the instructors in a more detailed manner, rather than struggling to comprehend the problem statement.

5 Limitations

We discuss a few limitations relating to this work and the related conclusions. The biggest limitation we faced was lack of student response rates. Many of the exploratory and justification questions that required student explanations beyond code were simply ignored in the Pre-Class and In-Class assignments. This made it challenging to fully assess if the students' comprehension of the material had increased with the inclusion of the visuals and guided questions. We faced a similar issue with the survey. Since we only used the responses of students who had completed the survey at both points of the semester, we had a sample size of 23 out of 41 students that completed the class. The small sample size, in both instances, made it very challenging to gauge the changes in student understanding from semester to semester, as well as conduct formal statistical tests about their confidence level differences within the semester. While we are able to make some claims about the students in this class, it is challenging to broaden these claims.

Related to the previous limition, in the interest of receiving more student responses to the survey, we simplified the amount of questions as well as the complexity of the questions. Initially, there were more open-ended questions related to completing a computational task and explaining the process of doing the task. We transitioned to strictly Likert scale questions, which was still helpful in gathering information on student confidence levels, but did not allow us to be as precise in our questions.

Additionally, the course was taught in a primarly in-person format with the option to join virtually. The other semesters that I taught the course were completely virtual. With this in mind, the results of our study might be impacted by the known differences of learning content online as opposed to in the classroom. Further, it might have affected my own perception of student understanding as it was easier to decipher group dynamics and general understanding in the classroom rather than through video call.

6 Conclusions

In this section, we highlight our main findings as well as mention some changes we would implement if this study were to continue.

Based on the survey as well as the completion of course assignments, students seem to have a grasp on indices as it relates to lists. Many are able to use descriptive langauge and create analogies for indices. However, there is a disconnect in for loops. Students report a moderate to high confidence to explaining for loops, especially to fellow classmates, but many students struggled with using indexing in for loops and differentiating between the two types. We note a few possible reasons for this difference. (1) Students may not be considering indices as a component of for loops in the survey response, especially since it is not stated explictly. (2) Students may feel comfortable with looping by value and thus, overestimate their confidence with all for loops. (3) This survey was given many weeks after students first learned for loops, and thus, they might understand more than they did at the time of completing the course assignments.

Secondly, in contrast to the reported low confidence in data analysis tasks, students were in fact more resourceful in class and asked more targeted questions regarding these types of tasks than in previous semesters. While

the larger data analysis questions still challenged students, they were stronger in smaller components of these larger tasks, such as data retrieval. Again, it would be interesting to analyze what students considered a data analysis task for the survey question and how heavily they considered indexing in relation to the task.

Embedding multiple visualizations throughout the class seemed to increase student understanding throughout the semester. The connections between arrays, lists, and data frames seemed more clear to students, specifically in relation to information retreival. Additionally, each of the visualizations gave students a conceptual representation of the code tool that they could continuously refer back at any time.

With this in mind, I do think these connections could be made more clear throughout the course material. I think we can place more emphasize on indexing in relation to for loops and use this description continuously with data frames and two-dimensional arrays. The repeated use of the visualizations through explanations with teaching assistants and faculty instructors can help stregthen these ideas in students. While some of the instructional staff used these kinds of visualizations previously, embedding them directly into the course materials ensures that all students have access to them and can use them to ask further questions, deepen their understanding, and communicate with their classmates.

Through completing the analysis on this project, I found myself having more questions that I did answers. If I were to use the survey in course again, I would adjust the wording of the questions to account for the specificity that I wished to study. Further, it would have been more beneficial to release this survey multiple times throughout the semester. By retrieving student responses every few weeks, their initial answers would be closer to when they first learned the coding tool and their changes in confidence levels would be easier to associate with a pointed time in the semester.

Related to the previous statement, due to the lack of responses, I struggled to capture how students would explain their thinking to another student or non-coder. Many did not take the time to complete these explanatory tasks, and thus, made it challenging for me to quantify their confidence levels. It might be fruitful to have a verbal component to the explanatory questions within the in-class assignments. For example, we could request that they chat with their groupmates, another group, and/or an instructor. Again, while some instructors do this, formalizing and adding structure to it in the in-class assignment might help it happen in all classrooms.

All in all, the modifications that were made to the CMSE 201 curriculum with the visual aids and guided conceptual questions seemed to strengthen student knowledge in comparison to previous semesters. I received much positive feedback from other instructors in the course based on my changes, and noticed an increase in clarity in student understanding. I am looking forward to asking these sorts of questions in future courses that I teach and gathering information on student perception, so that I can improve my course materials.

7 Mentor Contribution

I want to note that my mentor, Dr. Devin Silvia, was supportive of my research question and plans as well as provided many ideas for data analysis and visualization. He provided feedback on the changes that I made to the class assignments. Additionally, he shared many resources for visualizing the survey data that I collected as I did not have experience creating plots for this type of data before. Finally, he provided great feedback on this final written report. I am very thankful for his thoughts and support during this work.

References

Leema K Berland and Katherine L McNeill. For whom is argument and explanation a necessary distinction? a response to osborne and patterson. *Science Education*, 96(5):808–813, 2012.

Yuliya Cherenkova, Daniel Zingaro, and Andrew Petersen. Identifying challenging cs1 concepts in a large problem dataset. In *Proceedings of the 45th ACM technical symposium on Computer science education*, pages 695–700, 2014.

National Research Council. *A framework for K-12 science education: Practices, crosscutting concepts, and core ideas*. National Academies Press, 2012.

K Patricia Cross. *Classroom Assessment Techniques. A Handbook for College Teachers.* Jossey-Bass, Incorporated, 1993.

Simon P Davies. Expertise and display based strategies in computer programming. *PEOPLE AND COMPUTERS*, pages 411–411, 1993.

Sue Fitzgerald, Beth Simon, and Lynda Thomas. Strategies that students use to trace code: an analysis based in grounded theory. In *Proceedings of the first international workshop on Computing education research*, pages 69–80, 2005.

Michail N Giannakos, John Krogstie, and Nikos Chrisochoides. Reviewing the flipped classroom research: reflections for computer science education. In *Proceedings of the computer science education research conference*, pages 23–29, 2014.

Di Gong, Harrison H Yang, and Jin Cai. Exploring the key influencing factors on college students' computational thinking skills through flipped-classroom instruction. *International Journal of Educational Technology in Higher Education*, 17(1):1–13, 2020.

Marwah Ahmed Halwani, S Yasaman Amirkiaee, Nicholas Evangelopoulos, and Victor Prybutok. Job qualifications study for data science and big data professions. *Information Technology & People*, 2021.

Paul W Irving, Michael J Obsniuk, and Marcos D Caballero. P3: a practice focused learning environment. *European Journal of Physics*, 38(5):055701, 2017.

Cruz Izu, Cheryl Pope, and Amali Weerasinghe. Up or down? an insight into programmer's acquisition of iteration skills. In *Proceedings of the 50th ACM technical symposium on computer science education*, pages 941–947, 2019.

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

Craig S Miller and Amber Settle. Mixing and matching loop strategies: By value or by index? In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, pages 1048–1054, 2021.

Thomas L Naps, Guido Rossling, Vicki Almstrum, Wanda Dann, Rudolf Fleischer, Chris Hundhausen, Ari Korhonen, Lauri Malmi, Myles McNally, Susan Rodger, et al. Exploring the role of visualization and engagement in computer science education. In *Working group reports from ITiCSE on Innovation and technology in computer science education*, pages 131–152. 2002.

Juan David Ortega-Alvarez, Camilo Vieira, Nicolás Guarín-Zapata, and Juan Gómez. Flipping a computational modeling class: Strategies to engage students and foster active learning. In *2020 IEEE Frontiers in Education Conference (FIE)*, pages 1–4. IEEE, 2020.

Jonathan F Osborne and Alexis Patterson. Scientific argument and explanation: A necessary distinction? *Science Education*, 95(4):627–638, 2011.

Roy D Pea. Language-independent conceptual "bugs" in novice programming. *Journal of educational computing research*, 2(1):25–36, 1986.

David Perkins and Fay Martin. Fragile knowledge and neglected strategies in novice programmers. ir85-22. 1985.

Julia Phillippi and Jana Lauderdale. A guide to field notes for qualitative research: Context and conversation. *Qualitative health research*, 28(3):381–388, 2018.

Sawitree Pipitgool, Paitoon Pimdee, Somkiat Tuntiwongwanich, and Akan Narabin. Enhancing student computational thinking skills by use of a flipped-classroom learning model and critical thinking problem-solving activities: A conceptual framework. *Turkish Journal of Computer and Mathematics Education (TURCOMAT)*, 12(14):1352–1363, 2021.

Yizhou Qian and James Lehman. Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education (TOCE)*, 18(1):1–24, 2017.

Liam Rigby, Paul Denny, and Andrew Luxton-Reilly. A miss is as good as a mile: Off-by-one errors and arrays in an introductory programming course. In *Proceedings of the Twenty-Second Australasian Computing Education Conference*, pages 31–38, 2020.

Andrew Rudder, Margaret Bernard, Shareeda Mohammed, et al. Teaching programming using visualization. In *Proceedings of the Sixth IASTED International Conference on Web-Based Education*, pages 487–492, 2007.

Devin Silvia, Brian O'Shea, and Brian Danielak. A learner-centered approach to teaching computational modeling, data analysis, and programming. In *International Conference on Computational Science*, pages 374–388. Springer, 2019.