34th Australasian Association for Engineering Education Conference
3 - 6 December 2023 – Gold Coast, QLD

“I’m not a programmer, therefore I can’t program.” – Diverse perspectives of students’ difficulties with programming.

Vy Le; Lilly Borchardt: Wilson Kong; Marie Bodén
The University of Queensland
Corresponding Author Email: vy.le1@uqconnect.edu.au

ABSTRACT

CONTEXT

Learning to program is known to be a challenging and complex process and programming courses are notorious for high failure and drop-out rates. Programming is an inherently complex activity that requires a diverse set of cognitive skills and knowledge. Past literature has identified that students learning to program experience many difficulties such as understanding syntax and constructs, designing algorithms, writing programs, detecting, and handling errors, and many more. With several universities in Australia now teaching programming to all engineering students (e.g., The University of Queensland, Monash University), this presents a unique challenge to educators as these students may not share the same inherent affinity for programming as their programming-major counterparts would.

PURPOSE OR GOAL

Programming and computational thinking are becoming increasingly sought-after skills in graduates as we see digital transformations occurring across all industries. These trends have led to an influx of students from non-programming disciplines learning to program at university, such as engineering students who may not all go on to become programmers. Therefore, research into how universities can teach programming to engineering students effectively and how programming courses can continue to address the needs of students with diverse educational backgrounds and motivations is highly desirable. The present study’s aim is to understand teaching staff perspectives on the struggles of students learning to program and how staff perspectives align with the perspectives and expectations of students.

APPROACH OR METHODOLOGY/METHODS

The methodology will involve individual interviews with lecturers and head tutors involved in the delivery of two introductory programming courses at the University of Queensland. A questionnaire to gain insights into the struggles and expectations of students from those programming courses will also be distributed. Data from the interviews and questionnaire responses were analysed using NVivo, to identify overarching themes and concepts.

ACTUAL OR ANTICIPATED OUTCOMES

The results from the study have found three major themes and one minor theme: People-focused and practical help is valued and sought after; there is a steep learning curve and some difficulty keeping up with content; linking in-class content to assessment; and pre-content help. The findings provide us with insights for programming educators to better understand the challenges faced by students from non-programming backgrounds and guidance on how current teaching practices can effectively adapt to these issues.

CONCLUSIONS/RECOMMENDATIONS/SUMMARY

Conclusions from the study show that students from non-programming backgrounds face the same, if not more challenges as their programming-major counterparts. The courses are fast paced with a heavy load of new content and expectations on the course varies between students and course coordinators. Current teaching practices need to be able to adapt to the changing diversities in educational backgrounds and motivations of students learning to program at university.

KEYWORDS

Programming, teaching, learning, non-programmers, difficulties

Introduction

Being able to think and problem-solve like a programmer are becoming highly sought-after and marketable skills in engineering graduates (Yusoff et al., 2020; Terroso & Pinto, 2022; Vial & Bogdan, 2018), leading to a trend across universities where traditionally non-computer science disciplines are now required to study a compulsory introductory programming course as part of their program curricula. For example, all engineering students at the The University of Queensland and Monash University are now required to take at least one introductory programming course, regardless of their major.

Programming is an inherently complex activity that requires a diverse set of cognitive skills and knowledge (Yusoff et al., 2020; Ko, 2003), and programming education literature has identified it as an activity that many students find challenging to learn (Lahtinen et al., 2005; Nouri et al., 2020; Ozmen & Altun, 2014). Considering the increase in engineering students from non-programming backgrounds required to learn programming at university, further research into how traditional introductory programming education can continue to address the learning needs of students with diverse interests, motivations and educational backgrounds is highly desirable. Students from non-programming majors will have different motivations (Noshin & Ahmed, 2018) and perceptions of programming from programming majors, since they do not share the same inherent affinity towards programming (Terroso & Pinto, 2022; Ko, 2003). This study sought to gain further insights into the struggles and expectations of students learning introductory programming at The University of Queensland from both teaching and student perspectives. The courses we engaged with introduce students to Python programming, using lectures, tutorials and practical. The staff also offer individual help to students via a Teaching and Learning Centre, which is open to students every weekday. Assessments are made using a combination of assignments and examinations.

Methods

A mixed method design was used for this research project incorporating survey data (qualitative and quantitative) and interview data (qualitative). Qualitative methods were chosen to provide a richer and more in-depth analysis of student and teaching staff experiences in programming courses.

Participants

All participants worked or studied at The Engineering, Architecture and Information Technology Faculty (EAIT) at The University of Queensland (UQ). Research data was gathered from two groups of people, 51 students filled in an online questionnaire (see Figure 1) and 3 teaching staff (2 course coordinators and 1 lead tutor) were interviewed. The study had approved ethical clearance for both participating groups. The students were selected on a voluntary basis from the first-year introduction to programming courses offered to engineering and computing students. The teaching staff were contacted and asked for an interview, all staff volunteered to participate.

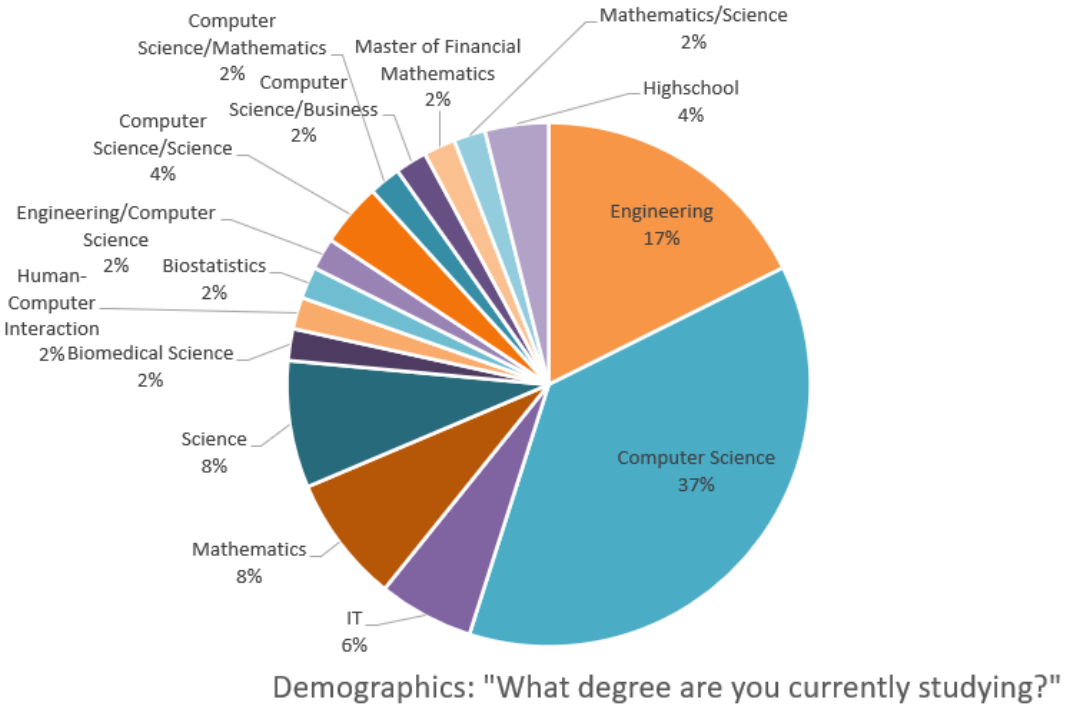

Figure 1. Demographic of students from the first-year programming course who responded to the questionnaire.

Data collection

Data was collected from participants using a questionnaire and semi-structured interviews. Student participants were asked to fill in an online questionnaire with 34 questions, including a mix of qualitative questions and free text responses. Participating Course Coordinators advertised the survey to their students in one of their lectures. Participation in the survey was voluntary and anonymous. The survey took approximately 15 minutes to complete, which included time for participants to sign off consent to participate in the research study.

Teaching staff were approached by the researcher and asked to participate in an interview. The interview questions were structured as semi-open to ensure we captured questions that we intended to explore but also not to miss important experiences from the teaching staff. After seeking signed consent from staff, the interviews took 40-60 minutes and were audio recorded. Participation in the interview was voluntary.

Data analysis

The surveys were analysed using word frequency and text search to complement a thematic analysis (Braun & Clarke, 2006) to identify student needs after coding open-ended questions using NVivo software. Analysis of the surveys started with researchers reading all the survey responses and noting potential insights. Open-ended questions were then coded and written up for discussion. Word frequency and text search were conducted using NVivo on other questions to supplement coding categories. This was then examined by researchers for insights and possible themes relating to the study's aims. The analysed data was then collated into themes and refined by researchers with qualitative data analysis experience.

Findings

Our study found that students' experiences and needs while learning programming could be categorised into three main themes and one minor theme:

1) People-focused and practical help is valued and sought after
2) There is a steep learning curve and some difficulty keeping up with content

3) Linking in-class content to assessment
4) Pre-content help (minor theme)

1. **People-focused and practical help is valued and sought after**

People-focused help came from teaching staff (including live-coding in lectures and tutorial and practical sessions), help-centre tutors, and their peers. Although help was widely available, especially from the tutors in the help centre, students' lack of confidence can be a barrier to accessing this help. The main benefit identified from in-person help was the ability to show/see someone else's thought process to problem-solving when coding.

Course tutorials and practicals were valued by most students (n=45) as good learning resources, in-part due to the expertise of the tutors. Students noted that it was sometimes hard to get time with tutors due to the constraints of time and ratio of tutors to students:

> *"The tutors were easily the most valuable resource, but it was often difficult to get access to them (especially around assignment time)."* (Student 34)

However, students also noted how difficult it was to seek help due to the perceived knowledge imbalance between them and their tutors:

> "*Asking for help. It feels a bit demoralising when I get stuck and am constantly told by tutors that although they can see I can have a grasp on a topic they don't understand how I am not "getting it*" (Student 16)

Teachers spoke about how it was difficult to get students attending help sessions:

> *"…they often say "I haven't done any coding before" and they feel very lacking in confidence. So they will often not even come to get tutor assistance for quite a while. And sometimes they'll say things like, "I just didn't even know where to start to ask questions". So they're very reluctant to start doing the actual work. … You know, for them, it's kind of like this mountain they just don't feel like they can get over." (Teaching Staff T)*

Students also sought more connection with their peers for accessing help:

> "*Better ways to connect with other students so 1) beginners maybe feel less alone and 2) to get help from more experienced peers"* (Student 15)

Over half of students surveyed listed 'in practicals', 'in tutorials', and/or 'with other people' as their most comfortable way of learning to code. Almost all students (n=48) listed practical or people-focussed teaching methods as their preferred method (tutorial exercises, practical sessions, working through an example problem as a class, learning from a friend, and/or tutor input). Students noted they wanted to see the problem-solving process:

> *"Tutorial solutions make sense when I read it, I just don't know what goes through your head sometimes to come up with it."* (Student 49)

Teaching staff also noted this and explained the benefit of in-person help was more than just getting a solution, but an opportunity to see the way tutors think about a programming problem:

> *[After going to the help centre] "I mean, first of all, they solved the problem, they'll say, 'Look, I can't get this program working'. So that relieves a lot of the stress, but additionally, they see the tutor, the way they think. You know, they'll say, well, 'Let's put in a print statement here to see what's happening' and 'See, ohh look, this variable hasn't been defined anywhere. You just defined it somewhere else and you thought it would be defined in this, and would be available in this function but it's not." or "You're printing something, but you're not returning it.". And that's so they think aloud and the thinking aloud often helps the students, I believe. And the other thing is, it's the opportunity to get feedback, you know, and we all know that feedback is one of the key drivers of learning. And you can get one-on-one feedback in the help centre."* and *"...we started a help centre, basically a drop-in centre, and that's really helped. Once the students go there, they get help from the tutors. They not only get help from the tutors, but they see the tutors modelling good practice."* (Course Teaching Staff T)

In-person help is also valuable for learning debugging practices. Debugging (including handling errors, finding bugs in your own code, and debugging) was the second most common programming challenge listed (see Figure 2) by students surveyed (n=38). It was also noted by teaching staff as one of the more challenging concepts students had to learn.

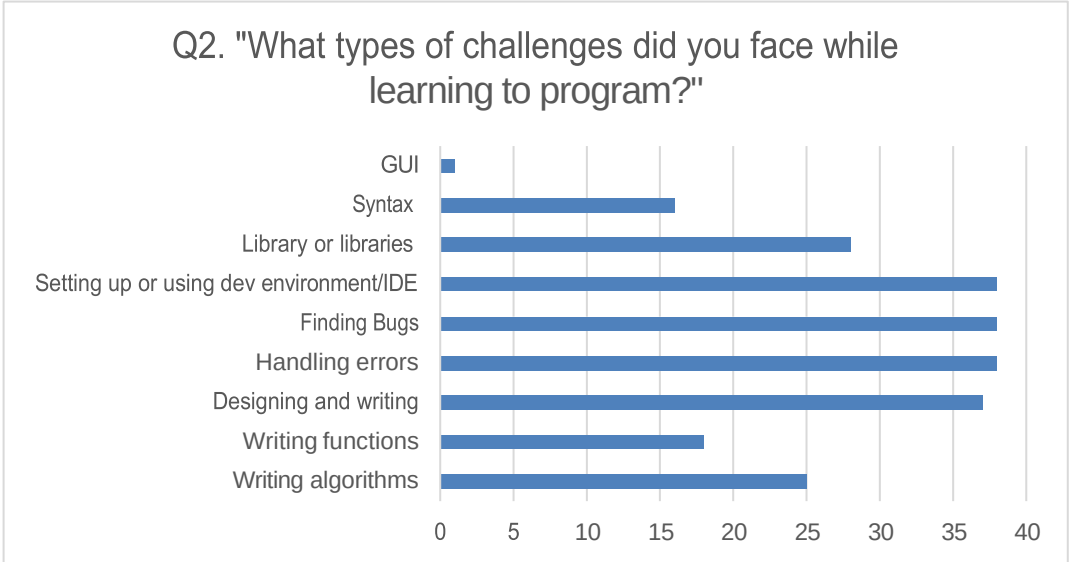

Figure 2: The challenges the students had when learning to program.

Staff noted de-bugging was a difficult concept to teach and they recommended the best learning was to practise as much as possible:

> *"It's more than to do with the problem-solving of it. You might have three different things wrong with your code. They saw one of them. The other two are still there, but then because it still doesn't work, they're like 'Ohh whatever I just did, didn't change anything', so might as well refer them back. So they actually reintroduced the error back. And then they're like, 'Oh, it still doesn't work'. And I'm like, 'Of course, it doesn't work, you just put the error back in' and then, you know, you help them get rid of that error again. And then it still doesn't work because you still have those two errors and then they're like, 'but you didn't help' and I'm like, 'no, like we now have a different error'. So I think people are very afraid to be, to try when something goes wrong because I don't know some kind to mentality of like, it has to work right away kind of thing…But I think a lot of the times when they start out coding they want to be right, right away."* (Course Teaching Staff C)

Another in-person strategy utilised by teaching staff is live coding exercises:

> *"I get a lot of feedback about my live coding in my C class. They're always like, 'That is the most helpful thing' to watch me program from literally nothing, and having done little to no preparation beforehand, because they don't know how to turn nothing into code."* (Course Teaching Staff N)

2. Steep learning curve and the difficulty keeping up with content

A steep learning curve and the subsequent difficulty of keeping up with new, weekly content was noted by both students and staff as a significant challenge when learning how to program at university. Students sought more content with explanations to see how to get from the problem to the solution. Additional videos from varied sources were suggested by students to explain complex concepts from multiple perspectives. The students found data structures the most challenging programming concept to learn (see Figure 3.) which is introduced in the second part of the course.

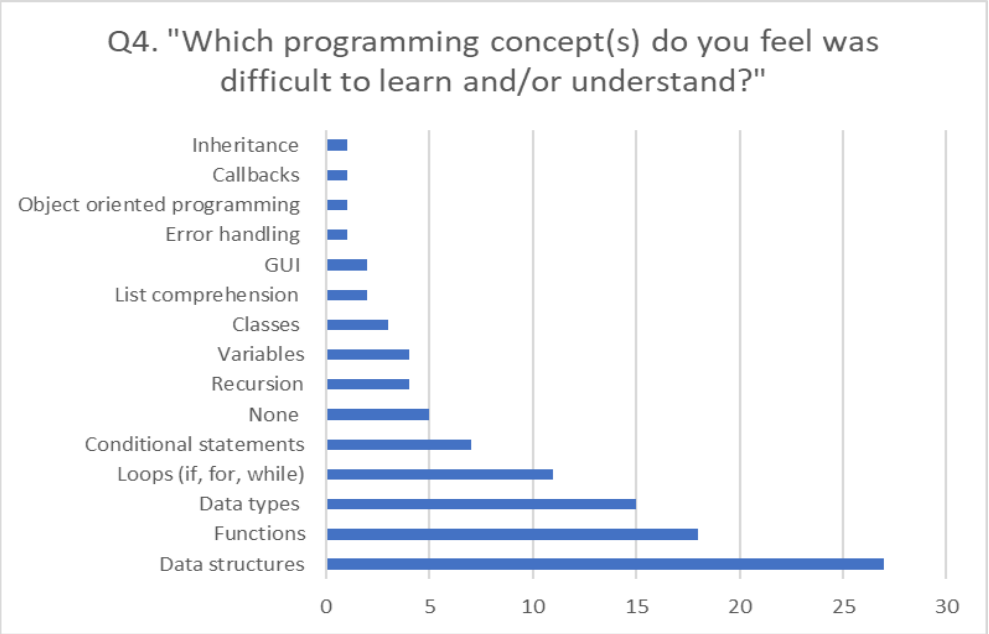

Figure 3: Data structures, functions and data types where ranked the top three top most difficult programming concepts to learn by students in the survey.

Confusion with programming concepts at the start of the semester tended to snowball as more advanced topics were introduced but students took opposing views on this. Some students found that while it was very hard at the start, by the end it became easier. Others wanted more compulsory weekly tasks (e.g., in the tutoring system) to motivate them to stay on track. Then again there were students who found the volume of work and time required as a beginner to complete all the necessary weekly activities already hard to get through and near impossible to keep up to date with:

> *"I feel like the concepts themselves are okay, it's just putting them into practice that can sometimes feel impossible (keeping up with lecture exercises, tutorial exercises, and [tutoring software]) on top of assignments. Although I recognise the value, it is easy to fall behind due to the amount of time it can take beginners to complete them."* (Student 7)

Students expected a slower start and pace of content for beginners and a smaller step up from high school to university programming content. They also found the course title inclusion of 'introduction' to be incongruent with the difficulty of the content. One teaching staff gave an example of the speed of learning programming at university in relation to learning calculus:

> *"It would be like trying to teach people calculus and starting them with counting. Like, let's learn the numbers and by the end of this, we'll do integration. That's kind of how I feel like the [introductory programming], right now." (Course Teaching Staff N)*

Teaching staff recommended practice, above all else as the best way to deal with the difficulties of learning to code for the first time:

> *"... do lots of practice. Go to the help centre as often as you can and start everything early. And it really is practice and feedback. So the practice means you start early and the feedback if you really are struggling, go to the help centre where you can get feedback." (Course Teaching Staff T)*

Students and teaching staff both reported the difficulty students had with asking for help, not knowing what questions to ask to get started:

> *"Especially in the first few weeks of going to tutorials I felt the learning curve was way too big that I sat there clueless to the point that I didn't even know what questions to ask."* (Student 25)

The steep learning curve also impacted the next theme relating to assessment, when students are not keeping up with content, it made approaching assessment difficult: "I felt like I was always behind and it made the tutorials and assignments difficult." (Student 46)

3. Linking in-class content to assessment

While students could understand some concepts in class and tutorials, they noted that the assignments appeared very different (and harder) than the content they were taught:

> *"I feel that there is some gap between the knowledge in class and the use of assignments. I have spent extra time outside of class to learn and supplement some basic knowledge, but still find it difficult to apply the knowledge to practical applications." (Student 34)*

Students asked for greater difficulty in practice questions before moving to the assignment and additional resources (e.g., videos) addressing assignment-specific problems. Teaching staff provided two possible reasons for why students feel like this: the timing of assessment and content, and creativity of thinking. Teaching staff noted that assessment was released before all the required content was covered:

> "*There's an expectation from the students that as soon as the assignment's released, they can finish it. But that's not the case, because they haven't learned the content. So I don't know where that comes from either. It could be high school, I don't know, like high school, I think is, once you learn the content, the assignment comes out."* (Course Teaching Staff C)

They also noted the requirement of creativity in thinking required when learning programming. Learning to program is not learning and regurgitating facts or formulas:

> *"Creativity is a problem of a lot of students who just want to follow a recipe, and it's like we can't give you a recipe for writing an algorithm"* (Course Teaching Staff N)

Again, practice was also noted as a very important part of learning programming, and one teaching staff likened it to learning how to play piano:

> *"It's like I'm a music teacher, this is what I tell them. I can show you how to press the keys on the piano, I can show you how to read music, but attending and rewatching me playing the piano is never going to imbue upon you any type of skill at playing the piano… the good students are always begging for more practice. And that's really what separates the people who can't do this from those who can't, the ones who are willing to practise and be confused."* (Course Teaching Staff N)

4. Pre-content help

While this relates to the steep learning curve, this theme looks at student needs before the content begins. Students requested specific help or step-by-step guide to set up their development environment and appropriate software (e.g., Visual Studio Code). While set-up was not identified as the most difficult problem for most, it was identified as a problem for most students surveyed (n=38).

Conclusions and Further Research

In conclusion, our research study found that students and staff agree the pace of teaching programming is intense and that there is a steep learning curve to understanding many of the concepts. Students said they appreciate hands-on and people-focused help but teaching staff found it difficult to make students attend practical sessions. Students also found it difficult to relate the in-class content to assessments and teachers said that students needed more practice to fully understand the programming concepts taught. The study also found there is a mismatch in expectations between students and teaching staff. Suggestions for further research on teaching introductory programming to engineering students from diverse educational backgrounds include how much new content is reasonable to teach in a first-year programming course; how to improve teaching of difficult programming concepts; the differences between engineering disciplines (e.g., electrical, mechanical, civil) in students' motivations for and perceptions of learning programming at university; and whether AI should be part of the teaching and practices of learning a first programming language or will this be seen as a risk to the students' learning and understanding of the subject.

References

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, *3*(2), 77–101. https://doi.org/10.1191/1478088706qp063oa

Ko, A. J. (2003). Preserving non-programmers' motivation with error-prevention and debugging support tools. IEEE Symposium on Human Centric Computing Languages and Environments, 2003. Proceedings. 2003, 271–272. https://doi.org/10.1109/HCC.2003.1260245

Lahtinen, E., Ala-Mutka, K., & Järvinen, H.-M. (2005). A study of the difficulties of novice programmers. *Annual Joint Conference Integrating Technology into Computer Science Education: Proceedings of the 10th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education : Caparica, Portugal*; 27-29 June 2005, 14–18. https://doi.org/10.1145/1067445.1067453

Noshin, J. A., & Ahmed, S. I. (2018) Teaching Programming to Non-Programmers at Undergraduate Level. *International Journal of Engineering and Management Research, 8*(3), 191-194. doi.org/10.31033/ijemr.8.3.26

Nouri, J., Zhang, L., Mannila, L., & Norén, E. (2020). Development of computational thinking, digital competence and 21st century skills when learning programming in K-9. Education Inquiry, 11(1), 1–17. https://doi.org/10.1080/20004508.2019.1627844

Özmen, B., & Altun, A. (2014). Undergraduate Students' Experiences in Programming: Difficulties and Obstacles. *Turkish Online Journal of Qualitative Inquiry, 5*(3). https://doi.org/10.17569/tojqi.2032

Terroso, T., & Pinto, M. (2022, June 2-3). *Programming for Non-Programmers: An Approach Using Creative Coding in Higher Education* [Paper presentation]. Third International Computer Programming Education Conference, ICPEC 2022, June 2-3, 2022, Polytechnic Institute of Cávado and Ave (IPCA), Barcelos, Portugal. https://doi.org/10.4230/OASIcs.ICPEC.2022.13

Vial, G., & Bogdan, N. (2018). *Teaching Programming to Non-Programmers: The Case of Python and Jupyter Notebooks* [Paper presentation]. Thirty ninth International Conference on Information Systems, San Francisco. https://aisel.aisnet.org/icis2018/education/Presentations/1

Yusoff, K. M., Ashaari, N. S., Wook, T. S. M. T., & Ali, N. M. (2020). Analysis on the requirements of computational thinking skills to overcome the difficulties in learning programming. International Journal of Advanced Computer Science & Applications, 11(3), 244–253. https://doi.org/10.14569/IJACSA.2020.0110329

Acknowledgements

We would like to thank the EAIT students and staff from the first-year programming courses who participated in this research study.