Motivation and vision: Alan Perlis said, *"Beware of the Turing tar-pit in which everything is possible but nothing of interest is easy"* [1]. Unfortunately, much of an introductory programming course can feel like the "Turing tar-pit." These courses focus on building generalizable programming knowledge by focusing on a language's syntax and semantics [2]. Assignments often involve close tracking of code's execution, typically in the context of 'toy' problems [3]. These activities can create a "perfect storm" for discouraging learners, because they require high cognitive load [4], leading to a low expectation of success [5], while also being disconnected from meaningful code, resulting in low value for the task [5].

With the increasing popularity of programming learning, there is a growing number of learners who do not plan to be software engineers [6]. As a result, there is a greater need to connect learning tasks to outcomes in the real world. Future software developers know that a deep understanding of how programming languages works is part of their professional practice. However, for learners who plan to become end-user programmers [7] (e.g. graphic designers) or conversational programmers [8] (e.g. managers), the reason to learn the details of code's syntax and semantics is less clear [9, 10]. These learners are more likely to be female, a population that has lower self-efficacy in programming learning [11].

New technologies are needed to create programming learning activities where novices can quickly and easily create authentic and meaningful code. Many learners may not care about exactly how a programming language works, but they do care about what code can achieve for them. A new approach that focuses on code's *purpose* rather than its *semantics* is required to help these learners remain engaged with programming learning.

Research Goals: To bridge this gap, **I will develop and evaluate a** "***purpose-first" programming learning environment***. In order to scaffold the code reading, writing, and debugging capabilities of novice programmers as they work with authentic code, I will provide information, structures, and real-time feedback that highlight how code achieves goals in a domain. This system will allow learners to quickly create and understand real-world code, without the need for a deep knowledge of programming language semantics.

This learning environment will decrease the cognitive cost necessary to complete meaningful programming tasks, while providing supports that can contribute to further learning. I will draw on the theory of programming plans [12] in order to design scaffolds (assistive mechanisms [13]) that clarify code's purpose. A programming plan is a commonly used code pattern, associated with a goal that code achieves. In "purpose-first" programming tasks, learners will always work with authentic code patterns used by practitioners. This "chunking" [14] of meaningful code is a scaffold that allows reasoning about code to occur more easily and aligns with what we know about the ways experts understand code [15]. Instead of writing code from scratch, purpose-first programmers will tailor and combine plans. Instead of tracing code line-by-line to debug or understand how code works, learners will infer code behavior across larger chunks of code.

The primary outcome of purpose-first programming is increased self-efficacy [16] for programming learners. Current approaches to programming learning result in lower self-efficacy for females than males [11], and studies of underrepresented minority groups in STEM have found lower general academic self-efficacy compared to majority groups [17]. Low self-efficacy is associated with lack of persistence in computing [18]. With purpose-first programming, learners see programming accomplishments quickly, which should increase self-efficacy [16].

Intellectual Merit: This work connects cognitive theories to theories of motivation in order to present a new approach to programming learning called purpose-first programming. I will build a novel programming learning intervention that allows me to extend theory to practice, and demonstrate its effectiveness. I will develop design implications for future research on supporting novice programmers as they engage in meaningful and authentic programming experiences. My findings will generate evidence for the impact of purpose-first programming on student self-efficacy and learning. These advances will improve computing education, create a new pathway to programming learning, and advance our knowledge of how students understand programs and programming.

Broader Impacts: This work opens new avenues to computing, inviting a broader group of learners to engage with programming, particularly those who are less likely to find code semantics interesting. By building a computational learning environment, I will be able to reach a large number of learners with a multifaceted approach that addresses the cognitive and motivational factors that can derail programming learning. I will improve learners' self-efficacy, a factor that disproportionately affects groups underrepresented in computing. During the development of purpose-first programming, I will work primarily with students at a non-R1 university; for example, DePaul University or Northeastern Illinois University, where African American constitute 9% and 10% of the student body, and Hispanic students constitute 15% and 36% [19,20].

Foundations: Purpose-first programming is a method of learning and performing programming tasks in a specific domain (e.g. web scraping), where tasks are facilitated with information, structures, and feedback that highlights how code achieves goals in that domain. In my thesis work, I laid the foundations for purpose-first programming by creating a curriculum for learning five frequently-used web scraping code patterns with "purpose-first" scaffolds [13]. These scaffolds include highlighting of "slots" where changes to each pattern should occur, natural language description of the goal each pattern achieves, and subgoal labels [21] that describe how parts of the pattern contribute to the overall goal.

Overview: In my postdoctoral work I will develop an interactive learning environment called *PURPLE* (PURPose-first Learning Environment), where novice programmers complete purpose-first programming. I will design and build *PURPLE* in multiple stages. First, I will use an off-the-shelf interactive learning platform to build *PURPLE* v1.0. Next, I will perform a formative study using *PURPLE* v1.0 to inform design decisions about new features. Then, I will use an iterative design process to build *PURPLE* v2.0 that incorporates real-time feedback mechanisms to support purpose-first programming. Finally, I will evaluate the effectiveness of *PURPLE* v2.0 for building self-efficacy and programming skills among novice programmers.

Milestone 1: *Building PURPLE v1.0: a purpose-first programming curriculum in a new context (Jan-Feb 2021)*
To create an initial version of an interactive learning environment for purpose-first programming, I will compile a set of programming patterns and associated tasks, in a new domain (e.g. creating shapes in Processing, or analyzing social media data with Pandas). Building on my experience identifying common patterns in the domain of web scraping with BeautifulSoup, I will identify patterns and annotate them with the purpose-first scaffolds I developed in my thesis work. I will then add the tasks to a Runestone interactive ebook [22], which will facilitate log data collection as well as enable basic interactive features, like textual feedback when a student selects a wrong answer.

Milestone 2: *Formative study on PURPLE v1.0 to inform design changes: Understanding problem-solving process and self-efficacy on tasks with purpose-first programming scaffolds (Mar-Aug 2021)*
In a series of formative studies, I will investigate the affordances that my initial set of plan-based scaffolds in *PURPLE* v1.0 provide for completion of programming tasks, as well as learners' affective experiences when completing tasks with extra support. By performing think-aloud studies [23] of novice programmers' code writing, debugging, and understanding when executing tasks with purpose-first scaffolds, I will gain an understanding of how novices use information about code's purpose to solve problems, and where they struggle to make use of the assistance provided. I will also interview students about their feelings of self-efficacy after completing tasks. I will run three studies with novice programmers, each focusing on a different programming task: debugging, code writing, and code explanation.

Milestone 3: *Iterative Development of PURPLE v2.0, a purpose-first programming environment with real-time feedback (Sept 2021-Aug 2022)*
Next, using the findings from Milestone 2, I will investigate how to design real-time feedback in *PURPLE* v2.0 to support novice programmers as they complete programming tasks. I will design new interactive feedback that extends the functionality of the initial scaffolds, to allow for an easier completion of programming tasks. My designs will modify the existing purpose-first scaffolds to add real-time feedback. For instance, while *PURPLE* v1.0 only provides static text describing what a code pattern achieves in general terms, *PURPLE* v2.0 may change that text dynamically to become more specific as a learner fills in slots in a code pattern. To develop *PURPLE* v2.0, I plan to either extend the capabilities of Runestone with new Javascript components, or create a new standalone system. I will develop three interactive, real-time feedback features. As I develop each feature, I will run a user study to evaluate the effectiveness of this feedback for improving novices' ability to complete programming tasks.

Milestone 4: *Evaluating the effectiveness of PURPLE v2.0: self-efficacy and learning (Sept-Dec 2022)*
Finally, I will evaluate *PURPLE* v2.0's ability to increase self-efficacy among new programmers, as well as to prepare learners to complete tasks without purpose-first scaffolds. At a local college, I will offer a workshop involving use of *PURPLE* v2.0 for students interested in learning programming, but who have no experience in the domain of the curriculum. Using a validated survey instrument (e.g. [24]), I will measure these learners' self-efficacy for programming before and after the workshop. At the end of the workshop, I will assess learners' ability to complete tasks similar to those in *PURPLE* v2.0, but without the purpose-first scaffolds. These findings will help us understand if purpose-first programming is a valuable early learning tool.

References

1. Perlis, A. J. (1982). Special feature: Epigrams on programming. ACM SIGPLAN Notices, 17(9), 7-13.
2. Sahami, M., Danyluk, A., Fincher, S., Fisher, K., Grossman, D., Hawthorne, E., ... & Cuadros-Vargas, E. (2013). Computer science curricula 2013: Curriculum guidelines for undergraduate degree programs in computer science. Association for Computing Machinery (ACM)-IEEE Computer Society.
3. Sorva, J. (2013). Notional Machines and Introductory Programming Education. Trans. Comput. Educ., 13(2), 8:1–8:31.
4. Cunningham, K., Ke, S., Guzdial, M., & Ericson, B. (2019, July). Novice Rationales for Sketching and Tracing, and How They Try to Avoid It. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 37-43).
5. Cunningham, K., Agrawal Bejarano, R., Guzdial, M., & Ericson, B. (2020). I'm not a computer: How identity informs value and expectancy during a programming activity. Proceedings of the 2020 International Conference of the Learning Sciences. Memphis, TN, United States.
6. Camp, T., Adrion, W. R., Bizot, B., Davidson, S., Hall, M., Hambrusch, S., ... & Zweben, S. (2017). Generation CS: the growth of computer science. ACM Inroads, 8(2), 44-50.
7. Ko, A. J., Abraham, R., Beckwith, L., Blackwell, A., Burnett, M., Erwig, M., ... & Rosson, M. B. (2011). The state of the art in end-user software engineering. ACM Computing Surveys (CSUR), 43(3), 1-44.
8. Chilana, P. K., Singh, R., & Guo, P. J. (2016, May). Understanding conversational programmers: A perspective from the software industry. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (pp. 1462-1472).
9. Dorn, B., & Guzdial, M. (2006, September). Graphic designers who program as informal computer science learners. In Proceedings of the second international workshop on Computing education research (pp. 127-134).
10. Wang, A. Y., Mitts, R., Guo, P. J., & Chilana, P. K. (2018, April). Mismatch of expectations: How modern learning resources fail conversational programmers. In Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (pp. 1-13).
11. Sylvia Beyer. 2008. Predictors of Female and Male Computer Science Students' Grades. Journal of Women and Minorities in Science and Engineering 14, 4 (2008), 377–409. https://doi.org/10.1615/JWomenMinorScienEng.v14.i4.30
12. Soloway, E., & Ehrlich, K. (1984). Empirical studies of programming knowledge. IEEE Transactions on software engineering, (5), 595-609.
13. Vygotsky, L. S. (1978). Socio-cultural theory. Mind in society.
14. Gobet, F., Lane, P. C., Croker, S., Cheng, P. C., Jones, G., Oliver, I., & Pine, J. M. (2001). Chunking mechanisms in human learning. Trends in cognitive sciences, 5(6), 236-243.
15. Ehrlich, K., & Soloway, E. (1984, February). An empirical investigation of the tacit plan knowledge in programming. In Human factors in computer systems (Vol. 16, pp. 113-134). Norwood, NJ: Ablex Publishing Co.
16. Bandura, Albert, W. H. Freeman, and Richard Lightsey. "Self-efficacy: The exercise of control." (1999): 158-166.
17. MacPhee, D., Farro, S. and Canetto, S.S. (2013), Academic Self-Efficacy and Performance of Underrepresented STEM Majors: Gender, Ethnic, and Social Class Patterns. Analyses of Social Issues and Public Policy, 13: 347-369.
18. Lin, G. Y. (2016). Self-efficacy beliefs and their sources in undergraduate computing disciplines: An examination of gender and persistence. Journal of Educational Computing Research, 53(4), 540-561.
19. DePaul University Marketing and Communications. Accessed June 17, 2020. https://offices.depaul.edu/university-marketing-communications/facts-stats/Pages/diversity.aspx
20. Northeastern Illinois University Institutional Research & Assessment. Student Profile Fall 2019. Accessed June 17, 2020. https://www.neiu.edu/sites/neiu.edu/files/documents/ysun2/Fall_2019_Enrollment_Fact_Sheet_0.pdf
21. Margulieux, L. E., Catrambone, R., & Guzdial, M. (2016). Employing subgoals in computer programming education. Computer Science Education, 26(1), 44-67.
22. Ericson, B. J., YeckehZaare, I., & Guzdial, M. J. (2019). Runestone Interactive Ebooks: A Research Platform for On-line Computer Science Learning.
23. Ericsson, K. A., & Simon, H. A. (1984). Protocol analysis: Verbal reports as data. the MIT Press.
24. Ramalingam, V., & Wiedenbeck, S. (1998). Development and validation of scores on a computer programming self-efficacy scale and group analyses of novice programmer self-efficacy. Journal of Educational Computing Research, 19(4), 367-381.