Codespec: A Computer Programming Practice Environment

Carl Haynes-Magyar
School of Information
University of Michigan
Ann Arbor, MI, USA
cchaynes@umich.edu

Nathaniel Haynes-Magyar
Center for Academic Innovation
University of Michigan
Ann Arbor, MI, USA
magyarn@umich.edu

Figure 1: Codespec logo.

CCS CONCEPTS

- **Applied computing** → **Interactive learning environments**;
- **Human-centered computing** → **Usability testing**.

KEYWORDS

Computer Programming Practice, Adaptive Learning Systems, Scaffolding

ACM Reference Format:
Carl Haynes-Magyar and Nathaniel Haynes-Magyar. 2022. Codespec: A Computer Programming Practice Environment. In *Proceedings of the 2022 ACM Conference on International Computing Education Research V.2 (ICER 2022), August 7–11, 2022, Lugano and Virtual Event, Switzerland.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3501709.3544278

1 ABSTRACT

Computing education researchers (CERs) theorize that learning how to program requires the development of at least four skills: code reading and tracing, code writing, pattern comprehension, and pattern application [see 30]. Developing these skills requires deliberate practice which involves engaging in tasks specifically designed to improve ones' skills in a particular domain [11]. Web-based interactive programming practice environments such as CloudCoder [23], CodeChef [7], CodeLab [1], CodeWars [8], CodingBat[25], Edabit [19], and LeetCode [17] can be considered informal and/or open-ended learning environments (OELEs) that support deliberate practice—the learner decides what to learn and when [3, 12, 24]. OELEs support self-directed learning [14, 16, 18] of computing topics by inspiring programmers to complete projects that are meaningful to them [21], by helping programmers build self-confidence, and by increasing programmers' sense of control over their learning process [5]. But the absence of a predefined curriculum and/or scaffolding, which are tenets of OELEs [12, 16, 18], has led to such environments being criticized for failing to help learners grasp basic computer programming concepts [4, 20]. Learners may form inadequate mental models of basic programming concepts [15], struggle to find good resources, and struggle to get help from real experts when learning in these environments [5]. Hence, Begel and Ko [3], in their chapter on informal learning, call for researchers to investigate whether learning technologies should "structure learning for learners" or whether learners should "be taught how to structure their own independent learning" *outside* of the classroom [p. 763].

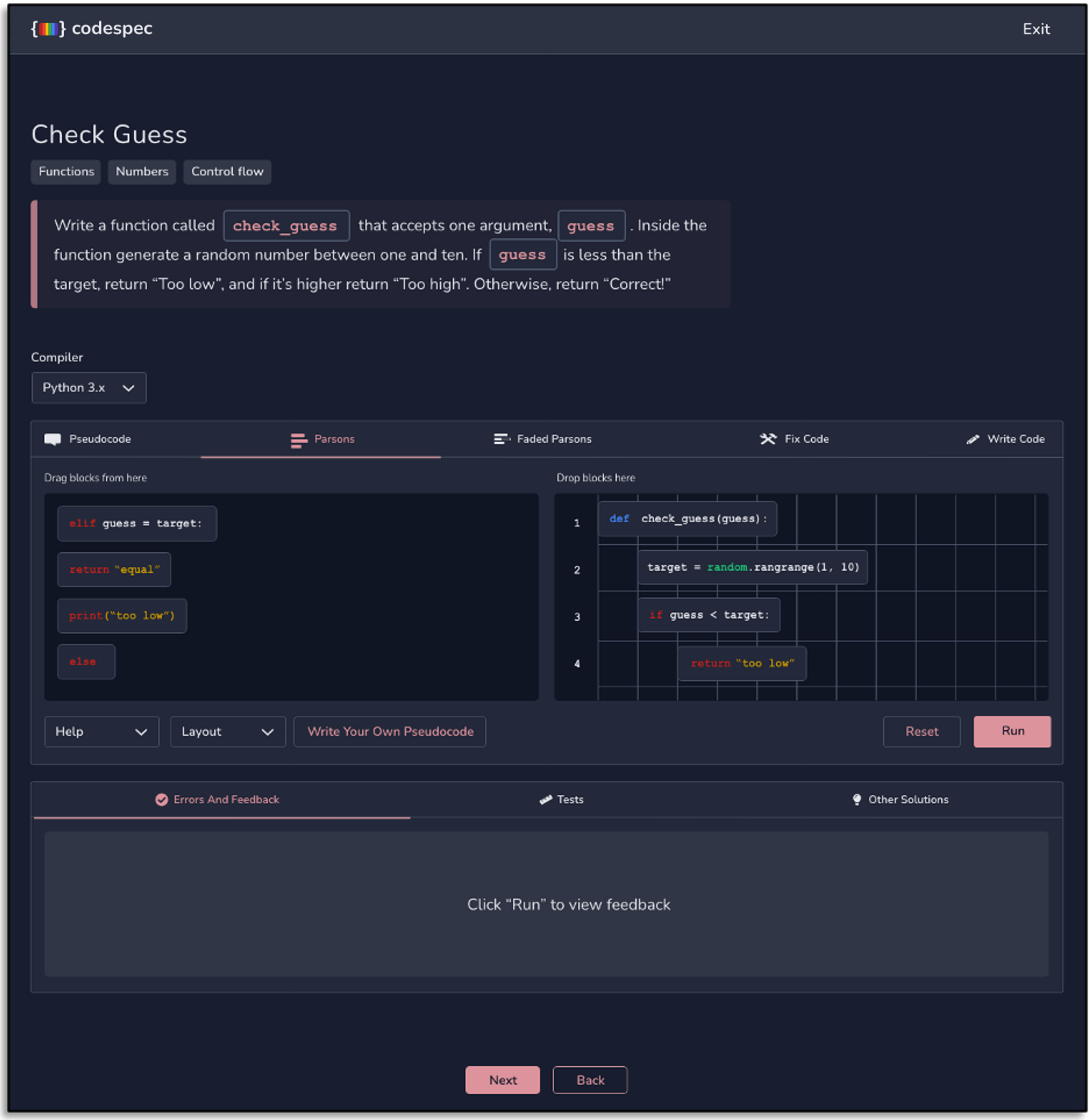
We are designing a self-directed learning environment for computer programming practice called Codespec. Its problem space area offers learners the option to switch between solving a problem as either a pseudocode problem (also described as subgoals and programming plans) [6, 22], a Parsons problem [9, 26], a Faded Parsons problem [28], a fix code problem [10], or a write-code problem. Runestone [13] is the only environment that plans to offer programmers the option to solve the same problem as a Parsons problem or write-code problem, and Crescendo aims to support scaffolding for independent learning [27].

Our prototype features a "Help" menu with the options to combine blocks, predict code, reveal distractors, show pseudocode, suggest changes, and provide indentation. Evidence shows programming plans (pseudocode) and purpose-first scaffolds can motivate novice programmers to learn programming [see 6]. We will explore how to incorporate pseudocode across each problem type. And while it is more efficient to solve Parsons problems than fixing and writing code [10], including the latter two types of problems can increase our potential to support a broad range of learners' abilities [29].

Our preliminary research questions are based on prior research on scaffolding self-directed learning with personalized learning and learner modeling in computer science education research (CSEd) [see 2, 18]. They are (1) how do we develop models for personalized learning across the different problem types for an introduction to programming? And (2) what is the effect of personalized scaffolding on self-directed learning in Codespec's adaptive learning environment? To evaluate our prototype and develop a model for personalized learning, we plan to interview stakeholders in computing education and conduct an experiment similar to [18].

ICER 2022, August 7–11, 2022, Lugano and Virtual Event, Switzerland

ACM ISBN 978-1-4503-9195-5/22/08.
https://doi.org/10.1145/3501709.3544278



Check Guess

Write a function called check_guess that accepts one argument, guess. Inside the function generate a random number between one and ten. If guess is less than the target, return "Too low", and if it's higher return "Too high". Otherwise, return "Correct!"

Drag blocks from here

```
elif guess = target:
return "equal"
print("too low")
else
```

Drop blocks here

```
def check_guess(guess):
    target = random.rangrange(1, 10)
    if guess < target:
        return "too low"
```

Click "Run" to view feedback

Figure 2: Codespec's practice area on the Parsons problem tab.

The goal is to design a system with diverse stakeholders that will 1) benefit programmers, instructors, and researchers, and 2) cultivate inclusion, diversity, equity, accessibility, and sexual orientation and gender identity awareness (IDEAS).

REFERENCES

[1] Valerie Barr and Deborah Trytten. 2016. Using turing's craft codelab to support CS1 students as they learn to program. *ACM Inroads* 7, 2 (2016), 67–75.

[2] Satabdi Basu, Gautam Biswas, and John S Kinnebrew. 2017. Learner modeling for adaptive scaffolding in a computational thinking-based science learning environment. *User Modeling and User-Adapted Interaction* 27, 1 (2017), 5–53.

[3] Andrew Begel and Amy J Ko. 2019. Learning outside the classroom. (2019).

[4] Vicki E Bennett, Kyu Han Koh, and Alexander Repenning. 2011. CS education re-kindles creativity in public schools. In *Proceedings of the 16th annual joint conference on innovation and technology in computer science education*. 183–187.

[5] Jonas Boustedt, Anna Eckerdal, Robert McCartney, Kate Sanders, Lynda Thomas, and Carol Zander. 2011. Students' perceptions of the differences between formal and informal learning. In *Proceedings of the seventh international workshop on Computing education research*. 61–68.

[6] Kathryn Cunningham, Barbara J Ericson, Rahul Agrawal Bejarano, and Mark Guzdial. 2021. Avoiding the Turing Tarpit: Learning Conversational Programming by Starting from Code's Purpose. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–15.

[7] Directi. 2021. CodeChef. https://www.codechef.com/

[8] Nathan Doctor and Jake Hoffner. 2021. Codewars. https://www.codewars.com/

[9] Barbara Ericson, Austin McCall, and Kathryn Cunningham. 2019. Investigating the Affect and Effect of Adaptive Parsons Problems. In *Proceedings of the 19th Koli Calling International Conference on Computing Education Research*. 1–10.

[10] Barbara J Ericson, Lauren E Margulieux, and Jochen Rick. 2017. Solving parsons problems versus fixing and writing code. In *Proceedings of the 17th koli calling international conference on computing education research*. 20–29.

[11] K Anders Ericsson, Ralf T Krampe, and Clemens Tesch-Römer. 1993. The role of deliberate practice in the acquisition of expert performance. *Psychological review* 100, 3 (1993), 363.

[12] Michael J Hannafin, Craig Hall, Susan Land, and Janette Hill. 1994. Learning in open-ended environments: Assumptions, methods, and implications. *Educational Technology* 34, 8 (1994), 48–55.

[13] Carl C Haynes and Barbara J Ericson. 2021. Problem-Solving Efficiency and Cognitive Load for Adaptive Parsons Problems vs. Writing the Equivalent Code. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–15.

[14] Malcolm S Knowles. 1975. Self-directed learning: A guide for learners and teachers. (1975).

[15] D Midian Kurland and Roy D Pea. 1985. Children's mental models of recursive LOGO programs. *Journal of Educational Computing Research* 1, 2 (1985), 235–243.

[16] Susan M Land and Michael J Hannafin. 1996. A conceptual framework for the development of theories-in-action with open-ended learning environments. *Educational Technology Research and Development* 44, 3 (1996), 37–53.

[17] LeetCode. 2022 [Online]. The World's Leading Online Programming Learning Platform. https://leetcode.com/

[18] Tobias Ley, Barbara Kump, and Cornelia Gerdenitsch. 2010. Scaffolding self-directed learning with personalized learning goal recommendations. In *International conference on user modeling, adaptation, and personalization*. Springer, 75–86.

[19] Matt MacPherson. 2021. Edabit. https://edabit.com/

[20] Richard E Mayer. 2004. Should there be a three-strikes rule against pure discovery learning? *American psychologist* 59, 1 (2004), 14.

[21] Robert McCartney, Jonas Boustedt, Anna Eckerdal, Kate Sanders, Lynda Thomas, and Carol Zander. 2016. Why computing students learn on their own: Motivation for self-directed learning of computing. *ACM Transactions on Computing Education (TOCE)* 16, 1 (2016), 1–18.

[22] Briana B Morrison, Lauren E Margulieux, Barbara Ericson, and Mark Guzdial. 2016. Subgoals help students solve Parsons problems. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. 42–47.

[23] Andrei Papancea, Jaime Spacco, and David Hovemeyer. 2013. An open platform for managing short programming exercises. In *Proceedings of the ninth annual international ACM conference on International computing education research*. 47–52.

[24] Seymour A Papert. 2020. *Mindstorms: Children, computers, and powerful ideas*. Basic books.

[25] Nick Parlante. 2021. CodingBat. https://codingbat.com/

[26] Dale Parsons and Patricia Haden. 2006. Parson's programming puzzles: a fun and effective learning tool for first programming courses. In *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52*. 157–163.

[27] Wengran Wang, Rui Zhi, Alexandra Milliken, Nicholas Lytle, and Thomas W Price. 2020. Crescendo: Engaging students to self-paced programming practices. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 859–865.

[28] Nathaniel Weinman, Armando Fox, and Marti A Hearst. 2021. Improving Instruction of Programming Patterns with Faded Parsons Problems. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–4.

[29] Jacob O Wobbrock, Shaun K Kane, Krzysztof Z Gajos, Susumu Harada, and Jon Froehlich. 2011. Ability-based design: Concept, principles and examples. *ACM Transactions on Accessible Computing (TACCESS)* 3, 3 (2011), 1–27.

[30] Benjamin Xie, Dastyni Loksa, Greg L Nelson, Matthew J Davidson, Dongsheng Dong, Harrison Kwik, Alex Hui Tan, Leanne Hwa, Min Li, and Amy J Ko. 2019. A theory of instruction for introductory programming skills. *Computer Science Education* 29, 2-3 (2019), 205–253.