# Steps Learners Take When Solving Programming Tasks, and How Learning Environments (Should) Respond to Them

Johan Jeuring*
Utrecht University
The Netherlands
j.t.jeuring@uu.nl

Hieke Keuning*
Utrecht University
The Netherlands
h.w.keuning@uu.nl

Samiha Marwan*
University of Virginia
Raleigh, USA
samarwan@ncsu.edu

Dennis Bouvier†
Southern Illinois University
Edwardsville / US Air Force Academy
USA
djb@acm.org

Cruz Izu
The University of Adelaide
Australia
cruz.izu@adelaide.edu.au

Natalie Kiesler
DIPF Leibniz Institute for Research
and Information in Education
Germany
kiesler@dipf.de

Teemu Lehtinen
Aalto University
Finland
teemu.t.lehtinen@aalto.fi

Dominic Lohr
Friedrich-Alexander-Universität
Germany
dominic.lohr@fau.de

Andrew Petersen
University of Toronto Mississauga
Canada
andrew.petersen@utoronto.ca

Sami Sarsa
Aalto University
Finland
sami.sarsa@aalto.fi

## ABSTRACT

Every year, millions of students learn how to write programs. Learning activities for beginners almost always include programming tasks that require a student to write a program to solve a particular problem. When learning how to solve such a task, many students need feedback on their previous actions, and hints on how to proceed. In the case of programming, the feedback should take the steps a student has taken towards implementing a solution into account, and the hints should help a student to complete or improve a possibly partial solution. Only a limited number of learning environments for programming give feedback and hints on intermediate steps students take towards a solution, and little is known about the quality of the feedback provided. To determine the quality of feedback of such tools and to help further developing them, we create and curate data sets that show what kinds of steps students take when solving programming exercises for beginners, and what kind of feedback and hints should be provided. This working group aims to 1) select or create several data sets with steps students take to solve programming tasks, 2) introduce a method to annotate students' steps in these data sets, 3) attach feedback and hints to these steps, 4) set up a method to utilize these data sets in various learning environments for programming, and 5) analyse the quality of hints and feedback in these learning environments.

## CCS CONCEPTS

• **Social and professional topics → Computer science education**; **Software engineering education**.

## KEYWORDS

Learning programming, tutoring systems, automated feedback

**ACM Reference Format:**
Johan Jeuring, Hieke Keuning, Samiha Marwan, Dennis Bouvier, Cruz Izu, Natalie Kiesler, Teemu Lehtinen, Dominic Lohr, Andrew Petersen, and Sami Sarsa. 2022. Steps Learners Take When Solving Programming Tasks, and How Learning Environments (Should) Respond to Them. In *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol 2 (ITiCSE 2022), July 8–13, 2022, Dublin, Ireland*. ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3502717.3532168

## 1 OVERVIEW

There are many learning environments that support beginners learning how to program, including intelligent tutoring systems (ITSs) [4], online environments[1], and serious games [6]. A number of these learning environments give feedback on potentially partial student solutions, and hints on how to proceed with a partial solution [5, 7, 8, 10, 13].

What can we say about the quality of learning environments that support learning programming step by step? How can a learning

*co-leader
†The views expressed are those of the author and do not reflect the official policy or position of the US Air Force, Department of Defense or the US Government.

[1]e.g. codecademy, Datacamp, Khan academy, code.org.


ACM ISBN 978-1-4503-9200-6/22/07.
https://doi.org/10.1145/3502717.3532168

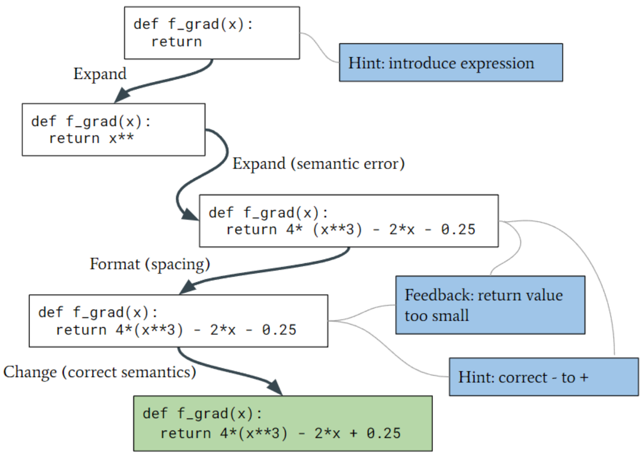


**Figure 1: Example of annotating student steps.**

environment best support a student solving a beginner's programming problem? One way would of course be to perform experiments with different environments, and to compare the learning outcomes. Setting up such experiments is not easy [1], and if we find a difference we would still like to know the cause(s) for that difference. Why does practicing in one environment result in better learning outcomes than in another?

Formative feedback and hints are essential aspects of learning [14]. ITSs that provide feedback and hints on steps of students show positive results [7, 8, 15]. One reason that might explain why some learning environments better support students is the quality of their feedback and hints. One way to compare the quality of feedback and hints is to create a data set consisting of student steps towards a solution, let experts annotate the data set with feedback and hints, thus creating a golden data set. We can then compare the feedback and hints of a learning environment with this golden data set [11].

Creating such a data set raises at least two questions. First, what do we consider to be a student step? What is the granularity of a step? Most teachers would consider neither a single keystroke nor a complete solution to be a single step: one is too small and the other too large. Second, how do we give feedback and hints on a step? To give a hint we need to know where a student should go, so we require that the steps in a data set are taken to solve a particular task. When we compare the behavior of a particular learning environment against a golden data set we need to answer additional questions, such as: can we mimic solving the exercise addressed in the golden data set in the learning environment, what if the learning environment uses a slightly different syntax than the golden data set, and how do we deal with potential differences of step granularities between the golden data set and the learning environment?

*Motivating example.* Figure 1 shows an example of steps taken by a student to solve a programming problem in Python. This example is taken from a data set that captures a snapshot of a student's current source code whenever a student changes the code and then does not apply further changes for at least two seconds [9].

Five program states are shown, and each step is annotated: an *expansion* (e.g. specifying a return value), a *formatting step* that does not change the semantics of the code (e.g. adding spacing for readability), and a *semantic change* by correcting an error (e.g. changing the minus operator to a plus operator). Additionally, the blue boxes contain expert hints and feedback on certain steps. Note that this is just an example of how such steps could be annotated; we will design the coding methodology in the working group.

*Goals.* This working group has five goals: 1) determine the desired characteristics of the data sets we want to use in our research, and collect existing data sets, or set up experiments in which such data sets can be obtained. A number of such data sets are already available [2, 3, 9, 10]. We will use, and if necessary extend, the ProgSnap2 format [12] to describe our data sets; 2) design a coding for characterising a student step, and annotate the steps in the data sets using this coding; 3) design a coding for annotating the steps in the data sets with feedback and hints; 4) set up a method to utilize the data sets in various learning environments, and 5) use the annotated data sets to evaluate learning environments for programming by comparing their feedback and hints with the expert-authored ones.

## REFERENCES

[1] Joseph E. Beck, Kai Min Chang, Jack Mostow, and Albert Corbett. 2008. Does help help? Introducing the Bayesian Evaluation and Assessment Methodology. In *Proceedings of the International Conference on Intelligent Tutoring Systems.*
[2] Neil Christopher Charles Brown, Michael Kölling, Davin McCall, and Ian Utting. 2014. Blackbox: A Large Scale Repository of Novice Programmers' Activity. In *Proceedings of the ACM Technical Symposium on Computer Science Education (SIGCSE).*
[3] code.org. 2014. Hoc4 and Hoc18 datasets. (2014). https://code.org/research.
[4] Tyne Crow, Andrew Luxton-Reilly, and Burkhard Wuensche. 2018. Intelligent Tutoring Systems for Programming Education: A Systematic Review. In *Proceedings of the Australasian Computing Education Conference.*
[5] Alex Gerdes, Bastiaan Heeren, Johan Jeuring, and L. Thomas van Binsbergen. 2017. Ask-Elle: an Adaptable Programming Tutor for Haskell Giving Automated Feedback. *International Journal of Artificial Intelligence in Education* 27, 1 (2017).
[6] Andreas Giannakoulas and Stelios Xinogalos. 2020. A review of educational games for teaching programming to primary school students. *Handbook of Research on Tools for Teaching Computational Thinking in P-12 Education* (2020).
[7] Samiha Marwan, Ge Gao, Susan Fisk, Thomas W. Price, and Tiffany Barnes. 2020. Adaptive Immediate Feedback Can Improve Novice Programming Engagement and Intention to Persist in Computer Science. In *Proceedings of the ACM Conference on International Computing Education Research (ICER).*
[8] Samiha Marwan, Joseph Jay Williams, and Thomas Price. 2019. An Evaluation of the Impact of Automated Programming Hints on Performance and Learning. In *Proceedings of the International Computing Education Research Conference.*
[9] Benjamin Paaßen. 2019. Python Programming Dataset. (2019). https://doi.org/10.4119/unibi/2941052 Bielefeld University.
[10] Thomas W. Price, Yihuan Dong, and Dragan Lipovac. 2017. iSnap: towards intelligent tutoring in novice programming environments. In *Proceedings of the ACM SIGCSE Technical Symposium on computer science education.*
[11] Thomas W. Price, Yihuan Dong, Rui Zhi, Benjamin Paaßen, Nicholas Lytle, Veronica Cateté, and Tiffany Barnes. 2019. A comparison of the quality of data-driven programming hint generation algorithms. *International Journal of Artificial Intelligence in Education* 29, 3 (2019).
[12] Thomas W. Price, David Hovemeyer, Kelly Rivers, Ge Gao, Austin Cory Bart, Ayaan M. Kazerouni, Brett A. Becker, Andrew Petersen, Luke Gusukuma, Stephen H. Edwards, and David Babcock. 2020. ProgSnap2: A Flexible Format for Programming Process Data. In *Proceedings of the ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE).*
[13] Kelly Rivers and Kenneth R Koedinger. 2017. Data-driven hint generation in vast solution spaces: a self-improving Python programming tutor. *International Journal of Artificial Intelligence in Education* 27, 1 (2017).
[14] Valerie J. Shute. 2008. Focus on formative feedback. *Review of Educational Research* 78, 1 (2008).
[15] Kurt VanLehn. 2011. The Relative Effectiveness of Human Tutoring, Intelligent Tutoring Systems, and Other Tutoring Systems. *Educ. Psych.* 46, 4 (2011).