"Early X or Late X" Questions for Discussing Curricular Practices in CS1 and CS2

Martin Henz
henz@comp.nus.edu.sg
National University of Singapore
Singapore

ABSTRACT

In teaching university entry-level calculus, it proved useful to distinguish early from late transcendentals depending on the time at which transcendentals such as the exponential and logarithmic functions are introduced. I suggest we pose analogous "early X or late X" questions for first-year computer science courses. I propose a tentative list of concepts for which the "early or late" question might be worthy a discussion and argue that the approach allows us to pay attention to pedagogical choices rather than the choice of the programming language for CS1 and CS2.

ACM Reference Format:
Martin Henz. 2023. "Early X or Late X" Questions for Discussing Curricular Practices in CS1 and CS2. In *Proceedings of the 54th ACM Technical Symposium on Computing Science Education V. 2 (SIGCSE 2023), March 15–18, 2023, Toronto, ON, Canada.* ACM, New York, NY, USA, 1 page. https://doi.org/10.1145/3545947.3573240

1 MOTIVATION

The first calculus textbook by late Torontonian and mathematics educator James Drewry Stewart [1] follows the traditional approach of *late transcendentals* where the transcendental logarithm function is introduced as an integral of $f(x) = 1/x$ and the transcendental exponential function as its inverse. Stewart and others realized that this approach deprives the concepts preceding integrals of these two transcendentals as interesting example functions. Stewart's *early-transcendentals* alternative [2] introduces them—along with transcendental trigonometric functions—informally in the first chapter and uses these transcendentals throughout the book.

Do we have analogous pedagogic choices in CS1 and CS2? An exhaustive approach would list all essential concepts in the first year in CS and analyse their prerequisite dependencies. To initiate a more focused discussion, I propose instead to limit the discourse to a handful of concepts, and explore their "early or late" question.

2 A LOADED QUESTION

"You say you teach CS1. What's your programming language?" The question deprives me of the opportunity to share essential features of my approach. When I oblige, the questioner is bound to draw unwarranted conclusions. Imagine the analogous question posed to Stewart: "You say you teach calculus. Do you use Mathematica, Maple, or Matlab?" The analogy is increasingly fitting, because like these tools, some of the most popular languages used in CS first-year classrooms are converging. Python, OCaml, Java, and JavaScript might differ in syntactic details, but they are all lexically-scoped and support object-oriented programming, the possibility of statically-checked type annotations, and higher-order functions.

3 POSSIBLE CANDIDATE CONCEPTS

To illustrate the "early or late" approach of discussing a CS first-year curriculum, I will present a few candidate concepts in my lightning talk, including:

Types: Early types establish generally useful programming habits but might slow down the teaching of essentials and might lack proper motivation.

Objects: Early objects provide a scaffolding for modeling real-world problems and prepare students for software design but might impose a conceptual overhead on programming.

Recursion: Early recursion provides useful examples for data structure traversal and for algorithmic complexity but might impose a significant learning obstacle; initial difficulties in mastering recursion might discourage some students.

Higher-order: Introducing higher-order functions early places due emphasis on programming abstractions but like recursion might impose unwarranted learning obstacles.

Assignment: Early imperative programming allows some students to draw on prior programming experiences but might complicate mental models of computation, and might lead to a lack of attention to mental models later on.

4 DISCUSSION

Seen in this way, any given pedagogical approach to the first year in CS can be roughly characterized by its coordinates in an n-dimensional design space, where each concept represents a dimension. This approach might let educators see curricular practices in a new light and gain new insights, possibly allowing them to improve their teaching. It also might help them question the choices they have made and in what ways these choices were constrained.

I hope that this lightning talk motivates a more in-depth empirical analysis of the choices that are manifested in the current pedagogical approaches in CS and fruitful discussions in the CS education community regarding the relative merits of the available choices, while avoiding the idiosyncrasies that arise around the choice of programming languages for CS1 and CS2.

REFERENCES

[1] James Stewart. 2015. *Calculus* (8 ed.). Cengage Learning, Boston, MA.
[2] James Stewart. 2015. *Calculus: Early Transcendentals* (8 ed.). Cengage Learning, Boston, MA.

SIGCSE 2023, March 15–18, 2023, Toronto, ON, Canada

ACM ISBN 978-1-4503-9433-8/23/03.
https://doi.org/10.1145/3545947.3573240