Leveraging the Potential of Abstraction in Programming Education

Abstraction is considered a fundamental part of computer science (Dijkstra, 1972), and can be interpreted either as "changing the resolution" or as separating the "what" (interface) from the "how" (implementation) (Statter & Armoni, 2020). Experts move flexibly between levels of abstraction and can see things simultaneously "in the large" and "in the small" (Knuth, in Hartmanis, 2007). It is possible that the majority of errors students make are in fact abstraction errors; such is certainly the case when students move between abstraction levels in elementary mathematics (Rich et al., 2019).

Teaching abstraction is a challenge, however. Novices especially tend to gravitate toward lower levels of abstraction: They get hung up on detail (such as syntax), focus on a particular case in itself (rather than a representative of something more general), and these tendencies are increased by unfamiliarity and discomfort (Hazzan & Zazkis, 2005). Meanwhile, teachers (and other experts) may operate on several levels of abstraction at once, often without being aware of it (Hazzan, 2003).

Armoni's (2013) framework for teaching abstraction to computer science novices aims to remedy this by recommending that teachers (a) call explicit attention to which level of abstraction they operate on, (b) similarly call attention to moves between such levels, and (c) give students opportunities to reflect on their own abstraction processes. This approach has been shown to improve both students' abstraction abilities and their general performance in computer science courses (Statter & Armoni, 2020).

Importantly, widely used programming languages such as Python makes abstraction visible and explicit on a structural level: Loops, functions, and classes all allow students to abstract away details when considering the higher-level program. A simple way of looking at it is that each time the indentation is increased, students move to a lower (more detailed) level of abstraction. Whether students interpret indentation in Python in this way is an open question, however.

How students think about or work with Python's built-in levels of abstraction is therefore an interesting avenue for further research. One can also look into how explicit attention to levels of abstraction by the teacher influences these processes in a programming context. How do students that demonstrate understanding of abstraction approach complex problems compared to those who do not, and to what extent are they able to generalise a good understanding of abstraction in one context (such as programming) to another (such as mathematics)?

Armoni, M. (2013). On Teaching Abstraction in CS to Novices. *Journal of Computers in Mathematics and Science Teaching*, *32*(3), 265–284.

Dijkstra, E. W. (1972). The humble programmer. *Communications of the ACM*, *15*(10), 859–866. https://doi.org/10.1145/355604.361591

Hartmanis, J. (2007). Turing award lecture: On computational complexity and the nature of computer science. In *ACM Turing award lectures* (p. 1993). Association for Computing Machinery. https://doi.org/10.1145/1283920.1283949

Hazzan, O. (2003). How Students Attempt to Reduce Abstraction in the Learning of Mathematics and in the Learning of Computer Science. *Computer Science Education*, *13*(2), 95–122. https://doi.org/10.1076/csed.13.2.95.14202

Hazzan, O., & Zazkis, R. (2005). Reducing Abstraction: The Case of School Mathematics. *Educational Studies in Mathematics*, *58*(1), 101–119. https://doi.org/10.1007/s10649-005-3335-x

Rich, K. M., Yadav, A., & Zhu, M. (2019). Levels of Abstraction in Students' Mathematics Strategies: What Can Applying Computer Science Ideas about Abstraction Bring to Elementary Mathematics? *Journal of Computers in Mathematics and Science Teaching*, *38*(3), 267–298.

Statter, D., & Armoni, M. (2020). Teaching Abstraction in Computer Science to 7th Grade Students. *ACM Transactions on Computing Education*, *20*(1), 8:1-8:37. https://doi.org/10.1145/3372143