Computational Thinking in Junior High School

Marcela Santillán
Universidad Pedagógica Nacional de México

Curriculum Approach

The three proposed pillars of computational thinking are not parallel, but in a hierarchy. Just as mathematical thinking and abilities are not all necessarily at the same level, so it is with all abilities and the knowledge and cognitive skill, which sustain them. Traditional computational thinking, and the abilities it is meant to enable, depends on both, traditional literacy, like reading and writing, and on a certain level of mathematical skills and maturity.

The second pillar, mathematical modelling through computer coding, assumes coding skills. Rests on a certain level of familiarity with the first pillar and requires more math maturity and skills than the first and a certain level of mathematical modelling skills, beyond the mathematics, it requires some ability applying the mathematics to model phenomena, and thus also some intuitive, if not formal, understanding of scientific models.

Finally, machine learning, the third pillar, is a fairly advanced subject in computer science, and it is a curiously specific one. It requires plenty of ease and familiarity with the first two pillars, but it also demands from the student immersion in a very particular and abstract paradigm of mathematical modelling through computers, adoption or acceptance of the mindset or perspective, which generates said paradigm, plus their acquisition of a certain bag of tricks.

It is vox populi in mathematical circles, that when levels of abstraction are piled on with education proposals, such as in mathematical education, the more abstract levels can hardly be attained if the more concrete levels are shaky. To put it another way, mathematical maturity takes time to develop, with intuition in a certain level required before it one can make sense on the more abstract levels, which depend on it. All this is to say that the students must reach a certain level of ability and intuition in the first pillar before they can appreciate anything beyond divulgation of the other two, let alone approach problems.

A sound approach to the first pillar is, therefore, necessary. A well-structured curriculum for it must be researched, adopted, and used to teach concepts such as: variables, conditional statements, control flow, data structures, control structures, and so on. Whether they are considered aspects of the first pillar or prerequisites, skills such as traditional literacy, familiarity with the use of computer (including the hardware, the operating system, relevant applications, et cetera), and mathematical thinking and abilities, need to be included. After a certain familiarity and their skills at problem solving show maturity, acquiring skills at mathematical modelling through coding is attainable. When, in turn, those skills show maturity, only then approaching the third pillar make sense. Once must here mention that many other advanced computer coding and science skills will also then make sense, and it seems somewhat arbitrary that machine learning be singled out as the third pillar.

Propose approach to teaching the preliminary skills mentioned above

We will use the python command line (and/or a similar command line console). It is a command line with a well thought out and simple interactivity. The language, python, is elegant, sparse on syntax, readable, high level, free (open source sense, and this openness is ideal for education) and particularly well suited for mathematics (it was developed by a mathematician and this has been one of the goals of the language since its inception).

We will use it as scientific calculator to teach mathematical concepts. We have a fair amount of experience with this approach to teach mathematics. The interpreter is use as a capable interlocutor, such that the students learn to make “mathematical utterances” in the same way that a child first acquires language, by saying them to this interlocutor and exploring the sense, they make through the feedback thus obtained. We will thus be building mathematical competency, while at the same time giving them familiarity with the computer and with the syntax of a very suitable computer language, used for mathematical modelling by experts.

At some point, we will make a subtly transition from using the command line as a scientific calculator to entering simple code, a function that this command line is built to facilitate. We then continue teaching simple mathematical problems through coding. We must then slowly make a transition to a programmers’ editor, while still relying on the command line, as an interlocutor, to explore the language by testing it.

A further transition involves switching from using coding to solve mathematical problems, to using coding to model natural phenomena. Here, again, we must, at first, keep it simple, because modelling is an open-ended activity, and it is easy to be bogged down in detail.

It is possible, of course, to somewhat short-circuit the long and elaborate process described, by, with appropriate software, providing a taste of the power of machine learning (or other advanced topics, for that matter). We mean, by appropriate software, one that hides much of the complexity while providing enough control to play, explore, and experiment. Still, while this provides motivation to explore these techniques, and perhaps some exposure to the basic concepts that underlie the techniques, any deeper learning and eventual mastery of the techniques would have to wait for the prerequisite skills to be developed and the maturity they imply. Nevertheless, since intrinsic motivation is central to learning, it is desirable to compile a list of software that fulfils this function, if there is any, as a resource for us and for the community at large.