How to teach Computer Programming to Children

A teacher's perspective

This is a set of observations and practical guidelines that I have come up with, based on my experience of teaching computer programming to school children. Teaching anything to anyone can be a huge challenge – unless of course you are a born teacher. Teaching programming to children is certainly a specialized activity – one that can be enjoyed and made effective through experience and continuous learning. Mostly, it's about realizing that students actually learn by themselves and the teacher can serve best by becoming a facilitator, mentor, and coach.

General Observations:

- *Programming is about a child teaching the computer, not the computer teaching the child*. As teachers, we must keep this in mind – we must allow children to choose what and how they want to teach their computers.

- *Theory when appropriate*: Generally computer training courses are laden with concepts and acronyms. Students get buried in theory quickly before getting a chance to appreciate the purpose of all of it. Instead of starting with a lot of theory and concepts, you should start with real problem-solving right away and explain concepts as and when necessary. Thus the approach would be to teach theory only when required.

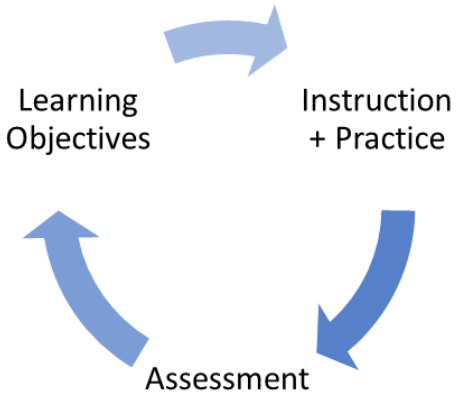
- *Understanding is important, facts are not*: For school-age children, understanding of underlying principles (why is it so, how does it work, etc.) is more important and interesting. Focus on these ideas and CS principles more than specific language commands and features.

- *The evolution of a solution is very interesting and insightful*. It is tempting (and often necessary due to the lack of time) for teachers to present experiments that validate facts and theories, and also to present readymade solutions to associated problems. All this is interesting. But students benefit immensely if they also get to learn how the experiments were devised or how the solutions were cooked up, because then they would have learnt problem-solving methods and the secret to creativity.

- *When you are learning English, you learn to read first.* When kids learn a language such as English, they first spend considerable time hearing conversations and reading books. Writing usually comes much later. Learning computer programming is thought to be similar to this. Students must get the opportunity to read good programs before starting to write their own. The best way to 'read' a program is to step through it during execution.

Teaching Methodology:

In general, the teaching/learning process looks like this:



This means every lesson must have learning objectives (e.g. "students will learn about looping") for which you will have material to explain the required concepts and practice assignments for students to apply their learning. Finally, you will have the means to assess their learning (which could simply be the students' ability to complete the assignment).

- As a general principle, spend about 20 to 30% time talking, and leave rest of the time for actual work (programming).

- Try to match your pace with the pace of learning. Every group of students is different, so it does not make sense to have a fixed "syllabus" for every period/session.

- Try to achieve two goals: (a) Every student should get a sense of learning/achievement through his/her work; (b) Every student should feel challenged. Luckily, in programming this is possible because a "smart" student can take on a harder approach or add more features to his/her program. As a teacher, you can simply have "optional" features in every assignment which should be taken up if possible.

- Update your notes after every session and record what was covered, what issues came up, is there some new/old topic that should be addressed next time, etc.

- Have a programming assignment for every session. This means, at the end of every session, students should be able to finish some "chunk" of work. If the program is complex, tell the students what part of it they should try to finish in that session, and leave the rest for subsequent sessions.

- Emphasize to students the truth (about programming) that there is no right or wrong – there can be different solutions to the same problem. Also, there is no "perfect" program; every program can be improved if there is time and interest. All the software out there has known

"bugs".

- It is important to let students know that it is OK to make mistakes. The truth is, even the best programmers can never get their programs working in the first pass. In fact, it is GOOD to make mistakes, because, that is how you make new discoveries, you learn new things.

Recommended Routine for Every Session:

Generally, students are eager to get to their programs and are not in a mood to listen to you. So, you have to carefully orchestrate the available time to get what you want done.

- Begin with a few review questions. This is to reinforce the main ideas and to ensure they have understood the concepts. Refer to "what we learnt" for the project to decide which concepts to review.
- Take questions from students if any.
- If project/program is finished, have students demo if they want to.
- Go to new topics.

Before Starting:

To achieve the objective of learning, it is necessary to make sure you have:

- Children's attention: So rules of shutting monitors off and coming closer to the teacher when teacher is talking, etc.
- Visibility of your demo: Every student must be able to see your demo screen (projector or LCD) clearly.
- Activity: Ensuring that (1) There are no issues with the computers, language (e.g. Scratch) software, working environment, etc., and that (2) Children get help when they are blocked in their work (program bug, conceptual understanding etc.).

For remote classes:

Challenges of teaching remotely:
(1) You don't have the benefit of looking over the shoulders of your students. So if they are stuck or need help, you won't know unless they tell you by email/WhatsApp.
(2) You need an adult supervisor to ensure kids are following your instructions, especially when you are presenting something.

Written by: Abhay B. Joshi (abjoshi@yahoo.com)
Last updated: 3 July 2017