VPython: Learning to code: tips

A.C. Norman
ACN.Norman@radley.org.uk

Adapted by ACN from *Part IB Introduction to computing* notes (2007) by D.J.C. Mackay and (2015) by C. Lester

Self-assess!

Coding is not taught as an exam-based for-credit course at Radley: it is either available as an extra-curricular activity, or taught as a non-examined skill inside of lessons. Therefore, when you are learning to code, you will mainly be self-assessing your progress. This is a great way to learn in general, provided that you are interested in learning, capable of choosing goals for yourself, and capable of self-evaluation. All of these are valuable for learning in the future when you are adults.

Programming is an essential skill, just like being able to draw and interpret graphs. If you don't know about graphs, what do you do? — practise, read books, talk to teachers and fellow students, practise some more. During your time at school, you will learn lots about graphs which will become second nature, and you should be able to use them in many of the subjects you study. Similarly, programming is a valuable tool for thinking, exploring, investigating and understanding almost any scientific or technical topic. You will find that programming can help you in many areas of study. You ought to do your best to get up to speed on programming, for the benefit of all of your courses.

There are different types of programmers. Some people like to get into the nuts and bolts and write programs from scratch, understanding every detail — like building a house by first learning how to make bricks from clay and straw. Other people are happy to take bricks as a given, and get on with learning how to assemble bricks into different types of building. Other people are happy to start with an existing house and just make modifications, knocking through a wall here, and adding an extension there. I don't mind what sort of programmer you become. All these different skills are useful. I encourage you to learn whatever skills interest you. There are opportunities for various styles of activity.

Many skills are best acquired gradually. Programming is one such skill, so ideally your learning should be spread out over the course of your school career. Don't try to rush through or cram the learning of this important skill.

The main form of assessment for all of this learning will be self-assessment. Ask yourself "have I mastered this stuff?" If not, take appropriate action. Think, read, tinker, experiment, talk to other people. Sort it out.

Collaboration

How you learn to program is up to you, but let me make a recommendation:

> I recommend that you do most of your programming work in **pairs**, *with the weaker programmer doing the typing*, and the stronger one looking over his/her shoulder.

Working in pairs greatly reduces programming errors. Working in pairs means that there is always someone there to whom you can explain what you are doing. Explaining is a great way of enhancing understanding. It's crucial to have the weaker programmer do the typing, so that both people in the pair understand everything that's going on.

Copying

When programming in real life, copying is strongly encouraged.

- Copying saves time.
- Copying avoids typing mistakes
- Copying allows you to focus on your new programming challenges.

Similarly, when learning to program, copying may well be useful. For example, copy a working program similar to what you want to do; then modify it. Feel free to copy programs from the internet. The bottom line is: "do you understand how to solve this sort of programming problem?" Obviously, copying someone else's perfect answer verbatim does *not* achieve the aim of learning to program. Always self-assess. If you don't understand something you've copied, tinker with it until you do.