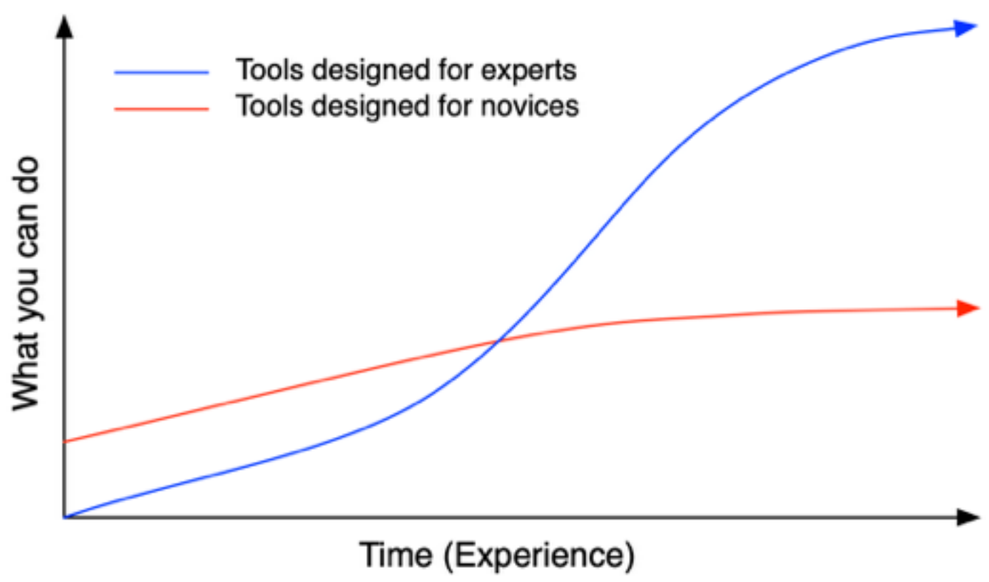
Programming Tools

Expert Tools

Programming is all about making something. Whenever you make something, you do well to invest time and effort into learning the tools that help you with that task. Over the years, programmers have developed a wide variety of tools that support development efforts in various ways. If you want to become a serious programmer, mastering these tools is crucial.

New programmers often wonder why they should invest the effort into learning these sorts of tools. It is *possible* to program with more-familiar seeming environments, which require less up-front effort to learn the basics. For example, you could write and compile your programs in a graphical environment, which will have a more familiar "buttons and menus'' interface. Choosing tools like these which are designed for the ease of novices represent a short-term benefit and long-term loss. If you are studying programming casually (e.g., just taking one or two required courses), then the time investment to learn these tools is not likely to be worthwhile. However, if your goal is to become a professional-level programmer, you will want to become well-versed in the tools of your trade.



The figure to the right shows the long-term tradeoff of using a tool designed for novices versus using a tool designed for experts. The x-axis of this graph represents time spent learning and using the tool. The y-axis represents proficiency (what you can do) with the tool.

The red line shows the progression of proficiency with a tool designed for a novice. At time 0—when you first start using the tool—you have a basic proficiency with it. This basic proficiency stems from the fact that the tool is setup to be easy for novices—it is "user friendly.'' As you spend more time with the tool, you learn more features and tricks, but your proficiency quickly plateaus as you reach the limits of your tool.

The blue line shows the progression with a tool designed for expert use. At time 0, the tool is difficult to use. It does not fit with the paradigms you are used to. As you spend time and effort learning the tool, your proficiency increases. At some point, your rate of learning increases too as you become familiar with the terminology and paradigms of the tool—you know what to look for, what to ask about, and where to look when you do not know something. As you continue to learn, your proficiency progresses past the plateau you could achieve in the tool designed for novices. You may eventually plateau, but when you do, that plateau will be much higher with the tool designed for experts. For some tools, you may never plateau—Drew has used Emacs for 20 years and still learns new things regularly.

This tradeoff is not unique to programming. Professional photographers, for example, use equipment that gives them full control over their art. They make decisions about shutter speed, aperture, and light sensitivity every few minutes. This level of control is likely overkill for taking casual photos. And a novice user might find that their first dozen photos taken on professional equipment are blurry, over-exposed, or that they missed the shot entirely while fiddling with camera settings. At the same time, few professional photographers would give up their professional equipment and replace it with a "point-and-click'' camera for any serious artistic undertaking.

Another reason to invest time and effort into learning programming tools (if you want to be a professional programmer) is the perception associated with your tool choices. Using the tools of an expert programmer (especially if you use them well) sets up the perception that you are an expert programmer. Several students have reported that when interviewing for jobs, the fact they used the tools described in this lesson was important in interviews.

Think of the photography analogy: would you hire a photographer who only knows how to use cheap disposable cameras?

Along those lines, we note that a recent grad thanked us for teaching Emacs, and said "I had a virtual interview recently for a machine learning position and **the interviewer was impressed I was using Emacs** to complete their coding challenges."

Introductions the Tools We'll be Using

Introduction to Linux

Introduction to Emacs

Introduction to Git

Mastery Learning Platform