THE SPECTRUM OF PROGRAMMING

In the beginning, everyone who worked with computer software was a programmer. It was the most natural thing. Software is the lines of code that make the computer chip, connected to its peripheral devices, do binary arithmetic and thus "process data" in ways that we want. To work with computers was to write those lines of code, in order to see what the results would be. Everyone was a programmer.

The people who handled the operating system became known as administrators, as they only need to work with and manipulate the instruction set that is given, and not invent lines of code from scratch.

Eventually, routine tasks became standardized and "etched in stone." The core of software that all computers need to run any program became known as the operating system. The people who handled the operating system became known as administrators, as they only need to work with and manipulate the instruction set that is given, and not invent lines of code from scratch. Key software applications, like word processing programs and spreadsheets and databases and web servers, were commoditized for great profit, and the running of them was left to operators — people skilled in using the program, but not needing to be capable of writing the program itself. These developments were hugely liberating for computers as tools for human beings, releasing them from the confines of the computer lab, and putting them in everyone's office and in everyone's home.

There is a similar evolution that occurred with automobiles. When the horseless-carriage was invented, powered by an internal combustion engine, the only people who could own or drive one were mechanics, the rich, and eccentrically-inclined enthusiasts. Only with the development of the electric starter and the affordability and reliability offered by Henry Ford's assembly line production technology was the automobile transformed into the 20th century's democratic icon of the age of speed. There still is a need for mechanics, who know in detail how an automobile is built and how it functions, but the overwhelming number of people are drivers only, who merely need some legally-required minimum of skill in the operation of the complex device that is an automobile.

Most people don't know binary arithmetic, and don't care to know. Boolean logic, with its terms like "logical and" and "logical not" and "logical exclusive or" are terms that belong to another language, as far as they are concerned. The fact that these terms are at the heart of everything their computer does as it carries out its functions is irrelevant, because this fundamental logic is not brought to their attention. The programming that underlies the operation of the computer is hidden behind slick graphical interfaces, command line "front ends", and other user-friendly features.

The job of being a programmer has become a specialization. There are large numbers of people who make their career supporting computers — system administrators, operators, help desk technicians, database administrators, and so on — who don't know a programming language. There is a sentiment that has grown up in the industry that the "classic" programs have all been written, and that the task of programmers is to improve upon them to exploit new hardware, to tweak them for customized purposes, and perhaps (if they are very lucky) come up with the next great thing — come up with a killer application like the World Wide Web.

It seems that there is little room in present day computer technology for a true Renaissance man. Where is there a place for an honest Jack-of-all-trades, as was absolutely essential in the world of computers from the 1940s until the 1970s? There are whispers of it still. A distribution of the Linux operating system includes all of the source code — but how many people look at it,

or are capable of manipulating it in a way that was commonplace before the great commoditization of computer operating systems and applications that occurred in the 1980s? Not many.

I would argue that programming should be understood as a spectrum of knowledge, and that doing so avoids the pitfalls of thinking of programming as something that only programmers do. Even ordinary home users want to automate routine tasks, and might figure out a way to get "scripts" to do so. For example, if you figure out a way to turn on your computer, and have it automatically open your electronic mail application and then check for messages without manual intervention, then you have done a little bit of programming. A system administrator can write down a series of tasks to be accomplished in a batch, and then schedule this job to be carried out at regular intervals — that's programming, and it can be done by writing a shell script, for example. A web hostmaster can code a lookup to a back-end database for presentation on a dynamic web page — that's programming, and it can be done by writing a Perl program, or other script using a programming language suited to the Common Gateway Interface used by web server applications. Someone looking to build a little stand-alone tool, or widget, might do so using the Java programming that can run on any platform that supports its run-time environment. It's all programming, even though it might not all be done by someone who has the title "Programmer" in the signature file at the bottom of their email messages.

Programming is a spectrum that runs the gamut from simple automation of routine actions to full blown development of stand-alone applications that are unprecedented. Programming is not just for programmers. Programming has a place for everyone that uses computers.

Source : https://www.exitcertified.com/blog/michael/2012/the-spectrum-of-programming/