COMPUTER LITERACY FOR EVERYONE*

Luby Liao and Jack W. Pope
University of San Diego
San Diego, CA 92110 USA
liao@sandiego.edu, pope@sandiego.edu

ABSTRACT

The authors have been teaching a very successful university computer literacy course since 1984. A computer literacy course should teach students to be passionate about and efficient with computers, and it should engender students with an appreciation of the power computing and networking tools provide today. In the authors' courses, students learn useful skills, but that is a secondary goal. The authors use the World Wide Web as a vehicle for proactive learning and publishing; and they encourage collaboration using web and programming applications tools to enable true literacy. This paper will review those aspects of the instructors' methodology that contribute to the success of the course.

COMPUTER LITERACY: PASSIONATE BUT NOT OBSESSIVE

For at least two decades, the term computer literacy has been controversial [4]. Wikipedia [10], however, gives a simple, but reasonable working definition of computer literacy. "Computer literacy is the knowledge and ability a person has to use computers and technology efficiently. Computer literacy can also refer to the comfort level someone has with using computer programs and other applications that are associated with computers. Another valuable component of computer literacy involves the knowledge of how computers work and operate."

Both authors have been teaching a university computer science course for non-Computer Science majors for quite some time and are aware of the evolution of the meaning of "computer literacy" over time. One author has taught such a course since 1984 and the other since the late 1980s. Throughout that time, both have always called this basic course a computer literacy course. Since the course does not satisfy any general education requirement nor does it count for the computer science major, students

who elect to take the course are doing so because they want to learn about the power of networking and computer technology rather than for a specific skill set.

In the authors' view, a computer literate person is not just efficient with computers, but is also passionate about them, just as a musically literate person is almost certainly passionate about music. Passion for computers ideally comes from an appreciation of the beauty and power of computer science. Rote learning, on the other hand, is known to kill passion.

Therefore, first and foremost, both authors convey and instill this passion for computers in the course through real life problem solving based on computer science principles and best practice. Students actively publish to the web not only to express themselves, but also as a way to communicate (e.g., posting something for others to use). Students are also taught the importance of networking and backup capabilities on the network using simple mechanisms such as ssh/scp, and other network technologies such as SMB (samba). We stress the paramount importance of automation using software tools or creative solutions through programming. The instructors make critical comparisons between open and proprietary software and also discuss new and important computing technologies such as VOIP system Skype, and social and server-side book marking services such as del.icio.us.

Increasingly, more and more people become obsessed with and addicted to computers. They spend an inordinate amount of time with computers and little time with human interaction. A good percentage of such people self-destroy by spending their useful hours playing games, chatting on line, emailing, etc; but spending little or no time to improve themselves . Students must understand the grave consequences of such obsession and recognize the signs of such addiction. Certainly, students should seek assistance in such cases. To quote a recent report entitled South Korea opens boot camp to confront cyberspace addiction from International Herald Tribune[5]:

Up to 30 percent of South Koreans under 18, or about 2.4 million people, are at risk of Internet addiction, said Ahn Dong Hyun, a child psychiatrist at Hanyang University who just completed a three-year government-financed survey of the problem.

COMPUTER LITERACY: WHAT TO TEACH

This section outlines the authors' two different approaches to teaching computer literacy. Both stress the importance of networking technology, software tools and programming. Though neither class is a programming course in its entirety, a substantial portion of the course is devoted to programming.

First Author's Approach

To fire up students' passion, one can choose among many topics that induce oohwow, and holycows, and that immediately empower students to solve real-life problems. Driven by this thinking, the first author's course consists of the following four components:

1. Use of client/server technology of the Internet. This includes using ssh/scp to manage user accounts and using an IMAP client such as Mozilla Thunderbird to

manage email. The course demonstrates how the HTML anchor element enables the Web, and the HTML form element facilitates eCommerce and dynamic web inquiries. It uses Dreamweaver, Mozilla Composer, iWeb, simple editors and blog to actively publish to the web throughout the semester.

2. Use of software tools. In previous years, these included résumé templates and publishing those to the web; mail merge; spreadsheets and charting. More recently, we use Google Docs & Spreadsheets to make students understand the impact of having documents stored on the server. (In fact, this paper practices the collaboration we teach. When writing this paper, one author was in Taiwan and the other in Palm Springs. They collaborated using Google Docs and Skype.)

In contrast to Microsoft Office, documents are now accessible wherever we have Internet access because the documents are stored on the server.

- Collaboration becomes natural and easy for the same reason. In my department, when it comes time to decide the next semester's teaching schedule, faculty would email their course and time preference to the whole department of twenty colleagues. This clutters our inboxes and wastes our time. A much better way is to use a program like Google Spreadsheet. Another example: if an instructor and her teaching assistants use Google Spreadsheet to manage their class's grade book, they no longer need to use email to communicate. While TAs enter assignment scores, the instructor can simultaneously enter the Midterm Exam scores, all without the latency or inconsistency caused by emailing.
- Versioning is integrated; this is useful even when we work alone on a document.

3. Programming using Python. Programs we write include an alarm clock simulator, an arithmetic tutor and a crude blackjack program. We can do so much in such a short time only because the Python programming language does not get in the way of the programmers, thus allows them to concentrate on programming. This is an extremely important point, but is often under-estimated. Allow me to use an analogy. I often hear people talk about the difficulty of the concept of object orientation or design patterns. I would beg their consideration that such things may become easy if we use a different language. Consider another example -- adding numbers. This is trivial when you use Arabic number system. What if you use Roman numerals?

4. Survey of recent important computing developments that empower us, such as the proliferation of podcasts and audio books, Google Docs and Spreadsheets, Skype, and del.icio.us.

There are other topics that can be covered instead of, or in addition to these. This author believes the topics chosen are not as important as the direction the instructor takes. The second author's different and equally successful approach proves this point succinctly. The fact is that it is impossible to teach everything we wish to teach, or we hope the students should learn. In the spirit of life-long learning, it is more useful to make students interested, teach them the skills to learn on their own, and entice them to continue to learn. In view of this, the first author's class is problem-driven and hands-on. All problems are real as opposed to being concocted. This is crucial for keeping students

interested and engaged. The author also assigns readings from fun books such as [1] or [6].

Second Author's Approach

The second author also seeks to instill in his class a sense of excitement about modern digital technology and its effect on not only learning, but also our culture. The course includes:

1. Reading assignments and in-class discussions on technologies and applications that impact our use of computers. These certainly include current hot topics such as podcasting, Google-earth, and social networking sites of various types - e.g., MySpace, Match.com. But there are also readings and discussions of the impact of technology on today's culture. For this, discussions point out the considerable differences in world view of authors such as Nicolas Negroponte in "Being Digital" [8] and Neil Postman in "Technopoly." [9]

2. A review of basic computing, web and Internet origins and technology including:

- the history and development of the modern Internet; the origins of the World Wide Web from text based browsers to modern day browsers;
- the origins of the World Wide Web from text based browsers to modern day browsers;
- basic components of a computer system and its reliance on digital (on-off, bits and bytes) storage of information;
- an introduction to the html markup language and cascading style sheets using group projects and requiring straight html coding. Later, the author allows the use of web coding tools such as Dreamweaver, but other simpler code editors are used first to encourage learning the basic tags and css style structure and hierarchy;
- a discussion of modern relational databases and how these fit into the context of dynamic information on the web. The course uses both Microsoft Access and, later, MySQL as examples of databases where the students get hands on experience with the design of a database and the use of SQL to extract information.
- an introduction to the PHP language as a middleware tool between the web server and the database. There are several group and individual projects that stress rudimentary programming skills where students learn to recognize and use loop structures and how the PHP language provides access to the back end database and allows for the creation of dynamic html code.

3. The course concludes with a web based essay or presentation that each student presents to the class and presides over a discussion about the topic. The topics are related to the use or pervasiveness of digital technology in our culture. Each student suggests a topic to the instructor and after some refinement of topic and careful review of what is to be covered, the students develops both the design and content for the essay.

The instructor's intended outcomes are twofold. First, the students should come out with a basic understanding of just how directly modern digital technology impacts and affects how we communicate. This should include recognition of the strength and weaknesses of digital communication. Secondly, the students should have a basic knowledge of how web servers work and interact with databases through the use of an application programming interface such as PHP.

III. BEST PRACTICES

As the authors solve different problems during the semester, they introduce appropriate contexts that encourage good computer and programmming habits. These habits invariably help people to become more productive. Here are some examples.

1. the authors advise students to never use white spaces, punctuations and capital letters in their file names. Such characters make inter-operation across operating systems and programs unexpectedly troublesome. An example is the conversion of a MS Access database to a Postgresql database. The problem is that Access database table and attribute names can contain spaces, punctuations and capitals -- none of which work naturally in Postgresql.

2. While the mouse is a useful navigational tool, it can be slow and inefficient, so keyboard shortcuts are encouraged in hands on sessions. For example, instead of mousing over menu items, use ctrl-s to save and ctrl-n to create a new object. We encourage use of universal key strokes such as control-C and control-V for copy and paste, shift-clicking for selecting a region, control-clicking for selecting non-adjacent regions, and shift-clicking for highlighting a region.

IV. WHAT A COMPUTER LITERACY CLASS SHOULD NOT BE

Computer literacy should empower students to solve problems. It is noble and intellectually challenging to teach Turing machines, limits of computation, Halting problems etc. But this is unlikely to inspire average students or empower them to solve their immediate and urgent and everyday problems. This is particularly true of students who may have an interest in basic concepts of networking and computing but are not likely to be computer science programming professionals. Even so, a number of our computer literacy students have continued their computer science and education by taking the majors' programming courses. A few have even become computer science majors.

A basic computer literacy course should avoid becoming a software training class. For example, a computer literacy class must not be a training class for software from a particular vendor, for example, Microsoft Excel or Adobe Dreamweaver. While the course may use these tools, it should emphasize fundamental principles that outlive particular software or vendors. Here are two examples.

One author uses Microsoft Excel in the course, but he is not training his students to use Excel per se. Instead, he takes the opportunity to illustrate that spreadsheets are, in the final analysis, computerized worksheets for calculation with sorting capability. If the demand for multiple views and extended queries for data subsets is paramount, a spreadsheet is not the most appropriate tools and a relational database should be used.

A spreadsheet's usefulness is severely limited by its basic design as a 'worksheet.'. On the other hand, for database chauvinists, some small problems are better suited for spreadsheet applications. Hence, the author provides hand on examples that encourage students to recognize the fundamental differences between the spreadsheet model and the database model. The author uses Google Docs & Spreadsheets in addition to, and in some cases instead of, Microsoft Word or Excel. Spreadsheet applications provide meaningful comparison between Microsoft office suite with free and open source alternatives such as Open Office.

The second author incorporates basic web design and mark up as part of a course component that leads to the study of databases and how the interaction facilitates dynamic web design and information delivery. While it is tempting to build the course component around a sophisticated design tool such a Dreamweaver, the students understand the concepts and are able to better assimilate the underlying HTML concepts by initially using a simple text editor and then augmenting the application and database server components with preset code that is carefully explained and broken. This is followed by a few basic programming examples that the students do on their own. Only at the end of the sequence do the students learn how to navigate a more technical web design tool - in this case Dreamweaver.

Finally, a computer literacy course must avoid becoming a computer training class. Instead, PCs or Macs with their respective operating systems are used to understand the fundamental concept of file systems and to run client/server and inter-operable programs. When students ask what computers we will be using, the authors' response is that the computer platform does not matter. Students understand this not abstractly, but as a result of using different PC and Mac labs and server platforms throughout the semester.

The authors contend that too many faculty teach computer literacy, and even some computer science courses, in such a way that the distinction between education and software training becomes unclear.

V. THE IMPACT OF COMPUTER LITERACY

The authors' own experience teaching a computer literacy course over the years has been very positive. But, more than that, the students themselves benefit from the course. For example, a former student of one author is now 'the' official web coordinator for the university and manages a team of Information Technology staff who help with the design and implementation of web policies at the University. Another former student, whose experience in the computer literacy course led her to take additional computer science course work is a technical support assistant for that group. And, indeed, over the years, a number of students from the non-majors course have become computer science majors based on their experience in this introductory course.

No one will argue that math competency is fundamental and that a demonstration of competence should be required for successful completion of a higher education degree. But computer competency is even more fundamental. It impacts people more than math and more directly than math. While most folks do not solve equations, inequalities or perform computations on a regular basis, most interact with computers daily.

The authors strongly believe that universities should require computer literacy course as a graduation requirement. Lack of such requirement sends a strong and, in the authors' opinion, harmful message to the students that computer literacy is not important, it can be self-taught, or attained naturally in some ways. The authors' twenty plus years of experience teaching computer literacy shows that none of this is true. Here are some of our observations:

1. On their own, students usually learn as little as they can. In the case of word processing software such as MS Word, students will learn to type and save, and little more. In the case of MS Excel, again they type and save and will not even learn about entering formulas. They will continue to pay for MS Office, get little out of this software other than typing and saving, and never consider open source and free alternatives.
2. The public tends to choose sub-optimal tools and programming languages. An example is the fact that MySQL was the most popular open source database management system. Yet it did not begin to offer any server-side features such as views, stored procedures and triggers until 2005. Postgresql, among others, offered all such features all along, but was much less favored.

As a recent session (2007) at the Educause Learning Initiative meetings pointed out: "Most universities do not require a computer literacy course in the core curriculum. Critical information technology competencies are often taken for granted, to the detriment of students who lack basic computer and Internet skills."[7] Sadly, this contrasts drastically with a popular faculty view - and here the authors paraphrase a colleague's opinion that as more faculty incorporate software into their courses, it will become increasingly unlikely that a student would graduate without having learned more about working with software in an academic environment.

Unfortunately, this is what the authors believe computer literacy should not be: software training. While software training is useful, will a random series of software training sessions suffice for a computer literacy education? Computer literacy is not the same as plumber training, and to treat it as such does not do justice to the breadth, depth and sophistication of computer science as a discipline. In practice, will such unplanned and uncoordinated training sessions suffice to prepare graduates to compete in this world made flat by information technology[5]?

The authors are not alone in their efforts to reconstitute basic computer literacy outside of rote programming and software skills. Thomas Cortina, for example, notes that "for non-technical non-majors (i.e. students in majors that were not purely mathematical, scientific or engineering oriented), the preciseness and detail needed to write computer programs correctly was overwhelming." This led Cortina to develop an introductory course for non-majors that emphasizes the understanding of algorithms and computational principles.[10]

In the courses reviewed here, students undoubtedly learn useful skills, but that is not the goal. Students learned to program to make things happen; but make no mistake that such short exposure does not make them programmers. Our courses offer students a starting point for life-long learning. It makes a statement that while skills are useful and important, proactive learning, publishing, collaborating and programming assist in learning key computing concepts and, as well, enable true literacy.

REFERENCES

[1] Comer, Doug, "The Internet Book" (4th edition), Prentice Hall, 2006

[2] Cortina, Thomas J, "An Introduction to Computer Science for Non-majors Using Principles of Computation", ACM SIGCSE Bulletin , Proceedings of the 38th SIGCSE technical symposium on Computer science education SIGCSE '07, March 2007

[3] Friedman, Thomas, "The world is Flat", Farrar, Straus and Giroux; April 18, 2006

[4] Harvey, Brian, "Stop Saying 'Computer Literacy'!", based on an article in Classroom Computer News, 1983, http://www.cs.berkeley.edu/~bh/stop.html

[5] International Herald Tribune, South Korea Opens Boot Camp to Confront Cyberspace Addiction, 2007-11-18, http://www.iht.com/articles/2007/11/18/business/boot.php

[6] Lehnert, Wendy, "Light on the Web", Addison Wesley, 2002

[7] Murray, Meg, Myers, Martha, Pérez, Jorge, "Learner-Centered Reflections on Computer Literacy in the Digital Age", Educause Learning Initiative Annual Conference, January, 2007

[8] Negroponte, Nicholas "Being Digital", Knopf, 1995

[9] Postman, Neil, "Technopoly - The Surrender of Culture to Technology", Vintage Books, 1993

[10] Wikipedia: Computer Literacy, http://en.wikipedia.org/wiki/Computer_literacy