Udacity CS101: Building a Search Engine
Unit 7: Fun Field Trips

Intro

A huge part of learning computer science is learning how to use it to solve everyday problems. In today's unit you will visit companies, universities and non-profits, as well as meet individuals who are using computer science to change the way people think about the world.

First, we'll review some of the main themes in the course.

Summary of Main Themes

Much of the course, and much of computing, revolves around three main themes:

Abstraction refers to a way of hiding details so that you can use one thing (like a procedure) in many different ways without needing to understand the details.

Example:
Procedural abstraction. You can write one procedure to do lots of different things, depending on what the inputs are. You do not need to know exactly what sequence of instructions you will execute. It is important to know what the procedure does.

Universality. Recall the toaster! Compared to a toaster that has one purpose, to toast, a computer is a universal machine because it only requires a few simple constructs to be able to define any program. Once you have a few simple operations, you have enough to program every possible algorithm. All you need is:

a. a way to make decisions; i.e. if statements
b. a way to keep track of things; i.e. variables, data
c. a way to keep going; i.e. while, recursive procedures

Recursive definitions enable you to use very simple rules to define infinitely many, complex things. Recursive definitions define rules for things in terms of simpler versions of themselves.

Example:

Expr *Expr* **+** *Expr*
Expr **Number**

Here is how the concepts you have learned throughout this course fit into the three main themes:

		Abstraction	**Universality**	**Recursive Definitions**
Unit	**Topic**			
1	Variables: x=3	x		
2	Procedures: def proc(x):	x	x	x (unit 6)
3	Lists: [3, [4,5]]	x (Data Abstraction)	x	x
4	Search Index	x		
	Network Protocols	x		
5	Cost	x	x	
	Hash Table	x	x	x
6	Recursive Procedures		x	x

Final Quiz

What is Computer Science?

a. Engineering
b. Science
c. Liberal Art

All three are true!

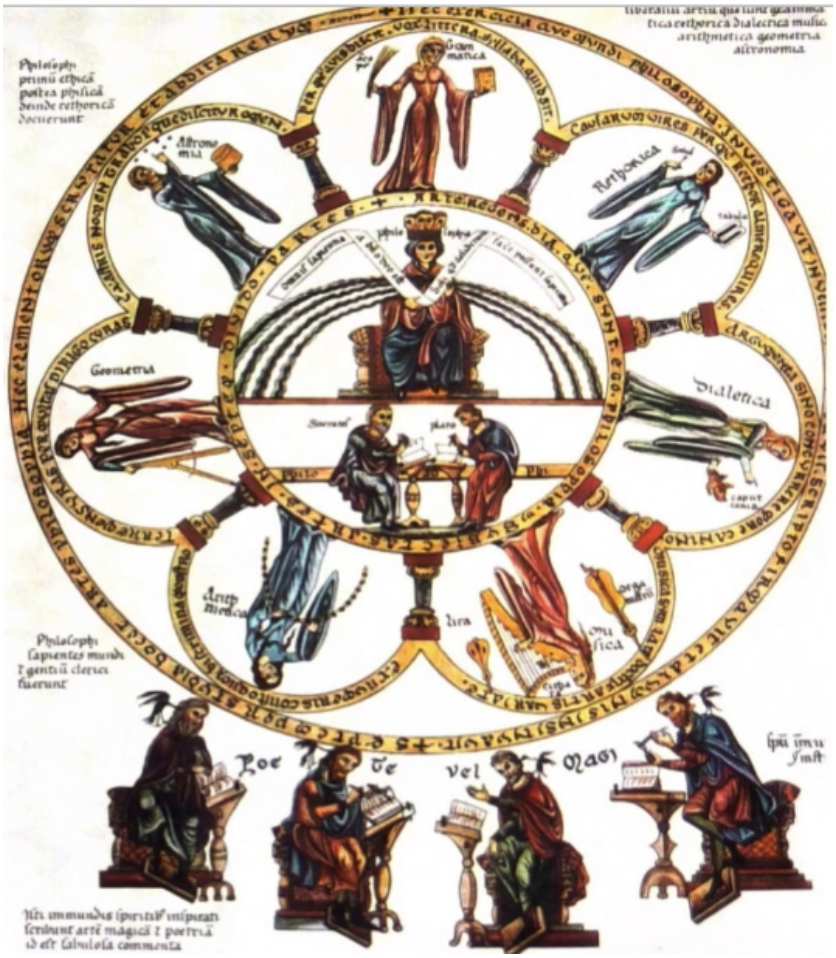
Engineering. Traditional engineering is about building complex structures, like a bridge, to solve physical problems, like crossing a bay that separates two cities. Computer science builds virtual structures, to solve problems like finding the best web page for a given query. Engineering is about "design under constraint". In traditional engineering, the main constraints come from physical properties of The questions posed within the discipline of engineering are always subject to the laws of physics and for this reason the field can be described as, "design under constraint."

While the laws of physics (for example, we have seen how the speed of light impacts network latency and computation time), are important and relevant to computer science, they are not the main constraints we face. In computer science the biggest constraint is the limits of human minds. The human mind is limited by creativity as well as the amount of information it can keep track of. The main tool we have for dealing with this is *abstraction*. By using abstraction well, we can hide details to focus only on what we need to understand at one time to be able to solve the problem.

Science. Science is about understanding nature through observation. Computing is a key tool in modern science, and most science today is about using computers to analyze data and building and doing experiments with computational models. On the other hand, computer science is mostly not about nature directly, but about abstract things like numbers, text, and procedures.

There is, however, lots of computation in nature. Examples include the growth of biological systems based on the programs encoded in their DNA, evolution itself as a computational process, and understanding how the neurons in the brain lead to complex thoughts. To understand the information processes in nature, we need to understand computer science. The deep questions in computing are also deep scientific questions. These are questions about what can and cannot be computed in our universe in a reasonable amount of time.

Liberal Art. The traditional liberal arts were envisions as what people who were wealthy enough to not need to learn a trade would study (hence the name liberal, meaning "free"). The traditional seven liberal arts (depicted in the drawing by Herrad von Landsberg) are the *trivium* (language) - grammar, rhetoric, and logic; and *quadrivium* (number) - arithmetic, geometry, music, and astronomy. All of these have strong connections with computing.



Grammar. Computing is all about using language to precisely describe processes, and grammar is a big part of that. We have seen many uses of grammars in this class, including how to precisely describe constructs in Python.

Expr *Expr* **+** *Expr*

Rhetoric. Rhetoric is about using language to communicate and persuade between speakers. In computing, we need interfaces to connect procedures to other procedures, and protocols to allow two hosts on a network to communicate. In Unit 4, we saw how network protocols like HTTP allow programs running on different computers to communicate with each other.

GET /index.html HTTP/1.1

Logic. Logic employs decision making constructs to formalize thinking. It is the art of thinking. Constructing programs is all about using logic. A simple example is the if statement:

if n==0: return 1

Arithmetic. We use arithmetic to perform computations with numbers.

return fibo(n-1) + fibo(n-2)

Geometry. We haven't done much in this course that is directly related to geometry, but it is important in computing especially for graphics (which requires understanding the geometry of light to synthesize images) and networking (which requires finding good paths across a network).

Music. Music has much in common with computing, including harmonic structures that are recursively defined. (Read Douglas Hofstadter's *Gödel, Escher, Bach* to see the deep connections between Bach's music, Escher's drawings, and Gödel's logic.)

Astronomy. Astronomy is number in space and time. This is the hardest one to claim as part of computing, although essentially all astronomy today is doing using computing.

Now for some field trips!

Computer History Museum

About

The Computer History Museum started in the 1970s in Boston and moved to California in the 1990s. The museum is located in Mountain View, California and chronicles the history of computers illustrating how computing has made an impact on the human experience. It is the world's largest history museum for the preservation and presentation of artifacts and stories of the Information Age.

Between 1750 and 1850, the Industrial Revolution developed technology to serve as a tool for human work in manufacturing, farming and transportation. Since that time, technology has shifted its focus, becoming a method for serving the human mind. It is the relics and developments from this period that are exhibited at the Computer History Museum. Read about the importance of preserving this history in Why a Computer History Museum.

Features

Revolution, the museum's permanent exhibition, which was opened in 2011 chronicles the first 200 years of computing –from the abacus to the smart phone.

The Babbage Engine, is one of the museum's prized relics. Envisioned by Charles Babbage in 1849, the machine's function is to produce mathematical tables without any human intervention, and send them all the way to the print shop. While the machine was not built in Babbage's lifetime, a working machine has since been built and is on display.

The World's First Hard Disk Drive was commissioned to IBM by the Air Force and was announced in 1956. The Air Force requested it to fulfill the need to store their 50,000 spare parts inventory records. With this drive records could be accessed in one second! It was the first product of IBM's laboratory in San Jose, CA. The machine has 50 disks, 24 inches in diameter, and stores a total of 5 million 6-bit characters (3.75 megabytes) - this is tiny by today's standards (a typical 1TB drive today holds over 290,000 times as much), but was huge for 1956.

More

If you can't come to California, there are other great computing museums around the world. Here are a few notable museums you may want to visit:

Bletchley Park (near London, United Kingdom) This is the site where Alan Turing and tens of thousands of others worked on breaking the Nazi encryption codes during during World War II. Some of the earliest electronic computers were developed here, including the bombe that Alan Turing designed to break the Enigma code.

National Cryptologic Museum (Fort Meade, near Washington D.C., USA) Visiting the National Cryptologic Museum is probably the closest you will get to the National Security Agency. The museum houses lots of exhibits on cryptography and early computing equipment that was design specifically to break codes.

Do you know of any interesting computer museums? Please post them in the forums!

Stanford's National Accelerator Center

The Stanford National Accelerator Center uses computing to solve the mysteries of the universe!

About

The center was founded to build a 2-mile long linear accelerator. An accelerator functions to accelerate particles creating new states of matter. Accelerating particles also lets scientists explore the nature of matter and encourages experimental science by outputting a huge amount data.

Computing is very important when it comes to analyzing the large quantity of data provided by the accelerator.

Features

Richard Mount, Head of ATLAS Computing at SLAC National Accelerator Laboratory, talked about the lab's current Higgs boson experiment. Mount says that the current particle physics experiment requires 300 petabytes of data (one petabyte is 2 ** 50 bytes, 1000 Terabytes, or one quadrillion bytes). These experiments produce data at a rate of one petabyte per second!

Not familiar with the Higgs boson? The Guardian explains here.

Although physics also involves searching this data to find events with interesting properties and correlations between those events, it is very different from the text searching that web search engines do. This is because the data is structured, and the queries are about complex numerical properties, not just looking for string matches.

Mozilla

About

The Mozilla Foundation is a non-profit organization, launched in 2003, that supports the open-source Mozilla project in the absence of the bygone Netscape. Mozilla was the name given to the free and open-source software projects founded in 1998 to create a next-generation Internet suite for Netscape. Mozilla is best known for the Firefox web browser, but also leads many other software projects.

The Foundation describes itself as, "a non-profit organization that promotes openness, innovation and participation on the internet." The Foundation sets policies that govern the development of open source software, as well as controls trademarks and intellectual property. The Mozilla Manifesto, 10 principles that Mozilla believes "are critical for the internet to continue to benefit the public good as well as commercial aspects of life,' guides the Foundation. The manifesto has been translated into over 30 languages! Read the manifesto.

Get Involved

There are many ways to contribute to open source projects, and you don't need to be an expert to get started. Working with experienced programmers by getting involved in open source projects is a great way to develop your programming skills.

Once project, called "universal subtitles" lets users provide translation for YouTube videos that are recorded in any language. Mozilla software then automatically creates subtitles based on the translation you provided.

The Mozilla open source project is a great way to get involved with Mozilla and free software!

Check out the GitHub open source Python projects.

Students can also come for weekends to supplement their education and professionals come to renew their skill sets.

Benetech

About

Benetech is a non-profit organization driven to create new technology that enables people to improve their lives. Jim Fruchterman, is the Founder and CEO of the organization, which sprung from the pioneering work of Arkenstone, the world's leader in reading machines for the blind. Their mission statement reads:

"Together with partners and supporters worldwide, we use technology innovation and business expertise to solve unmet social needs. Leveraging the intellectual capital and resources of Silicon Valley, we create solutions that are truly life-changing. Our global endeavors have been instrumental in improving literacy, human rights and landmine detection. With expanded support we can accomplish much more."

Jim Fruchterman: “I think that being educated in a profession like software development, you have a responsibility to give back to society. Many people do this by getting a job and making products that have value. But the thing I want to talk to engineers about is: as you go out there and you see problems that are important to society and they can take advantage of your skills, do not ignore them just because they don't make a lot of money. I think that there are ways to make a living or volunteer to actually help these issues and I think that that is really an area of terrific satisfaction – as a toolmaker when you see the great things that people do with your tools. Taking big social problems and changing the world for the better, not just for more money.”

Get Involved

There are three ways that you can get involved with Benetech. The new "Social Coding for Good" project will provide ways to match up developers with projects like wikimedia or the guardian project that need help. The project is still in the pilot phase, but in the next year it aims to become a place like github where programmers can volunteer and gain some credit for doing something good.
Also see, volunteer opportunities at Benetech.

University of California, Berkeley

The University of California, Berkeley is one of the top computer science departments in the world. We visit with graduate students there working in a variety of computing areas to get a glimpse into the future of computing.

Brian Gawalt

Brian is a researcher interested in text mining and large-scale optimization. He's a big fan of the Bay Area's skiing and surfing, but especially the vibrant tech scene. Once as a goof, he had some software read the works of Shakespeare and keep tabs on word transition. That software now can generate new lines that sound an awful lot like the original material, though are in fact complete gibberish, and Brian pushes these fake quotes to the novelty Twitter account @Fakespearean

Sara Alspaugh

Sara Alspaugh grew up near Muncie, Indiana and went to the University of Virginia for her undergraduate studies, where she studied computer science and mathematics. At the time of this filming, she was an third year in the CS PhD program at the University of California, Berkeley. There, she works on ways to apply computer systems principles to electric demand management, as well as ways to increase the energy efficiency of computer systems themselves. In her free time, she does more work. And sometimes she plays indoor volleyball in a few bay area adult leagues.

Adrienne Felt

Isabelle Stanton

Isabelle is a PhD student in the Theory group at UC Berkeley. She received her Bachelors from a small women's liberal arts college, Mary Baldwin College, and Masters in Mathematics and Computer Science from the University of Virginia. At Berkeley, her research focuses on algorithmic questions related to social networks and voting protocols. Outside of Computer Science, she enjoys taking advantage of the natural beauty of the Bay Area from surfing to skiing to hiking.

Brielin Brown

Brielin Brown entered UC Berkeley's PhD program in 2011 after completing Physics and Computer Science degrees at the University of Virginia. Brown started taking computer science courses as an undergraduate physics major because he thought they would be "practical" in the event that being a physicist did not pan out. After taking his first theory of computing class he began to realize that, just like physics, computing underlies almost every aspect of scientific inquiry in a non-trivial way. Since then Brown has worked on quantum computing theory. Most recently he has been exploring computation biology. Brown is a self-diagnosed audiophile who enjoys surfing and playing ultimate frisbee, among other things.